Assignment 10

Lakshya J ee19b035

May 10th, 2021

Introduction

In this assignment we are asked to analyze the spectra of non-periodic signals. I begin by working through the examples that sir had illustrated.

```
from pylab import *
from scipy.linalg import lstsq
import numpy as np
```

This is the list of libraries I imported for the same.
We make use of the hamming window given by

```
wnd = fftshift(0.54+0.46*cos(2*pi*n/63))
```

This is the hamming window created for a non periodic signal which is sampled at 64 discrete points.

Question 2

In this question we are asked to plot the spectrum of $cos^3(0.86t)$ with and without a hamming window.

```
t=linspace(-pi,pi,65)
t=t[:-1]
dt=t[1]-t[0]
fmax=1/dt
y=cos(0.86*t)**3
y=fftshift(y)
Y=fftshift(fft(y))/64.0
w=linspace(-pi*fmax,pi*fmax,65)
w=w[:-1]
```

$fmax$ is chosen such that the Nyquist rate is met. After doing the fourier shifts and computations I plot the magnitude and phase angle plots as follows:

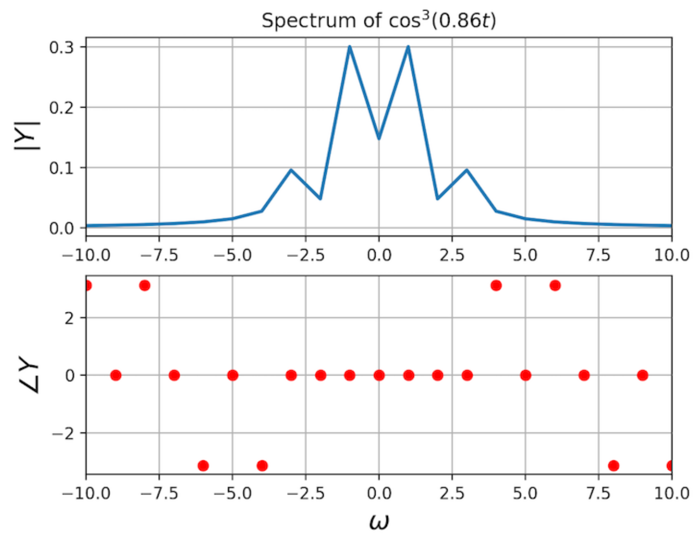

Figure 1: Spectrum of $cos^3(0.86t)$ without hamming window

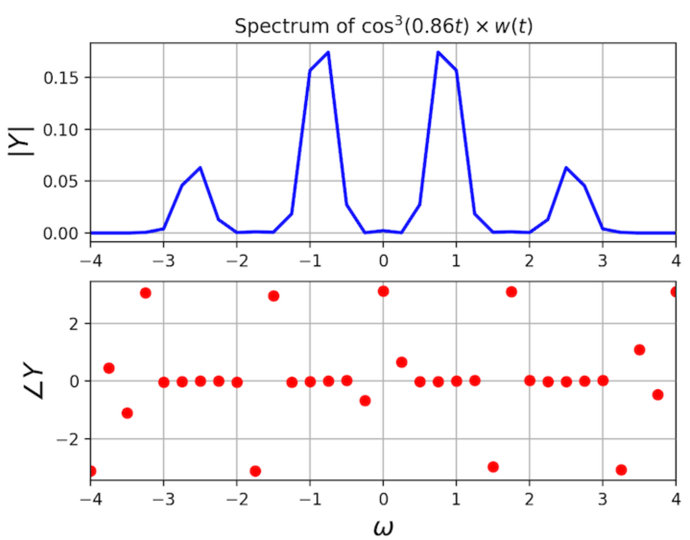

Figure 2: Spectrum of $cos^3(0.86t)$ with hamming window

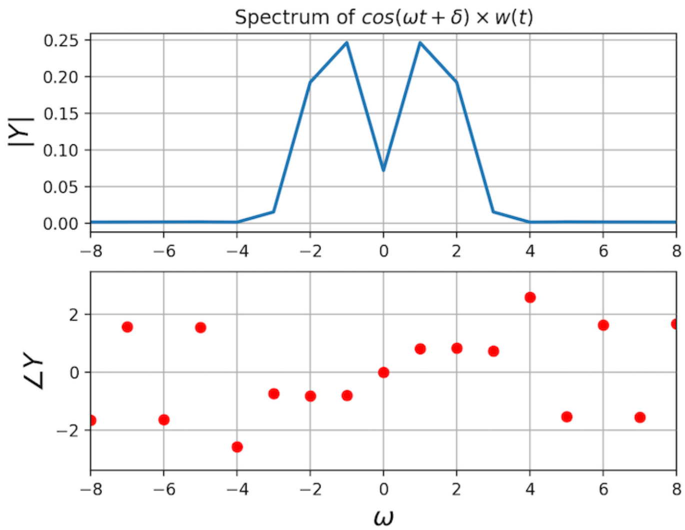

Figure 3: Spectrum of $cos(\omega \times t + \delta)$ with $\omega = 1.35$ and$\delta = 0.8$

Q3

