EE2703 Assignment 9: Spectra of Non-Periodic Signals

Anvith Pabba [EE19B070]

30th May 2021

1 Abstract

The goal of this Assignment is to:

1. Find the Spectra of various non-periodic functions using numpy.fft.fft() and fftshift().
2. Using window functions (in this case, Hammings Window) to obtain a more accurate DFT.
3. Plotting and analysing the relevant graphs.

2 Introduction

In the previous assignment we found the DFTs of periodic signals using the numpy.fft.fft() command and it gave us very accurate results. In this assignment we see the limitations of numpy.fft.fft() in finding the DFT of non-periodic signals and we understand how we can utilize **Windowing functions** in order to get accurate DFTs.

Initially, we take a look at the Spectra of $sin(\sqrt{2}t)$ considering 64 samples over a range of $-\pi$ to π.

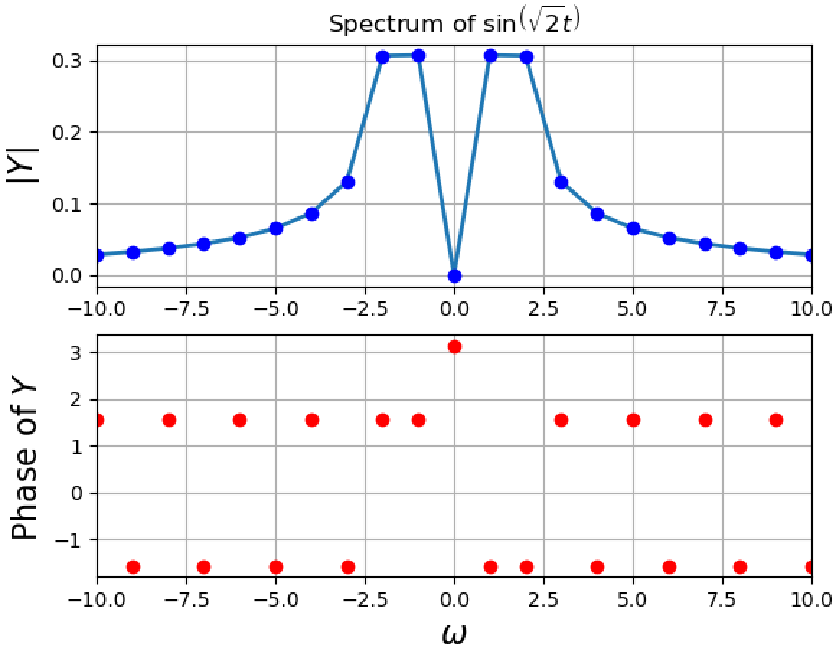


Figure 1: Spectra of $sin(\sqrt{2}t)$ with 64 samples

2.1 Analysing the output Spectrum

- We can clearly see that there are **4** peaks when there should only be **2** peaks at $\omega = \sqrt{2}$.
- We see that the phase plot also follows a similar trend.
- The reason for this is because the DFT only works for functions that have a time period of 2π. Which $sin(\sqrt{2}t)$ does not follow.
- We can see the non-periodic nature of $sin(\sqrt{2}t)$ in the following **Figure 2**

2.2 Non-periodic nature of $sin(\sqrt{2}t)$

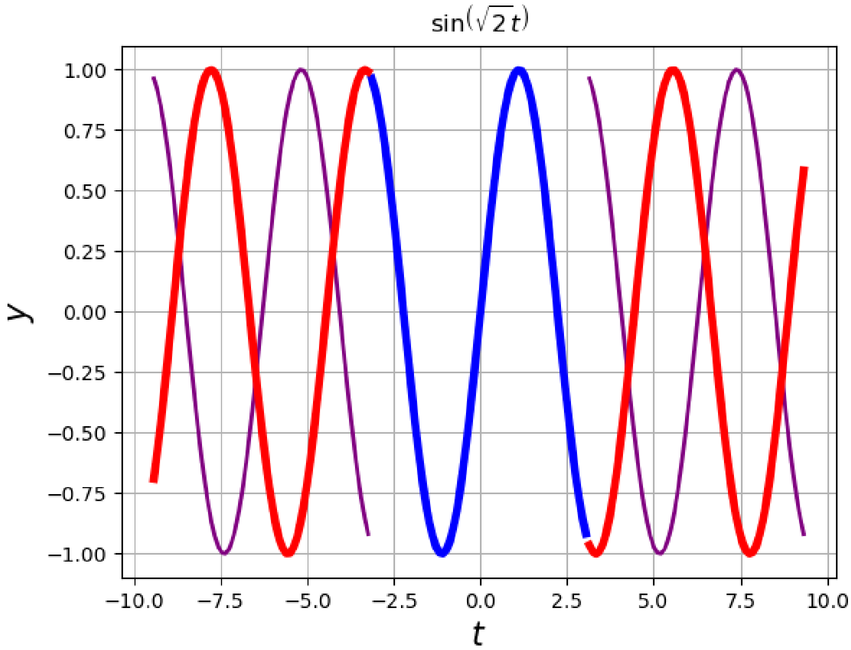


Figure 2: Non-periodic nature of $sin(\sqrt{2}t)$

2.3 Analysing the above graph

As we can clearly see, there are 3 colors of lines, these are:

1. Blue: This is the plot of $sin(\sqrt{2}t)$ over a period of $-\pi$ to π.
2. Red: This is actual remaining plot of $sin(\sqrt{2}t)$ vs time.
3. Purple: This is the shifted plot of $sin(\sqrt{2}t)$ over a period of $-\pi$ to π but it has been shifted for all the remaining 2π intervals as well.

- We can clearly see that the signal is not periodic with a periodic of 2π.
- Infact, we can see that we are **Actually finding the DFT of the signal created by repeating the middle blue plot** (i.e the blue and purple plot).
- This is known as the **Periodic Extension of** $sin(\sqrt{2}t)$ **in the range of** $(-\pi,\pi)$.

