EE2703 : Applied Programming Lab
Assignment 10
Spectra of Non periodic Signals

Bachotti Sai Krishna Shanmukh
EE19B009

May 22, 2021

1 Introduction

In this assignment, we further explore our usage of np.fft module to get spectra for non-periodic signals, the challenges we face like the *Gibbsphenomenon* and solutions like *windowing*.

2 Examples - Building the spectrum of $sin(\sqrt{2}t)$

2.1 DFT using regular method for periodic signal

By using the previous assignment's method for finding DFT spectrum, we get the following:

```
def dft(tstart,tend,N,f,xlim,title):
    """
    DFT spectrum without any windowing
    tstart : Start Time
    tend : End time
    N : No. of samples
    f : Signal function
    xlim : Plotting limits of x axis
    title : Title of plot
    """
    t = np.linspace(tstart,tend,N+1)
    t = t[:-1]
    dt = t[1] - t[0]
    fmax = 1/dt
    w = np.linspace(-np.pi*fmax,np.pi*fmax,N+1)
```

```
    w = w[:-1]
    y = f(t)
    y[0] = 0
    y = np.fft.fftshift(y)
    Y = np.fft.fftshift(np.fft.fft(y))/N
    fig ,ax = plt.subplots(figsize=(7,7))
    plt.subplot(2,1,1)
    plt.plot(w,np.abs(Y),'b')
    plt.xlim([-xlim,xlim])
    plt.ylabel(r"Magnitude$\longrightarrow$")
    plt.title(r"Spectrum of " + title)
    plt.grid()
    plt.subplot(2,1,2)
    plt.plot(w,np.angle(Y),'ro')
    plt.ylabel(r"$\phi$ $\longrightarrow$")
    plt.xlim([-xlim,xlim])
    plt.xlabel(r"$\omega \longrightarrow$")
    plt.grid()
```

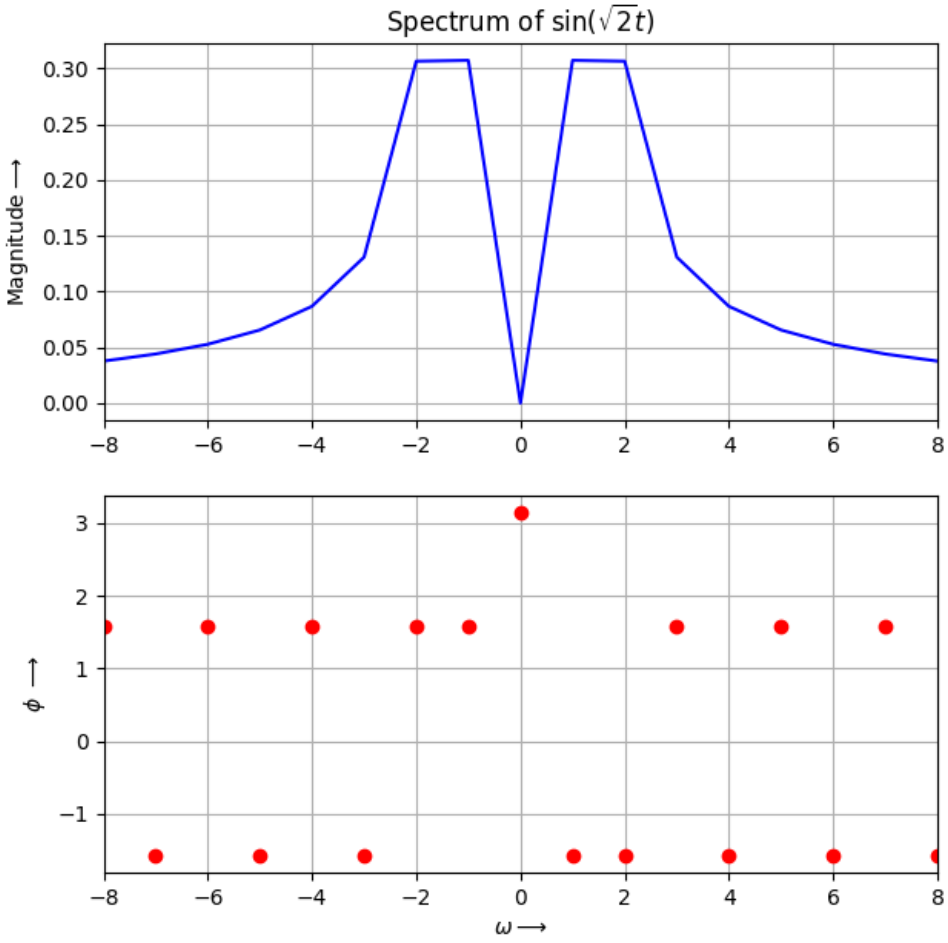

Figure 1: Spectrum of $sin(\sqrt{2}t)$

2.2 Gibbs's phenomenon

From the spectrum above we can observe that peaks are broad and different from what we expected. We expect two sharp peaks at the frequency of signal. This error occurs because we are finding the spectrum for a different signal. To get a clear picture, let's plot the signal we wish to see the spectrum and the signal for which we get the spectrum by the above method. Below is the plot of the signal $sin(\sqrt{2}t)$

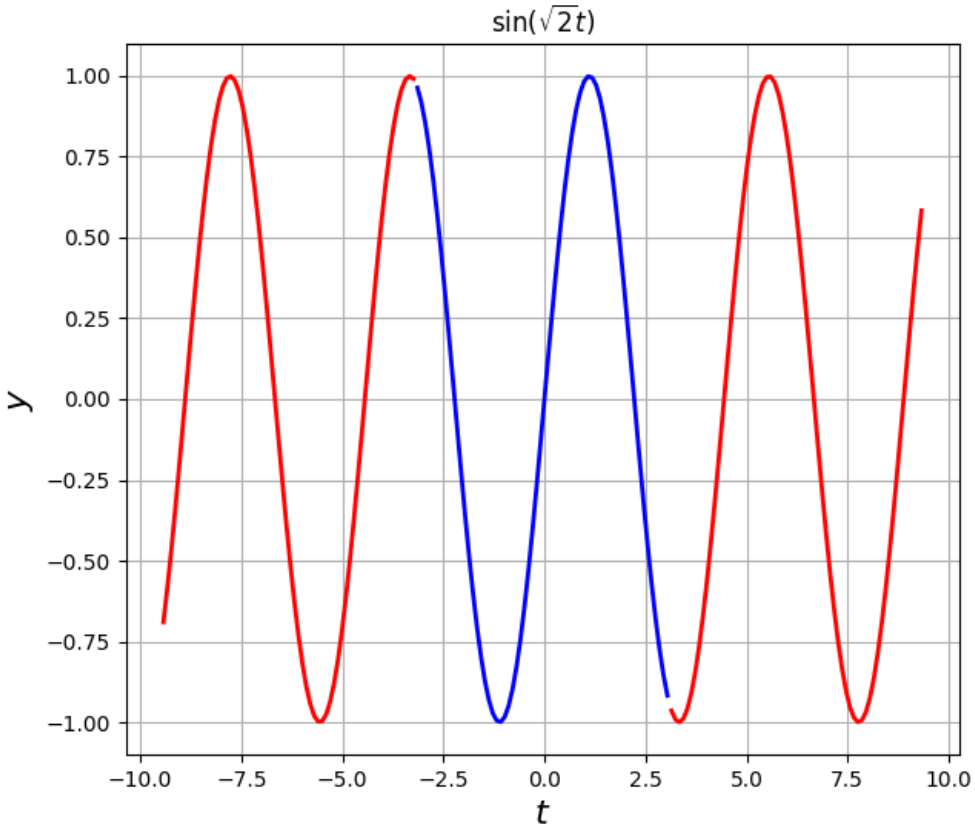

Figure 2: Signal $sin(\sqrt{2}t)$

However, if we look at a 2π wrapped version of this signal, we observe discontinuities. This causes what we call as Gibbs's phenomenon and this results in a broad peak.

