Assgn10_EE17B114

April 10, 2019

Spectra of non-periodic signals

In this assignment, the fourier transform of some functions, with and without a windowing function is found out and their graphs plotted. Without windowing, for some functions, whose peak is not at any point in the discrete time interval taken, the peak appears at, and not entirely accurate, whereas with windowing, for those function, the peaks become visible, though they appear broad as a multiplication in time domain, done by the Windowing function, results in a convolution in the frequency domain, which results in broader peaks.

Libraries Used

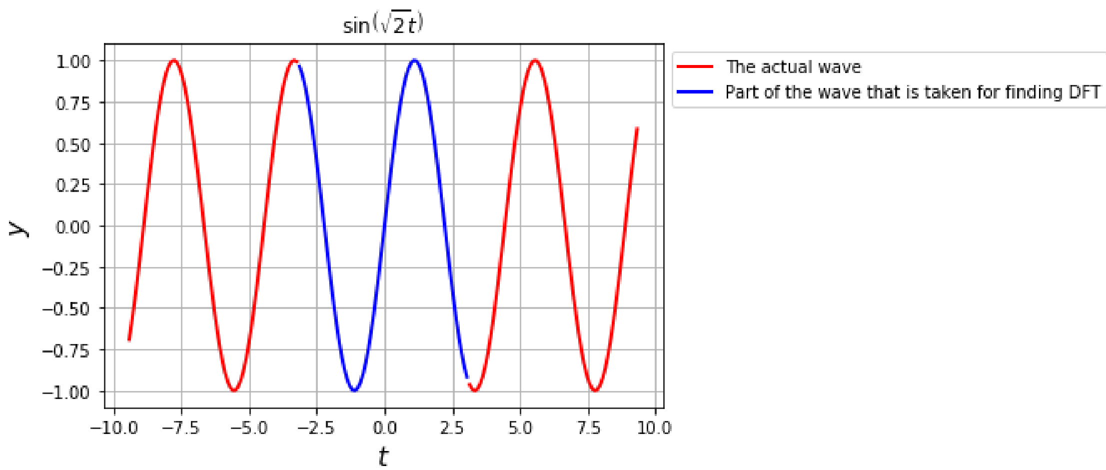
```
In [1]: import numpy as np
        import numpy.fft as f
        import matplotlib.pyplot as plt
        import math
        import mpl_toolkits.mplot3d.axes3d as p3
        pi = math.pi
        np.random.seed(5)
```

```
In [2]: def plot_function(w,Y,x_lim, function):
            fig, axes = plt.subplots(2, 1, figsize=(15, 7), sharex = True)

            plt.suptitle("The DFT plots for " + function, fontsize=18)
            # The magnitude plot is plotted
            axes[0].plot(w,abs(Y),'b',w,abs(Y),'bo',lw=2)
            axes[0].set_xlim([-x_lim,x_lim])
            axes[0].set_ylabel(r"$|Y|$",size=16)
            axes[0].set_title("Spectrum of " + function, fontsize=14)
            axes[0].grid(True)

            # The Phase plot is plotted
            ii=np.where(abs(Y)>1e-3)
            axes[1].plot(w[ii],np.angle(Y[ii]),'ro',lw=2)
            axes[1].set_xlim([-x_lim,x_lim])
            axes[1].set_ylim([-4,4])
            axes[1].set_ylabel(r"Phase of $Y$",size=16)
            axes[1].set_title("Phase Plot of " + function, fontsize=14)
            axes[1].set_xlabel(r"$\omega$",size=16)
            axes[1].grid(True)
            plt.show()
```

