EE2703: Assignment 9

Sagar, EE20B115

April 16, 2022

1 Aim

In this assignment we aim to analyse DFT's of non-periodic functions using *numpy.fft*, and solve problems associated with them, namely the *Gibbs Phenomenon* by using *windowing*.

2 Procedure and Observations

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

2.1.1 Code snippet:

```
t = linspace(-pi, pi, 65)[:-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)[:-1]

t = linspace(-pi, pi, 65)[:-1]
dt = t[1]-t[0]
fmax = 1/dt
n = arange(64)
y = sin(sqrt(2)*t) * hammingWindow(n)
y[0] = 0
y = fftshift(y)
Y = fftshift(fft(y))/64.0
w = linspace(-pi*fmax, pi*fmax, 65)[:-1]
```

2.1.2 Plots:

Using the method we followed for obtaining the DFT of a periodic signal, we get the following spectrum for $sin(\sqrt{2}t)$:

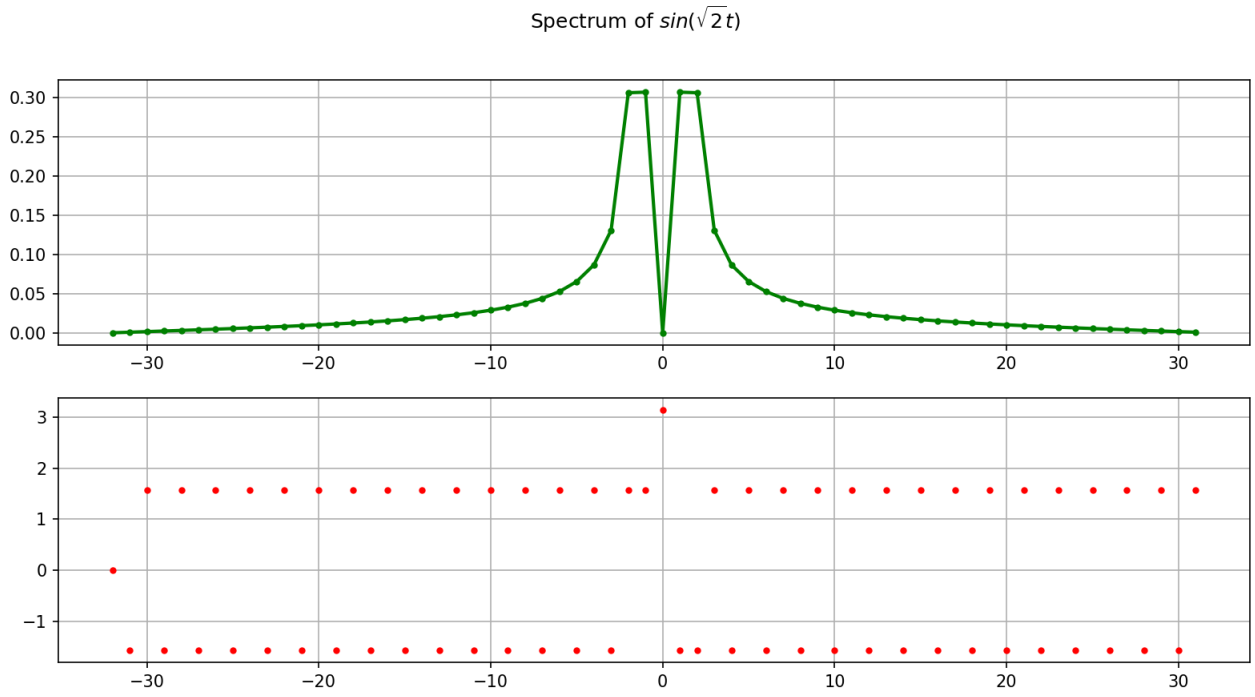

Figure 1: Spectrum of $sin(\sqrt{2}t)$

The above plot is not what was expected. This error occurs because the discontinuities in the function has led to the *Gibb's Phenomenon*. So, we do not observe a sharp peak, but rather a flat one.

This problem can be fixed by **windowing**. Windowing is an operation in which we multiply the time-domain function with a suitable window function. In this assignment, we choose the *Hamming Window*, defined as:

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

The result in time domain is that the magnitude of the jump is greatly reduced, thus minimizing the effect of Gibb's phenomenon in the frequency domain:

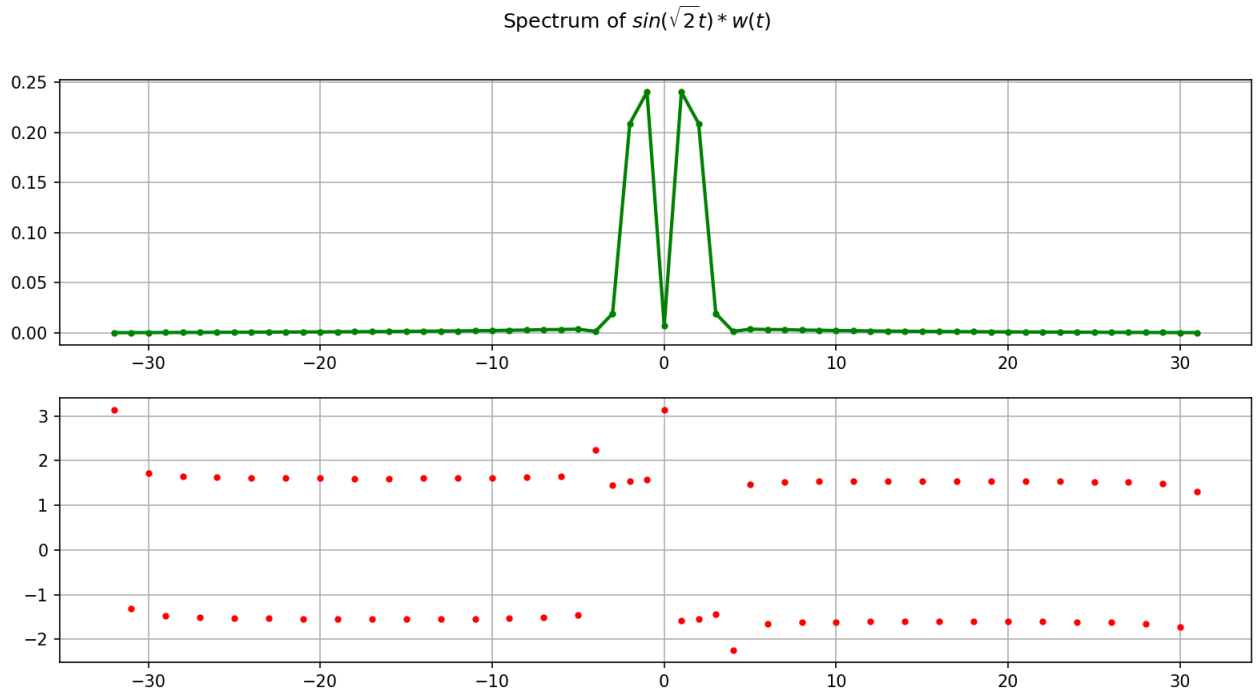

Figure 2: Spectrum of windowed $sin(\sqrt{2}t)$

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

2.2.1 Code snippet:

```
t = linspace(-4*pi, 4*pi, 257)[:-1]
dt = t[1]-t[0]
fmax = 1/dt
y = cos(0.86*t)**3
y[0] = 0
y = fftshift(y)
Y = fftshift(fft(y))/256.0
w = linspace(-pi*fmax, pi*fmax, 257)[:-1]

t = linspace(-4*pi, 4*pi, 257)[:-1]
dt = t[1]-t[0]
fmax = 1/dt
n = arange(256)
y = (cos(0.86*t))**3 * hammingWindow(n)
y[0] = 0
y = fftshift(y)
Y = fftshift(fft(y))/256.0
w = linspace(-pi*fmax, pi*fmax, 257)[:-1]
```

2.2.2 Plots:

We can observe the effect of windowing even better for $cos^3(0.86t)$. We get the following spectra before and after windowing:

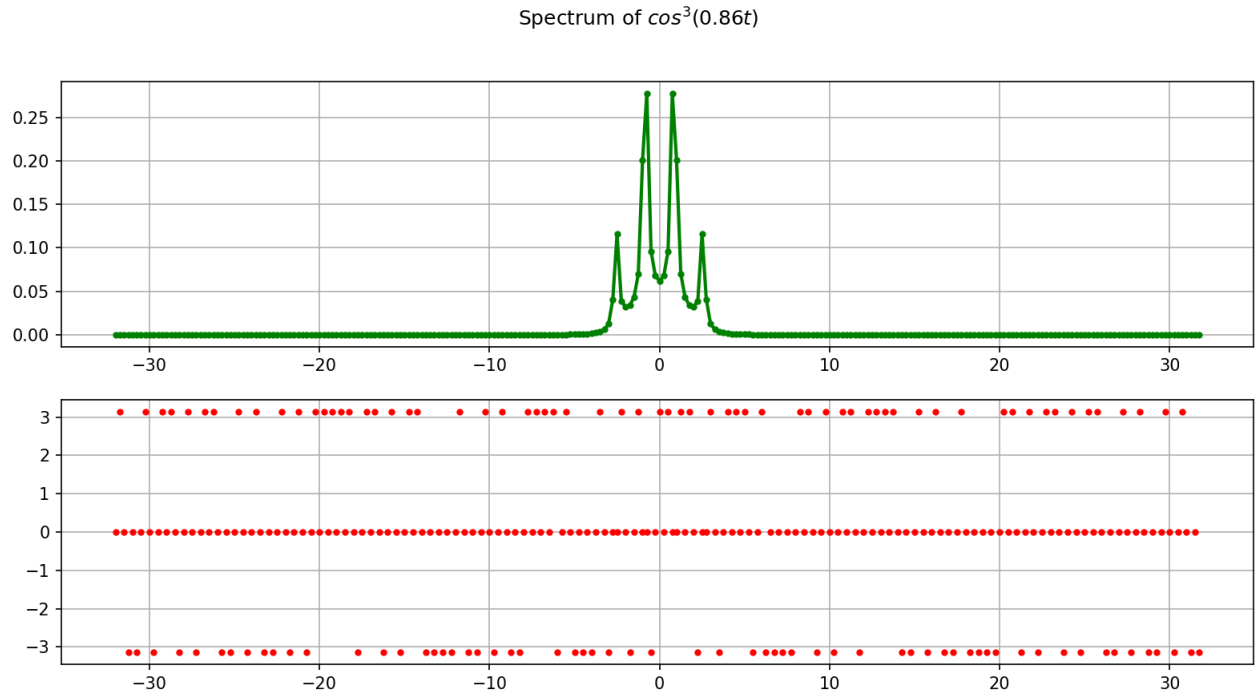

Figure 3: Spectrum of $cos^3(0.86t)$ without windowing

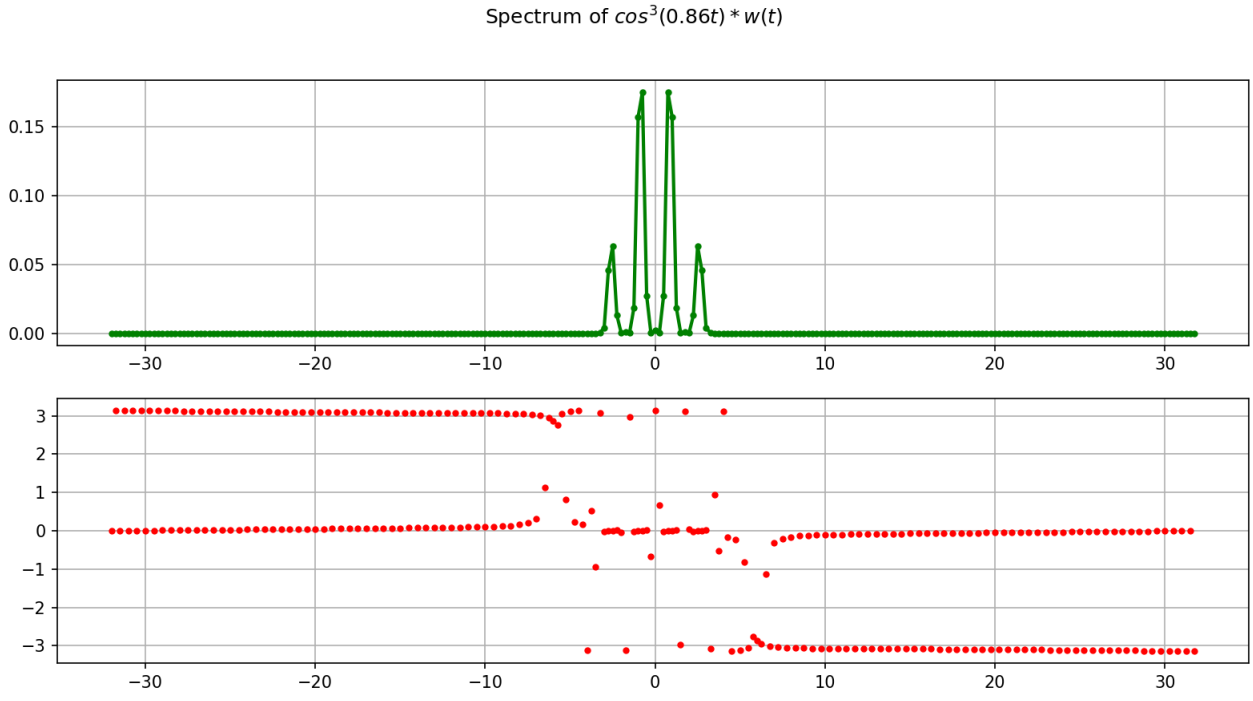

Figure 4: Spectrum of $cos^3(0.86t)$ after windowing

2.2.3 Observations:

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

2.3 Parameter Estimation using DFT

2.3.1 Code snippet:

```
wo = 1.35
d = pi/2

t = linspace(-pi, pi, 129)[:-1]
trueCos = cos(wo*t + d)
fmax = 1.0/(t[1]-t[0])
n = arange(128)
y = trueCos.copy()*hammingWindow(n)
y = fftshift(y)
Y = fftshift(fft(y))/128.0
w = linspace(-pi*fmax, pi*fmax, 129)[:-1]
spectrum(r"Spectrum of $cos(\omega_o t + \delta) \cdot w(t)$", w, Y)
estimateWandD(w, wo, Y, d, pow=1.75)

trueCos = cos(wo*t + d)
noise = 0.1*randn(128)
n = arange(128)
y = (trueCos + noise)*hammingWindow(n)
fmax = 1.0/(t[1]-t[0])
y = fftshift(y)
Y = fftshift(fft(y))/128.0
w = linspace(-pi*fmax, pi*fmax, 129)[:-1]
spectrum(r"Spectrum of $(cos(\omega_o t + \delta) + noise) \cdot w(t)$", w, Y)
estimateWandD(w, wo, Y, d, pow=2.5)
```