2.4 Code

```
#DFT of sin(sqrt(2)t)
t=linspace(-pi,pi,65);t=t[:-1]
dt=t[1]-t[0];fmax=1/dt
y=sin(sqrt(2)*t)
y[0]=0 # the sample corresponding to -tmax should be set zeroo
y=fftshift(y) # make y start with y(t=0)
Y=fftshift(fft(y))/64.0
w=linspace(-pi*fmax,pi*fmax,65);w=w[:-1]

subplot(2,1,1)
plot(w,abs(Y),w,abs(Y),'bo',lw=2)
xlim([-10,10])
ylabel(r"$|Y|$",size=16)
title(r"Spectrum of $\sin\left(\sqrt{2}t\right)$")
grid(True)

subplot(2,1,2)
plot(w,angle(Y),'ro',lw=2)
xlim([-10,10])
ylabel(r"Phase of $Y$",size=16)
xlabel(r"$\omega$",size=16)
grid(True)
show()


#plots of sin(sqrt(2)t) for different time ranges
t1=linspace(-pi,pi,65);t1=t1[:-1]
t2=linspace(-3*pi,-pi,65);t2=t2[:-1]
t3=linspace(pi,3*pi,65);t3=t3[:-1]

# y=sin(sqrt(2)*t)
plot(t1,sin(sqrt(2)*t1),'b',lw=4)
plot(t1+2*pi,sin(sqrt(2)*t1),'purple',lw=2)
plot(t1-2*pi,sin(sqrt(2)*t1),'purple',lw=2)
plot(t2,sin(sqrt(2)*t2),'r',lw=4)
plot(t3,sin(sqrt(2)*t3),'r',lw=4)
ylabel(r"$y$",size=16)
xlabel(r"$t$",size=16)
title(r"$\sin\left(\sqrt{2}t\right)$")
grid(True)
show()
```

3 Windowing Function: The Hamming Window

We can clearly see in Figure 2 that at the ends of the periodic interval, for example at the end of π, there are huge discontinuities. This is the reason our DFT results are not as accurate as we expect.

In order to fix this, we damp the function near there, i.e, we multiply our function sequence f [n] by a "window" sequence w[n]:

$$g(n) = f(n)w(n) \tag{1}$$

We get the new spectrum by convolving the two Fourier transforms.

$$G_k = \sum_{n=0}^{N-1} F_n W_{k-n} \tag{2}$$

The window we use is called the Hamming window, which is given by:

$$w[n] = \begin{cases} 0.54+0.46\cos(\frac{2\pi n}{N-1}) & |n| < \text{N} \\ 0 & \text{else} \end{cases}$$

3.1 Plot of $sin(\sqrt{2}t)$ after multiplying with the Hamming Window

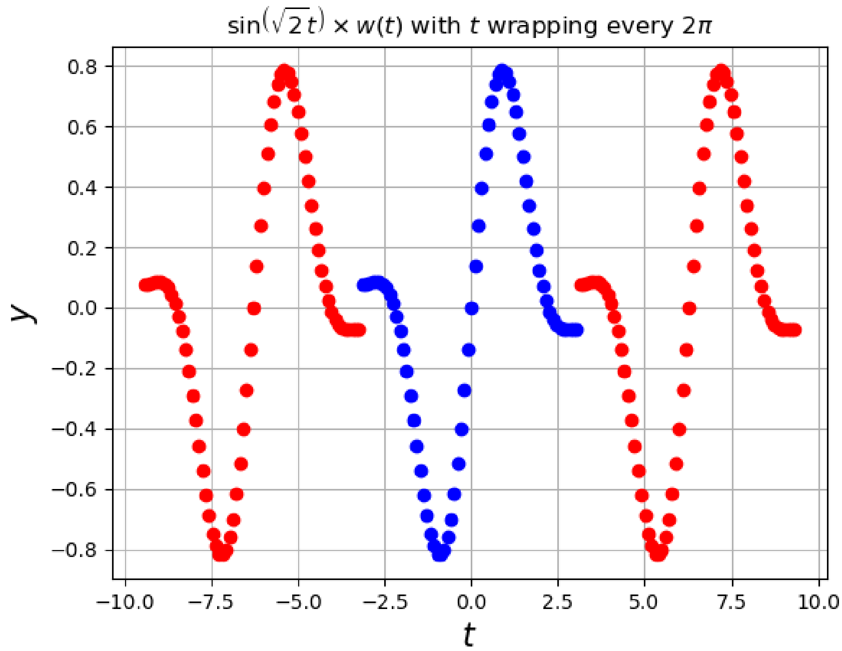


Figure 3: Plot of $sin(\sqrt{2}t)$ after Windowing

3.2 Code for the above

```
#plot of sin(sqrt(2)t) after windowing:
t1=linspace(-pi,pi,65);t1=t1[:-1]
t2=linspace(-3*pi,-pi,65);t2=t2[:-1]
t3=linspace(pi,3*pi,65);t3=t3[:-1]
n=arange(64)
wnd=fftshift(0.54+0.46*cos(2*pi*n/63))
y=sin(sqrt(2)*t1)*wnd
figure(3)
plot(t1,y,'bo',lw=2)
plot(t2,y,'ro',lw=2)
plot(t3,y,'ro',lw=2)
ylabel(r"$y$",size=16)
xlabel(r"$t$",size=16)
title(r"$\sin\left(\sqrt{2}t\right)\times w(t)$ with $t$ wrapping every $2\pi$
")
grid(True)
show()
```