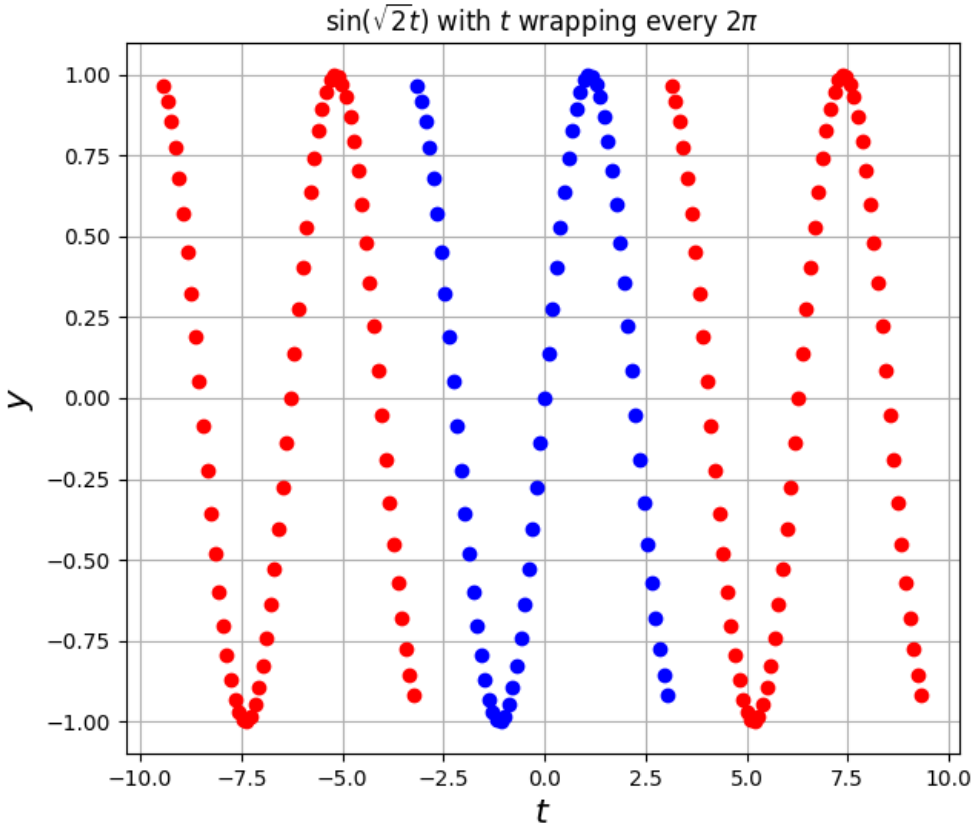

Figure 3: Signal $sin(\sqrt{2}t)$

Before we go into fixing the problem caused by Gibbs's phenomenon, let's look at why we don't see the spikes in more detail.
Consider a ramp signal. It's Fourier coefficients decay slowly i.e proportional to $\frac{1}{w}$. The DFT is just like the Fourier series, except that both time and frequency are samples. So, if the time samples are like a ramp, the frequency samples will decay as $\frac{1}{w}$.

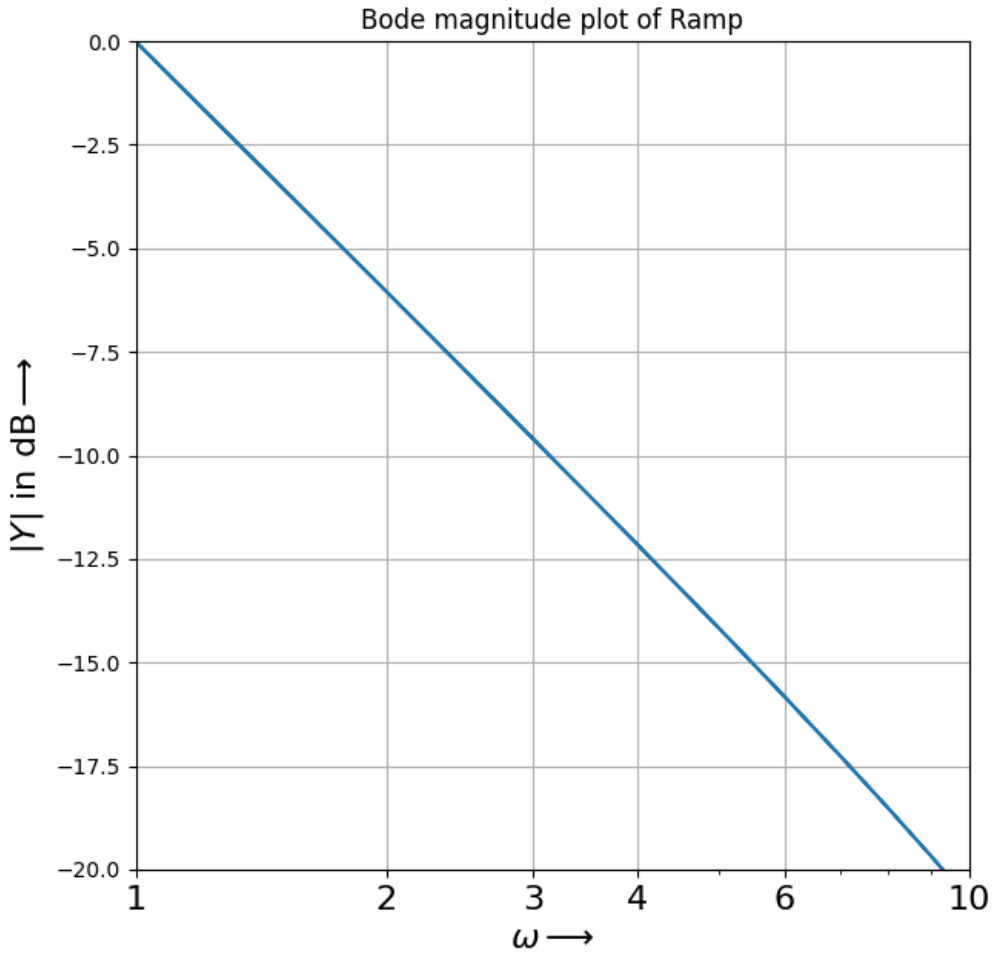

Figure 4: Bode plot of ramp signal

Clearly the spectrum decays as 20 dB per decade, which corresponds to $\frac{1}{w}$. The big jumps at nπ force this slowly decaying spectrum, which is why we don't see the expected spikes for the spectrum of $sin(\sqrt{2}t)$

2.3 Windowing

This problem can be fixed by windowing, an operation where we multiply the signal in time domain by a window signal. Here, we choose the Hamming window signal. It is defined as:

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

Let's look at our signal multiplied to hamming window signal in the below plot

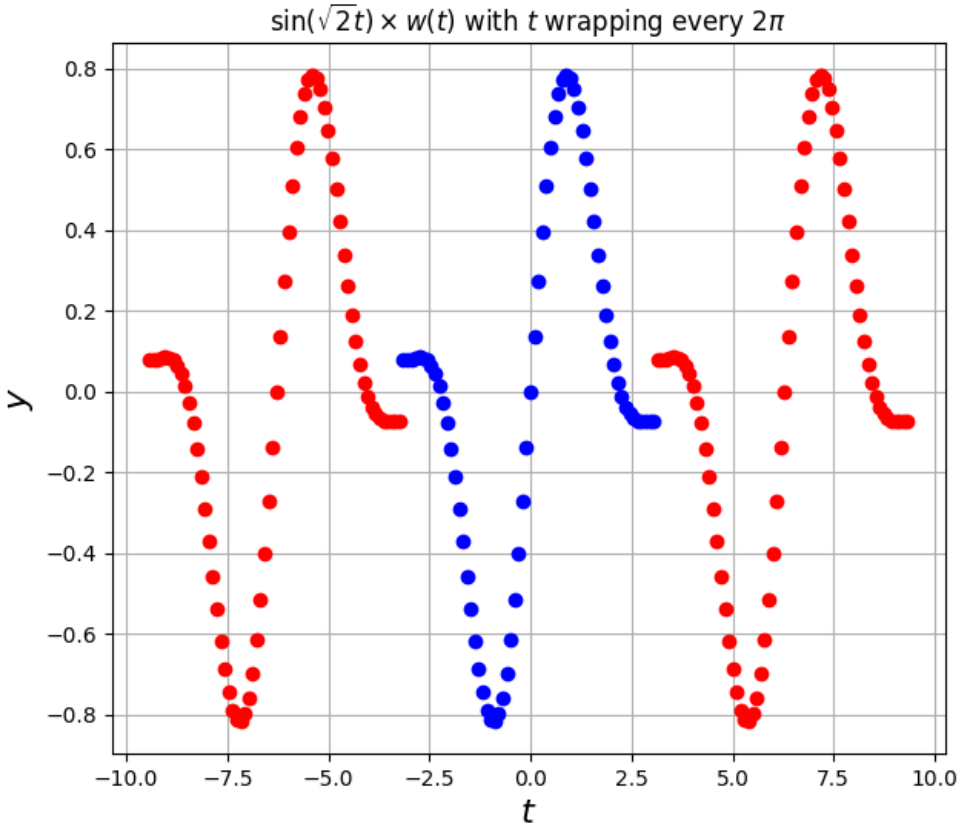

Figure 5: Plot of $sin(\sqrt{2}t) \times w(t)$

The jump is still there, but it is much reduced and the effect of Gibbs's phenomenon is reduced greatly. Below is the code and spectrum for windowed version of signal.

```
def window_dft(tstart,tend,N,f,xlim,title,antisymmetry = True):
    """
    DFT spectrum with WINDOWING
    tstart : Start Time, tend : End time, N : No. of samples
    f : Signal function, xlim : Plotting limits of x axis, title : Title of plot
    antisymmetry : if true then y[0] is initialized to zero before fftshift
    """
    t = np.linspace(tstart,tend,N+1)
    t = t[:-1]
    dt = t[1] - t[0]
    fmax = 1/dt
    n=np.arange(N)
    wnd=np.fft.fftshift(0.54+0.46*np.cos(2*np.pi*n/(N-1)))
    w = np.linspace(-np.pi*fmax,np.pi*fmax,N+1)
    w = w[:-1]
    y = f(t)*wnd
    if antisymmetry:
        y[0] = 0
```

```
    y = np.fft.fftshift(y)
    Y = np.fft.fftshift(np.fft.fft(y))/N
    fig ,ax = plt.subplots(figsize=(7,7))
    plt.subplot(2,1,1)
    plt.plot(w,np.abs(Y),'b',lw=2)
    plt.xlim([-xlim,xlim])
    plt.ylabel(r"Magnitude$\longrightarrow$")
    plt.title(r"Spectrum of "+ title +r"$\times w(t)$")
    plt.grid()
    plt.subplot(2,1,2)
    plt.plot(w,np.angle(Y),'ro')
    plt.ylabel(r"$\phi$ $\longrightarrow$")
    plt.xlim([-xlim,xlim])
    plt.xlabel(r"$\omega \longrightarrow$")
    plt.grid()
```

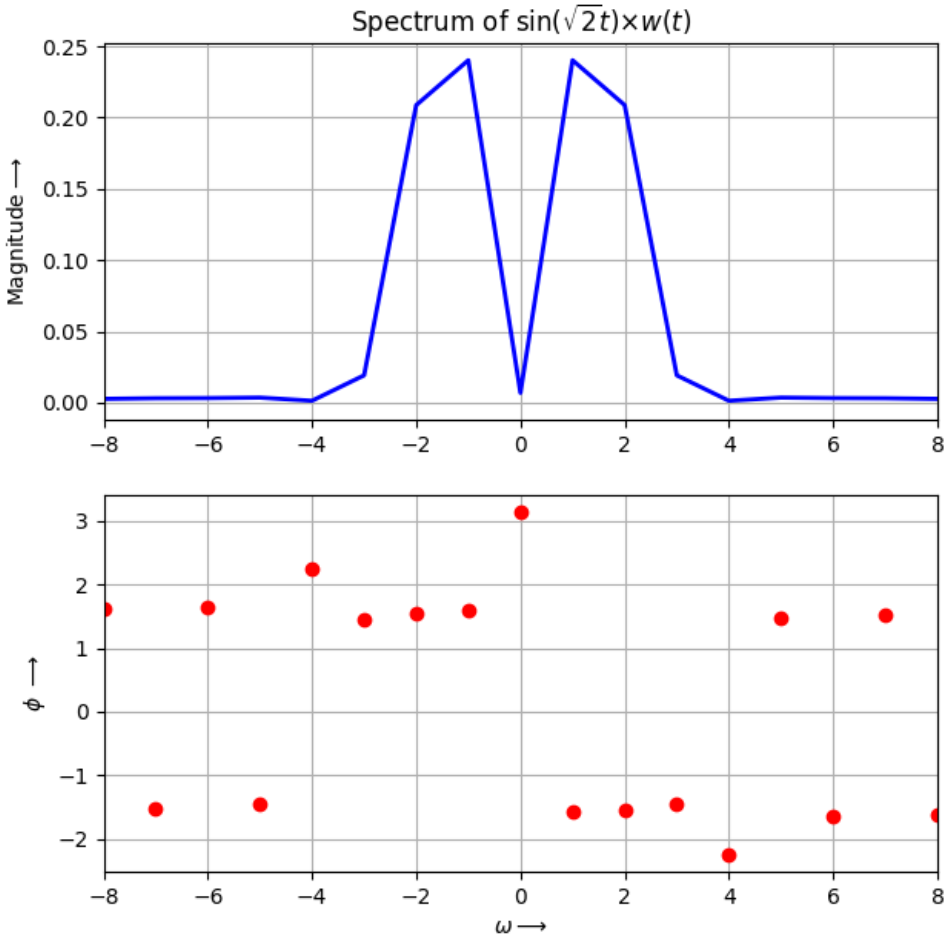

Figure 6: Spectrum of windowed $sin(\sqrt{2}t)$

2.4 Increasing the window-size and samples

If we increase the window size and the number of sample without altering the sampling frequency, we can observe a sharper peak as shown below.

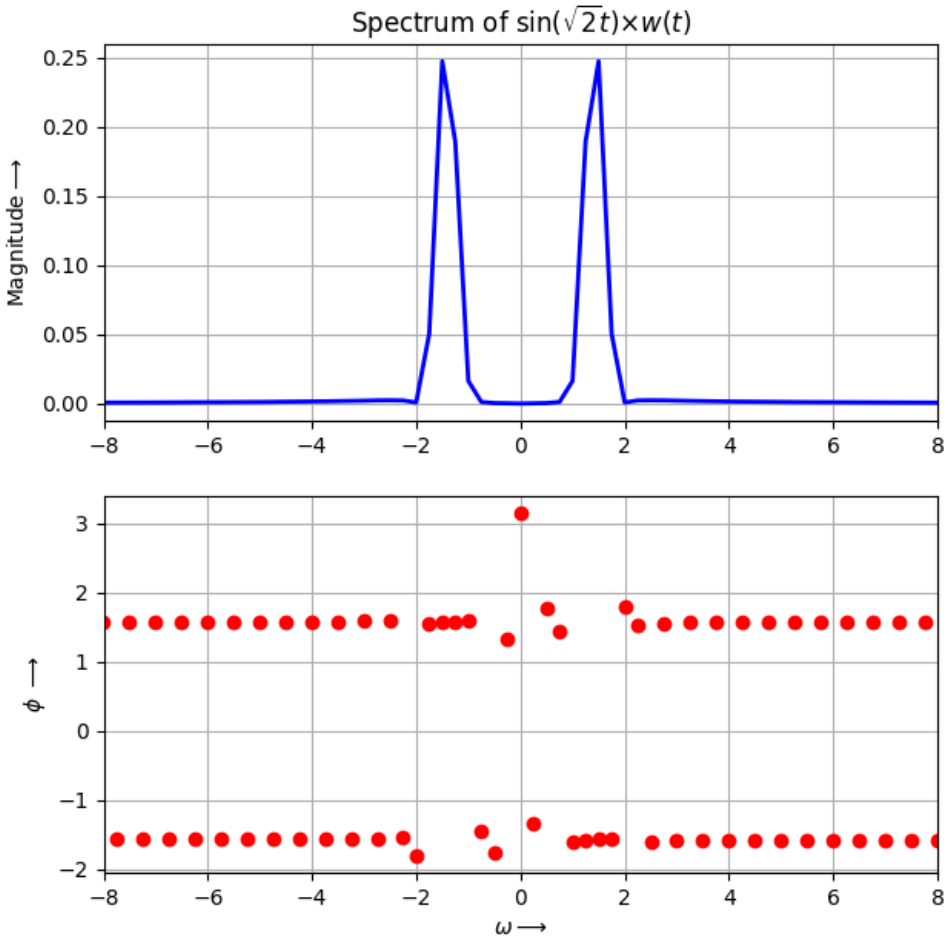

Figure 7: Spectrum of windowed $sin(\sqrt{2}t)$ with larger window

2.5 Spectrum of $sin(1.5t)$ with windowing

Below is the spectrum of $sin(1.5t)$ with windowing. It's peaks are sharp as expected with windowing too.