2.3.2 Plots:

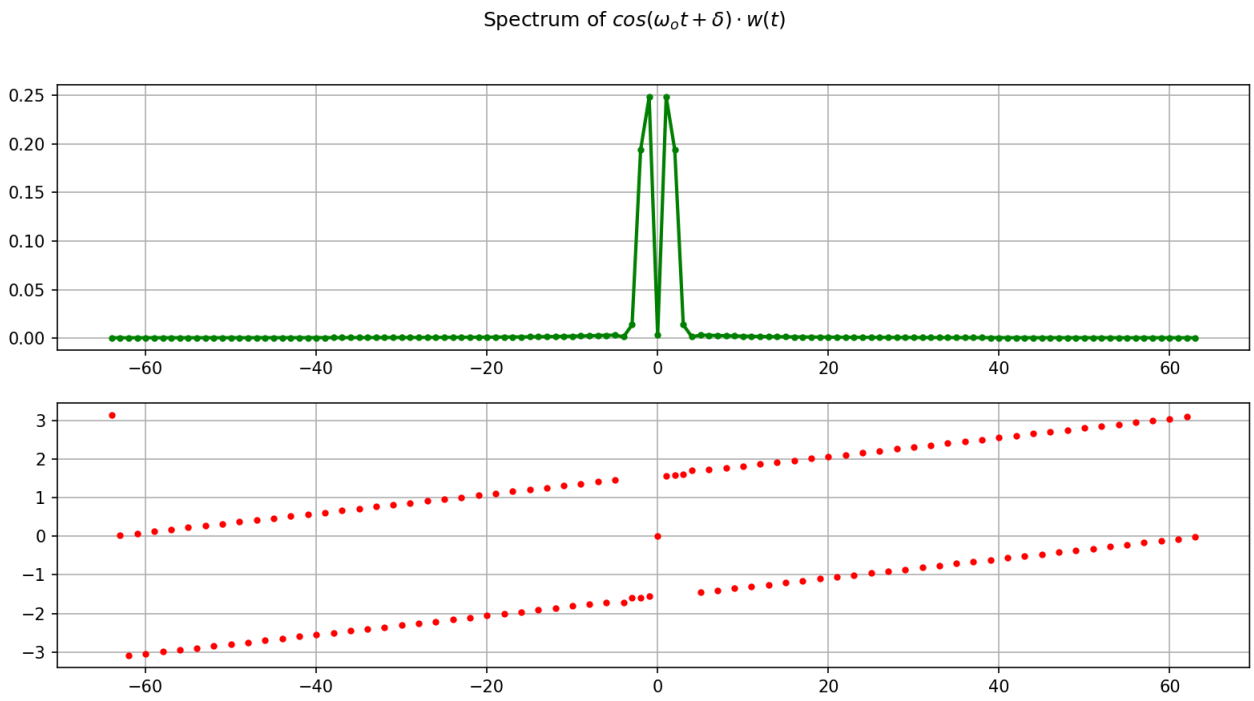

Figure 5: Spectrum of windowed $cos(\omega_0 * t + \delta) * w(t)$

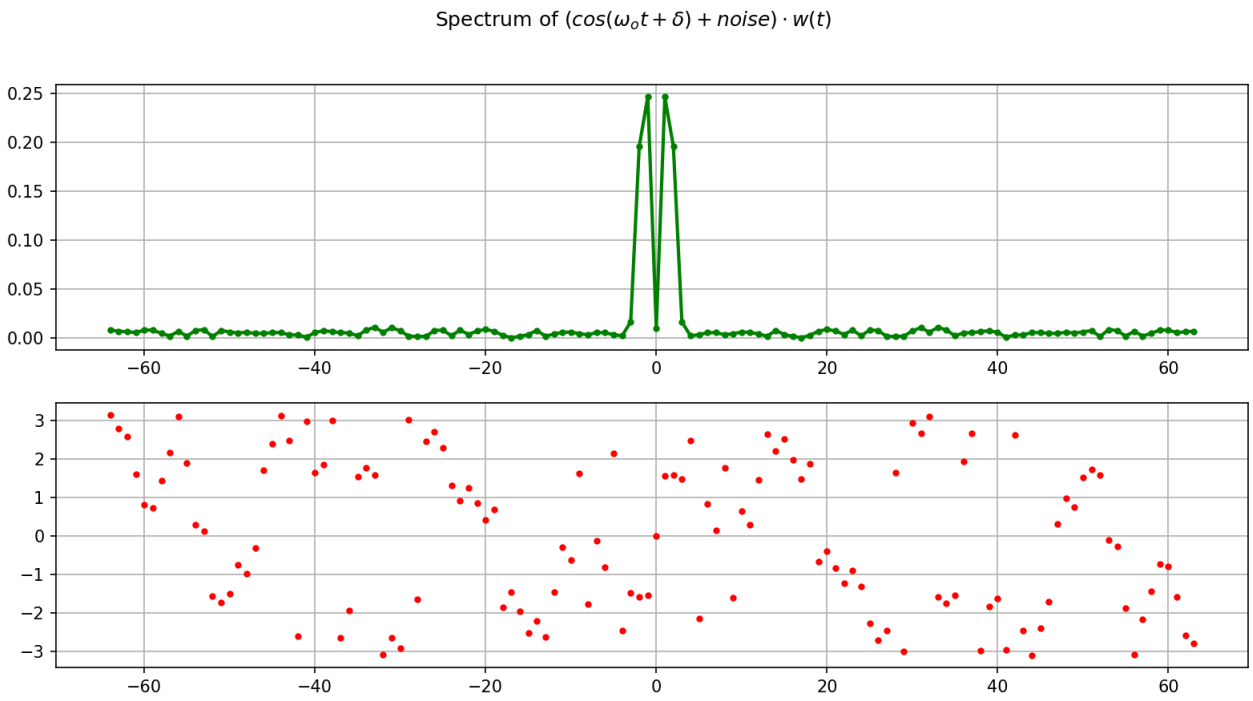

Figure 6: Spectrum of windowed $(cos(\omega_0 * t + \delta) + noise) * w(t)$

We are given a 128-element vector with sampled values of the signal $cos(\omega_o t + \delta)$, where ω_o and δ are to be estimated.

We cannot extract ω_o from the DFT spectrum because the sampling rate is very low (only 128 points). The peaks will overlap and so we will not be able to extract their values.

We can estimate values using equations given below where least square method is used to estimate A and B:

$$\omega_{o_{est}} = \frac{\sum_\omega |Y(j\omega)|^k \cdot \omega}{\sum_\omega |Y(j\omega)|^k}$$
$$y(t) = Acos(\omega_o t) - Bsin(\omega_o t)$$
$$\delta_{est} = \arctan(-\frac{B}{A})$$

After that we can vary the parameter k manually to find out which is suitable for the given range of ω_o.

With $k = 1.7$, for estimation in absence of noise, and $k = 2.4$, for estimation in presence of noise, we get the following results ($\omega_o = 1.35$ and $\delta = \frac{\pi}{2}$ in both cases) :

Noise	ω_o	$\omega_{o_{est}}$	δ	δ_{est}
0	1.35	1.453	1.571	1.571