3.3 DFT of the signal after Windowing

3.3.1 Spectrum of the signal only taking 64 points

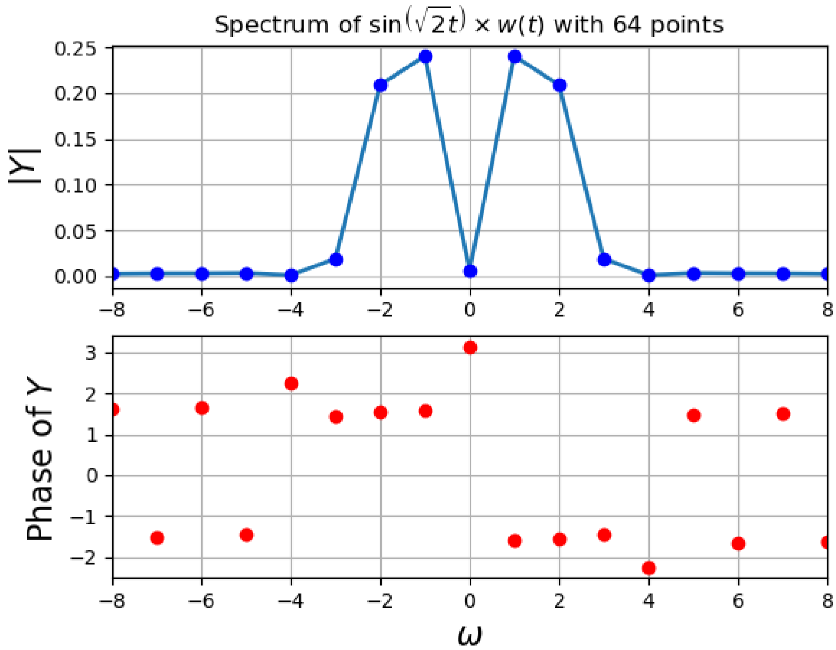


Figure 4: DFT of $sin(\sqrt{2}t)$ after Windowing with 64 points

We now find the Spectrum after increasing the number of samples taken to 256.

3.3.2 Spectrum of the signal only taking 256 points

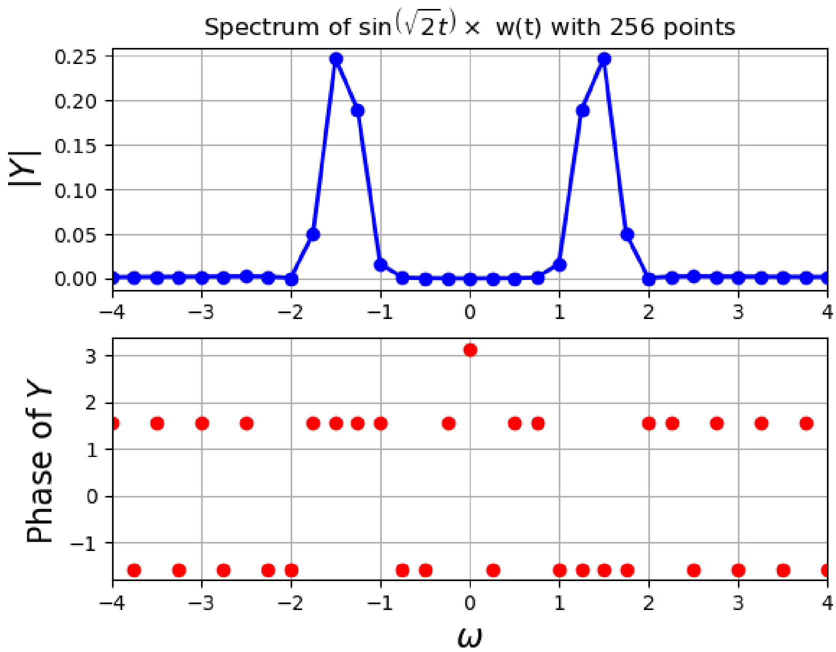


Figure 5: DFT of $sin(\sqrt{2}t)$ after Windowing with 256 points

We can clearly see that the Spectrum has become more accurate.

3.4 Code

```
#DFT of the function after windowing

t=linspace(-pi,pi,65);t=t[:-1]
dt=t[1]-t[0];fmax=1/dt
n=arange(64)
wnd=fftshift(0.54+0.46*cos(2*pi*n/63))
y=sin(sqrt(2)*t)*wnd
y[0]=0 # the sample corresponding to -tmax should be set zeroo
y=fftshift(y) # make y start with y(t=0)
Y=fftshift(fft(y))/64.0
w=linspace(-pi*fmax,pi*fmax,65);w=w[:-1]

subplot(2,1,1)
plot(w,abs(Y),w,abs(Y),'bo',lw=2)
xlim([-8,8])
ylabel(r"$|Y|$",size=16)
title(r"Spectrum of $\sin\left(\sqrt{2}t\right)\times w(t)$ with 64 points")
grid(True)

subplot(2,1,2)
plot(w,angle(Y),'ro',lw=2)
xlim([-8,8])
ylabel(r"Phase of $Y$",size=16)
xlabel(r"$\omega$",size=16)
grid(True)
show()

#DFT after windowing with 256 points


t=linspace(-4*pi,4*pi,257);t=t[:-1]
dt=t[1]-t[0];fmax=1/dt
n=arange(256)
wnd=fftshift(0.54+0.46*cos(2*pi*n/256))
y=sin(sqrt(2)*t)
# y=sin(1.25*t)
y=y*wnd
y[0]=0 # the sample corresponding to -tmax should be set zeroo
y=fftshift(y) # make y start with y(t=0)
Y=fftshift(fft(y))/256.0
w=linspace(-pi*fmax,pi*fmax,257);w=w[:-1]

subplot(2,1,1)
plot(w,abs(Y),'b',w,abs(Y),'bo',lw=2)
xlim([-4,4])
ylabel(r"$|Y|$",size=16)
title(r"Spectrum of $\sin\left(\sqrt{2}t\right)\times$ w(t) with 256 points")
grid(True)

subplot(2,1,2)
plot(w,angle(Y),'ro',lw=2)
xlim([-4,4])
ylabel(r"Phase of $Y$",size=16)
xlabel(r"$\omega$",size=16)
grid(True)
show()
```

4 Q2: Spectrum of $cos^3(0.86t$ before and after windowing

We plot the spectrum's in the figures below:

4.1 Spectrum of $cos^3(0.86t$ before Windowing

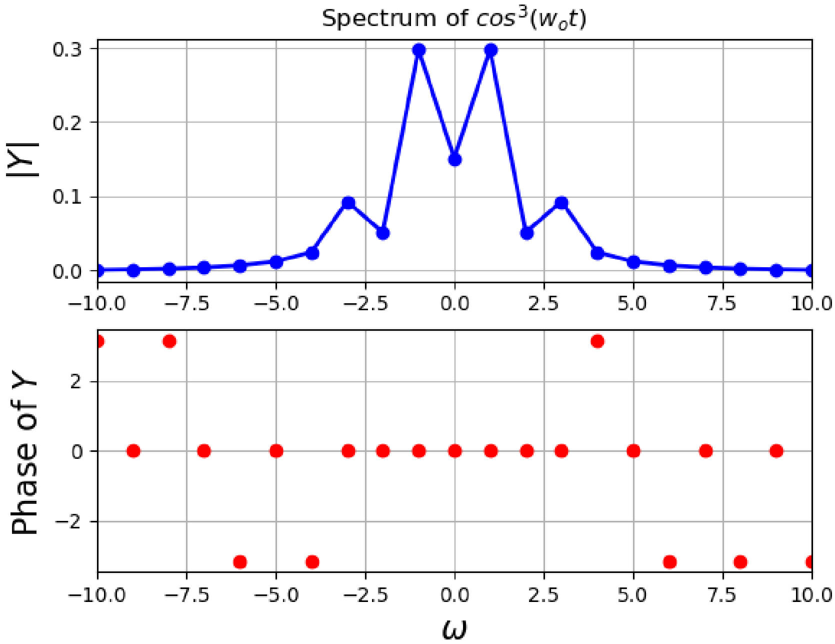


Figure 6: Spectrum of $cos^3(0.86t$ before Windowing

4.2 Spectrum of $cos^3(0.86t$ after Windowing

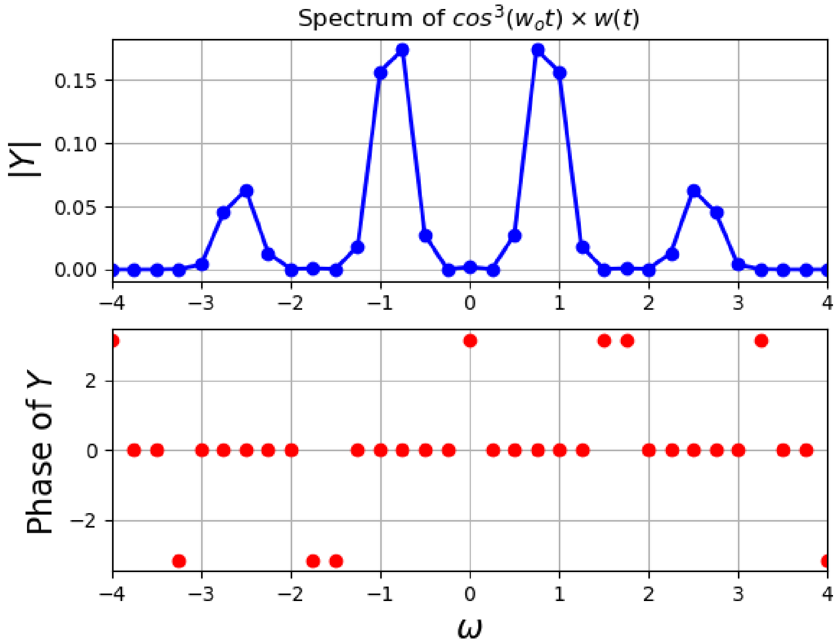


Figure 7: Spectrum of $cos^3(0.86t$ after Windowing

4.3 Code

```
#Question 2:

#DFT of cos^3(wot) where wo=0.86 without hamming window

t=linspace(-pi,pi,257);t=t[:-1]
dt=t[1]-t[0];fmax=1/dt
wo=0.86
y=(cos(wo*t))**3
y[0]=0 # the sample corresponding to -tmax should be set zeroo
y=fftshift(y) # make y start with y(t=0)
Y=fftshift(fft(y))/256.0
w=linspace(-pi*fmax,pi*fmax,257);w=w[:-1]

subplot(2,1,1)
plot(w,abs(Y),'b',w,abs(Y),'bo',lw=2)
xlim([-10,10])
ylabel(r"$|Y|$",size=16)
title(r"Spectrum of $cos^3(w_ot)$")
grid(True)


subplot(2,1,2)
plot(w,angle(Y),'ro',lw=2)
xlim([-10,10])
ylabel(r"Phase of $Y$",size=16)
xlabel(r"$\omega$",size=16)
grid(True)
show()
```

```
#DFT of cos^3(wot) where wo=0.86 with the hamming window


t=linspace(-4*pi,4*pi,257);t=t[:-1]
dt=t[1]-t[0];fmax=1/dt
n=arange(256)
wnd=fftshift(0.54+0.46*cos(2*pi*n/256))
wo=0.86
y=(cos(wo*t))**3
# y=cos^3(wo*t)
y=y*wnd
y[0]=0 # the sample corresponding to -tmax should be set zeroo
y=fftshift(y) # make y start with y(t=0)
Y=fftshift(fft(y))/256.0
w=linspace(-pi*fmax,pi*fmax,257);w=w[:-1]


subplot(2,1,1)
plot(w,abs(Y),'b',w,abs(Y),'bo',lw=2)
xlim([-4,4])
ylabel(r"$|Y|$",size=16)
title(r"Spectrum of $cos^3(w_ot)\times w(t)$")
grid(True)


subplot(2,1,2)
plot(w,angle(Y),'ro',lw=2)
xlim([-4,4])
ylabel(r"Phase of $Y$",size=16)
xlabel(r"$\omega$",size=16)
grid(True)
show()
```

