# Assignment No 10: Spectra of non-periodic signals

Sakthi Harish D T (EE19B054)

May $7^{th}$, 2021

## Abstract:

In this assignment, we aim to,

- obtain the DFT of non-periodic signals
- improve the DFT by the use of Hamming window
- visualize the spectra using plots
- retrieve the signal parameters from the plots
- show how the frequency of signal varies with time

## 1 Introduction:

- We will use windowing to get better DFT of non-periodic signal.
- Windowing causes the signal to get suppressed in the edges, so that the breaks are not that significant.
- We use the Hammimg window function which is generally used in narrow-band applications.

$$W_N[n] = \begin{cases} 0.54 + 0.46\ cos(\frac{2\pi n}{N-1}),\ |n| < N \\ 0,\ otherwise \end{cases}$$

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

### 2.1 Without Hamming Window:

- Let us plot the DFT of $sin(\sqrt{2}t)$ without the hamming window.
- The following Python code does this function

```
t= np.linspace(-1*np.pi, np.pi, 65); t= t[:-1]
dt= t[1]-t[0]
fmax= 1/dt
y= np.sin(np.sqrt(2)*t)
y[0]= 0
y= fft.fftshift(y)
Y= fft.fftshift(fft.fft(y))/64.0
w= np.linspace(-1*np.pi*fmax, np.pi*fmax, 65)[:-1]
spectrum_plot(w,Y,r"Spectrum of $sin(\sqrt{2}t)$",xLimit=[-10, 10])
```

The obtained spectrum is as follows:

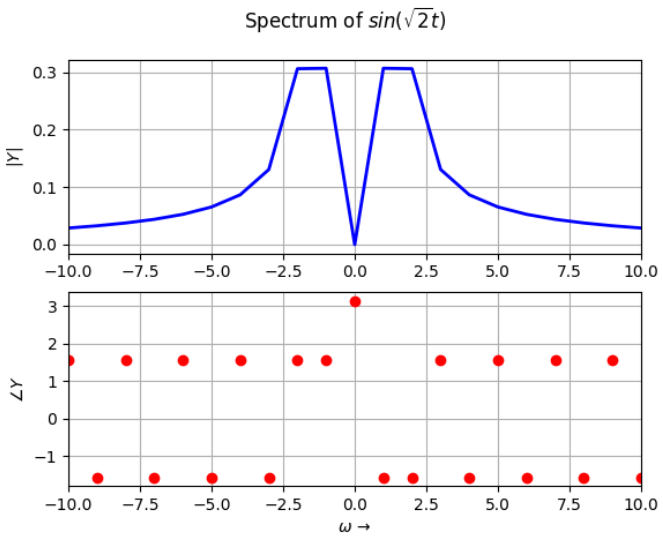


Figure 1: Spectrum of $sin(\sqrt{2}t)$ without windowing

## 2.2 With Hamming Window:

- Let us plot the DFT of $sin(\sqrt{2}t)$ with the hamming window.
- The following Python code does this function

```
t= np.linspace(-4*np.pi, 4*np.pi, 257);t=t[:-1]
dt= t[1]-t[0]; fmax= 1/dt
n= np.arange(256)
wnd= fft.fftshift(0.54+0.46*np.cos(2*np.pi*n/(n.size-1)))          #Hamming Window
y= np.sin(np.sqrt(2)*t)
y= y*wnd
y[0]= 0
y= fft.fftshift(y)
Y = fft.fftshift(fft.fft(y))/256.0
w = np.linspace(-np.pi*fmax, np.pi*fmax, 257);w= w[:-1]
spectrum_plot(w,Y,r"Spectrum of $sin(\sqrt{2}t)*w(t)$",magStyle='b-o',xLimit=[-4, 4])
```

The obtained spectrum is as follows:

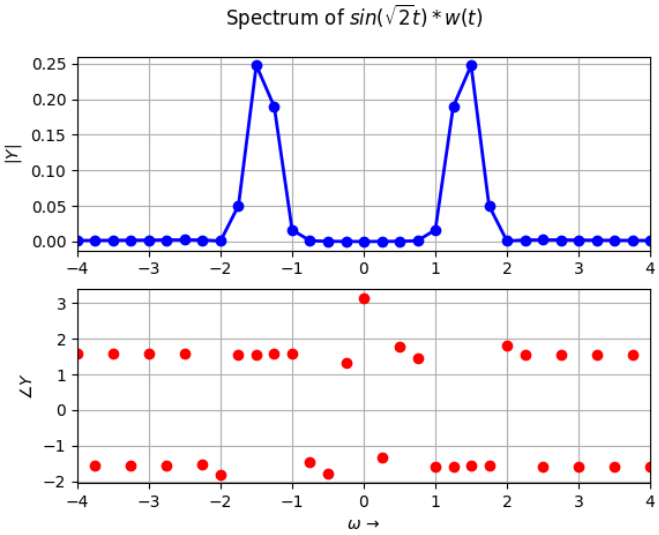


Figure 2: Spectrum of $sin(\sqrt{2}t)$ with windowing

## 3 DFT of $cos^3(0.86t)$ :

Now, we look at one more function, $cos^3(0.86t)$. We obtain the spectrum for this function with and without a Hamming window.
The obtained plots are as shown below:

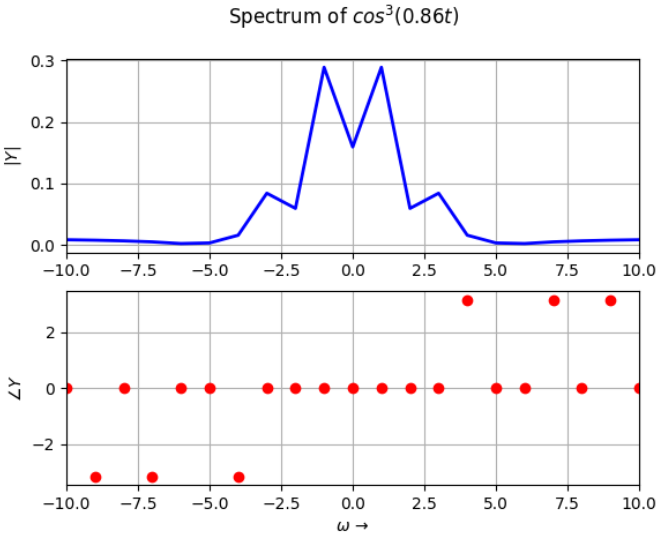


Figure 3: Spectrum of $cos^3(0.86t)$ without windowing

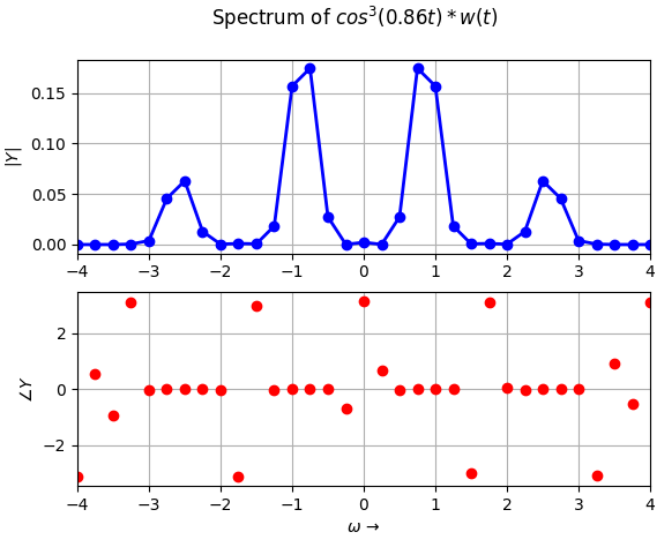


Figure 4: Spectrum of $cos^3(0.86t)$ with windowing

We can see narrower and sharper peaks at the frequencies that are present in the signal.