Example plots are plotted

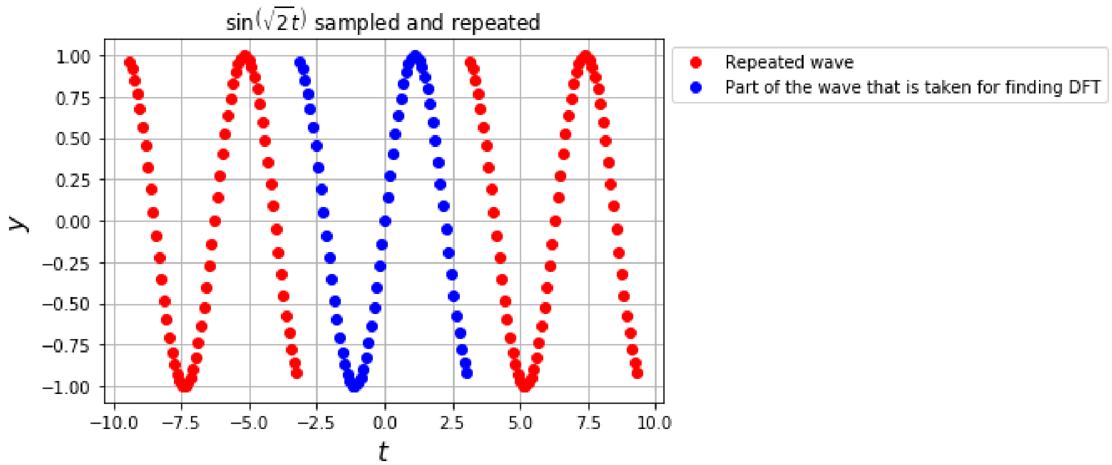
The example plots given in the question (\$ sin($\sqrt{}(2)x)\$)isplottedwithHammingwindowandwithoutHammingw$
, and hence if we plot the points that are sampled by the program, repeated periodically (whose $\sqrt{}$ 2x) , as shown below. fourier transform is actually found) the function is not the same as the continuous time.

```
In [3]: t1=np.linspace(-pi,pi,65);t1=t1[:-1]    # The interval used to find the DFT.
        t2=np.linspace(-3*pi,-pi,65);t2=t2[:-1]
        t3=np.linspace(pi,3*pi,65);t3=t3[:-1]

        plt.plot(t2,np.sin(np.sqrt(2)*t2),'r',lw=2)
        plt.plot(t1,np.sin(np.sqrt(2)*t1),'b',lw=2)    # The interval t1 is plotted in a differen
        plt.plot(t3,np.sin(np.sqrt(2)*t3),'r',lw=2)

        plt.legend(('The actual wave','Part of the wave that is taken for finding DFT'),bbox_to_
        plt.ylabel(r"$y$",size=16)
        plt.xlabel(r"$t$",size=16)
        plt.title(r"$\sin\left(\sqrt{2}t\right)$")
        plt.grid(True)
        plt.show()
```




```
In [4]: t1=np.linspace(-pi,pi,65);t1=t1[:-1]     # The interval used to find the DFT.
        t2=np.linspace(-3*pi,-pi,65);t2=t2[:-1]
        t3=np.linspace(pi,3*pi,65);t3=t3[:-1]

        y = np.sin(np.sqrt(2)*t1)

        # The function on interval t1 is repeated periodically.
        plt.plot(t2,y,'ro',lw=2)
        plt.plot(t1,y,'bo',lw=2)
        plt.plot(t3,y,'ro',lw=2)
```

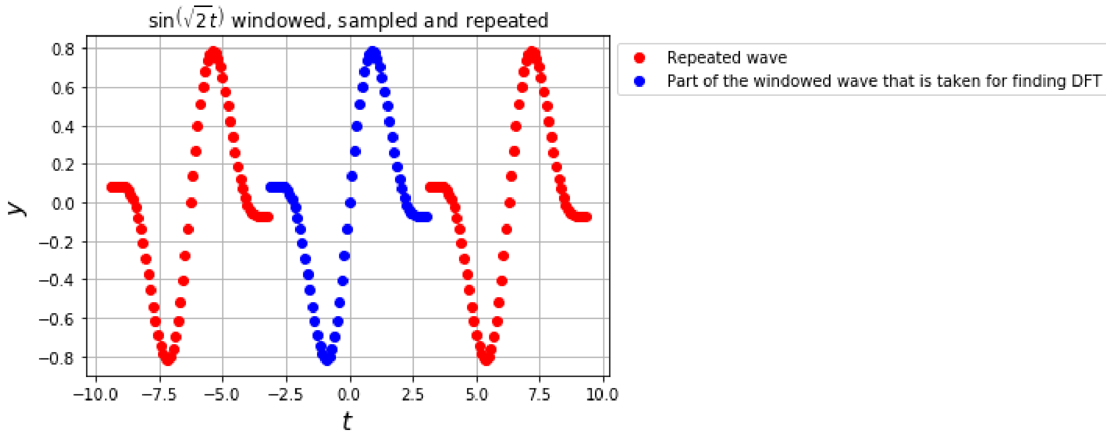
```
plt.legend(('Repeated wave','Part of the wave that is taken for finding DFT'),bbox_to_an
plt.ylabel(r"$y$",size=16)
plt.xlabel(r"$t$",size=16)
plt.title(r"$\sin\left(\sqrt{2}t\right)$ sampled and repeated")
plt.grid(True)
plt.show()
```



Thus, it can be seen that, in the actual plot $\sqrt{}$ that is used to and DFT, there is a huge discontinuity between successive periods, which is mainly due to the fact that 2 is an irrational number and therefore can't be plotted correctly in a discrete time waveform. It is this discontinuity that poses problems when calculating DFT, as Gibb's phenomenon occurs here. The main aim of windowing is to reduce this discontinuity without aecting the rest of the graphs very much, and as seen from the DFT plots below, it does work to a large extent. Before that, just to show, the same graph above, after windowing, the plot in time domain is as follows:

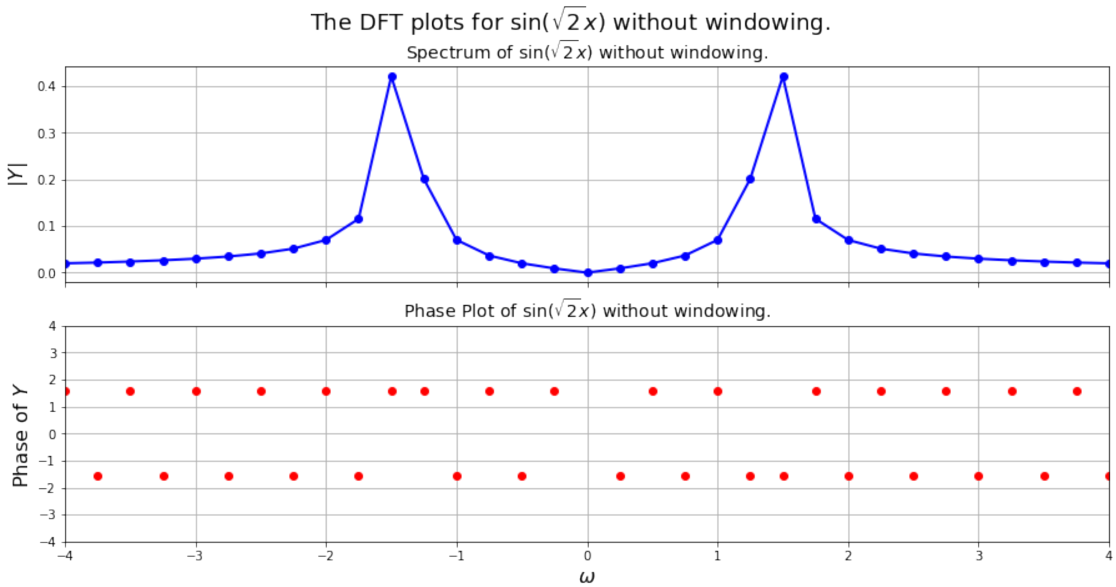
```
In [5]: t1=np.linspace(-pi,pi,65);t1=t1[:-1]    # The interval used to find the DFT.
        t2=np.linspace(-3*pi,-pi,65);t2=t2[:-1]
        t3=np.linspace(pi,3*pi,65);t3=t3[:-1]

        n=np.arange(64)
        wnd=f.fftshift(0.54+0.46*np.cos(2*pi*n/63))
        y=np.sin(np.sqrt(2)*t1)*wnd

        # The function on interval t1 is windowed and repeated periodically.
        plt.plot(t2,y,'ro',lw=2)
        plt.plot(t1,y,'bo',lw=2)
        plt.plot(t3,y,'ro',lw=2)
        plt.legend(('Repeated wave','Part of the windowed wave that is taken for finding DFT'),b
        plt.ylabel(r"$y$",size=16)
        plt.xlabel(r"$t$",size=16)
        plt.title(r"$\sin\left(\sqrt{2}t\right)$ windowed, sampled and repeated")
```

```
plt.grid(True)
plt.show()
```



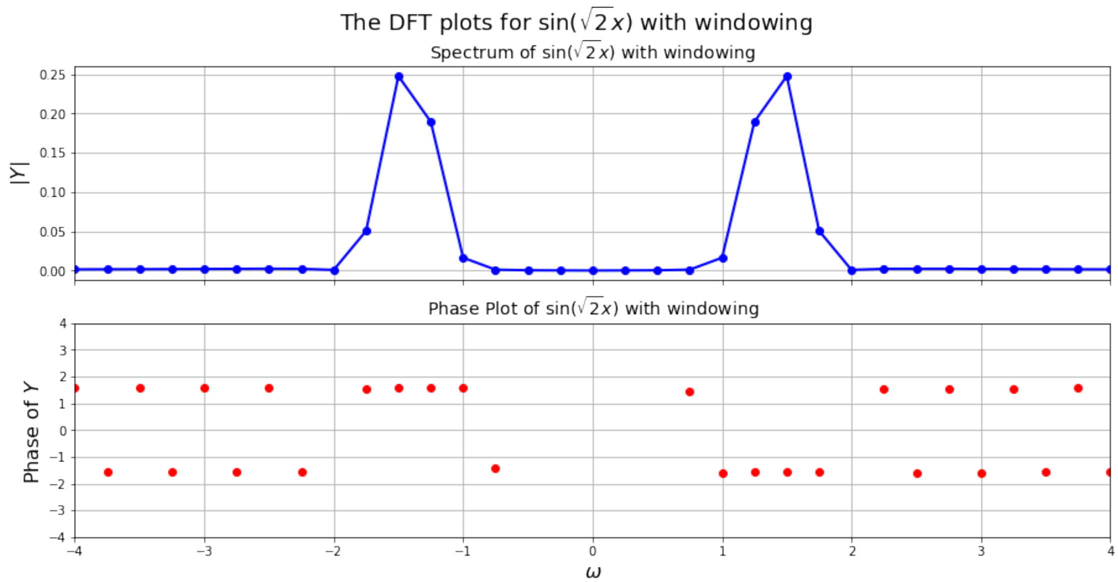
Thus, it is seen that, after windowing, the overall shape of the graph is preserved with a much reduced discontinuity. Now, the DFT of the function is calculated with and without windowing and observations are noted

```
In [6]: t = np.linspace(-4*pi,4*pi,257)    # Time interval declared
        t = t[:-1]
        dt = t[1]-t[0]
        fmax = 1/dt
        y = np.sin(np.sqrt(2)*t)           # Function defined
        y[0]=0
        y=f.fftshift(y)
        Y=f.fftshift(f.fft(y))/256.0
        w=np.linspace(-pi*fmax,pi*fmax,257);w=w[:-1]
        plot_function(w,Y,4,'$\sin(\sqrt{2}x)$ without windowing.')
```