5 Q3: Estimating the ω_o and δ of a $cos(\omega_o t + \delta)$ signal

- In this part of the code, we have a function that **Randomly generates a ω_o and a δ** and plots the DFT of the randomly obtained function.
- **NOTE: the values of ω_o and δ change every time you run the code as they are RANDOMLY GENERATED**
- According to the conditions given in the Question, the resolution of the code will not be clear enough that we can directly find the peaks and call that ω_o, instead, we have to extract all the information we can from the graoh to get these values.
- Hence, we find the **Weighted mean** of all the points with magnitude greater than 0.
- the Equation is as follows:

$$\omega_o = \frac{\Sigma \omega_k |Y(\omega_k)|^2}{\Sigma |Y(\omega_k)|^2} \ \forall\, w_k > 0 \tag{3}$$

- To estimate δ, we just find the phase of the DFT at the points close to the point of maximum amplitude.
- This approach is based on the fact that for $\delta = 0$, the phase of the maximum point is 0. But for a non-zero δ, the δ we obtain is the required δ.
- We multiply it with the Hamming window to get accurate results.

5.1 Outputs

The outputs one gets is:

1. A line telling them the actual ω_o and δ.
2. A line telling them the estimated ω_o.
3. A line telling them the estimated δ.
4. Spectrum plots of the function

5.2 An Example plot

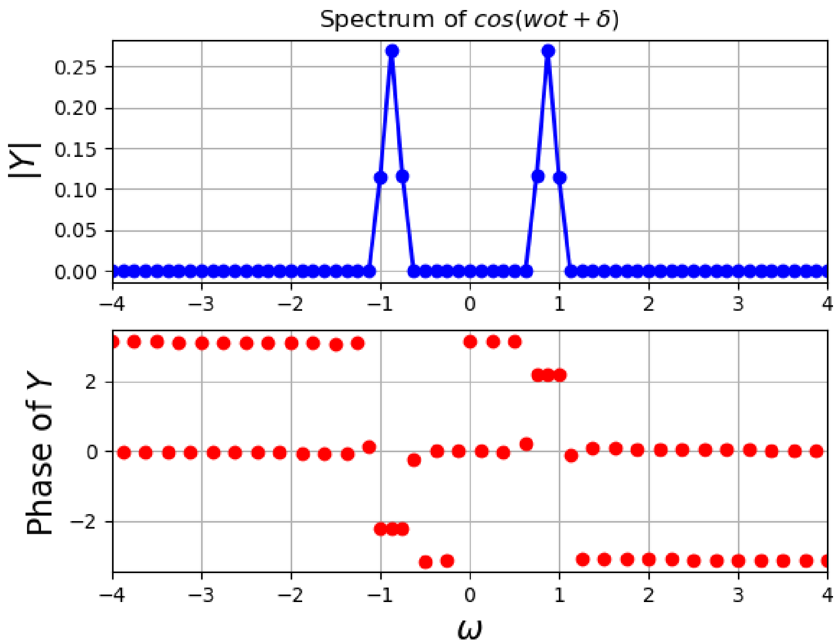
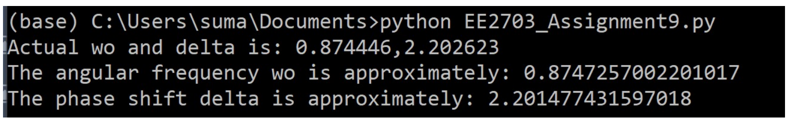


Figure 8: Spectrum of $\cos(\omega_o t + \delta)$

5.3 Terminal Output



```
(base) C:\Users\suma\Documents>python EE2703_Assignment9.py
Actual wo and delta is: 0.874446,2.202623
The angular frequency wo is approximately: 0.8747257002201017
The phase shift delta is approximately: 2.201477431597018
```

Figure 9: Output of terminal

As we can clearly see,

1. the actual randomly generated ω_o is 0.874446
2. the actual randomly generated δ is 2.202623

The estimated ω_o and δ are:

1. estimated $\omega_o = 0.8747257002201017$

2. estimated $\omega_o = 2.201477431597018$

SUCCESS!!, We can clearly see that the estimated value are very very similar to the actual randomly generated values.

6 Q4: Estimating the ω_o and δ of a $cos(\omega_o t + \delta)$ Noisy signal

We add a gaussian white noise of amplitude 0.1 to make the signal noisy.

AGAIN NOTE, the values of ω_o and δ change every time you run the code as they are RANDOMLY GENERATED.

The process for estimated the values here is extremely similar to the process we used in the previous question, but here we need to take the weighted mean of signals with magnitude greater than 0.15. i.e,

$$\omega_o = \frac{\Sigma \omega_k |Y(\omega_k)|^{1.6}}{\Sigma |Y(\omega_k)|^{1.6}} \ \forall\, w_k > 0.15 \tag{4}$$

This power of 1.6 was obtained through trial and error until I got the lowest error between actual and estimated values of the variables for HIGHEST number of cases.

6.1 Outputs

The outputs one gets is:

1. A line telling them the actual ω_o and δ.
2. A line telling them the estimated noisy ω_o.
3. A line telling them the estimated noisy δ.
4. Spectrum plots of the noisy function