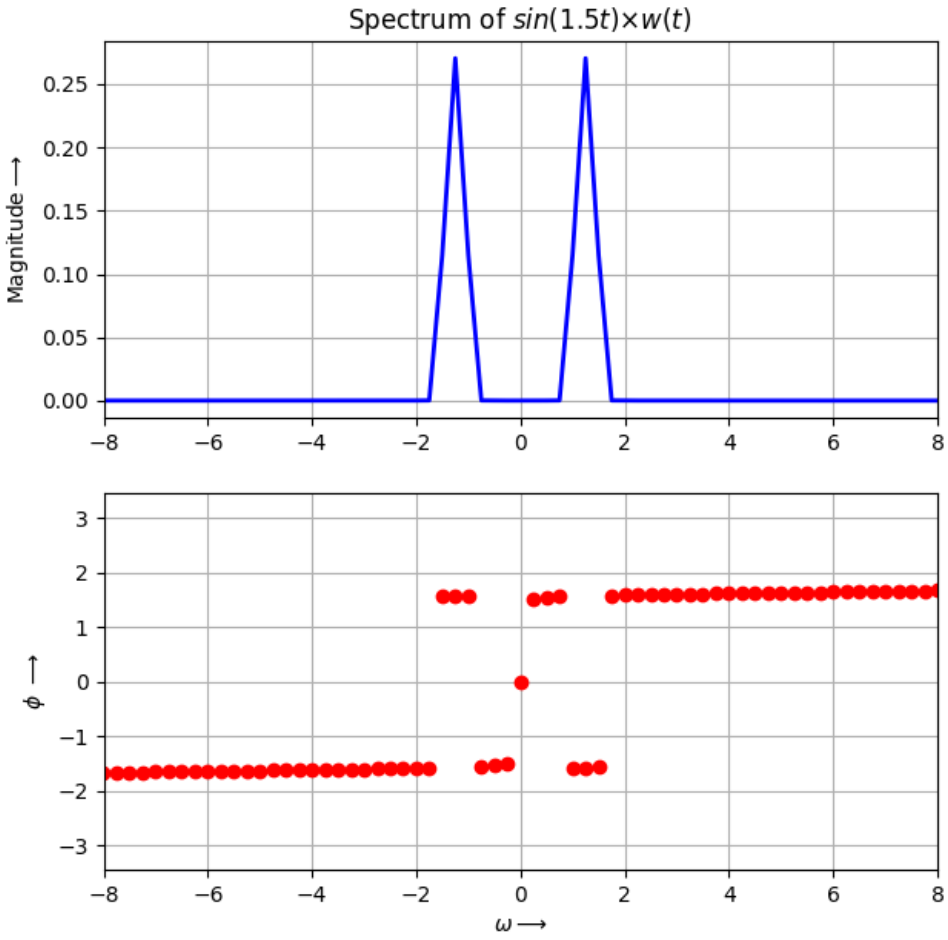

Figure 8: Spectrum of windowed $sin(1.5t)$

3 Spectrum of $cos^3(0.86t)$

We can see that the effect of windowing is even better for $cos^3(0.86t)$. The peaks are narrower and sharper in windowed version.

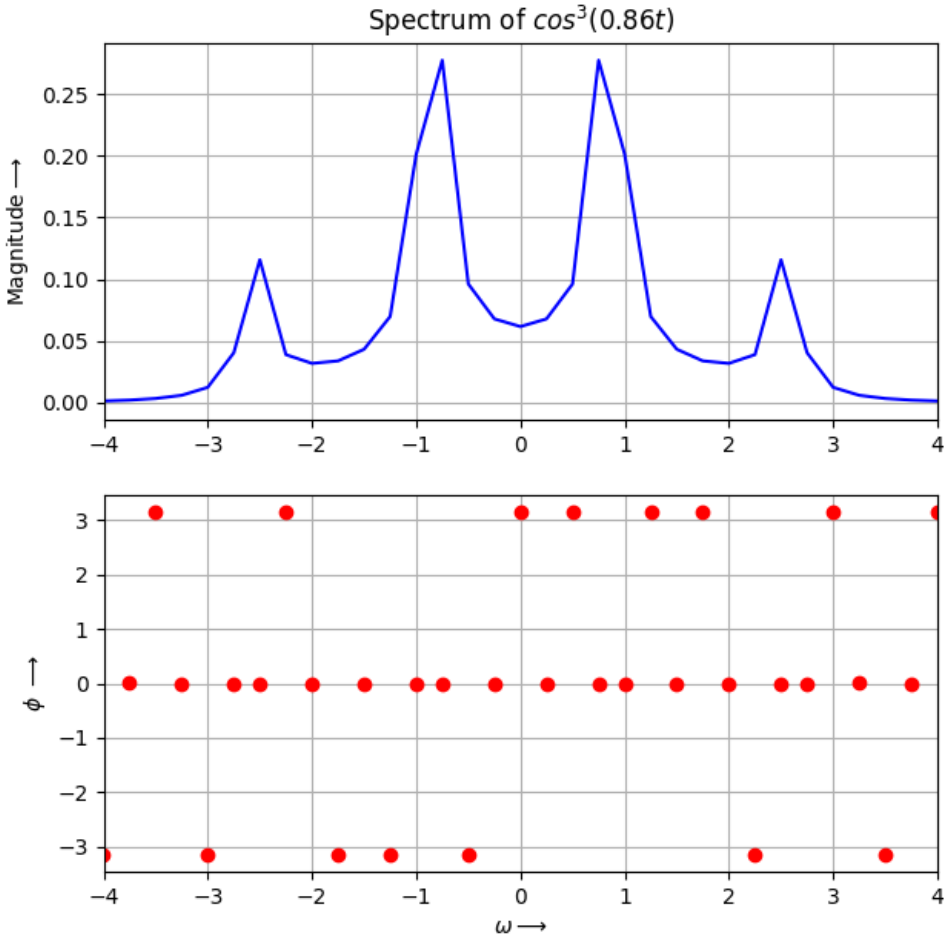

Figure 9: Spectrum of $cos^3(0.86t)$

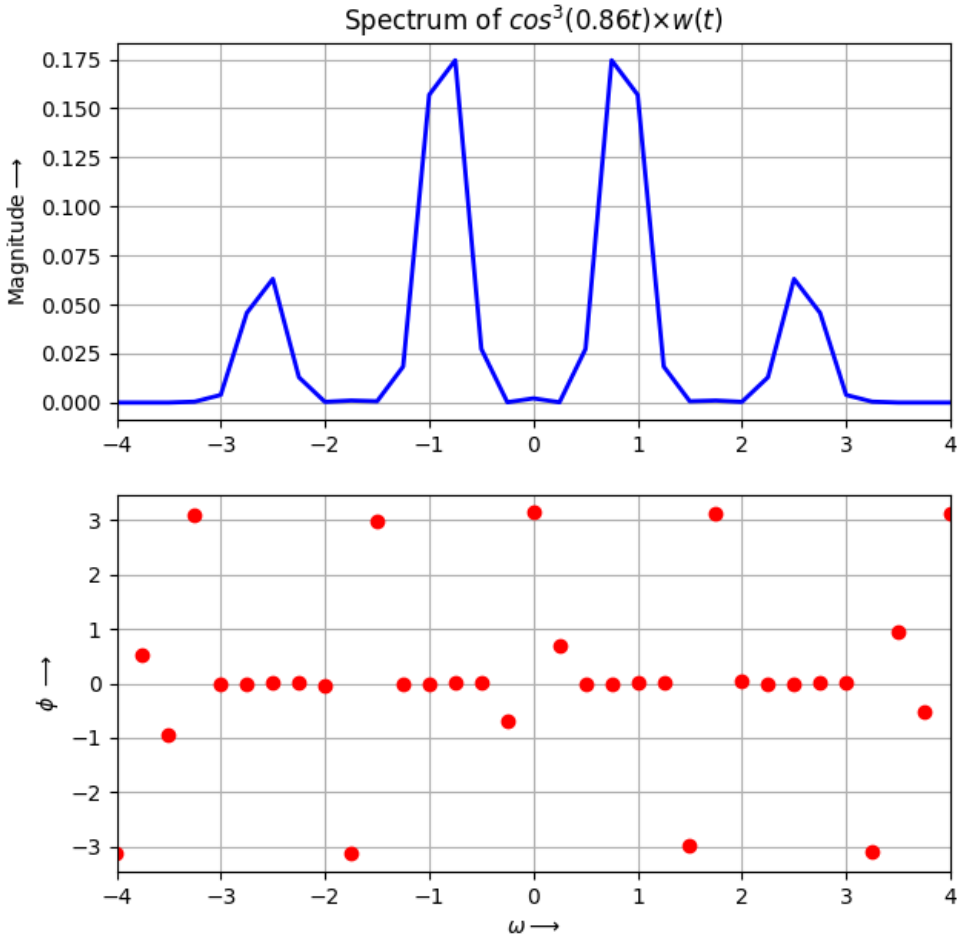

Figure 10: Spectrum of windowed $cos^3(0.86t)$

4 Estimating ω and δ using DFT

Given, a 128 element vector which are samples of the signal $cos(\omega_o t + \delta_o)$, where ω_0 and δ_o are to be estimated. We cannot extract ω_o from the DFT spectrum because the number of samples are too low and the peaks are not sharp impulses. To get an estimate, we can use weighted average approach. Given below is the equation for finding ω_{est}.