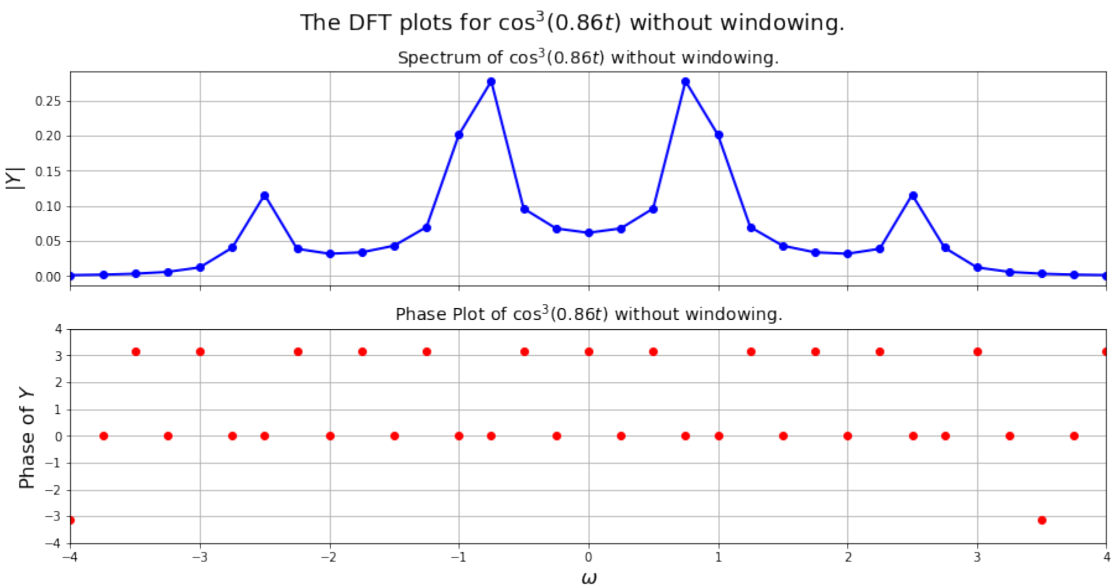
Thus, from the plots above, it can be seen that, without windowing, due to the high discontinuity, though the peaks exist, it doesn't reduce to 0 fast, and a phase exist for the wave even in between the peaks where its supposed to be 0. From the latter plot, with windowing, it is observed that, although the peak is broader (which is expected as multiplication in time domain by the windowing function is equivalent to convolution in frequency domain, which results in a broader peak), the magnitude does go to 0 fast, and there is no phase in between the peaks. Also, the obtained peak is in between 1 and 2, which is expected as ideally the peak is at \$ $\sqrt(2)$\$ ≈ 1.414.

```
In [7]: t = np.linspace(-4*pi,4*pi,257)
        t = t[:-1]
        dt = t[1]-t[0]
        fmax = 1/dt
        n = np.arange(256)
        wnd=f.fftshift(0.54+0.46*np.cos(2*pi*n/255))     # Hamming window declared.
        y = np.sin(np.sqrt(2)*t)
        y = y * wnd              # Hamming window multipled in the time domain
        y[0]=0
        y=f.fftshift(y)
        Y=f.fftshift(f.fft(y))/256.0
        w=np.linspace(-pi*fmax,pi*fmax,257);w=w[:-1]
        plot_function(w,Y,4,'$\sin(\sqrt{2}x)$ with windowing')
```