6.2 An Example noisy plot

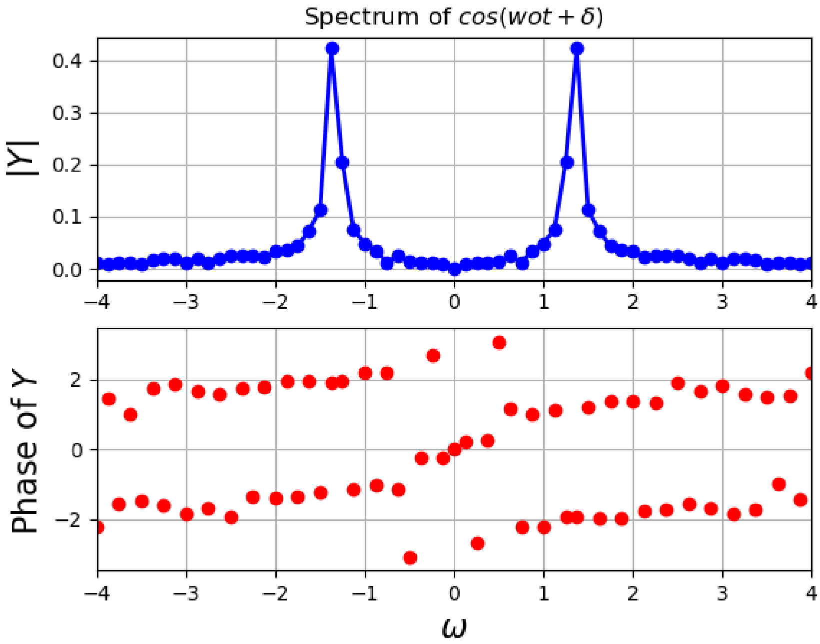


Figure 10: Spectrum of noisy $\cos(\omega_o t + \delta)$

6.3 Terminal Output

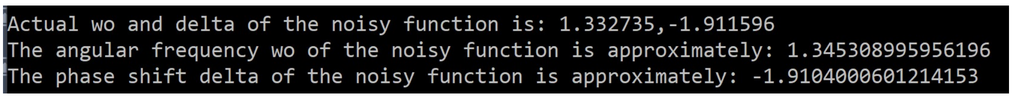


Figure 11: Output of terminal

As we can clearly see,

1. the actual randomly generated noisy ω_o is 1.332735
2. the actual randomly generated noisy δ is -1.911596

The estimated ω_o and δ are:

1. estimated noisy ω_o = 1.345308995956196
2. estimated noisy ω_o = -1.9104000601214153

SUCCESS!!, We can clearly see that the estimated value are very very similar to the actual randomly generated values.

6.4 Code for Q3

```
#Question 3: finding wo and delta from a randomly generated vector

N=128

wo = random() + 0.5
```

```
delta = random()*2*pi - pi
t=linspace(-8*pi,8*pi,N+1);t=t[:-1] #time period of -pi to pi has very low
                                                        precision
dt = t[1]-t[0];fmax=1/dt
n = arange(N)
wnd = fftshift(0.54+0.46*cos(2*pi*n/N))
vec = cos(wo*t + delta)
vec = vec*wnd
vec[0]=0 # the sample corresponding to -tmax should be set zeroo
y=fftshift(vec) # make y start with y(t=0)
Y=fftshift(fft(y))/N
w=linspace(-pi*fmax,pi*fmax,N+1);w=w[:-1]
print('Actual wo and delta is: %f,%f'%(wo,delta))

subplot(2,1,1)
plot(w,abs(Y),'b',w,abs(Y),'bo',lw=2)
ylabel(r"$|Y|$",size=16)
xlim([-4,4])
title(r"Spectrum of $cos(wot+\delta)$")
grid(True)

subplot(2,1,2)
plot(w,angle(Y),'ro',lw=2)
ylabel(r"Phase of $Y$",size=16)
xlabel(r"$\omega$",size=16)
xlim([-4,4])
grid(True)
show()

max = abs(Y).max()
ii = where(abs(abs(Y)-(max))<0.01)
ii2 = where(abs(Y)>0)
wapprox = sum(abs(Y[ii2])*abs(Y[ii2])*abs(w[ii2])) / sum(abs(Y[ii2])*abs(Y[ii2]
                                                        ))
print('The angular frequency wo is approximately:',wapprox)
print('The phase shift delta is approximately:',angle(Y[ii][1]))
```

6.5 Code for Q4

```
#Question 4: Same as above, but now we find it for a noisy signal

N=256

wo = random() + 0.5
delta = random()*2*pi - pi
t=linspace(-8*pi,8*pi,N+1);t=t[:-1] #time period of -pi to pi has very low
                                                        precision
dt = t[1]-t[0];fmax=1/dt
vec = cos(wo*t + delta)
vec = vec + 0.1*randn(N)
vec[0]=0 # the sample corresponding to -tmax should be set zeroo
y=fftshift(vec) # make y start with y(t=0)
Y=fftshift(fft(y))/N
w=linspace(-pi*fmax,pi*fmax,N+1);w=w[:-1]
print('\nActual wo and delta of the noisy function is: %f,%f'%(wo,delta))

subplot(2,1,1)
plot(w,abs(Y),'b',w,abs(Y),'bo',lw=2)
xlim([-4,4])
ylabel(r"$|Y|$",size=16)
title(r"Spectrum of $cos(wot+\delta)$")
grid(True)
```

```
subplot(2,1,2)
plot(w,angle(Y),'ro',lw=2)
xlim([-4,4])
ylabel(r"Phase of $Y$",size=16)
xlabel(r"$\omega$",size=16)
grid(True)
show()

max = abs(Y).max()
ii = where(abs(abs(Y)-(max))<0.01)
ii2 = where(abs(Y)>0.15)
wapprox = sum(abs(Y[ii2])**1.6*abs(w[ii2])) / sum(abs(Y[ii2])**1.6)
print('The angular frequency wo of the noisy function is approximately:',
                                   wapprox)
print('The phase shift delta of the noisy function is approximately:',angle(Y[
                                   ii][1]))
```

7 Q5: Spectrum of the Chirped Signal

The **Chirped signal** is given by:

$$y(t) = cos(16(1.5 + \frac{t}{2\pi})t) \tag{5}$$

its frequency continuously changes from 16 radians per second near $-\pi$ to 32 radians per second near π.

