

ASSIGNMENT 9: SPECTRA OF NON-PERIODIC SIGNALS

NITHIN BABU [EE18B021]

May 4, 2020

Abstract

The goal of this assignment is the following:

- Obtaining the DFT of non-periodic functions.
- To see how the use of windowing functions (e.g Hamming Window) can help in making the DFT better.
- To plot graphs to understand this.

Introduction

- We will explore digital fourier transform (DFT) with windowing. This is used to make the signal square integrable, and more specifically, that the function goes sufficiently rapidly towards 0 , also to make infinitely long signal to a finite signal, since to take DFT we need finite aperiodic signal.
- Windowing a simple waveform like $\cos(\omega t)$, causes its fourier transform to develop non-zero value at frequencies other than ω . This is called *Spectral Leakage*. This can cause in some applications the stronger peak to smear the weaker counter parts. So choosing proper windowing functions is essential. The windowing function we use is called **Hamming window** which is generally used in *narrow – band applications*.

Setting Up Of Modules and Functions

We must declare the relevant modules and also set up the functions to efficiently run the program.

```

# Importing the necessary modules.
from pylab import *
from mpl_toolkits.mplot3d import Axes3D

# Declaring the function to plot a spectrum graph.
def spectrum_plot(fig_no,w,Y,xlimit,Title,ylabel1,ylabel2,Xlabel,Grid=True):
    figure(fig_no)
    subplot(2,1,1)
    plot(w,abs(Y),lw=2)
    xlim([-xlimit,xlimit])
    ylabel(ylabel1,size=16)
    title(Title)
    grid(grid)
    subplot(2,1,2)
    plot(w,angle(Y), 'ro',lw=2)
    xlim([-xlimit,xlimit])
    ylabel(ylabel2,size=16)
    xlabel(Xlabel,size=16)

```

DFT of $\sin(\sqrt{2}t)$

Without Hamming Window

- We will first plot the DFT of $\sin(\sqrt{2}t)$ without the Hamming Window.
- The python code snippet to calculate and plot the DFT of $\sin(\sqrt{2}t)$ is as shown below:

```

# The below piece of code is for Question.1.
# We to plot a spectrum of  $\sin(\sqrt{2}t)$ , in the most basic approximate way.
t = linspace(-pi,pi,65); t = t[:-1]
dt = t[1]-t[0]; fmax = 1/dt
y = sin(sqrt(2)*t)
y[0] = 0
y = fftshift(y)
Y = fftshift(fft(y))/64.0
w = linspace(-pi*fmax,pi*fmax,65); w = w[:-1]

spectrum_plot(0,w,Y,10,r"Spectrum of  $\sin(\sqrt{2}t)$ ",
r" $|Y|$   $\rightarrow$ ",r"Phase of  $Y$   $\rightarrow$ ",r" $\omega$   $\rightarrow$ ")

show()

```

The spectrum plot for $\sin(\sqrt{2}t)$ without windowing is as follows:

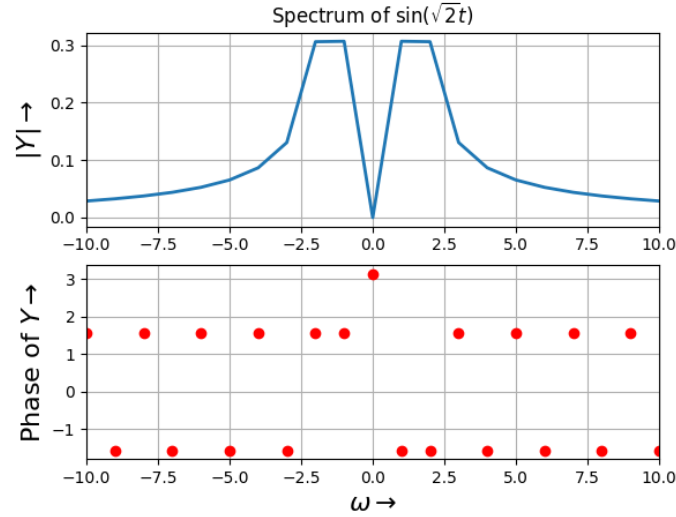


Figure 1: Spectrum of $\sin(\sqrt{2}t)$ without windowing

Without Hamming Window

- We will first plot the DFT of $\sin(\sqrt{2}t)$ with the Hamming Window.
- The python code snippet to calculate and plot the DFT of $\sin(\sqrt{2}t)$ is as shown below:

```
# The below piece of code is to plot a spectrum of sin(sqrt(2)t), in a better way a.
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)*wnd
y[0] = 0
y = fftshift(y)
Y = fftshift(fft(y))/256.0
w = linspace(-pi*fmax,pi*fmax,257); w = w[:-1]

spectrum_plot(1,w,Y,4,r"Improved Spectrum of $\sin\left(\sqrt{2}t\right)$",r"$|Y|\backslash r$
r"Phase of $Y\rightarrow$",r"$\omega\rightarrow$")

show()
```