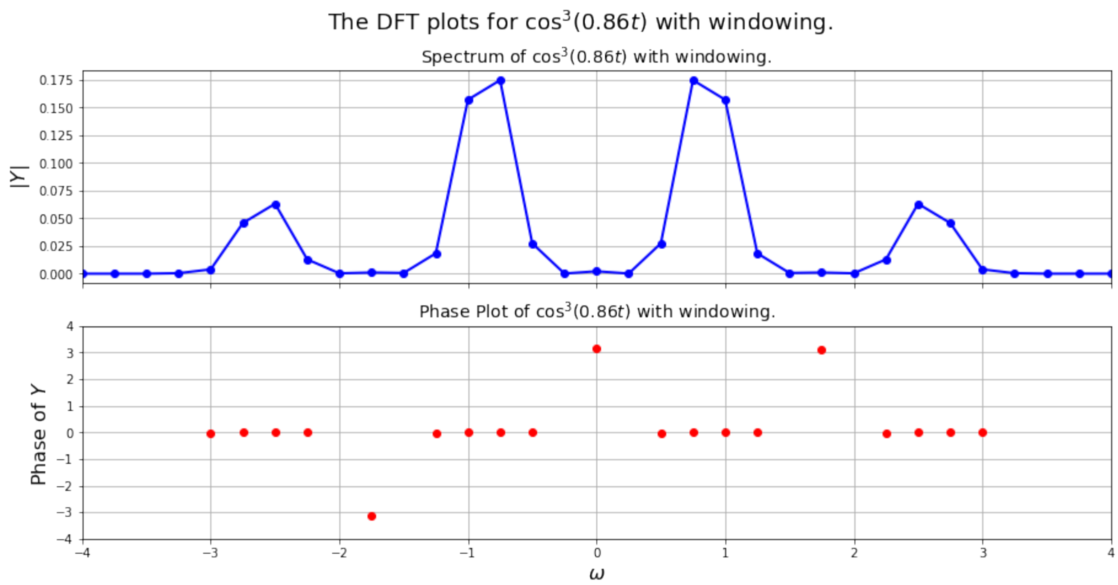


Just like the previous example, 0.86 is also not in the sampled points and hence, it will also have a discontinuity, when the interval for which DFT is taken is repeated periodically, which is seen in its DFT as a wide peak, which doesn't go to 0 anywhere in between. This is because of the Gibb's phenomenon at the discontinuity, because of which, a lot of frequency components would be present in the DFT. This is very much reduced when the function is windowed as seen below.

```
In [8]: t = np.linspace(-4*pi,4*pi,257)
        t = t[:-1]
        dt = t[1]-t[0]
        fmax = 1/dt
        y = np.cos(0.86*t) ** 3
        y[0]=0
        y=f.fftshift(y)
        Y=f.fftshift(f.fft(y))/256.0
        w=np.linspace(-pi*fmax,pi*fmax,257);w=w[:-1]
        plot_function(w,Y,4,'$\cos^3(0.86t)$ without windowing.')
```




```
In [9]: t = np.linspace(-4*pi,4*pi,257)
        t = t[:-1]
        dt = t[1]-t[0]
        fmax = 1/dt
        n = np.arange(256)
        wnd=f.fftshift(0.54+0.46*np.cos(2*pi*n/255))
        y = np.cos(0.86*t) ** 3
        y = y * wnd
        y[0]=0
        y=f.fftshift(y)
        Y=f.fftshift(f.fft(y))/256.0
        w=np.linspace(-pi*fmax,pi*fmax,257);w=w[:-1]
        plot_function(w,Y,4,'$\cos^3(0.86t)$ with windowing.')
```



To do this, first the DFT of the function (modelled as a 128 element vector) is plotted after windowing so as to get accurate peaks in the magnitude plot. Since, it is a cos function, the phase would be 0 normally. So δ would be the phase of the spectrum at its peaks and ω 0 would be approximately equal to the average of elements near the peak, as the peak is broadened due to windowing. Some examples are considered below:

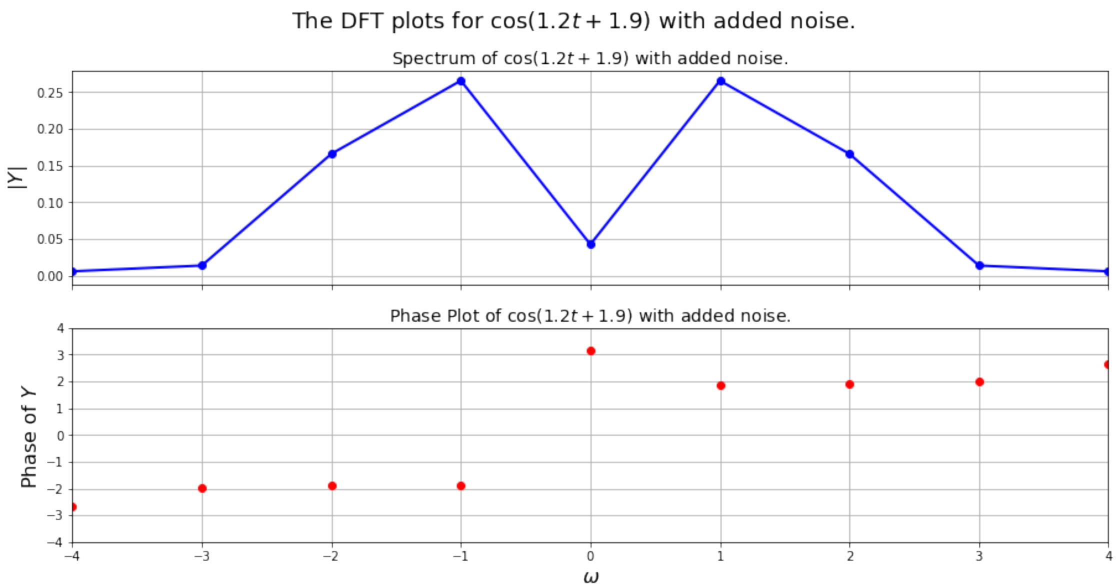
```
In [ ]: # A random delta and omega declared to find error
        delta = 1.9
        omega = 1.2
        t = np . linspace ( - pi , pi ,129)
        t = t [: -1]
        dt = t [1] - t [0]
        fmax = 1/ dt
        n = np . arange (128)
        wnd = f . fftshift (0.54+0.46* np . cos (2* pi * n /127) )
        y = np . cos ( omega * t + delta )
        y = y * wnd
        y [0]=0
        y = f . fftshift ( y )
        Y = f . fftshift ( f . fft ( y ) ) /128.0
        w = np . linspace ( - pi * fmax , pi * fmax ,129) ; w = w [: -1]
        plot_function (w ,Y ,4 , '$ \ cos (1.2 t + 1.9) $ without noise . ')

In [11]: ii = np.where(w>0)[0]     # All values of DFT above w=0 are taken (only one half of plot
         ii = np.where((abs(Y) == max(abs(Y[ii]))))    # The maximum in this region is taken as t
         est_delta = abs(np.angle(Y[ii])[0])    # The phase of the graph at the peak is the delta

         ii = np.where((abs(Y) > 3.5e-2) & (w >= 0))[0]     # The points greater than or equal to
         est_omega = abs(Y[ii]*w[ii])
```

```
        est_omega = sum(est_omega)/(sum(abs(Y[ii])))  # As peak is spread out, omega is estimat
        print ('Without noise, the calculated delta is %.6f and the error in the calculated del
        print ('Without noise, the Calculated omega is %.6f and the error in the calculated ome
```

```
Without noise, the calculated delta is 1.890923 and the error in the calculated delta is 0.00907
Without noise, the Calculated omega is 1.247086 and the error in the calculated omega is 0.04708
```

Thus, with windowing, we are able to estimate the original values to a reasonable level of accuracy.

```
In [16]: t = np.linspace(-pi,pi,129)
         t = t[:-1]
         dt = t[1]-t[0]
         fmax = 1/dt
         n = np.arange(128)
         wnd=f.fftshift(0.54+0.46*np.cos(2*pi*n/127))
         y = np.cos(omega*t + delta)
         y = y + 0.1*np.random.randn(128)
         y = y * wnd
         y[0]=0
         y=f.fftshift(y)
         Y=f.fftshift(f.fft(y))/128.0
         w=np.linspace(-pi*fmax,pi*fmax,129);w=w[:-1]
         plot_function(w,Y,4,'$\cos(1.2t + 1.9)$ with added noise.')
```



Thus, it can be seen that, when the noise was added, the error increased, though not by a lot.

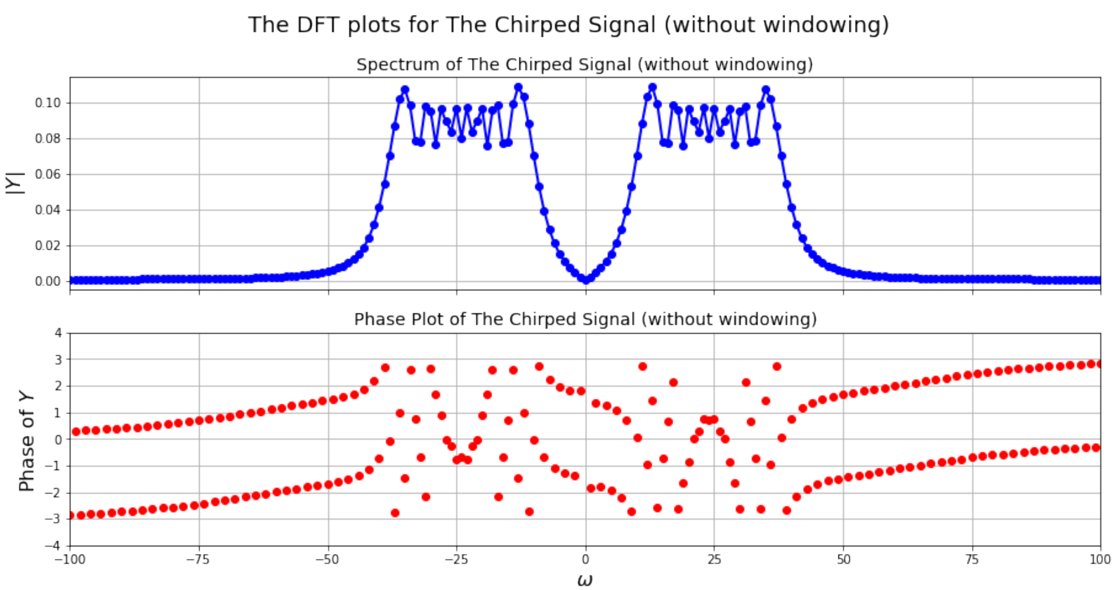
```
In [17]: ii = np.where(w>0)[0]
         ii = np.where((abs(Y) == max(abs(Y[ii]))))
```

```
est_delta = abs(np.angle(Y[ii])[0])    # The phase of the graph at the peak is the delta

ii = np.where((abs(Y) > 3.5e-2) & (w >= 0))[0]
est_omega = abs(Y[ii]*w[ii])
est_omega = sum(est_omega)/(sum(abs(Y[ii])))  # As peak is spread out, omega is estimat
print ('With noise, the calculated delta is %.6f and the error in the calculated delta
print ('With noise, the Calculated omega is %.6f and the error in the calculated omega
```

