EE2703 : Applied Programming Lab
Assignment 10
Spectra of non-periodic signals

Arun Krishna A M S
EE19B001

2nd June 2021

Objective

In the previous assignment we looked at functions that were periodic and extracted their spectra. The approach was:

- Sample the signal so that **Nyquist Criteria** is met. Generate the frequency axis from $-f_{max}/2$ to $f_{max}/2$
- Obtain the **Discrete Fourier Transform**. Rotate the samples so that f ranges from $f = -f_{max}/2$ to $f = f_{max}/2 - \Delta f$.
- Plot the magnitude and phase of the spectrum.

In this assignment our aim is to

- Obtain and analyze the frequency response of non-periodic discrete time signals
- Try to minimize the errors that arise from **Gibbs Phenomenon** by performing **Windowing** using a technique called as **Hamming**

Analysis of Frequency Response of $\sin\left(\sqrt{2}t\right)$

The Fast Fourier Transform any for non-periodic-function can be obtained by

```
def FastFourierTransform(function, interval, N, boolOdd):
    time = linspace(interval[0], interval[1], N + 1)[:-1]
    w = linspace(-pi*N/(interval[1]-interval[0]),
                  pi*N/(interval[1]-interval[0]), N + 1)[:-1]
    y = function(time)
    if boolOdd == True:
        y[0] = 0
    y = fftshift(y)
    Y = fftshift(fft(y))/N
    phase = angle(Y)
```

In case of function $\sin\left(\sqrt{2}t\right)$, the signal is sampled in the interval $[\pi, \pi)$ and the `fft()` and `fftshift()` functions are used to obtain the **DFT** of the said function.

In case of any odd function, we can observe that the code initializes the first element `y[0] = 0` to zero. This is because although the function is odd, the samples are not Anti-symmetric. Un-symmetric samples can result in residual components in phase.

For $i \in \{1, 2, 3...\frac{N}{2} - 1\}$:

$$y[i] = y[N - i]$$

$$y[\frac{N}{2}] = y[-t_{max}] = \sin\left(t_{\frac{N}{2}}\right)$$

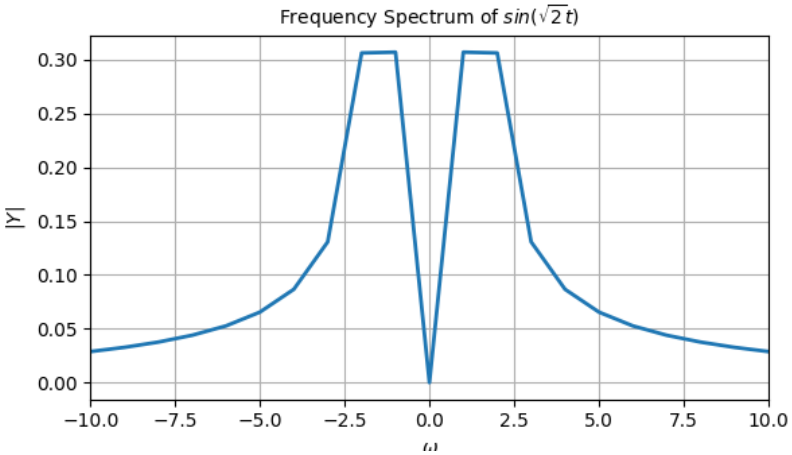
$$Y[k] = \sum_{n=0}^{N-1} y[n] exp(j\frac{2\pi kn}{N})$$

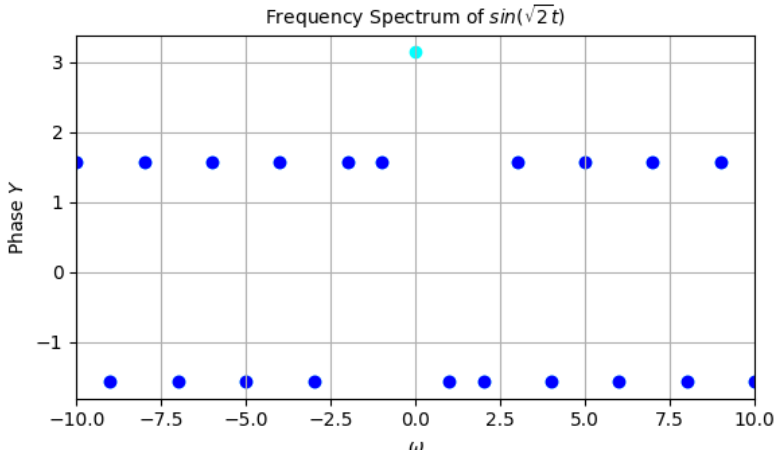
$$Y[k] = \sum_{n=0}^{\frac{N}{2}-1} y[n](exp(j\frac{2\pi kn}{N}) - exp(-j\frac{2\pi kn}{N}) + (y[\frac{N}{2}]exp(j\pi k))$$

