EE2703 Assignment 9 : Spectra of non-periodic signals

Aman Kumar EE19B066

May 29, 2021

1 Introduction

In this assignment we will:

1. Learn how to obtain DFT of non-periodic functions
2. See the advantages of using a windowing function in cases where functions that the DFT is trying to analyse has discontinuities.
3. Learn about and use the famous **Hamming window** given by:

$$w[n] = \begin{cases} 0.54 + 0.46cos(\frac{2\pi n}{N-1}) & |n| \leq \frac{N-1}{2} \\ 0 & else \end{cases} \tag{1}$$

4. Extract the spectrum from a given vector of 128 elements known to contain $cos(\omega_0 t + \delta)$ and also estimate the values of ω_0 and δ from the spectrum with and without noise.

2 The Assignment

2.1 The `Spectrum()` function

I have made this function to plot the spectrums of different functions it is such that the plots are highly customizable. The phase in the plots is in degrees. Its arguments are:

1. **fmax** : 1/dt = 1/(t[1] - t[0])
2. **y** : function
3. **Title** : title of the plot
4. **fig_no** : number of the figure
5. **Xlim** : x-axis range in the plot
6. **wind** : True if windowing is done, False if not done. (Default = False)
7. **Xticks** : ticks along x-axis. (Default = None)
8. **Yticks** : ticks along y-axis. (Default = None)

It returns ω, the frequency axis of the spectrum and Y the transform. Here is the function:

```
def Spectrum(fmax,y,Title,fig_no,Xlim,wind=False,Xticks=None,Yticks=None):

    N = len(y)                          #Number of samples
    if wind:                            #Checking if windowing is to be done
        n = p.arange(N)
        wnd = p.fftshift(0.54 + 0.46*p.cos(2*pi*n/(N-1)))  #The Hamming window
        y = y*wnd                #Multiplying the function to window in time domain
    y[0] = 0                #the sample corresponding to -tmax should be set zero
    y = p.fftshift(y)
    Y = p.fftshift(p.fft(y))/N                    #Finding the Transform
    w = p.linspace(-pi*fmax,pi*fmax,N+1)[:-1]     #Frequency vector

    p.figure(fig_no)
    p.subplot(2,1,1)                              #Magnitude spectrum
    p.title(Title)
    p.plot(w,abs(Y),'bo',linestyle = "dashed",lw = 1,markersize = 3)
    p.xlim(Xlim)
    p.yticks(Yticks)
    p.xticks(Xticks)
    p.ylabel(r"|$Y$|")
    p.grid(True)

    p.subplot(2,1,2)                                  #Phase spectrum
    p.plot(w,p.angle(Y)*180/pi,'ro',markersize = 3) #Plotting phase in degrees
    p.yticks(p.arange(-180,181,90))                   #Plotting phase in degrees
    p.xticks(Xticks)
    p.xlim(Xlim)
    p.ylabel(r"Phase of $Y$")
    p.xlabel(r"$\omega$")
    p.grid(True)
    p.savefig("Figure "+str(fig_no)+".png")  #Saving the figure
    p.show()

    return(w,Y)
```

2.2 Spectrum of $sin(\sqrt{2}t)$

2.2.1 Without Hamming window

Without the Hamming window, the discontinuities at $2n\pi$ won't be suppressed. So actually, the DFT will be calculated of some different signal as it will wrap after 2π. Here's the code for plotting:

```
t = p.linspace(-pi,pi,65)[:-1]                 #64 points from -pi to pi
dt = t[1] - t[0];fmax = 1/dt                   #getting fmax
y = p.sin(p.sqrt(2)*t)                         #The given function
Title = r"Spectrum of $sin(\sqrt{2}t)$"
w,Y = Spectrum(fmax,y,Title,0,[-10,10])        #Calling the function Spectrum to
                                               #plot its spectrum
```

We observe that:

1. We expected two spikes, but what we got were two peaks each with two values and a gradually decaying magnitude.
2. Since $\sqrt{2} = 1.4142$ is not in the resolution of ω therefore we get a peak that is two samples wide from 1 to 2.
3. The magnitude is gradually decreasing due the presence of large discontinuities in the 2π wrapped signal.
4. The phase is correct.