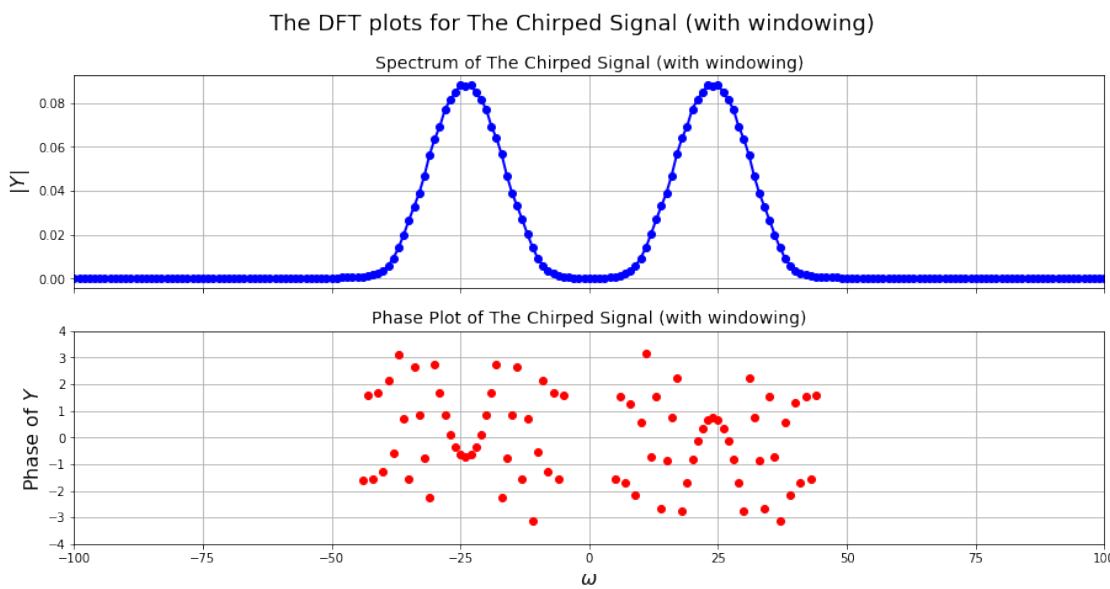
```
With noise, the calculated delta is 1.884112 and the error in the calculated delta is 0.015888
With noise, the Calculated omega is 1.260343 and the error in the calculated omega is 0.060343
```

In this section, the DFT plots of the chirped signal (dened below) is plotted, both with and without windowing. Though, windowing is not really required here, as the function would be continuous even after sampling it, as the beginning and ending values of the function in the interval $[-\pi, \pi]$ are the same

```
In [18]: t = np.linspace(-pi,pi,1025)
         t = t[:-1]
         dt = t[1]-t[0]
         fmax = 1/dt
         y = np.cos(16*t*(1.5 + (t/(2*pi))))
         y[0]=0
         y=f.fftshift(y)
         Y=f.fftshift(f.fft(y))/1024.0
         w=np.linspace(-pi*fmax,pi*fmax,1025)
         w=w[:-1]
         plot_function(w,Y,100,'The Chirped Signal (without windowing)')
```



In [21]:
```
t = np.linspace(-pi,pi,1025)
t = t[:-1]
dt = t[1]-t[0]
fmax = 1/dt
n = np.arange(1024)
wnd=f.fftshift(0.54+0.46*np.cos(2*pi*n/1023))
y = np.cos(16*t*(1.5 + (t/(2*pi))))
y = y * wnd
y[0]=0
y=f.fftshift(y)
Y=f.fftshift(f.fft(y))/1024.0
w=np.linspace(-pi*fmax,pi*fmax,1025)
w=w[:-1]
plot_function(w,Y,100,'The Chirped Signal (with windowing)')
```