$$\omega_{est} = \frac{\sum_\omega |Y(j\omega)|^k \cdot \omega}{\sum_\omega |Y(j\omega)|^k} \tag{2}$$

Here k is a parameter which can be tweaked to get the best estimate. Usually we get the best estimation for k ranging between 1.5 to 3.
We can use the least squares approach for finding $delta_{est}$

$$cos(\omega_o t + \delta_o) = Acos(\omega_{est} t) + Bsin(\omega_{est} t)$$

$$A = cos(\delta_{est})$$
$$B = -sin(\delta_{est})$$
$$cos(\omega_o t + \delta_o) = Acos(\omega_{est} t) + Bsin(\omega_{est} t)$$
$$\delta_{est} = arctan(-B/A) \tag{3}$$

Below is the code for estimating both the parameters

```
def estimator(y, hamming, N=128,power = 2,plotspectrum=False):
    """
    y : input vector
    hamming : If true then hamming window is applied to signal
    N : 128 samples default value
    power : default value set to 2
    plotspectrum : If false then spectrum is not plotted
    """
    t = np.linspace(-np.pi,np.pi,N+1)
    t = t[:-1]
    fmax = 1.0/(t[1]-t[0])
    n = np.arange(128)
    wnd=np.fft.fftshift(0.54+0.46*np.cos(2*np.pi*n/(N-1)))
    w = np.linspace(-np.pi*fmax,np.pi*fmax,N+1)
    w = w[:-1]
    if hamming:
        y = y*wnd
    y[0] = 0
    y = np.fft.fftshift(y)
    Y = np.fft.fftshift(np.fft.fft(y))/N
    # Prediction of w
    w_pred = np.sum(np.abs(w)*np.abs(Y)**power)/np.sum(np.abs(Y)**power)
    # Prediciton of delta by least squares method
    c1 = np.cos(w_pred*t)
    c2 = np.sin(w_pred*t)
    A = np.c_[c1,c2]
    params = slg.lstsq(A, y)[0]
    cosd = params[0]
    sind = params[1]
    d_pred = np.arctan(-sind/cosd)
    #print(w_pred,d_pred)
    if plotspectrum:
        fig ,ax = plt.subplots(figsize=(7,7))
        plt.subplot(2,1,1)
        plt.plot(w,np.abs(Y),'b')
        plt.xlim([-4,4])
        plt.ylabel(r"Magnitude$\longrightarrow$")
        plt.title(r"Spectrum of y")
        plt.grid()
        plt.subplot(2,1,2)
        plt.plot(w,np.angle(Y),'ro')
```

```
        plt.ylabel(r"$\phi$ $\longrightarrow$")
        plt.xlim([-4,4])
        plt.xlabel(r"$\omega \longrightarrow$")
        plt.grid()
    return w_pred, d_pred
```

We can generate a 128-size vector for required parameters and pass the vector to above function and find the error in estimation.
For $\omega_o = 1.3$ and $\delta_o = -\frac{\pi}{2}$ we get:

```
Predicted w and delta of the signal are :
1.301233065273084  and  -1.5641547910529252
We achieve the best estimation at k =  2.875
Error in w prediction is  0.0012330652730838665
Error in delta prediction is  0.0066415357419713494
```

Below plot contains the spectrum of given sequence.

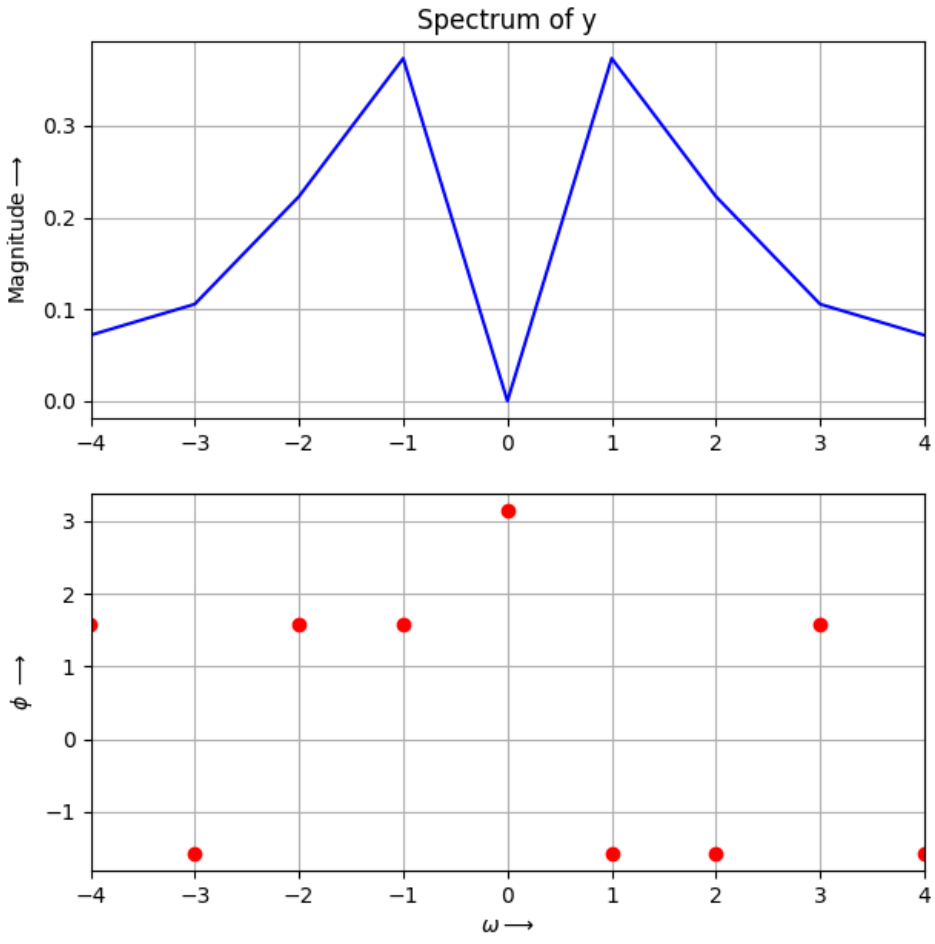

Figure 11: Spectrum of given signal sequence

5 Estimating ω and δ in presence of White noise

Given that the signal has a random white noise with amplitude 0.1. We can use the same estimation function defined above. We get a slightly larger error in the predicted parameters. For $\omega_o = 1.3$ and $\delta_o = -\frac{\pi}{2}$ we get:

```
Predicted w and delta of the signal are
1.3613855407159534  and  1.5078489778220687
We achieve the best estimation at k=  2.875
Error in w prediction is  0.049969946149938105
Error in delta prediction is  0.053745199266615806
```

Below plot contains the spectrum of given sequence with added white Gaussian noise. We can hardly observe any change in the spectra. This is due to the low amplitude of noise compared to main signal.