$$Y[k] = \sum_{n=0}^{\frac{N}{2}-1} -2jy[n]sin(\frac{2\pi}{N}kn) + (-1)^k(y[\frac{N}{2}])$$

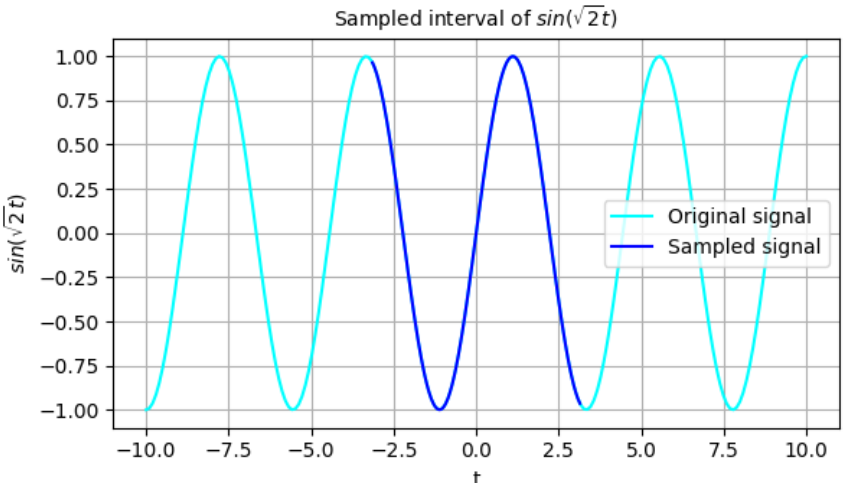
This system is no longer purely imaginary which is evident from **CTFT**. For the system to be purely imaginary as expected $y[0] = 0$ to give us purely imaginary phase.

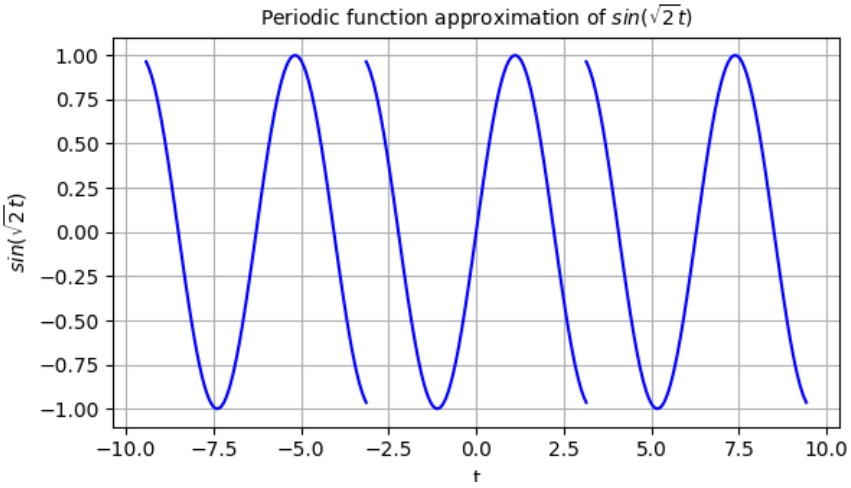
If the DFT of the function is obtained we observe the following:





One could observe that frequencies are spread throughout instead of just two peaks at $\pm\sqrt{(2)}$. This is because when the signal is sampled, the constructed periodic function would be





As one can observe, the periodic function of the sampled interval is not same as the original function. Even though the function $\sin(\sqrt{2}t)$ is periodic, the portion between $-\pi$ and π is not the region that can be replicated to obtain the original periodic function. The function showed in the second graph is what the DFT is trying to fourier analyse.

The discontinuities in the second graph observed for each section of 64 samples results in the **Gibbs Phenomenon**. This phenomenon results in higher frequencies having significant contributions.