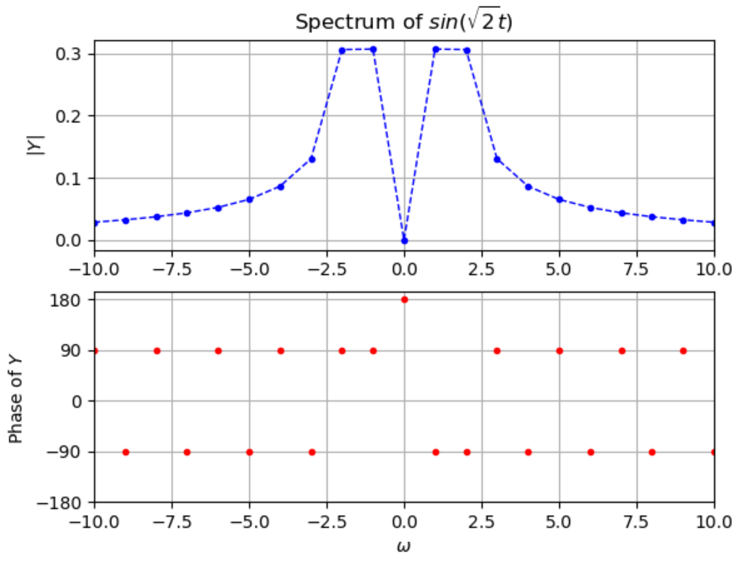

Figure 1: Spectrum of $sin(\sqrt{2}t)$ without windowing

5. What went wrong is due to the Gibbs phenomenon. Thus we have a need of windowing the original signal to suppress the discontinuities.

2.2.2 With Hamming window

Now we plot the spectrum of the windowed signal, using the Hamming window which is defined in Equation 1. Python code snippet for plotting:

```
Title = r"Spectrum of $sin(\sqrt{2}t)w(t)$"
w,Y = Spectrum(fmax,y,Title,1,[-8,8],True,Yticks=p.arange(0,0.26,0.05))
```

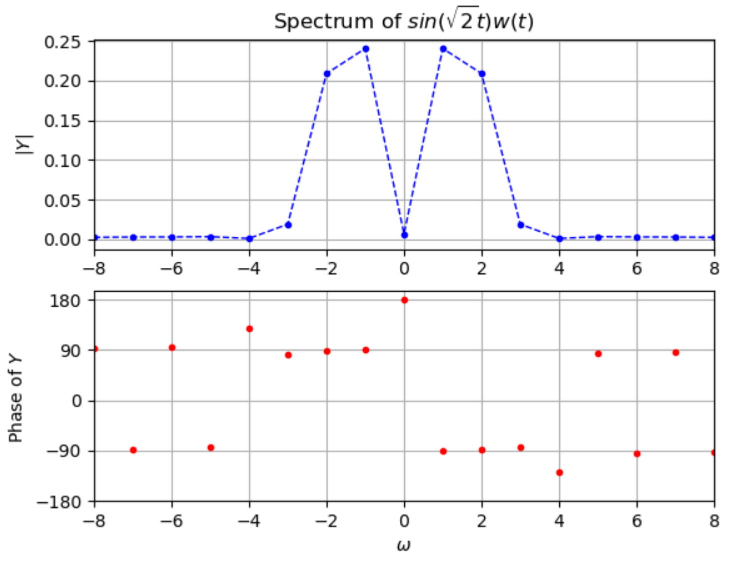

Figure 2: Spectrum of $sin(\sqrt{2}t)w(t)$ with 64 points

We notice that:

1. The magnitude spectrum has greatly improved.

2. We still have a peak that is 2 samples wide, but that is because $\sqrt{2}$ lies between 1 and 2, which are the two fourier components available.