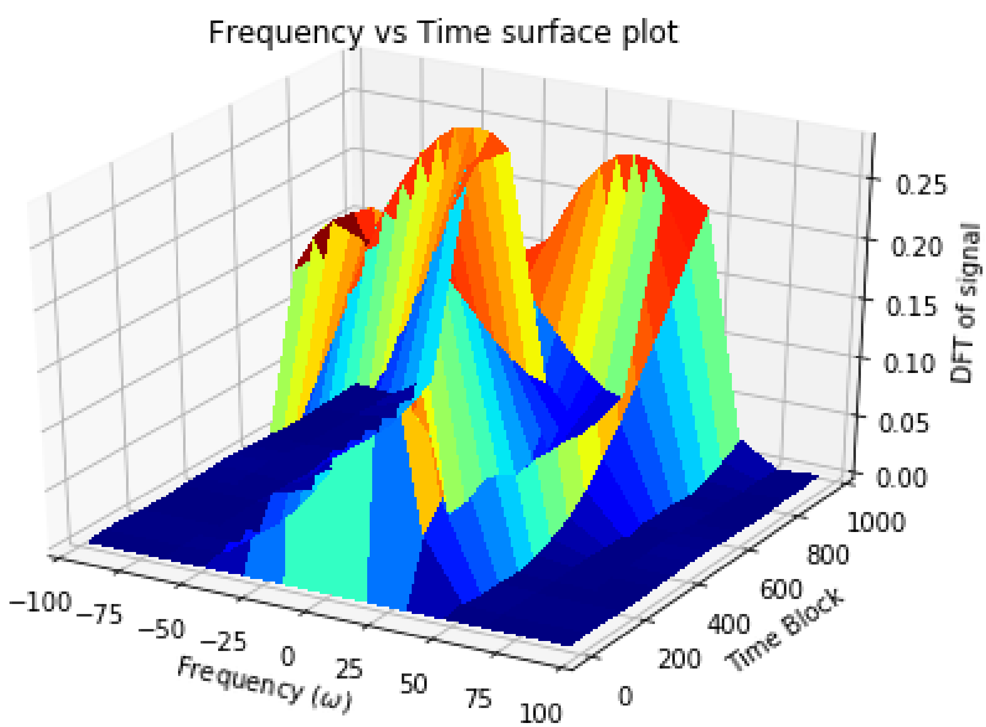
Thus, again, it can be seen that when the signal is windowed the peaks are more clearer (distinct), even though they are broader, whereas without windowing the peaks are jagged and not clear at all. Also, the magnitude also decreases at a much slower rate in the one without windowing, as seen from the fact that a phase is present throughout, for the case without windowing, whereas, for the one with, it decreases faster.

In [22]:
```
t = np.linspace(-pi,pi,1025)
t = t[:-1]
dt = t[1]-t[0]
fmax = 1/dt
t = np.array(np.split(t, 16))     # The entire 1024 elements are split into 16 disjoint
n = np.arange(64)
wnd=f.fftshift(0.54+0.46*np.cos(2*pi*n/63))
y = np.cos(16*t*(1.5 + (t/(2*pi))))
```

```
    y = y * wnd
    y[0]=0
    y=f.fftshift(y)
    Y=f.fftshift(f.fft(y))/64.0
    w=np.linspace(-pi*fmax,pi*fmax,65)
    w=w[:-1]
```

```
In [23]: n = np.arange(0,1024,64)
    fig1 = plt.figure(4)
    ax = p3.Axes3D(fig1)
    plt.title('Frequency vs Time surface plot')
    ax.set_xlabel('Frequency ($\omega$)')
    ax.set_ylabel('Time Block')
    ax.set_xlim([-100,100])
    ax.set_zlabel('DFT of signal')
    x,y = np.meshgrid(w,n)
    x[x>100]= np.nan      # Without this and the next line, the surface plot overflows due
    x[x<-100]= np.nan
    surf = ax.plot_surface(x, y, abs(Y), rstride=1, cstride=1,cmap=plt.cm.jet,linewidth=0,
    plt.show()
```

```
d:\python\lib\site-packages\ipykernel_launcher.py:11: RuntimeWarning: invalid value encountered
  # This is added back by InteractiveShellApp.init_path()
```



Thus, it is seen that at lesser time instants, the peaks of the DFT are closer to each other, while as the time instant (from which the 64 time points are taken) increases, the peaks become more wide apart. Because the graph was plotted with sets that are disjoint, this variation is not that clearly seen in the surface plot.

CONCLUSION

The observations and conclusions for each section is written in the respective sections. Some general conclusions are explained below. 13The normal DFT need not provide accurate peaks of the function, because of the fact that, it depends on the rate at which sampling is done on the continous time signal. If the rate of sampling is not a multiple of the signal frequency, a mismatch between signals occur after sampling. Because of this mismatch, periodic continuous time signals become discontinuous periodic signals after sampling, and this discontinuity is the one that causes the ambiguity in the fourier transform in the form of Gibb's Phenomenon. One way to reduce this ambiguity is to reduce the discontinuity in the initial function, which is done by mulitplying the function with a windowing function, which, as seen above, reduces the discontinuity, and makes the peaks more visible in the DFT. Though these peaks do become broader, because of the fact that, multiplication in the time domain leads to convolution in the frequency domain, which leads to broadening of the peaks. Thus, in general windowing helps in better distinguishing the function which becomes discontinuous after sampling.