7.1 Spectrum of the Chirped Signal without hamming window

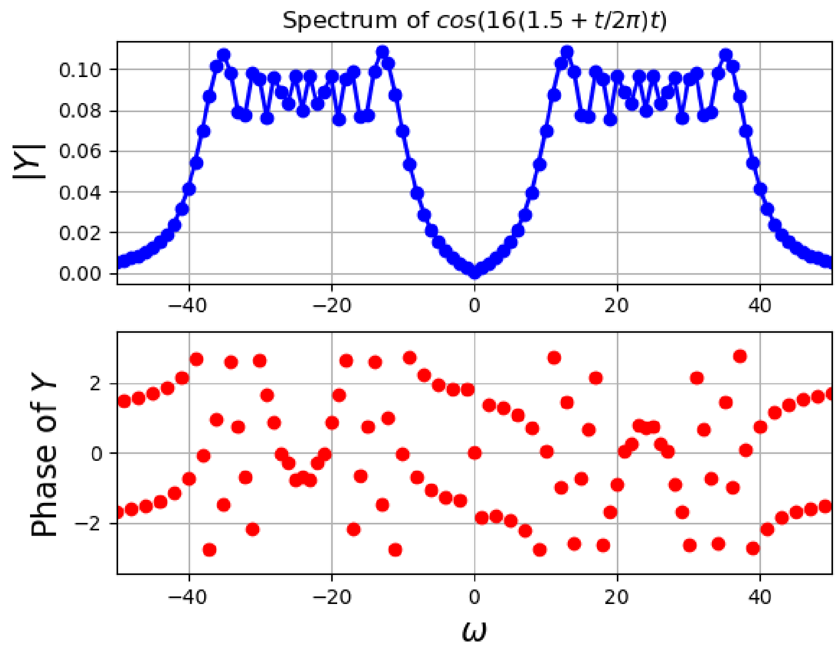


Figure 12: Spectrum of the Chirped Signal without hamming window

7.2 Spectrum of the Chirped Signal with hamming window

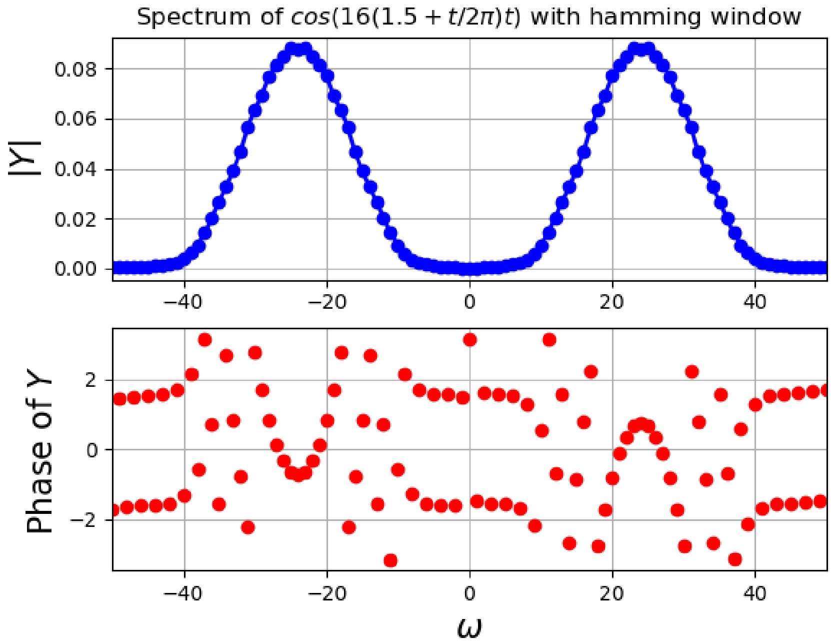


Figure 13: Spectrum of the Chirped Signal with hamming window

7.3 Code

```
#Question 5: DFT of a 'chirped' signal

N=1024

t=linspace(-pi,pi,N+1);t=t[:-1] #time period of -pi to pi has very low
                                            precision
dt = t[1]-t[0];fmax=1/dt
y = cos(16*t*(1.5 + (t/(2*pi))))
y[0]=0 # the sample corresponding to -tmax should be set zeroo
y=fftshift(y) # make y start with y(t=0)
Y=fftshift(fft(y))/N
w=linspace(-pi*fmax,pi*fmax,N+1);w=w[:-1]

subplot(2,1,1)
plot(w,abs(Y),'b',w,abs(Y),'bo',lw=2)
xlim([-50,50])
ylabel(r"$|Y|$",size=16)
title(r"Spectrum of $cos(16(1.5+t/2\pi)t)$")
grid(True)


subplot(2,1,2)
plot(w,angle(Y),'ro',lw=2)
xlim([-50,50])
ylabel(r"Phase of $Y$",size=16)
xlabel(r"$\omega$",size=16)
grid(True)
show()

#with hamming window
N=1024
```

```
t=linspace(-pi,pi,N+1);t=t[:-1] #time period of -pi to pi has very low
                                        precision
dt = t[1]-t[0];fmax=1/dt
n = arange(N)
wnd = fftshift(0.54+0.46*cos(2*pi*n/N))
y = cos(16*t*(1.5 + (t/(2*pi))))
y = y*wnd
y[0]=0 # the sample corresponding to -tmax should be set zeroo
y=fftshift(y) # make y start with y(t=0)
Y=fftshift(fft(y))/N
w=linspace(-pi*fmax,pi*fmax,N+1);w=w[:-1]

subplot(2,1,1)
plot(w,abs(Y),'b',w,abs(Y),'bo',lw=2)
xlim([-50,50])
ylabel(r"$|Y|$",size=16)
title(r"Spectrum of $cos(16(1.5+t/2\pi)t)$ with hamming window")
grid(True)


subplot(2,1,2)
plot(w,angle(Y),'ro',lw=2)
xlim([-50,50])
ylabel(r"Phase of $Y$",size=16)
xlabel(r"$\omega$",size=16)
grid(True)
show()
```

8 Q6: Surface Plot of Variation of frequency with time - Chirped Signal

We now use a surface plot to visualise how the frequency of the signal varies with time.

1. We first break the 1024 vector into pieces that are 64 samples wide.
2. We then Extract the DFT of each and store as a column in a 2D array (variable name is 'subvec'.)
3. Unfortunately, we need to use atleast 1 for loop to execute what we want.
4. Finally, plot the array as a surface plot to show how the frequency of the signal varies with time.