# 4 Estimating $\omega_o$ and $\delta$ from the spectrum:

## 4.1 Without noise:

The spectrum of $cos(\omega_o t + \delta)$ with $\omega_o$= 1 and $\delta$= $\pi/4$ is as follows:

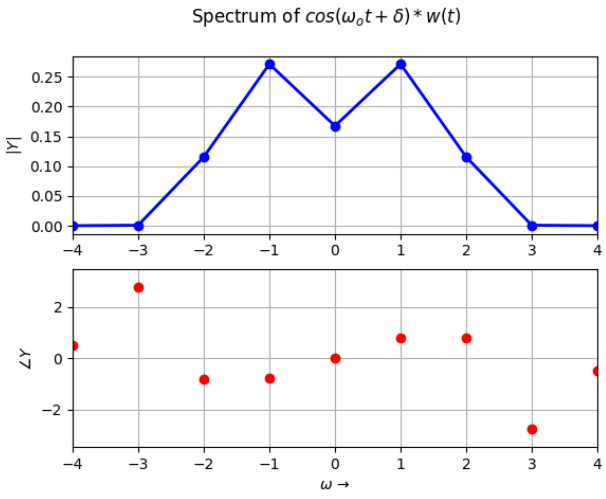


Figure 5: Spectrum of $cos(\omega_o t + \delta)$ with windowing

After plotting the spectrum for $cos(\omega_o t + \delta)$ with $\omega_o$= 1 and $\delta$= $\pi/4$, we can see that extracting the values of $\omega_o$ and $\delta$ will not be accurate because the resolution of the frequency axis is not sufficient.
So, we find the value of $\omega_o$ by using `weighted mean`.

$$\omega_{o_{est}} = \frac{\sum_\omega |Y(j\omega)|^2 \cdot \omega}{\sum_\omega |Y(j\omega)|^2} \forall \omega > 0$$

$\delta$ can be found by calculating the phase of the DFT at $\omega$ nearest to the estimated $omega_o$ using the above method.
The python code to perform the estimation part is given below:

```
#Estimation of wo & d
wo_estim= np.sum((np.abs(Y)**2*np.abs(w))/(np.sum(abs(Y)**2)))
i= np.abs(w-wo_estim).argmin()
d_estim= np.angle(Y[i])
```

**Estimated values:**

- Estimated value of $\omega_o$ : 0.9973353652918199
- Estimated value of $\delta$ : 0.7888010081342315

The results obtained are fairly accurate.

## 4.2 With noise:

Now, we add `white gaussian noise` to the same signal given above. We use the `randn()` function to generate the noise given by `0.1*randn(N)`, where N is the number of samples. The spectrum is as given below:

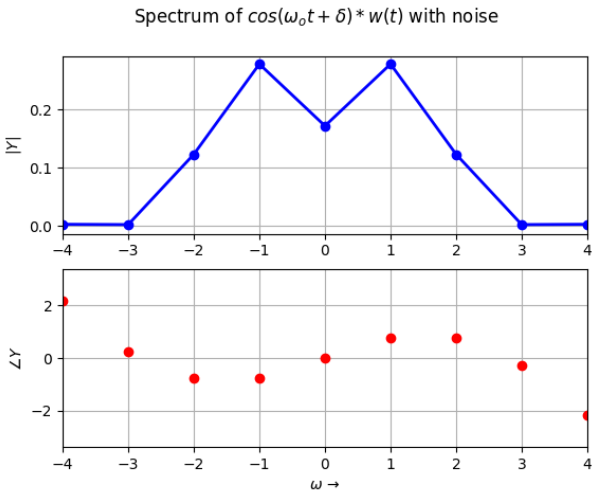


Figure 6: Spectrum of $cos(\omega_o t + \delta)$ with noise

We use the same method for obtaining the estimated values of $\omega_o and \delta$ .
**Estimated values:**

- Estimated value of $\omega_o$ : 1.6477300472784084
- Estimated value of $\delta$ : 0.787591957607207

The results obtained are slightly inaccurate when compared to the case with no noise added.