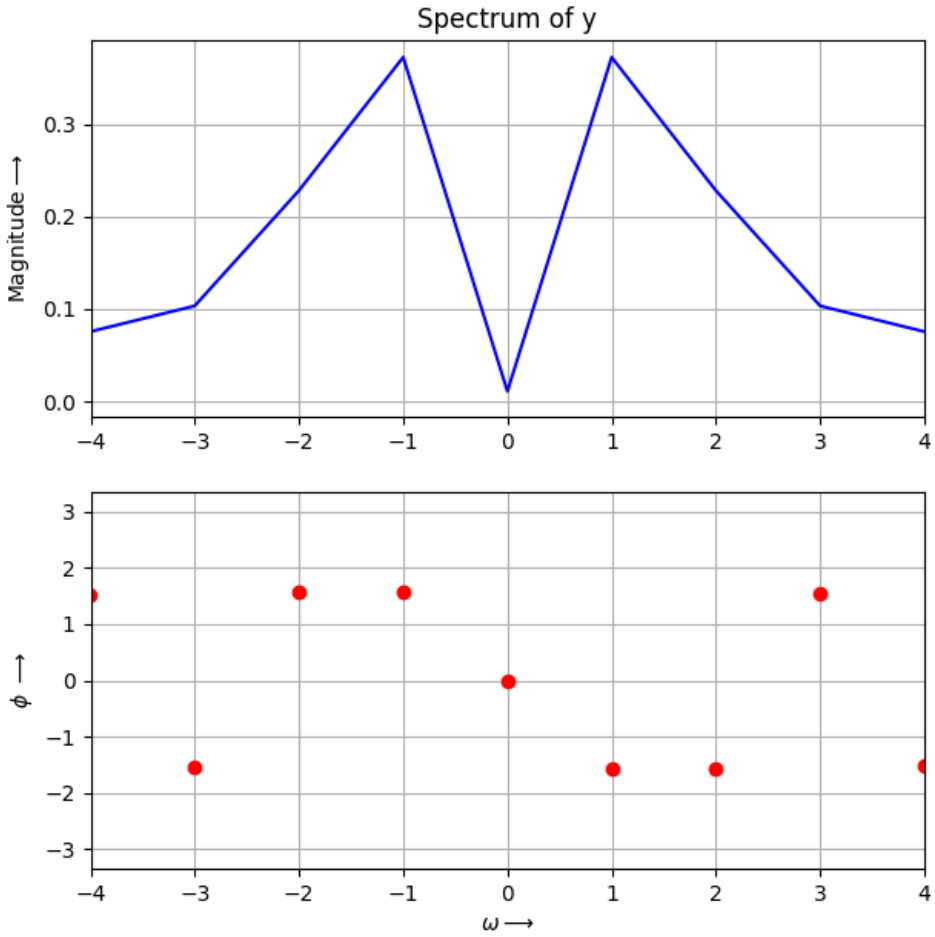

Figure 12: Spectrum of given signal sequence with added white noise

6 DFT of Chirp signal

A chirped signal's frequency is also a function of time. Below is one such signal

$$chirp(t) = cos(16(1.5 + \frac{t}{2\pi})t) \tag{4}$$

Below is the plot of the signal.

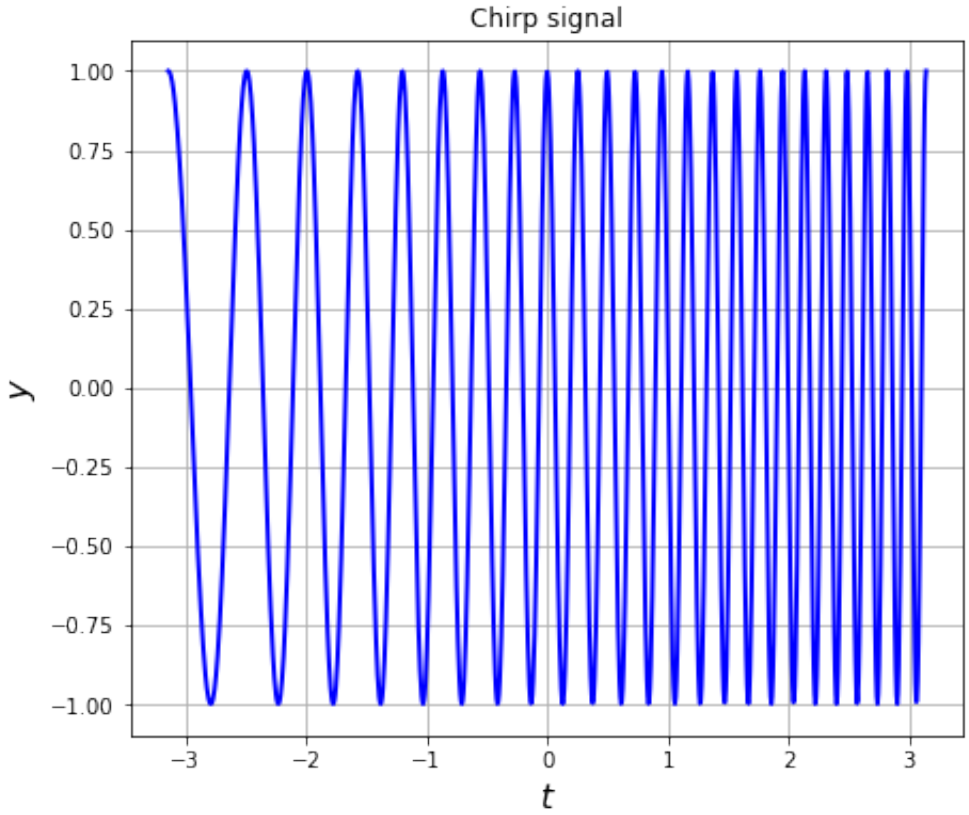

Figure 13: Plot of chirped signal

Below is the plot of the spectrum without hamming window

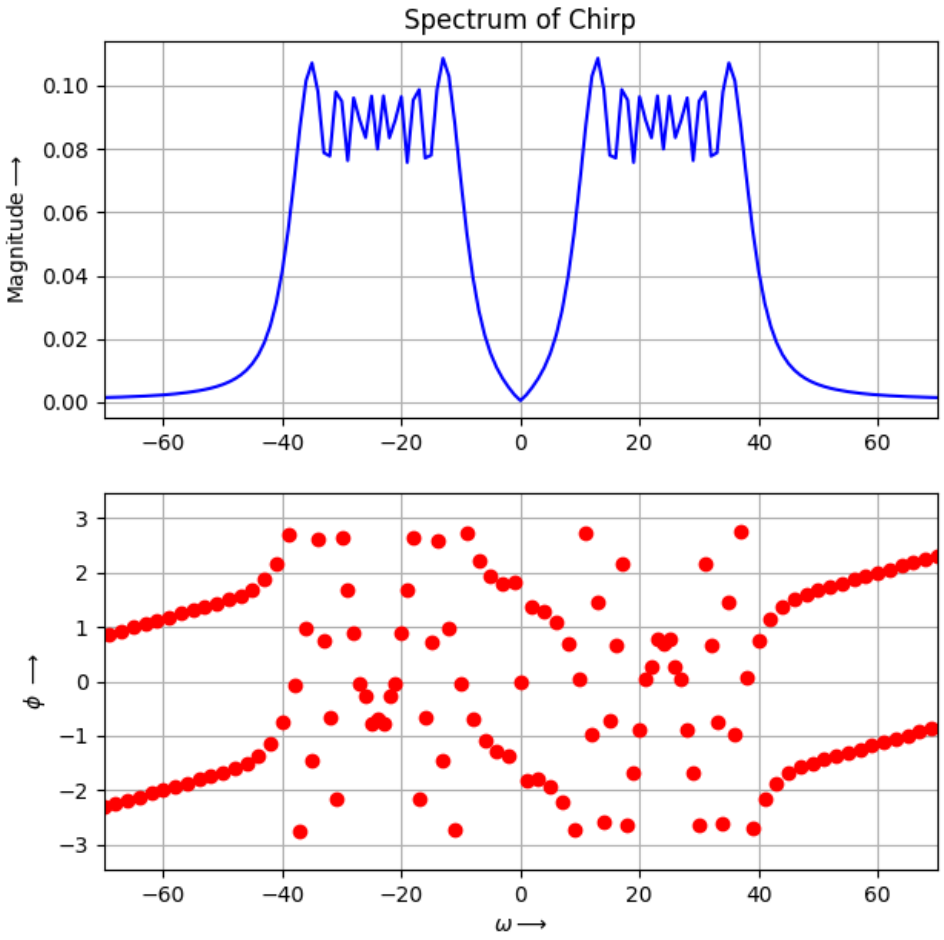

Figure 14: Spectrum of chirped signal

We can see that the spectrum has many peaks with intermediary falls, due to variation of frequency of signal.
If we multiply the chirp signal with hamming window, the variations are smoothed out and we get only two broad peaks that are approximately confined to frequency range of 16-32 rad/s.