Windowing is a technique to reduce the discontinuities by damping the function near the discontinuities. For this we shall multiply the given function sequence by a **window sequence** w[n]:

$$g(n) = w(n) * f(n)$$

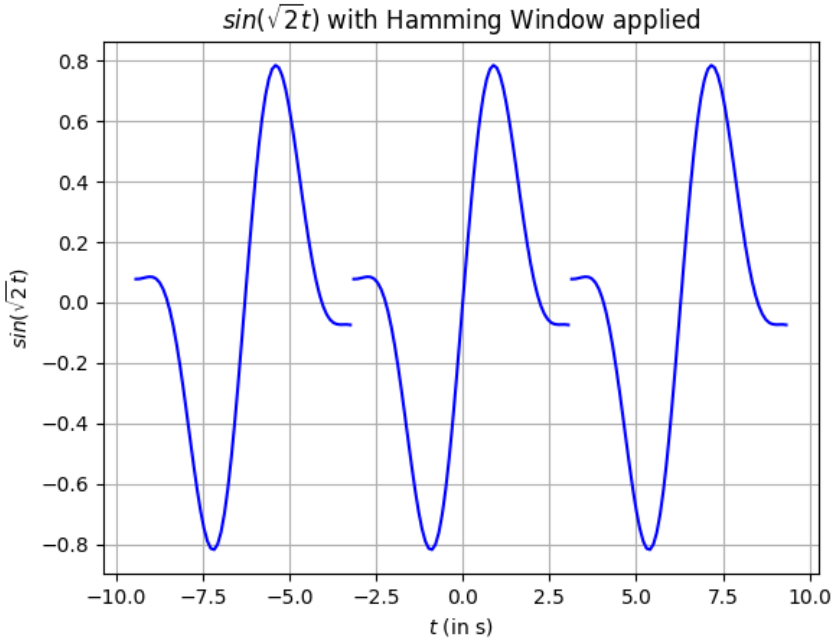
This windowing will result in the spikes smeared out by W_k. So we expect to get broader peaks with frequencies completely suppressed at the end of the window. The function that we use for windowing is called **Hamming Window**

$$f(z) = \begin{cases} 0.54 + 0.46cos(\frac{2\pi n}{N-1}) & \text{for} \quad |n| \leq \frac{N-1}{2} \\ 0 & \text{for} \quad else \end{cases}$$