The spectrum plot for $\sin(\sqrt{2}t)$ with windowing is as follows:

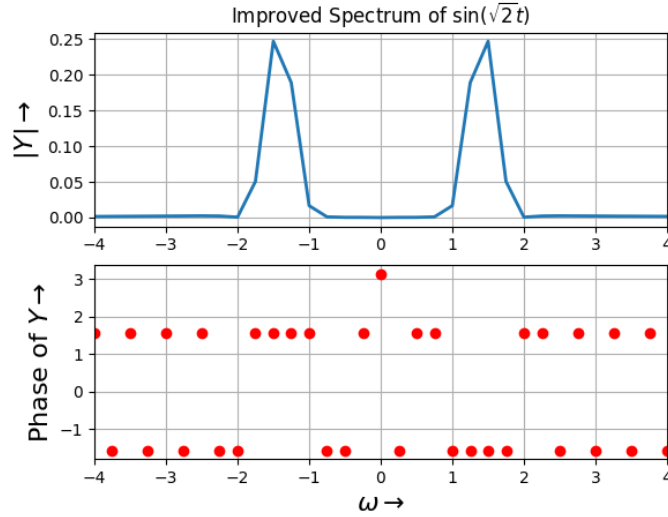


Figure 2: Spectrum of $\sin(\sqrt{2}t)$ with windowing

DFT of $\cos^3(\omega_0 t)$

- Consider the function $\cos^3(\omega_0 t)$. Obtain its spectrum for $\omega_0 = 0.86$ with and without a Hamming window.
- The python code snippet to calculate and plot the DFT of $\cos^3(\omega_0 t)$ with and without a Hamming Window is as shown below:

Code:

```
# The below piece of code is for Question.2.
# We to plot a spectrum of cos^3(0.86t), with and without windowing.
n = arange(256)
wnd = fftshift(0.54+0.46*cos(2*pi*n/256))
y = cos(0.86*t)**3
y1 = y*wnd
y[0]=0
y1[0]=0
y = fftshift(y)
y1 = fftshift(y1)
Y = fftshift(fft(y))/256.0
Y1 = fftshift(fft(y1))/256.0
```

```
spectrum_plot(2,w,Y,4,r"Spectrum of  $\cos^3(0.86t)$  without Hamming window",
r" $|Y| \rightarrow$ ",r"Phase of  $Y \rightarrow$ ",r" $\omega \rightarrow$ ")
```

```
spectrum_plot(3,w,Y1,4,r"Spectrum of  $\cos^3(0.86t)$  with Hamming window",
r" $|Y| \rightarrow$ ",r"Phase of  $Y \rightarrow$ ",r" $\omega \rightarrow$ ")
```

```
show()
```

The plots for the spectrum of $\cos^3(0.86t)$ with and without windowing are as follows:

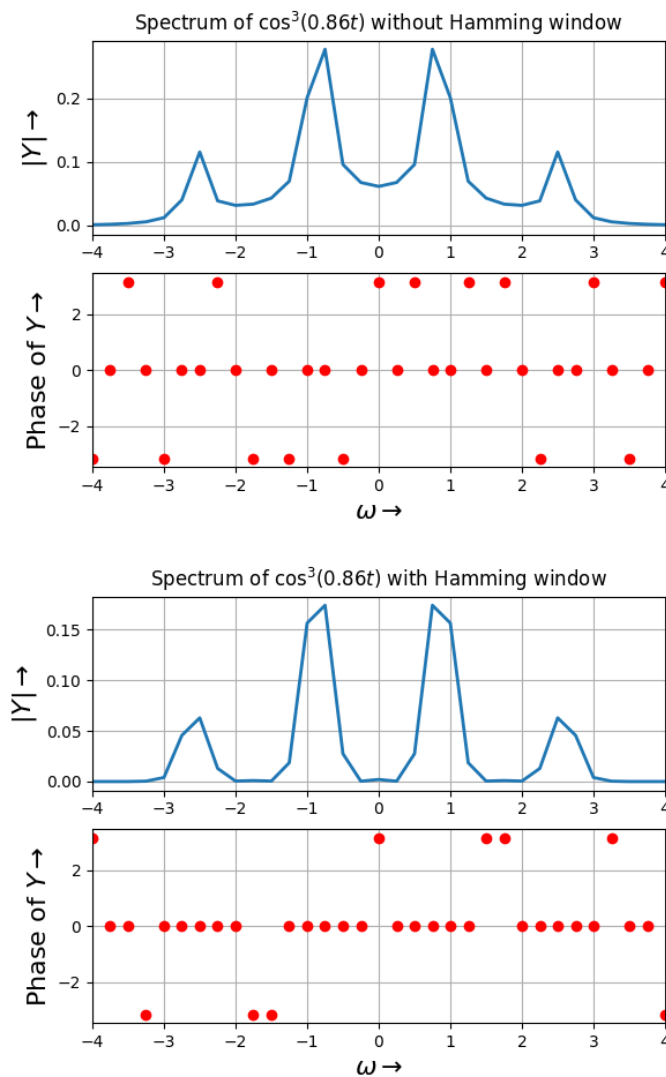


Figure 3: Spectrum of $\cos^3(\omega_0 t)$ with and without windowing

Estimating ω_0 and δ

- According to the question if the spectra is obtained, the resolution is not enough to obtain the ω_0 directly. The peak will not be visible clearly because of the fact that resolution of the frequency axis is not enough. So a statistic is necessary to estimate value of ω_0 . Hence, we can obtain ω_0 by taking a weighted average of all the ω weighted with the magnitude of the DFT.

$$\omega_0 = \frac{\sum \omega_i |Y(\omega_i)|^2}{\sum |Y(\omega_i)|^2} \quad \forall \omega_i > 0$$

- δ can be found by calculating the phase of the discrete fourier transform at ω_0 nearest to estimated ω using the above statistic.
- This works because the phase of $\cos(\omega_0 t + \delta)$ when $\delta = 0$ is 0, so when its not its δ , so we can estimate it by this approach.
- The python code snippet to carry out the above is as follows:

```
# The below piece of code is for Question.3.
# We have to find the values of w0 and delta from the spectrum of the signal.
# Let w0 = 1.5 and delta = 0.5.
w0 = 1.5
d = 0.5

t = linspace(-pi,pi,129)[: -1]
dt = t[1]-t[0]; fmax = 1/dt
n = arange(128)
wnd = fftshift(0.54+0.46*cos(2*pi*n/128))
y = cos(w0*t + d)*wnd
y[0]=0
y = fftshift(y)
Y = fftshift(fft(y))/128.0
w = linspace(-pi*fmax,pi*fmax,129); w = w[: -1]
spectrum_plot(4,w,Y,4,r"Spectrum of $\cos(w_0t+\delta)$ with Hamming window",
r"$|Y|\rightarrow$",r"Phase of $Y\rightarrow$",r"$\omega\rightarrow$")

# w0 is calculated by finding the weighted average of all w>0.
# Delta is found by calculating the phase at w closest to w0.
ii = where(w>=0)
w_cal = sum(abs(Y[ii])**2*w[ii])/sum(abs(Y[ii])**2)
i = abs(w-w_cal).argmin()
delta = angle(Y[i])
```

```

print("Calculated value of w0 without noise: ",w_cal)
print("Calculated value of delta without noise: ",delta)

show()

```

Results:

- Calculated value of ω_0 without noise: 1.473
- Calculated value of δ without noise: 0.502

The plot of the DFT spectra of $\cos(\omega_0 t + \delta)$ for the respective input values is as shown:

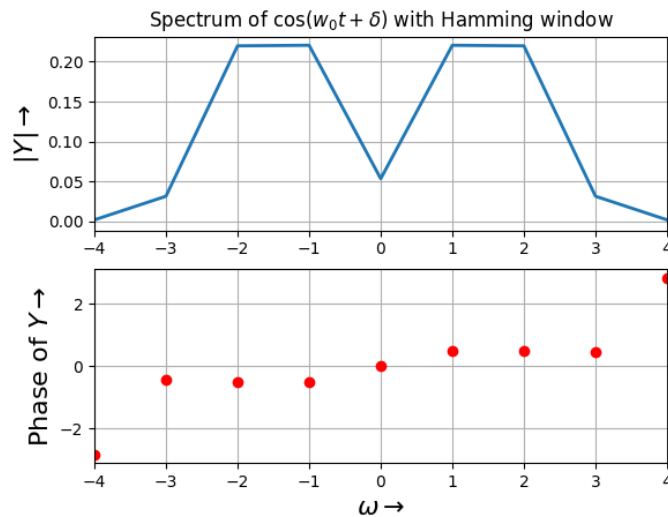


Figure 4: Spectrum of $\cos(\omega_0 t + \delta)$

Estimating ω_0 and δ in the presence of noise

- Now we add **white gaussian noise** to data in Q_3 . This can be generated by `randn()` in python. The extent of this noise is 0.1 in amplitude (i.e., $0.1 * \text{randn}(N)$, where N is the number of samples).
- The python code snippet is as shown:

```

# The below piece of code is for Question.4.
# We have to find the same for a noisy signal.
y = (cos(w0*t + d) + 0.1*randn(128))*wnd
y[0]=0

```

```

y = fftshift(y)
Y = fftshift(fft(y))/128.0
spectrum_plot(5,w,Y,4,r"Spectrum of a noisy  $\cos(\omega_0 t + \delta)$  with Hamming window",
r" $|Y| \rightarrow$ ",r"Phase of  $Y \rightarrow$ ",r" $\omega \rightarrow$ ")

#  $\omega_0$  is calculated by finding the weighted average of all  $\omega > 0$ .
# Delta is found by calculating the phase at  $\omega$  closest to  $\omega_0$ .
ii = where(w>=0)
w_cal = sum(abs(Y[ii])**2*w[ii])/sum(abs(Y[ii])**2)
i = abs(w-w_cal).argmin()
delta = angle(Y[i])
print("Calculated value of  $\omega_0$  with noise: ",w_cal)
print("Calculated value of  $\delta$  with noise: ",delta)

show()

```

Results:

- Calculated value of ω with noise: 2.052
- Calculated value of δ with noise: 0.510
- This error is slightly higher compared to the case without noise as we expected.

The plot of the DFT spectra of a noisy $\cos(\omega_0 t + \delta)$ for the respective input values is as shown:

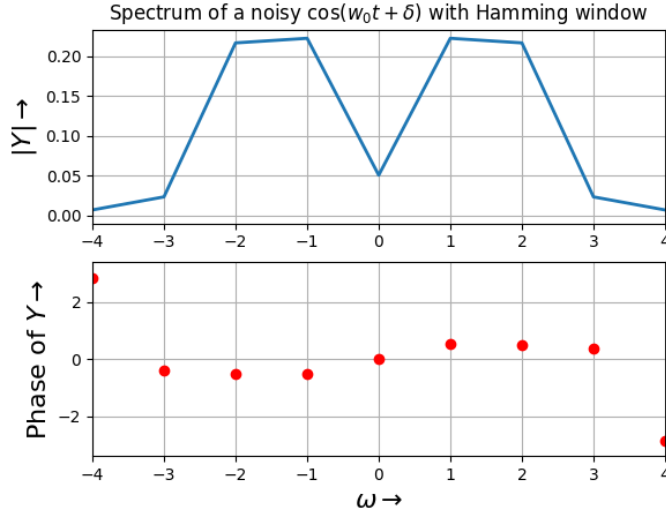


Figure 5: Spectrum of a $\text{noisycos}(\omega_0 t + \delta)$

Analysis of Chirped Signal Spectrum

- Plot the DFT of the function $\cos(16 (1.5 + \frac{t}{2\pi}) t)$ where $-\pi \leq t \leq \pi$ in 1024 steps. This is known as a *chirped* signal.
- Its frequency continuously changes from 16 to 32 radians per second. This also means that the period is 64 samples near π and is 32 samples near $+\pi$.
- The python code snippet for calculating and plotting the DFT of a "chirped" signal is as follows:

```
# The below piece of code is for Question.5.
# We have to plot the spectrum of a "chirped" signal.
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/1024))
y = cos(16*t*(1.5 + t/(2*pi)))*wnd
y[0]=0
y = fftshift(y)
Y = fftshift(fft(y))/1024.0
w = linspace(-pi*fmax,pi*fmax,1025); w = w[:-1]
spectrum_plot(6,w,Y,100,r"Spectrum of chirped function",r"$|Y|\rightarrow$",
r"Phase of $Y\rightarrow$",r"$\omega\rightarrow$")

show()
```