3. If we use four times the number of points i.e. 256, we should get better results.

Using 256 points in $t = [-4\pi, 4\pi)$, thus the fmax remains unchanged. That means the limits of frequency axis remains unchanged. But the number of points in the frequency axis has become four times. Thus resolution increases. Python code snippet:

```
t = p.linspace(-4*pi,4*pi,257)[:-1]
dt = t[1] - t[0];fmax = 1/dt
y = p.sin(p.sqrt(2)*t)
Title = r"Spectrum of $sin(\sqrt{2}t)w(t)$ with four times the number of points"
w,Y = Spectrum(fmax,y,Title,2,[-4,4],True,Yticks=p.arange(0,0.26,0.05))
```

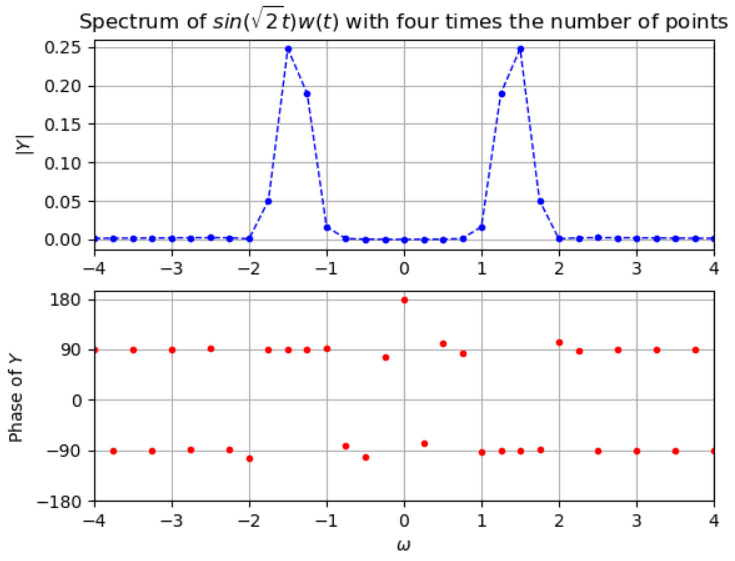

Figure 3: Spectrum of $sin(\sqrt{2}t)w(t)$ with 256 points

It is quite a bit better since we are now zoomed in and see a lot more detail. But again the peak does not look like a delta function and has some width. The reason for that is $w(t)$. Multiplication in time is convolution in frequency and vice versa. So by multiplying with $w(t)$, we got rid of the $1/f$ decay. But the delta function is now replaced by the shape of the DFT of $w[n]$. That gives us a factor of two broadening over the peak when there is no window, which is why we still see a peak whose width is two samples.

2.3 Spectrum of $cos^3(\omega_0 t)$

2.3.1 Without Hamming window

First we plot the spectrum without windowing the function with Hamming window. I am using 256 points on the frequency axis t0 have good resolution. Here $\omega_0 = 0.86$. The python code snippet:

```
y = p.cos(0.86*t)**3
Title = r"Spectrum of $cos^3(\omega_0 t)$"
w,Y = Spectrum(fmax,y,Title,3,[-5,5],Xticks=p.arange(-5,6,1),Yticks = p.arange(0,0.3,0.05))
```

We notice:

1. We have four peaks as expected but not exactly where they should have been due to less resoltuion.

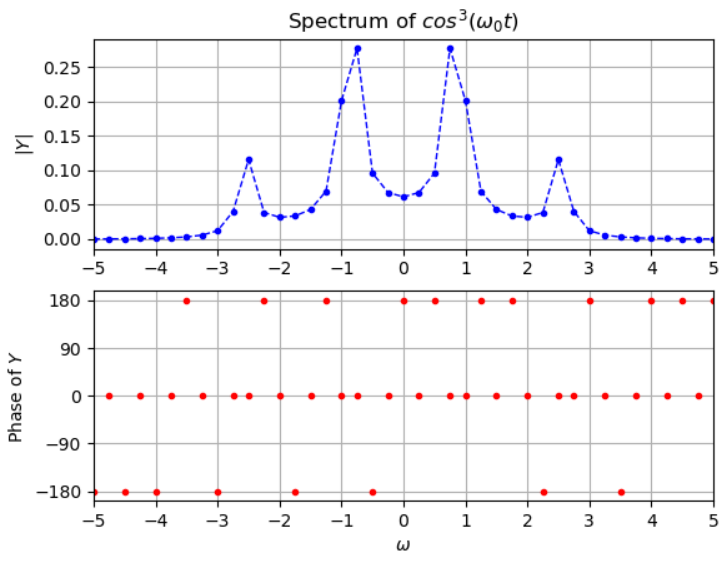

Figure 4: Spectrum of $cos^3(\omega_0 t)w(t)$

2. Magnitude is **not** going to zero for frequencies other than peak. This is becauuse we haven't windowed the function.
3. Phase is correct.

2.3.2 With Hamming window

Python code snippet for plotting the spectrum of windowed signal:

```
Title = r"Spectrum of $cos^3(\omega_0 t)w(t)$"
w,Y = Spectrum(fmax,y,Title,4,[-5,5],True,Xticks=p.arange(-5,6,1))
```

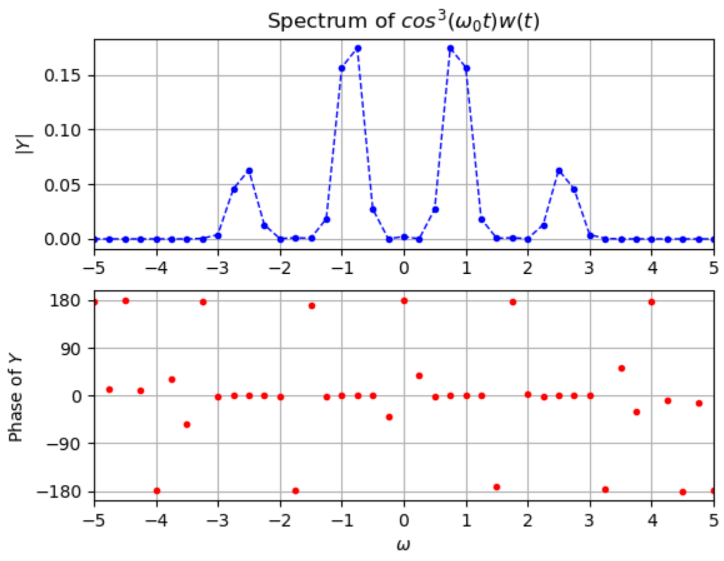

Figure 5: Spectrum of windowed $cos^3(\omega_0 t)$

Magnitude has improved. But still the peaks are not like delta functions but has some width due to reasons mentioned in the previous section.

2.4 Extracting the spectrum and estimating ω_0 and δ

We have a 128 element vector known to contain $cos(\omega_0 t + \delta)$ for arbitrary δ and $0.5 < \omega_0 < 1.5$. The values of t go from $-\pi$ to π. We have to extract the digital spectrum of the signal, find the two peaks at $\pm\omega_0$, and estimate ω_0 and δ.

1. 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, we have to obtain ω_0 by taking a weighted average of all the ω weighted with the magnitude of the DFT.

$$\omega_{cal} = \frac{\sum \omega_i |Y(\omega_i)|^2}{\sum |Y(\omega_i)|^2}; \forall \omega_i > 0 \tag{2}$$

2. δ can be found by calculating the phase of the discrete fourier transform at ω nearest to estimated ω_{cal}. 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.