The surface plots for the chirped signal are given in the plots below:

8.1 Surface Plot of Variation of frequency with time - Chirped Signal without Windowing

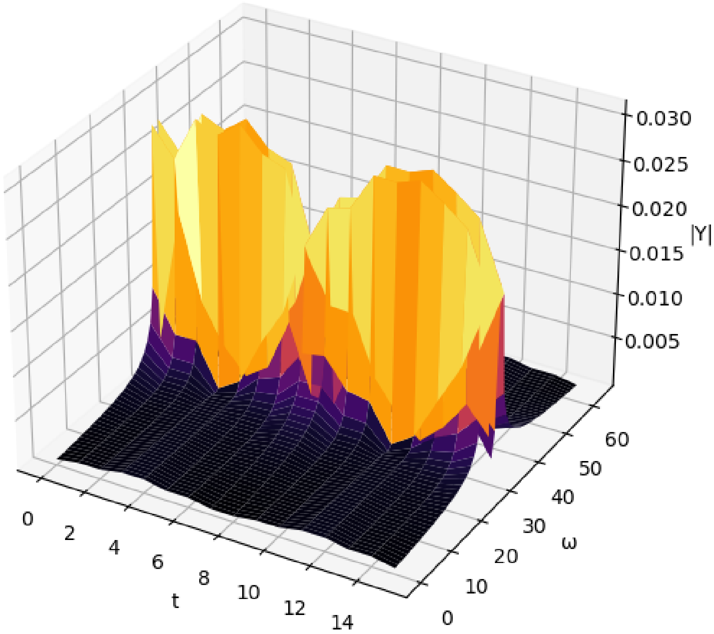


Figure 14: Surface Plot of Variation of frequency with time - Chirped Signal without Windowing

8.2 Surface Plot of Variation of frequency with time - Chirped Signal with Windowing

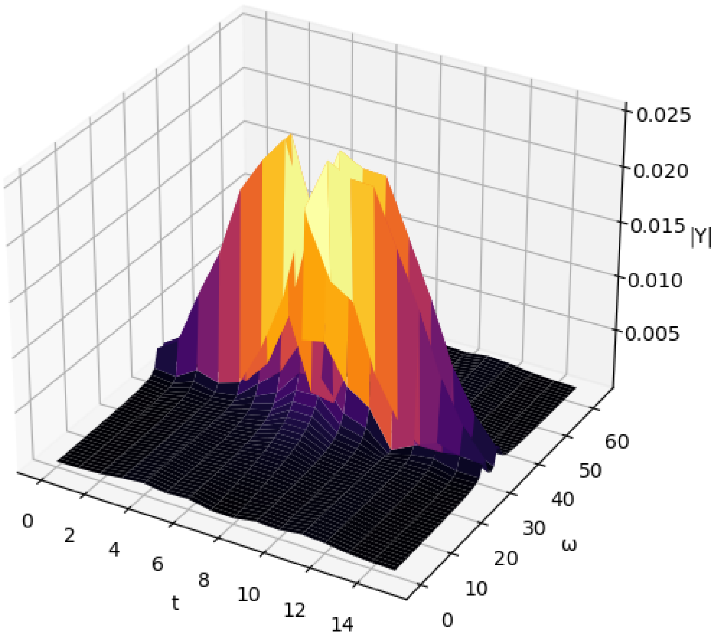


Figure 15: Surface Plot of Variation of frequency with time - Chirped Signal with Windowing

8.3 Code for without windowing

```
#Question 6: Surface Plot of Variation of frequency with time for both types of
                                                         chirped signals
#Without hamming window:

N=1024

t=linspace(-pi,pi,N+1);t=t[:-1] #time period of -pi to pi has very low
                                                        precision
dt = t[1]-t[0];fmax=1/dt
y = cos(16*t*(1.5 + (t/(2*pi))))

subvec = zeros((16,64))
for n in arange(16):
    subvec[n]=y[64*n:64*n + 64]
    subvec[n][0] = 0

y=fftshift(subvec) # make y start with y(t=0)
Y=fftshift(fft(y))/N
w=linspace(-pi*fmax,pi*fmax,N+1);w=w[:-1]
n=arange(64)


t1 = np.array ( arange (16) )
t1,n = meshgrid ( t1 , n )
ax = Axes3D(figure())
surf = ax.plot_surface ( t1 ,n ,abs(Y).T , rstride =1 , cstride =1 , cmap ='
                                                        inferno')
ylabel('\u03C9')
xlabel('t')
title(" Surface Plot of Variation of frequency with time - Chirped Signal
                                                        without Hamming Window ")
ax.set_zlabel ('|Y|')
show()
```

8.4 Code for with windowing

```
#With hamming window:
N=1024

t=linspace(-pi,pi,N+1);t=t[:-1] #time period of -pi to pi has very low
                                                        precision
dt = t[1]-t[0];fmax=1/dt
n = arange(N)
wnd = fftshift(0.54+0.46*cos(2*pi*n/N))
y = cos(16*t*(1.5 + (t/(2*pi))))
y = y*wnd

subvec = zeros((16,64))
for n in arange(16):
    subvec[n]=y[64*n:64*n + 64]
    subvec[n][0] = 0


y=fftshift(subvec) # make y start with y(t=0)
Y=fftshift(fft(y))/N
w=linspace(-pi*fmax,pi*fmax,N+1);w=w[:-1]
n=arange(64)
```

```
t1 = np.array ( arange (16) )
t1,n = meshgrid ( t1 , n )
ax = Axes3D( figure())
surf = ax.plot_surface ( t1 ,n ,abs(Y).T , rstride =1 , cstride =1 , cmap ='
                                                inferno')
ylabel('\u03C9')
xlabel('t')
title(" Surface Plot of Variation of frequency with time - Chirped Signal with
                                                Hamming Window ")
ax.set_zlabel ('|Y|')
show()

#end
```