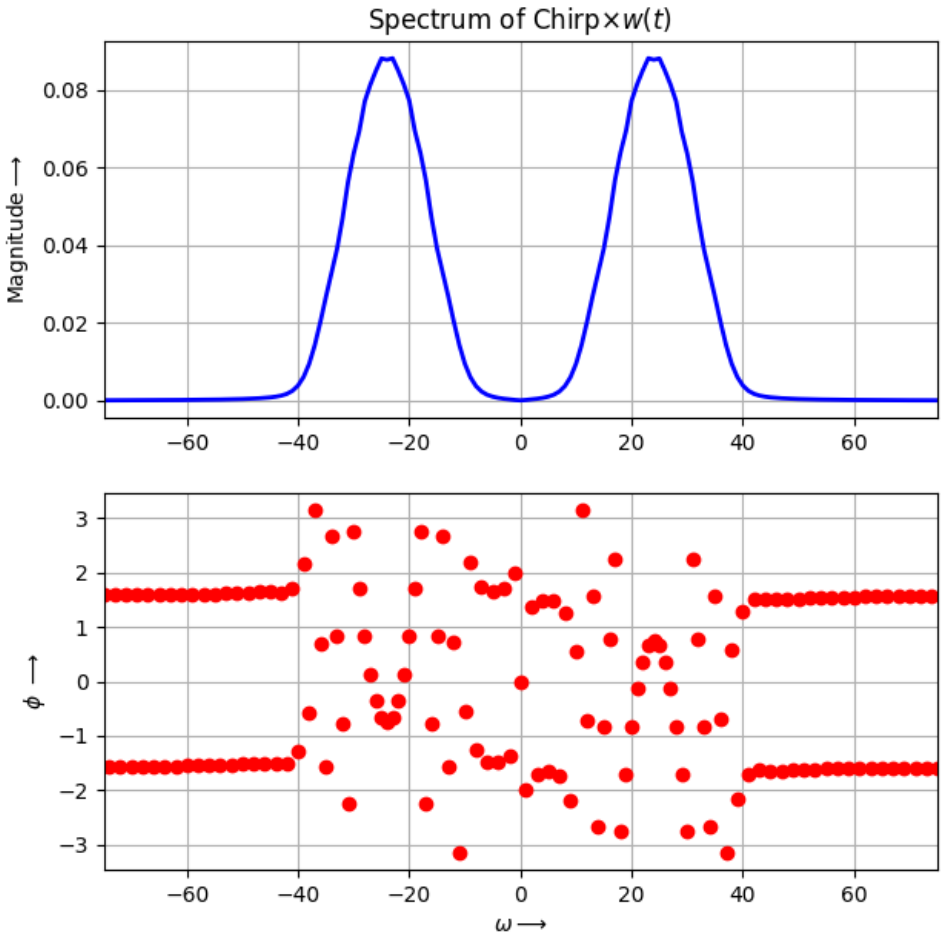

Figure 15: Spectrum of chirped signal with hamming window

7 Time-Frequency plot of Chirp signal

We split the chirp signal in the time interval $[-\pi, \pi]$ into smaller sections and also record how frequency of signal varies with time. Initially we take 1024-size vector of chirp signal sequence. We split it into 64-size vectors, compute the DFT of each section and plot a time-frequency surface plot to observe the variation of frequency with time. Below is the plot for chirp signal with no windowing.

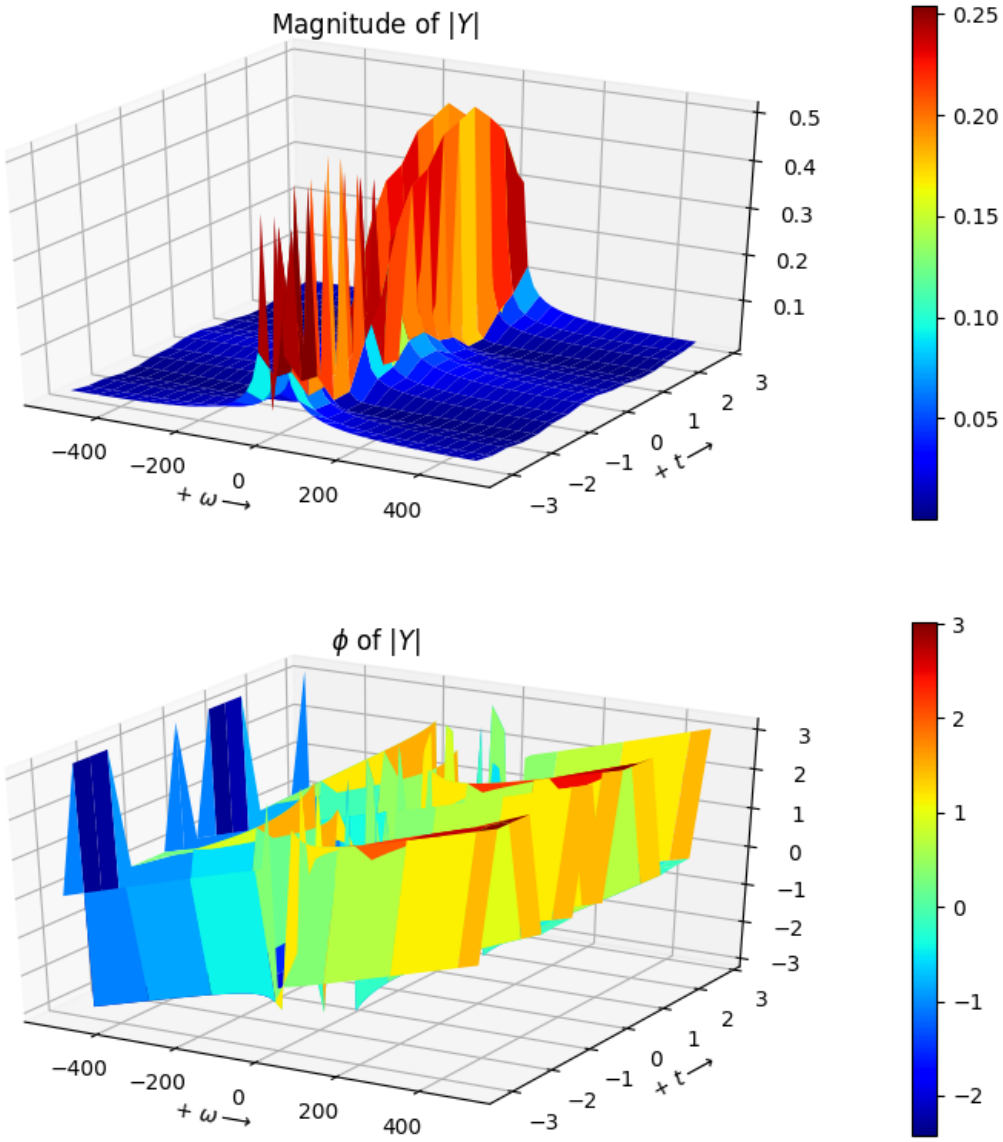

Figure 16: Time-Frequency plot of Chirp signal with No Windowing

Below is a rotated view of the same plot

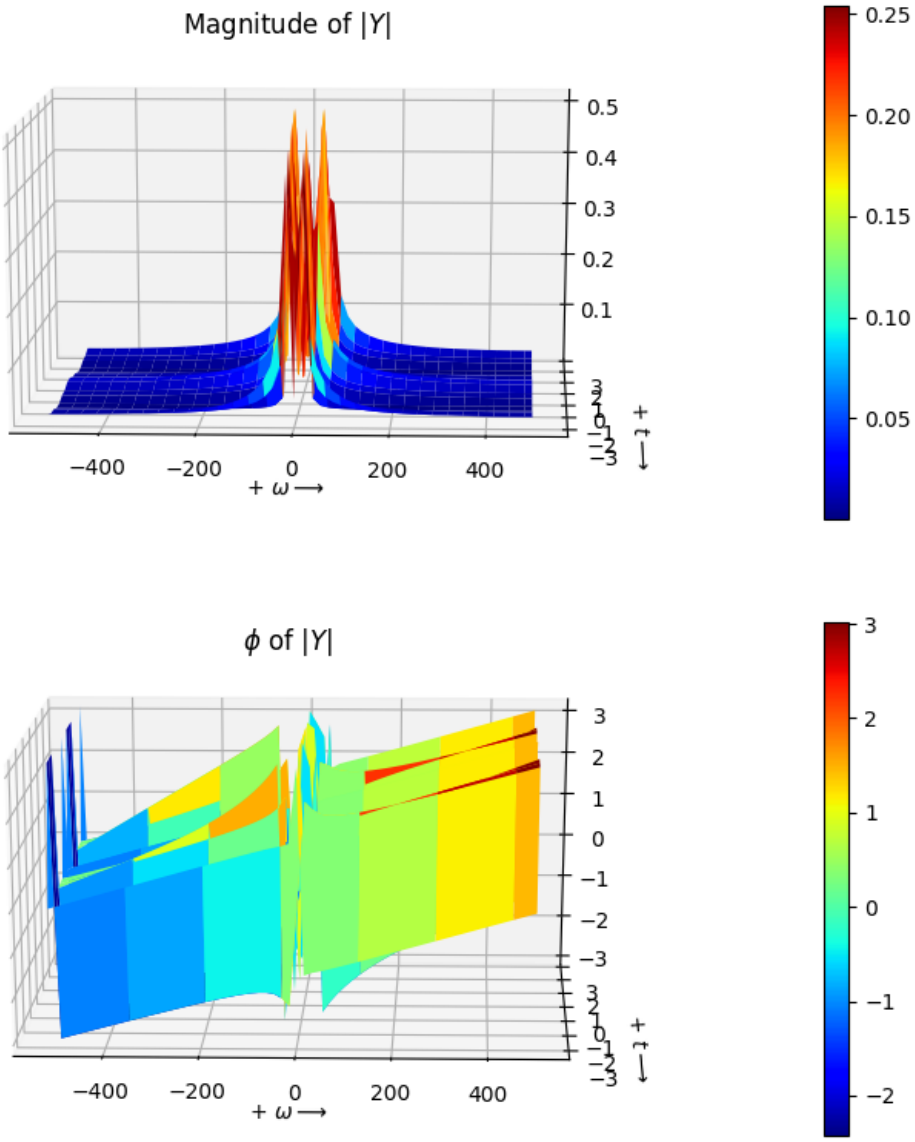

Figure 17: Time-Frequency plot of Chirp signal with No Windowing

Let's also look at the time-frequency surface plots with windowed version of Chirp.