# 5 DFT of Chirped Signal:

- We will plot the DFT of the function $cos(16t(1.5 + \frac{t}{2\pi}))$ where $-\pi \leq t < \pi$ in 1024 steps. This signal is known as a chirped signal.
- the signal's 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 spectrum of the Chirped signal is as shown below:

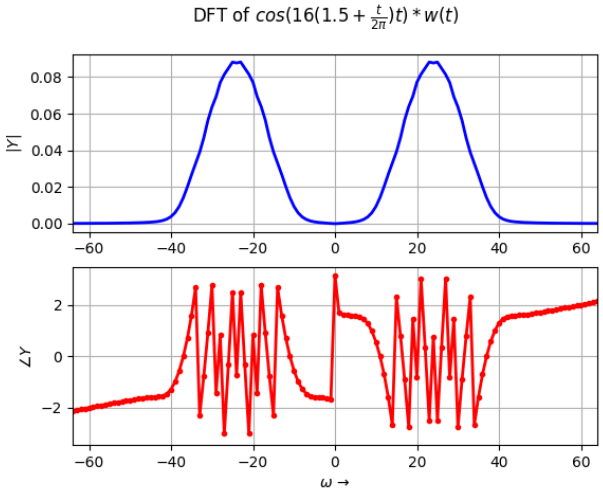


Figure 7: Spectrum of chirped signal with windowing

Now, we break the same signal into pieces that are 64 samples wide, and calculate the DFT for each of the piece and store them as columns in 2-D array.
We then plot the 2-D array as a surface plot to show how the frequency of the signal varies with time. The below Python code does the discussed function:

```
#Question 6: Variation of frequency of the above signal with time.
y_piece = np.split(y, 1024//64)          #breaking the signal into pieces
X = np.zeros((1024//64, 64), dtype=complex)
for i in range(1024//64):                #Finding DFT for the split signal
    X[i] = fft.fftshift(fft.fft(y_piece[i]))/64

t = t[::64]      #Surface plot of the 2D array containing the DFT of each piece
w = np.linspace(-fmax*np.pi, fmax*np.pi, 65);w=w[:-1]
t, w = np.meshgrid(t, w)
fig = plt.figure(fig_no)
ax = fig.add_subplot(111, projection='3d')
surface = ax.plot_surface(w, t, abs(X).T, cmap='jet')
fig.colorbar(surface)
plt.xlabel(r"Frequency $\to$")
plt.ylabel(r"Time $\to$")
plt.title(r"Magnitude $\|Y\|$")
plt.show()
```

The obtained surface plot is given below:

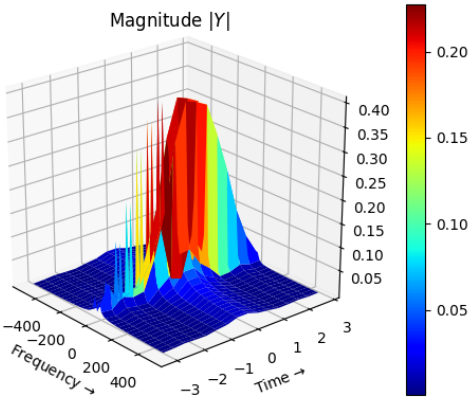


Figure 8: Surface plot of spectrum of chirped signal

# 6 Conclusion:

In this assignment,

1. we obtained the DFT of non-periodic function.
2. we saw how the usage of Hamming window improved the results
3. we extracted the signal parameters from the spectrum of signals
4. we plotted the surface plot of the chirped signal to show the time dependence of frequency of the signal.