In this part we try to plot the spectrum of $cos(\omega \times t + \delta)$. We also try to approximate the values of ω and δ from the obtained spectrum. Here is the code for the same:

```
t=linspace(-pi,pi,129)
t=t[:-1]
dt=t[1]-t[0]
fmax=1/dt
n=arange(128)
wnd=fftshift(0.54+0.46*cos(2*pi*n/127))
wo = 1.35
delta = 0.8
y=cos(wo*t+delta)
y=y*wnd
y=fftshift(y)
Y=fftshift(fft(y))/128.0
w=linspace(-pi*fmax,pi*fmax,129)
w=w[:-1]
```

Here is the approximation of ω and δ:
wo Estimate: 1.381 wo Actual value: 1.350
delta Estimate -2.788 delta Actual value: 0.800
Clearly there is deviation from the actual values.
Here is the code for approximating the same:

```
wEstimate = sum(abs(Y)**1.75 * abs(w))/sum(abs(Y)**1.75)
c1 = cos(wEstimate*t)
```

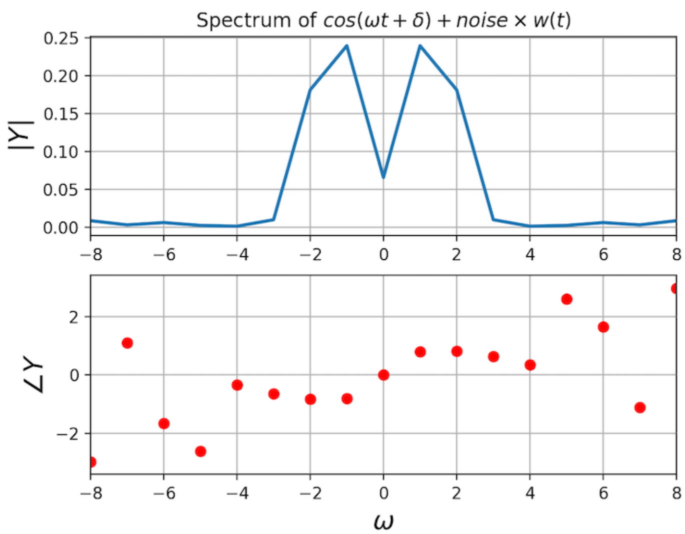

Figure 4: Spectrum of $cos(\omega \times t+\delta)$ with $\omega = 1.35$ and$\delta = 0.8$ with added noise

```
c2 = sin(wEstimate*t)
A = np.c_[c1, c2]
vals = lstsq(A, y)[0]
dEstimate = arctan2(-vals[1],vals[0])
```

Q4

Here I plot $cos(\omega \times t + \delta)$ with some added white Gaussian noise. This is the code I used to add the noise and produce the spectrum:

```
noise = 0.1*np.random.randn(128)
t=linspace(-pi,pi,129)
t=t[:-1]
dt=t[1]-t[0]
fmax=1/dt
n=arange(128)
wnd=fftshift(0.54+0.46*cos(2*pi*n/127))
wo = 1.35
delta = 0.8
y=cos(wo*t+delta)+noise
y=y*wnd
y=fftshift(y)
Y=fftshift(fft(y))/128.0
w=linspace(-pi*fmax,pi*fmax,129)
w=w[:-1]
```