0.1*`rand(128)`	1.35	1.437	1.571	1.571

2.4 DFT of chirp $cos(16t(1.5 + \frac{t}{2\pi}))$

2.4.1 Code snippet:

```
def chirped(t):
    return cos(16*(1.5*t + (t**2)/(2*pi)))
```

```
t = linspace(-pi, pi, 1025)[:-1]
x = chirped(t)
signal(t, x, r"$cos(16(1.5 + \frac{t}{2\pi})t)$")
fmax = 1.0/(t[1]-t[0])
X = fftshift(fft(x))/1024.0
w = linspace(-pi*fmax, pi*fmax, 1025)[:-1]
spectrum(r"DFT of $cos(16(1.5 + \frac{t}{2\pi})t)$", w, X, 'g.-', 'r.')

n = arange(1024)
x = chirped(t)*hammingWindow(n)
signal(t, x, r"$cos(16(1.5 + \frac{t}{2\pi})t) \cdot w(t)$")
X = fftshift(fft(x))/1024.0
spectrum(r"DFT of $cos(16(1.5 + \frac{t}{2\pi})t) \cdot w(t)$", w, X, 'g.-', '

def STFT(x, t, batchSize=64):
    t_batch = split(t, 1024//batchSize)
    x_batch = split(x, 1024//batchSize)
    X = zeros((1024//batchSize, batchSize), dtype=complex)
    for i in range(1024//batchSize):
        X[i] = fftshift(fft(x_batch[i]))/batchSize
    return X
```

2.4.2 Plots:

A chirp is a signal in which the frequency increases or decreases with time. We are going to analyse the chirped signal $cos(16t(1.5+\frac{t}{2\pi}))$. First, we shall plot the signal versus time, to get an idea of how the signal looks like:

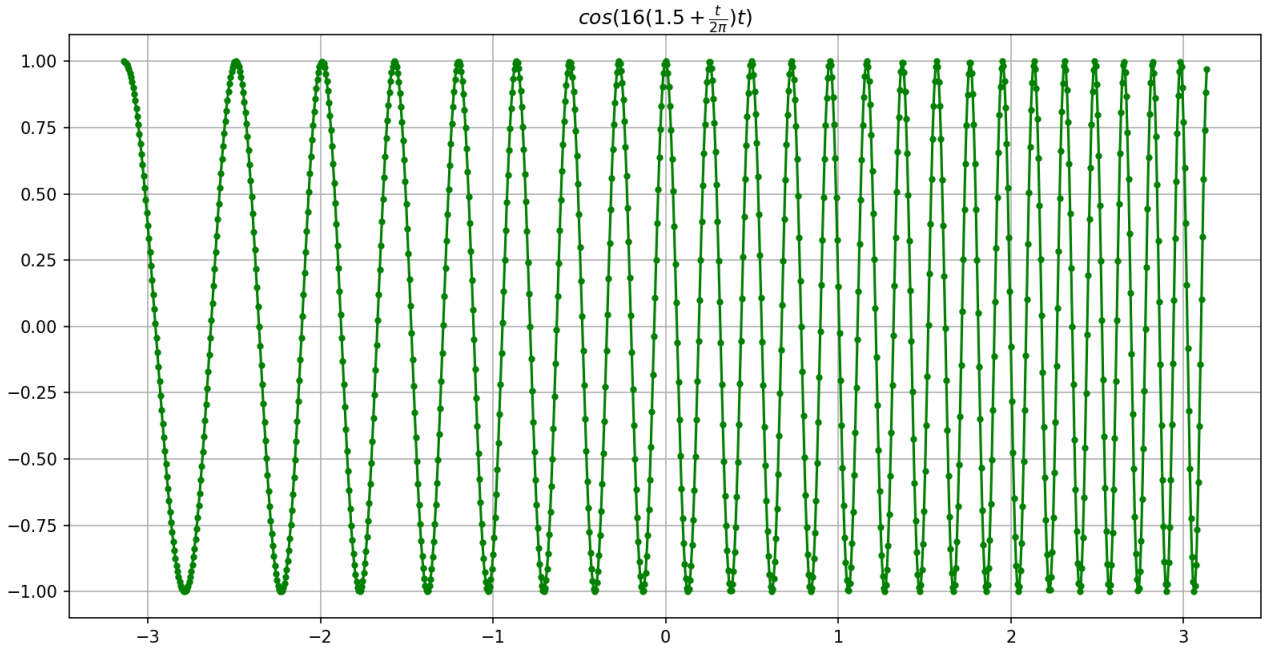

Figure 7: $cos(16t(1.5+\frac{t}{2\pi}))$

We observe that the frequency varies from 16 rad/sec to 32 rad/sec as t goes from $-\pi$ sec to π sec. On finding the DFT of the above signal, we get:

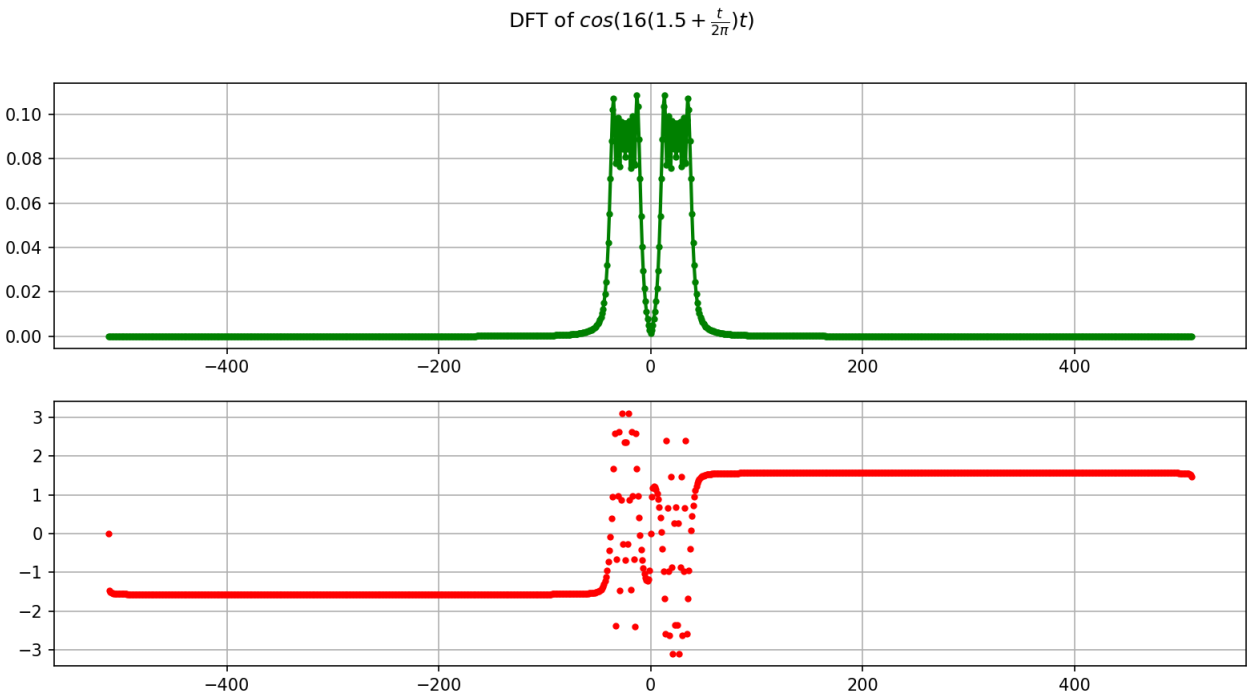

Figure 8: Spectrum of $cos(16t(1.5 + \frac{t}{2\pi}))$

Applying the Hamming Window to the chirp results in the following:

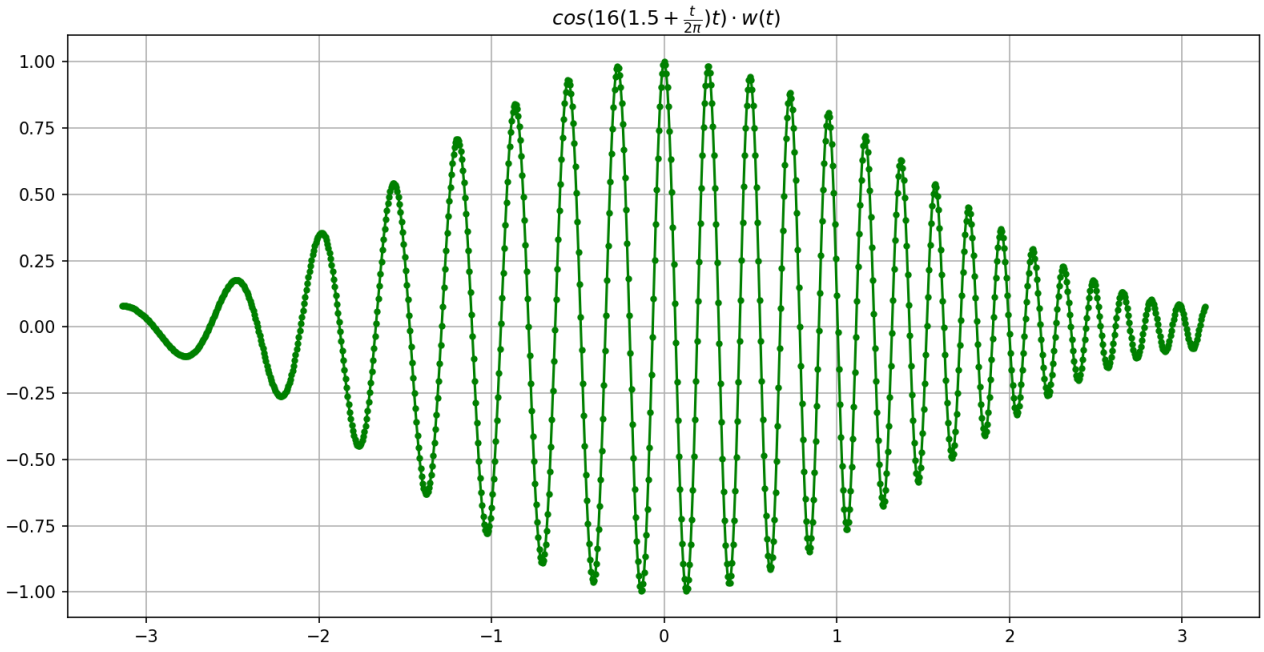

Figure 9: Windowed $cos(16t(1.5 + \frac{t}{2\pi}))$

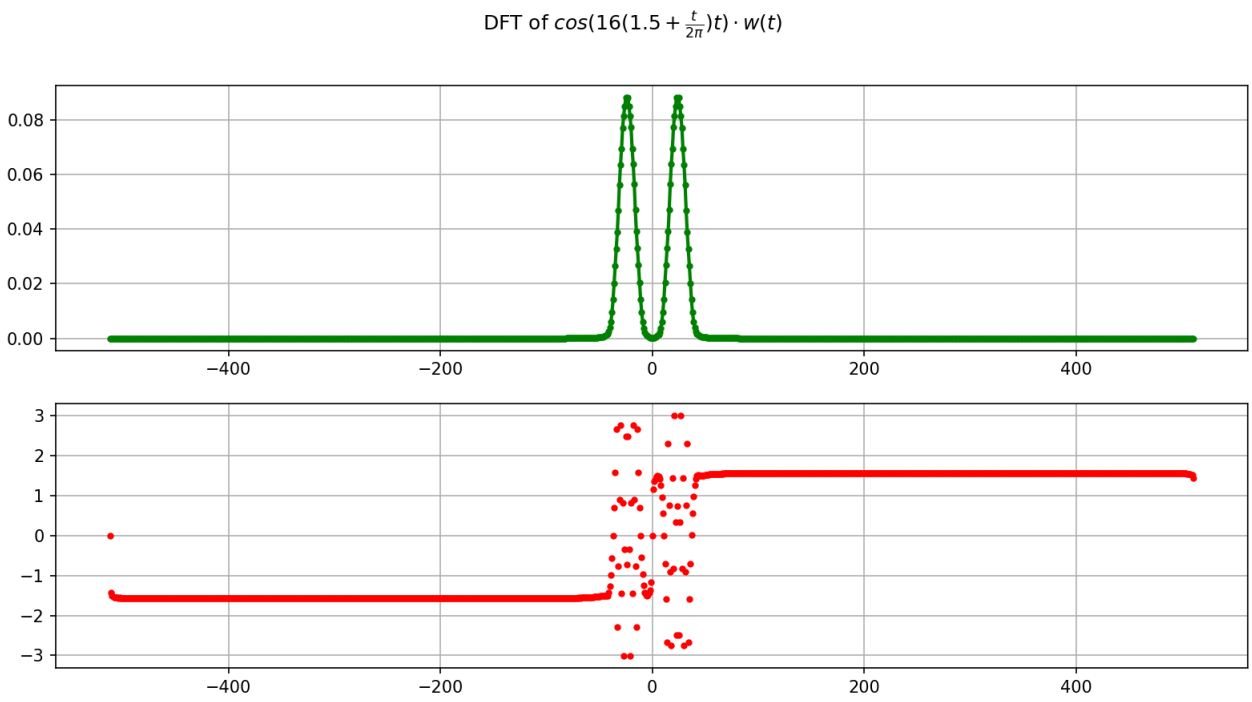

Figure 10: Spectrum of windowed $cos(16t(1.5 + \frac{t}{2\pi}))$

2.4.3 Observations:

1. We can observe that variations in the frequency have been smoothed out by the window.
2. We can observed that the frequencies are more accurately confined to the range 16-32 rad/sec.

2.5 Time-frequency plot of $cos(16t(1.5 + \frac{t}{2\pi}))$

2.5.1 Code snippet:

```
def STFT3d(t, w, X):
global figNum

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

fig = plt.figure(figNum)
ax = fig.add_subplot(211, projection='3d')
surf = ax.plot_surface(w, t, abs(X).T)

ax = fig.add_subplot(212, projection='3d')
surf = ax.plot_surface(w, t, angle(X).T)
figNum+=1

x = chirped(t)
X = STFT(x, t)
STFT3d(t, w, X)

x = chirped(t)*hammingWindow(arange(1024))
X = STFT(x, t)
```

STFT3d(t , w, X)

2.5.2 Plots:

We can split the chirp in the time interval $[-\pi,\ \pi]$ into smaller intervals of time, and observe how the frequency of the signal varies with time.