I have written a function w0_delta() that takes in as arguments ω (the frequency axis of transform) and Y (transform), and estimates and prints the values of ω and δ.

```
def w0_delta(w,Y,prnt=""):
    ii = p.where(w > 0)                                        #Indexes of all w>0
    #Taking weighted average of all w>0 with magnitude of DFT as weights
    w_estimate = sum(w[ii]*abs(Y[ii])**2)/sum(abs(Y[ii])**2)
    #Finding index of nearest w to the estimated w. This is done for finding delta
    i = abs(w - w_estimate).argmin
    #delta(estimated) = angle of Y at the index found above.
    delta_estimate = p.angle(Y[i])

    print("\nEstimated w0"+prnt+" = ",w_estimate)      #Printing the estimated w
    print("Estimated delta"+prnt+" = ",delta_estimate) #Printing the estimated
                                                       #delta
```

2.4.1 Without noise

Python code for extracting spectrum:

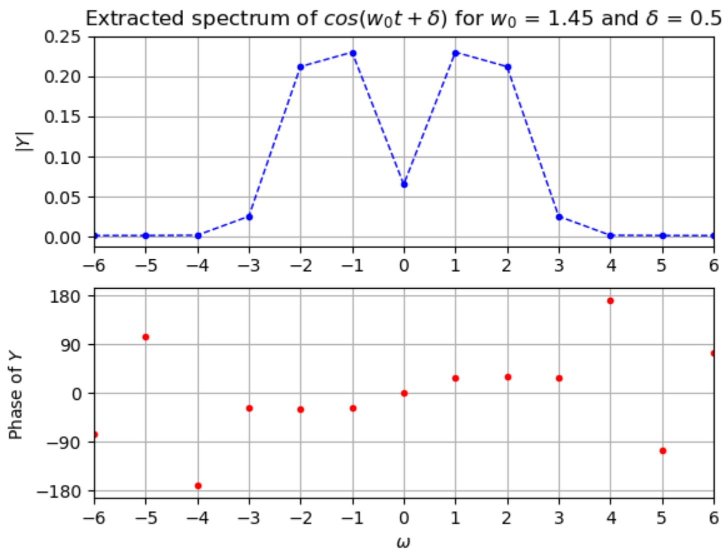

Figure 6: Extracted spectrum of $cos(w_0 t + \delta)$ without noise.

```
w0,delta = 1.45,0.5                           #Arbitrary values for w0 and delta
t = p.linspace(-pi,pi,129)[:-1]               #Creating 128 element time axis
dt = t[1] - t[0];fmax = 1/dt
y = p.cos(w0*t + delta)                       #Creating the 128 element vector
Title = r"Extracted spectrum of $cos(w_0 t + \delta)$ for $w_0$ = " + str(w0) + " and
        $\delta$  = " + str(delta)
w,Y=Spectrum(fmax,y,Title,5,[-6,6],True,Xticks=p.arange(-6,7,1),Yticks=p.arange(0,0.26,0.05))
```

With this we will get ω and Y which are needed as arguments in the `w0_delta()` function to estimate ω_0 and δ. Then to get the estimation of ω_0 and δ we do `w0_delta(w,Y," with noise")` The estimated values are (in noiseless case):

```
    Estimated w0 =  1.472374597597627
    Estimated delta =  0.49406546876219665
```

These values are very close to the actual values: $\omega_0 = 1.45$ and $\delta = 0.5$ since there is no noise in the data.

2.4.2 With noise

Python code snippet for extracting the spectrum:

```
y = p.cos(w0*t + delta) + 0.1*p.randn(128)           #Adding noise to the previous
                                                     #vector
Title = r"Extracted spectrum of noisy $cos(w_0 t + \delta)$ for $w_0$ = " + str(w0) + " and
        $\delta$ = " + str(delta)
w,Y=Spectrum(fmax,y,Title,6,[-6,6],True,Xticks=p.arange(-6,7,1),Yticks=p.arange(0,0.26,0.05))
```

