Assignment 9

K.R.Srinivas - EE18B136

January 30, 2021

Abstract

This week's Python assignment focuses on the following topics.

- To compute DFT of non-periodic signals.
- Use Windowing functions like Hamming window to improvize DFT plot.

1 Introduction & Theory

1.1 Non-Periodic Signals Windowing

The DFT is equivalent to the Discrete Time Fourier Series of the N-periodic extension of the input sequence, the DFTs of non-periodic sequences suffer from Gibbs Phenomenon.To remove the high frequency components that arise due to Gibbs' Phenomenon, such non-periodic signals are multiplied by a window in time domain. This is used to make the signal square integrable, also to make infinitely long signal to a finite signal, since to take DFT we need finite aperiodic signal.

Hamming Windows used in our assignment shows high-side-band suppression and hence the discontinuities at the end of signal are suppressed. Hamming Window belongs to the class of Cosine-Sum windows. The Hamming window function is given by

$$x[n] = 0.54 + 0.46cos(\frac{2\pi n}{N-1}), \tag{1}$$

where $|n| \leq \frac{N-1}{2}$.

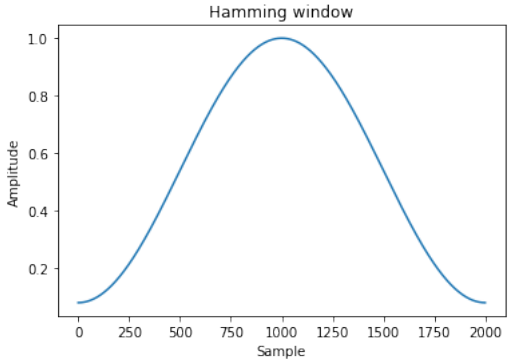

Figure 1: Hamming Window - time domain plot

1.2 Estimation of frequency and phase of sinusoidal signal from its DFT

For a sinusoidal signal $\cos(\omega t + \delta)$, we need to estimate ω and δ. We sample the signal for sufficient time at a rate greater than Nquist rate to obtain the DFT. The peak frequency is assumed to be the ω required, and the phase corresponding to it as δ.

1.3 Chirped Signal

A chirp is a signal in which the frequency increases (up-chirp) or decreases (down-chirp) with time. It is generally of the form $\sin(\phi(t) + \delta)$. The instantaneous angular frequency, , is defined as the phase rate as given by the first derivative of phase.

$$\omega(t) = \frac{d\phi(t)}{dt}, \tag{2}$$

In our problem we consider a chirp signal given by,

$$f(t) = cos(16t(1.5 + \frac{t}{2\pi}))$$

2 Excercise

2.1 Analysis of $sin(\sqrt{2}t)$ signal

The DFT is computed using the FFT algorithm, implemented using `numpy.fft` module. To centre the fft vector about the 0-frequency bin, the vector is re-arranged using the `fftshift()` function. In order to get a purely imaginary FFT, the signal is made anti-symmetric by setting the N/2 th element 0. The python code snippet and the corresponding plot are give below.

```
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]
```

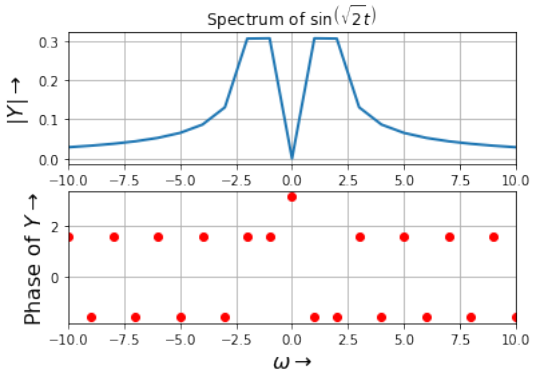

Figure 2: Spectrum Plot without using Window

In order to obtain a better DFT plot, the given signal can be multiplied by the Hamming window.

```
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
```