The plot of the DFT spectra of a "chirped" signal is as shown:

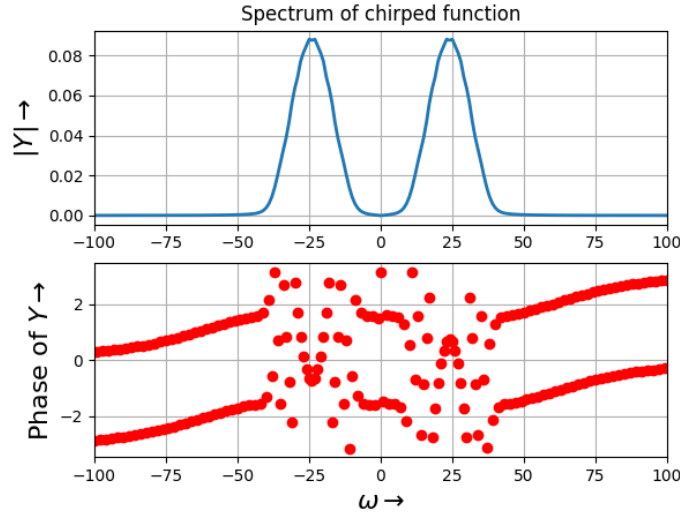


Figure 6: Spectrum of chirped signal with windowing

Surface Plot of chirped signal

- For the same chirped signal, break the 1024 vector into pieces that are 64 samples wide. Extract the *DFT* of each and store as a column in a 2D array.
- Then plot the array as a surface plot to show how the frequency of the signal varies with time.
- Plot and analyse the **time frequency** plot, where we get localized *DFTs* and show how the spectrum evolves in time.
- The python code snippet for executing the above is as shown:

```
# The below piece of code is for Question.6.
# We have to plot a surface plot with respect to t and w.
t_array = split(t,16)
Y_mag = zeros((16,64))
Y_phase = zeros((16,64))

for i in range(len(t_array)):
    n = arange(64)
    wnd = fftshift(0.54+0.46*cos(2*pi*n/64))
    y = cos(16*t_array[i]*(1.5 + t_array[i]/(2*pi)))*wnd
```

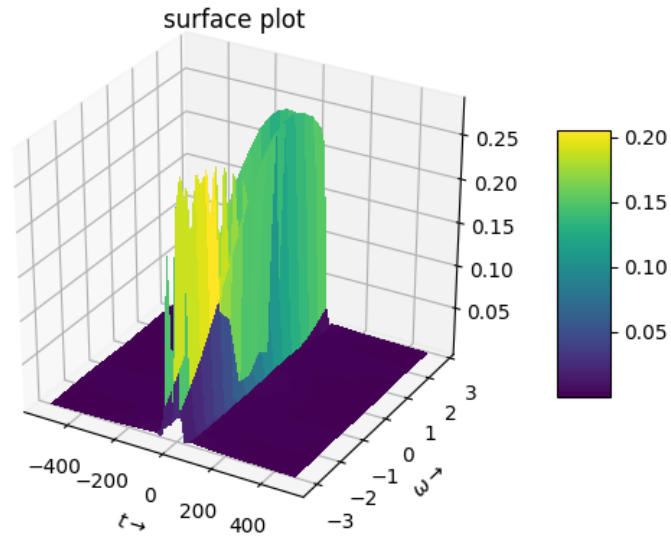


Figure 7: Surface plot of the magnitude of broken chirped signal

```

y[0]=0
y = fftshift(y)
Y = fftshift(fft(y))/64.0
Y_mag[i] = abs(Y)
Y_phase[i] = angle(Y)

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

fig1 = figure(7)
ax = fig1.add_subplot(111, projection='3d')
surf=ax.plot_surface(w,t,Y_mag.T,cmap='viridis',linewidth=0, antialiased=False)
fig1.colorbar(surf, shrink=0.5, aspect=5)
ax.set_title('surface plot');
ylabel(r"$\omega\rightarrow$")
xlabel(r"$t\rightarrow$")

show()

```

Conclusion :

- We Obtained the DFT of non-periodic functions.
- We saw how the use of windowing functions (e.g Hamming Window) improved the results.
- We made a 3-D surface plot of the chirped signal.
- To plotted graphs to understand the same.