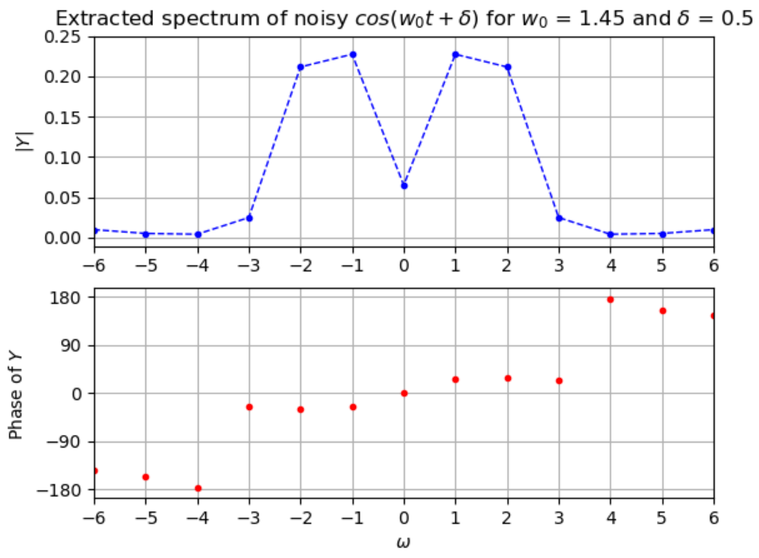

Figure 7: Extracted spectrum of $cos(w_0 t + \delta)$ with noise.

With this we will get ω and Y which are needed as arguments in the `w0_delta()` function to estimate ω_0 and δ. Then to get the estimation of ω_0 and δ we do `w0_delta(w,Y," with noise")`

```
    Estimated w0 with noise =  2.226896537304723
    Estimated delta with noise =  0.5199696772341339
```

These values are not quite close to the actual values as noise is added to the data. The more there is noise, the more difficult it is to extract the peak frequency and phase.

2.5 Chirped Signal : $cos(16(1.5 + \frac{t}{2\pi})t)$

Python code for plotting the spectrum (using Hamming window). It's given that $t = [-\pi, \pi)$ in 1024 steps.

```
t = p.linspace(-pi,pi,1025)[:-1]         #time axis : 1024 points between [-pi,pi)
dt = t[1] - t[0];fmax = 1/dt
y = p.cos(16*(1.5 + t/(2*pi))*t)
Title = r"Spectrum of $cos(16(1.5 + \frac{t}{2\pi})t)w(t)$"
w,Y = Spectrum(fmax,y,Title,7,[-50,50],True,Xticks = p.arange(-50,51,10))
```

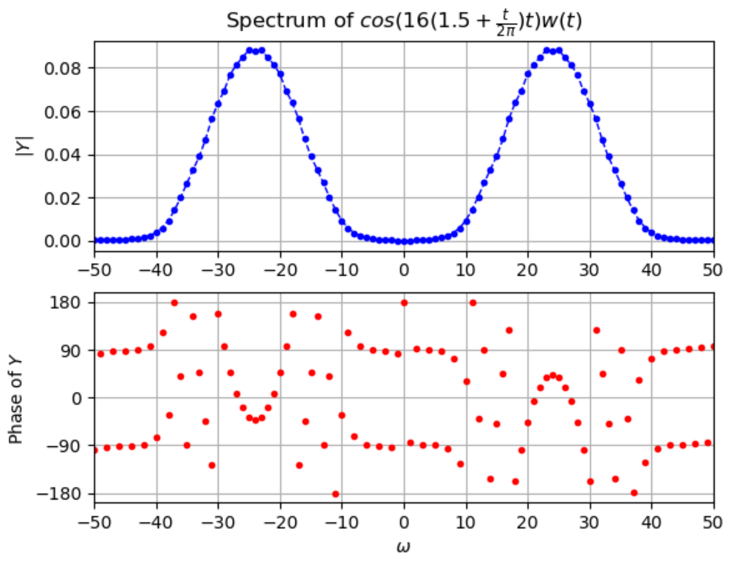

Figure 8: Spectrum of $cos(16(1.5 + \frac{t}{2\pi})t)w(t)$

Its frequency continuously changes from 16 to 32 rad/sec. This also means that the period is 64 samples near $-\pi$ and is 32 samples near $+\pi$. In the plot $w(t)$ is the windowing function.