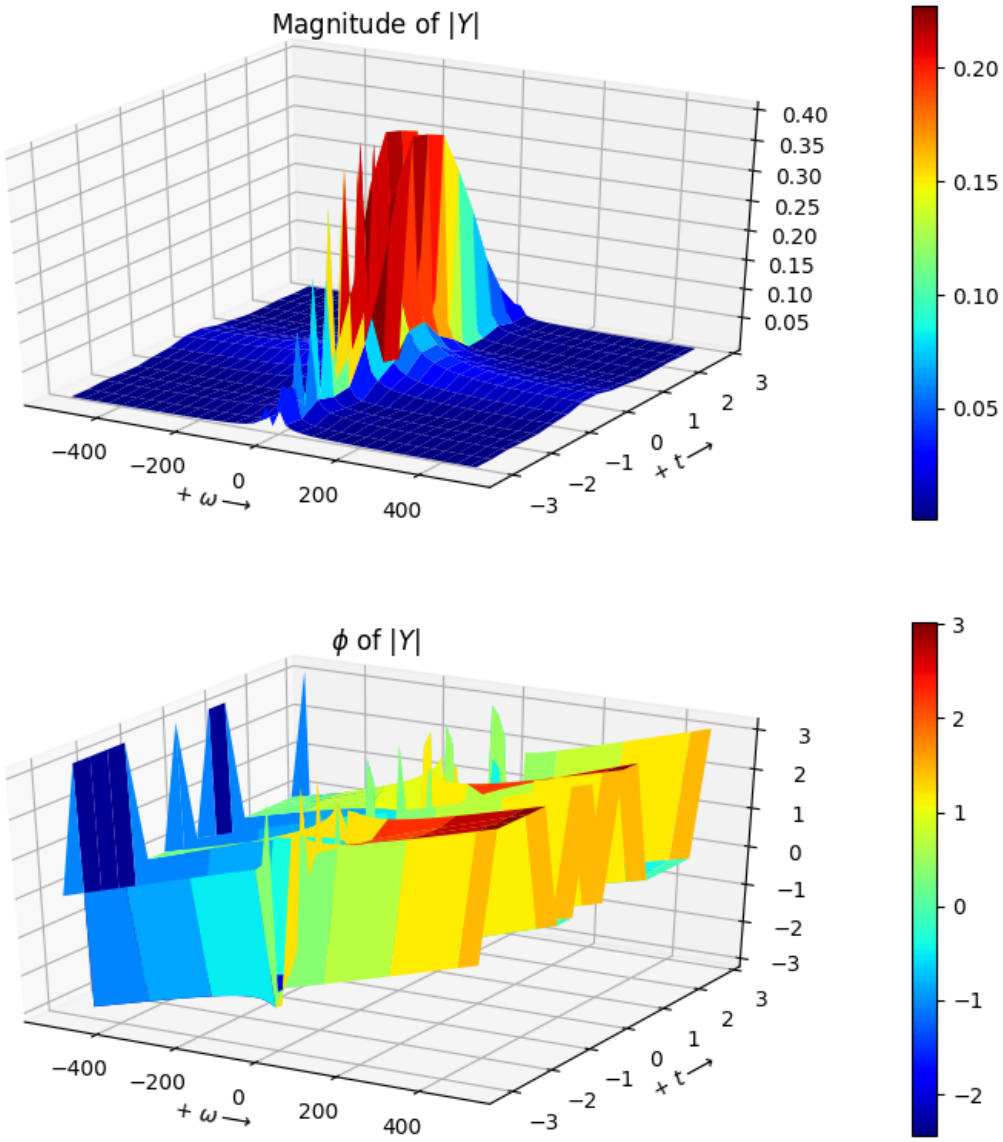

Figure 18: Time-Frequency plot of Chirp signal with Windowing

Below is a rotated view of the same plot

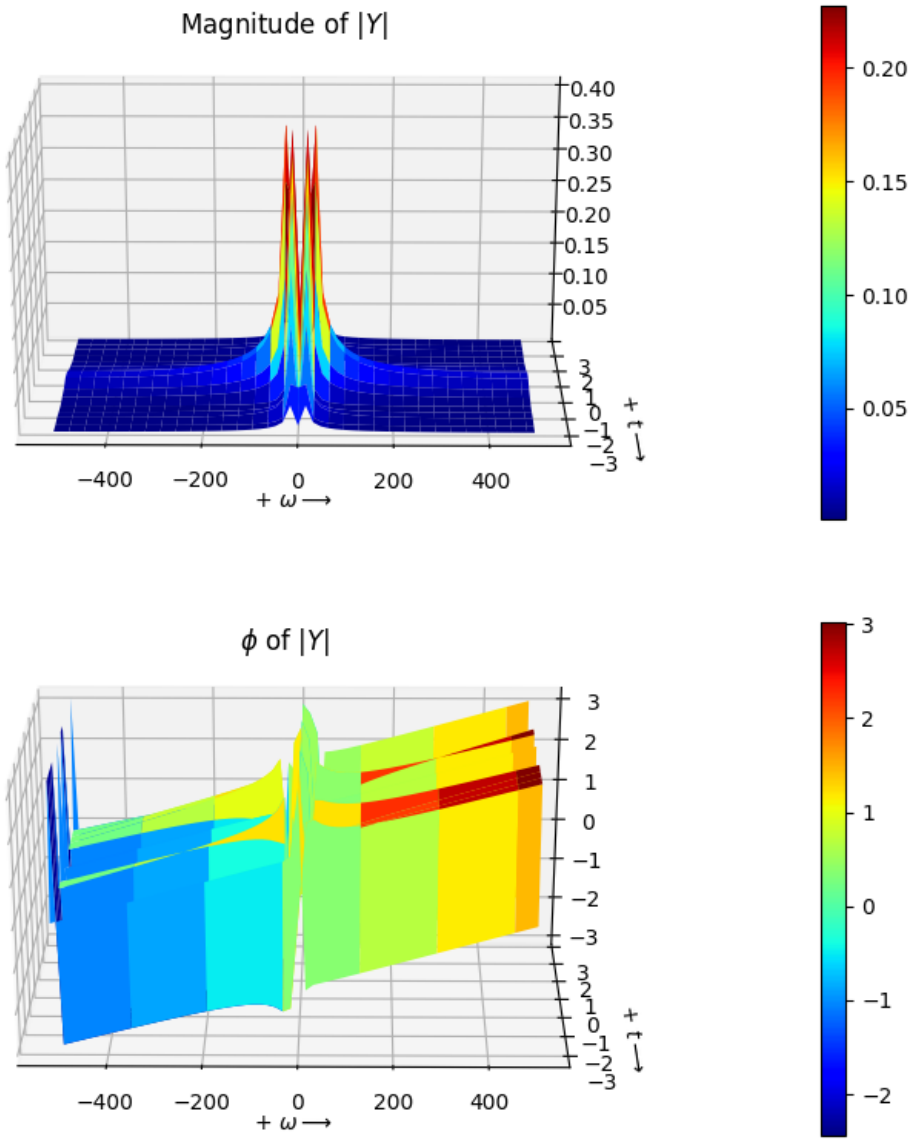

Figure 19: Time-Frequency plot of Chirp signal with Windowing

We can observe that the change in magnitude in windowed version is more smooth and steep along ω-axis compared to it's counterpart non windowed version.

Conclusion

In the default way of finding DFT, we actual find the DFT for the 2π periodic extension of original signal. This results in jump discontinuities and Gibbs's phenomenon. To rectify this error in spectra, we use a hamming window signal. We also estimated the parameters of signal from given data and they turned out to be fairly good methods. At last, we analysed the spectrum of chirp signal and it's time-frequency plots