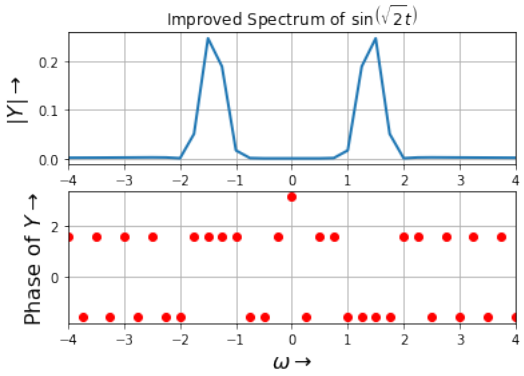

Figure 3: Spectrum Plot using Window

2.2 DFT of signal $cos^3(0.86t)$

Computation of DFT of $cos^3(0.86t)$ is very similar to previous problem and the procedure is same. However ,unlike the sine signals, $cos^3(0.86t)$ does not have discontinuities at the ends of the interval $(-\pi, \pi)$, Gibbs Phenomenon is not observed even when a Hamming window is not used.

```
# without windowing
y = cos(0.86*t)**3
y[0] = 0
y = fftshift(y)
Y = fftshift(fft(y))/256.0

# with windowing
y1 = y*wnd
y1[0] = 0
y1 = fftshift(y1)
Y1 = fftshift(fft(y1))/256.0
```

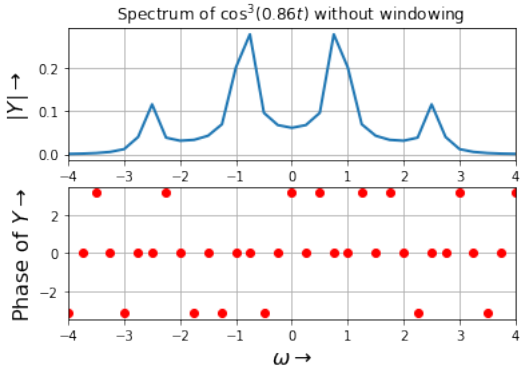

Figure 4: Spectrum Plot without Windowing

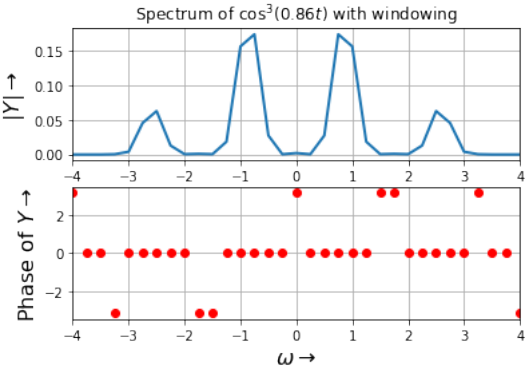

Figure 5: Spectrum Plot with Windowing

2.3 Frequency and Phase estimation of $cos(\omega_o t + \delta)$

For our purposes, ω_o and δ are randomly chosen from a uniform distribution over their respective range as specified. The DFT of the resulting signal is plotted. The estimate of ω_o is taken to be that value for which the frequency response has maximum magnitude and the δ as the phase corresponding to that frequency in the DFT plot.

```
#  Spectrum of cos(w0t +delta) and extraction of 0 and .

w0 = random.uniform(0.5, 1.5)
delta = random.uniform(-pi, pi)

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 + delta)*wnd
y[0]=0
y = fftshift(y)
Y = fftshift(fft(y))/128.0

w = linspace(-pi*fmax,pi*fmax,129); w = w[:-1]

i = where(w>=0)
w_estimate = argmax(abs(Y[64:]))
n = abs(w-w_estimate).argmin()
phase = angle(Y[n])

print("Calculated value of w0 : ",w_estimate)
print("Calculated value of delta : ",phase)
```

Calculated value of w0 : 1
Calculated value of delta : 1.281793623998017

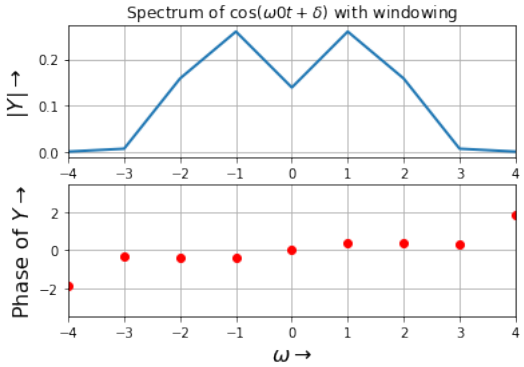

Figure 6:

The above procedure of estimating frequency and phase can also be done when the signal is corrupted by white gaussian noise as well.

```
y = (cos(w0*t + delta) + 0.1*randn(128))
```

Calculated value of w0 in the presence of noise : 1 Calculated value of delta in the presence of noise : 1.267422333220794

2.4 Chirp Signal

We plot the DFT of the function where $cos(16t(1.5 + \frac{t}{2\pi}))$ 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 $-\pi$ and is 32 samples near $+\pi$. The DFT of the corresponding signal without using Hamming window is plotted below,

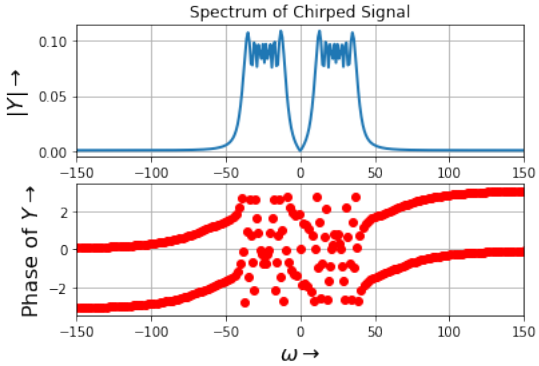

Figure 7:

The DFT of the corresponding signal using Hamming window is plotted below,

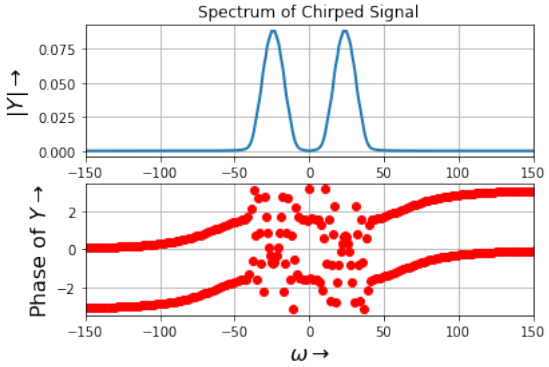

Figure 8:

To compute the Time-Frequency Plot of this signal the 1024-long vector is divided into 16 sub-vectors each of length 64. The FFT of each of the sub-vectors are computed and then plotted on a surface plot. Before computing its FFT, the subvector is multiplied by a Hamming window.

```
# Surface plot: frequency of the signal vs time

Y_magnitude = zeros((16,64))
Y_phase = zeros((16,64))
time_segment = split(t,16)
```

```
n = arange(64)
wnd = fftshift(0.54+0.46*cos(2*pi*n/64))

for i in range(len(time_segment)):

        y = cos(16*time_segment[i]*(1.5 + time_segment[i]/(2*pi)))*wnd
        y[0] = 0
        y = fftshift(y)
        Y = fftshift(fft(y))/64.0
        Y_magnitude[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)
```

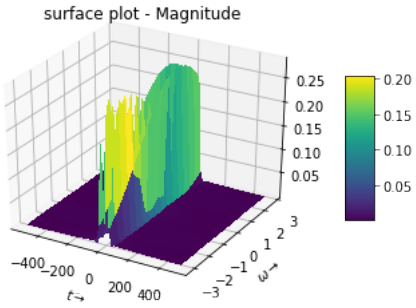

Figure 9:

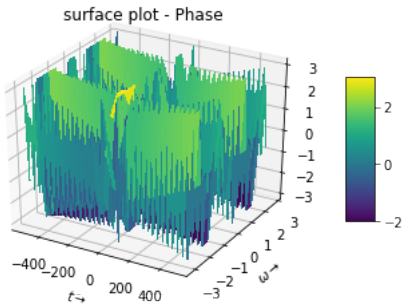

Figure 10:

3 Conclusion

In this assignment we have looked at the need of windowing in the case of computing DFT for non-periodic signals.Windowing is essential to supress the high frequency components causing Gibbs phenomenon due discontinuities at signal ends. We have also looked at fourier spectra of Chirp signals in time frequency plots.