Surface Plot

The following code is for making the surface plot to show how the frequency of the signal varies with time.

```
t_array = p.array(p.array_split(t,16))          #Dividing the t array into 16
                                                #sections of 64 elements each
y_array = p.cos(16*(1.5 + t_array/(2*pi))*t_array)   #Calculating and storing the
                                                     #function values at those
                                                     #values of t


n = p.arange(64)
wnd = p.fftshift(0.54 + 0.46*p.cos(2*pi*n/63))
y_array = y_array*wnd                           #Multiplying all 16 sections of y
                                                #with window function
y_array[:,0] = 0        #the sample corresponding to -tmax should be set zero in
                        #all 16 sections
Y_array = p.fftshift(p.fft(y_array))/64         #Calculating the transform

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

```
t,w = p.meshgrid(t,w)

#Plotting the surface plot
fig8 = p.figure(8)
ax = fig8.add_subplot(111,projection = '3d')
surf=ax.plot_surface(w,t,abs(Y_array[::-1]).T,cmap='viridis',linewidth=0, antialiased=False)
fig8.colorbar(surf,shrink = 0.5,aspect = 5)
ax.set_title(r"Surface Plot")
p.ylabel(r"$\omega \rightarrow$")
p.xlabel(r"$t \rightarrow$")
p.savefig("Figure 8.png")  #Saving the figure
p.show()
```

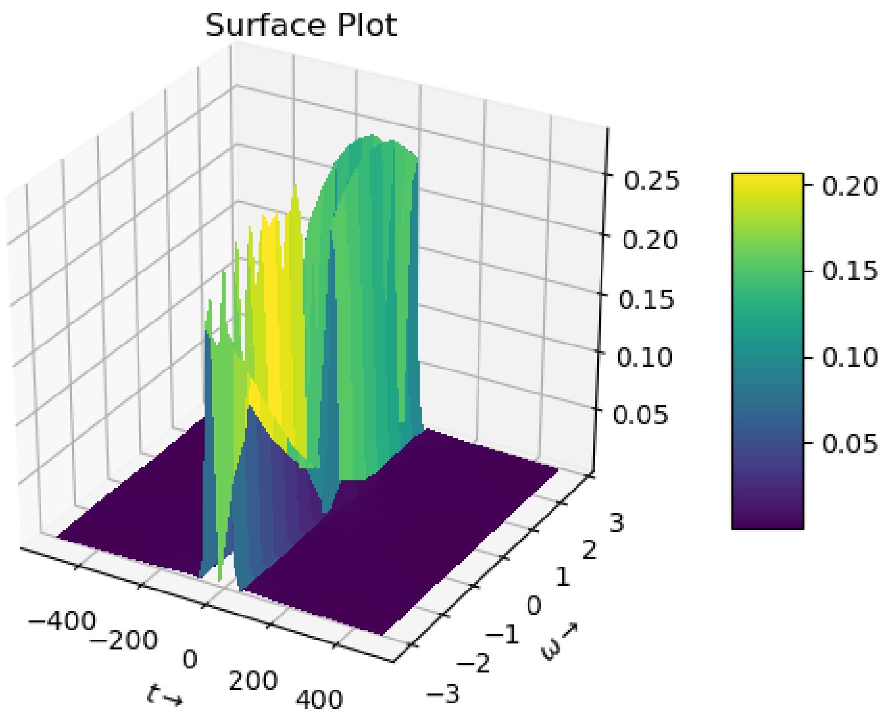

Figure 9: Surface plot of magnitude

3 Conclusion

In this assignment:

1. We learnt how to get DFT and plot spectrum of non-periodic functions using windowing.
2. We learnt about the Hamming window.
3. We learnt how to estimate the peak frequency and phase from a vector that is known to contain a function value.
4. We also made a 3D plot of magnitude of broken Chirped signal