Here is the approximation of ω and δ:
wo Estimate: 2.775 wo Actual value: 1.350

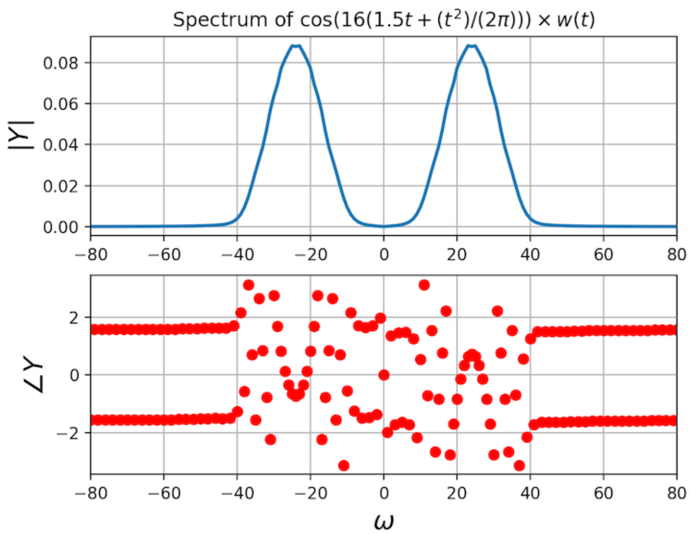

Figure 5: Spectrum of $\cos(16(1.5t + (t^2)/(2\pi))) \times w(t)$

delta Estimate 0.530 delta Actual value: 0.800

Q5

In this question we find the DFT of a chirped signal. Here is the code for the same:

```
t=linspace(-pi,pi,1025)
t=t[:-1]
dt=t[1]-t[0]
fmax=1/dt
n=arange(1024)
wnd=fftshift(0.54+0.46*cos(2*pi*n/1023))
y=cos(16*(1.5*t + (t**2)/(2*pi)))
y=y*wnd
y=fftshift(y)
Y=fftshift(fft(y))/1024.0
w=linspace(-pi*fmax,pi*fmax,1025)
w=w[:-1]
```

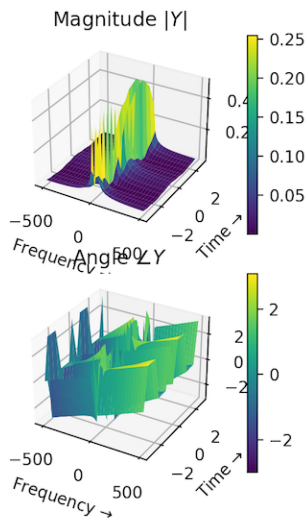

Figure 6: 3D PLOT without multiplying by the hamming window

Q6

In this question I take the same chirped signal, break the 1024 vector into pieces that are 64 samples wide. I then extract the DFT of each and store as a column in a 2D array. I plot the array as a surface plot to show how the frequency of the signal varies with time. Here is the code for how I achieve the same. In the first case I do it without the hamming window.

```
# First calculating the DFT of x taking every batch size sample
x = cos(16*(1.5*t + (t**2)/(2*pi)))
t_batch = split(t, 1024//64)
x_batch = split(x, 1024//64)
X = np.zeros((1024//64, 64), dtype=complex)
for i in range(1024//64):
    X[i] = fftshift(fft(x_batch[i]))/64
#Plotting the 3D PLOT without multiplying by the hamming window
t = t[::64]
w = linspace(-fmax*pi,fmax*pi,65)[:-1]
t, w = meshgrid(t, w)
```

Now I do the same with the hamming window. Here is the code used to achieve it.

```
t = linspace(-pi, pi, 1025)[:-1]
x = cos(16*(1.5*t + (t**2)/(2*pi))) * wnd
t_batch = split(t, 1024//64)
x_batch = split(x, 1024//64)
X = np.zeros((1024//64, 64), dtype=complex)
for i in range(1024//64):
    X[i] = fftshift(fft(x_batch[i]))/64
```

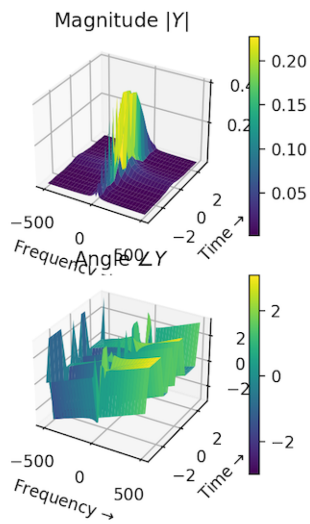

Figure 7: 3D PLOT with multiplying by the hamming window

```
t = t[::64]
w = linspace(-fmax*pi,fmax*pi,65)[:-1]
t, w = meshgrid(t, w)
```