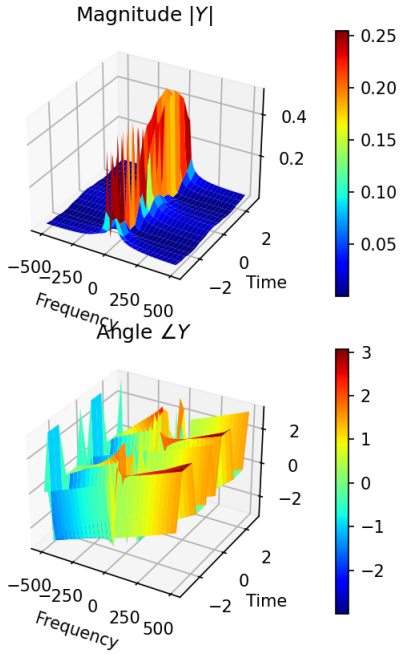

Figure 11: Spectrum of windowed $cos(16t(1.5 + \frac{t}{2\pi}))$

Now the plot but with a windowed version of the chirped signal.

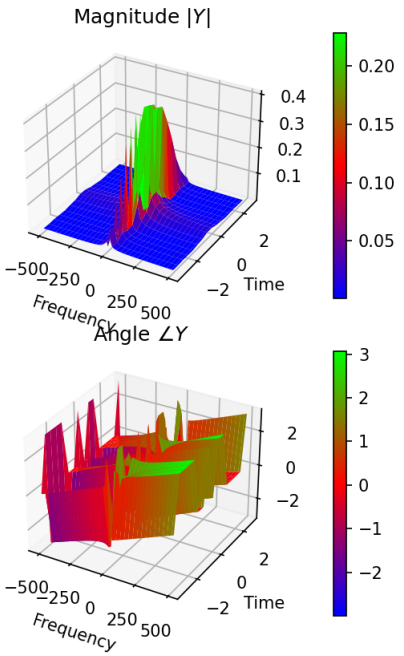

Figure 12: Spectrum of windowed $cos(16t(1.5 + \frac{t}{2\pi}))$

2.5.3 Observations:

We can see that the change in the magnitude is more gradual in the windowed case.

3 Conclusions

1. The DFT was obtained using a 2π periodic extension of the signal, and thus the spectrum was found to be erroneous for a non periodic function.
2. The spectrum was rectified by the using a windowing technique, by employing the Hamming window.
3. Given a vector of cosine values in the a time interval, the frequency and phase were estimated from the DFT spectrum, by using the expectation value of the frequency and a parameter tuning for optimum values.
4. The DFT of a chirped signal was analysed and its time-frequency plot showed the gradual variation of peak frequency of the spectrum with time.