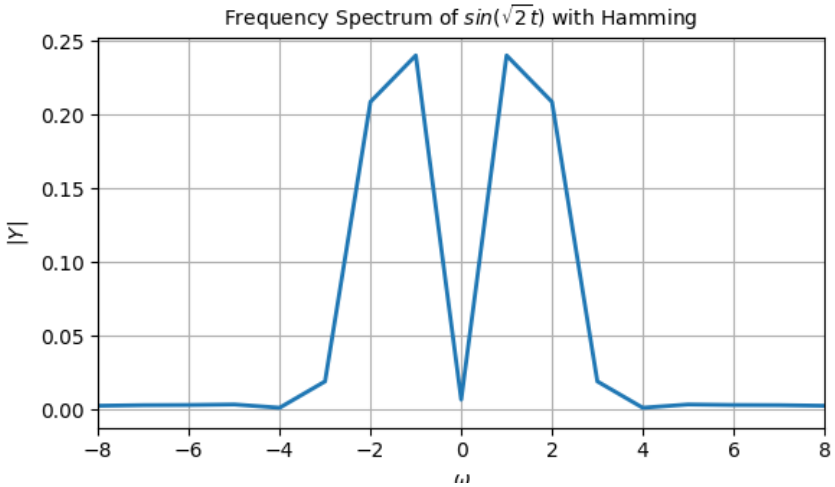
The hamming function is given by

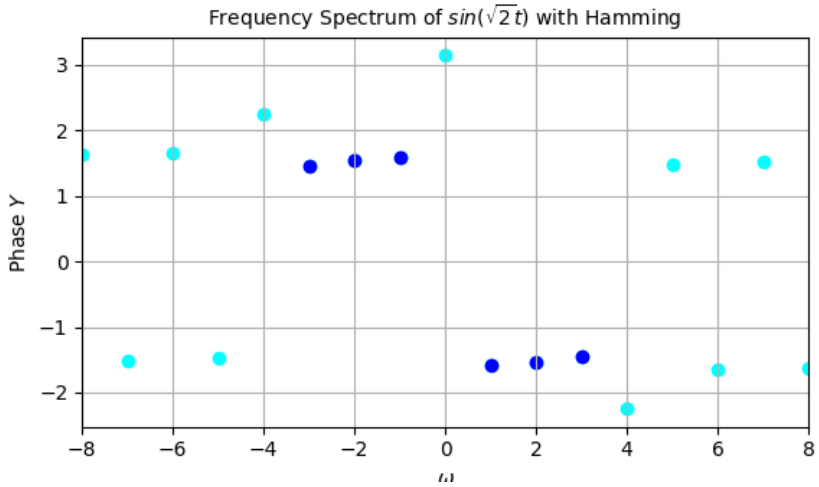
```
def sinroot2(x):
    return sin(sqrt(2) * x)

# Function to return Hamming Window sequence
def hammingFunction(n, a = 0.54,b = 0.46):
    return fftshift(a + b*cos((2*pi*n)/(len(n) - 1)))

def hammingsinroot2(x):
    return sinroot2(x)*hammingFunction(arange(len(x)))
```

After applying the hamming function one can observe that the obtained graph is

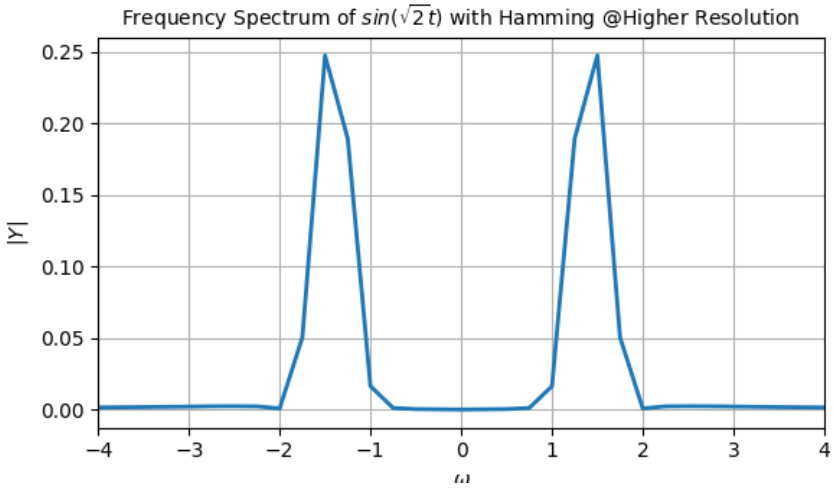


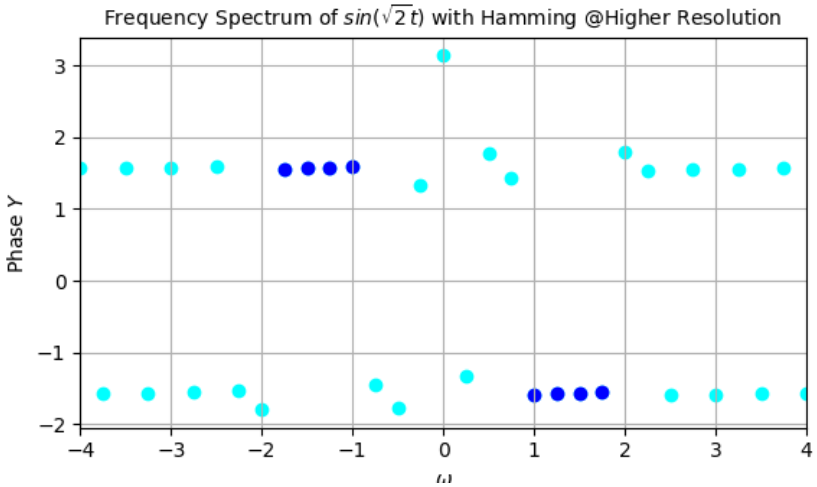
Although the function still has discontinuities, it has been greatly reduced by multiplying with the hamming function. We next try to plot the frequency spectrum of the $sin(\sqrt{2}nT_s)w[n]$ function.





As expected the magnitude plot is closer to our expectations. But it is still too wide. We need better resolution to get better defined peaks. So we increase the time interval to $[4\pi, 4\pi)$ and the samples to 256.

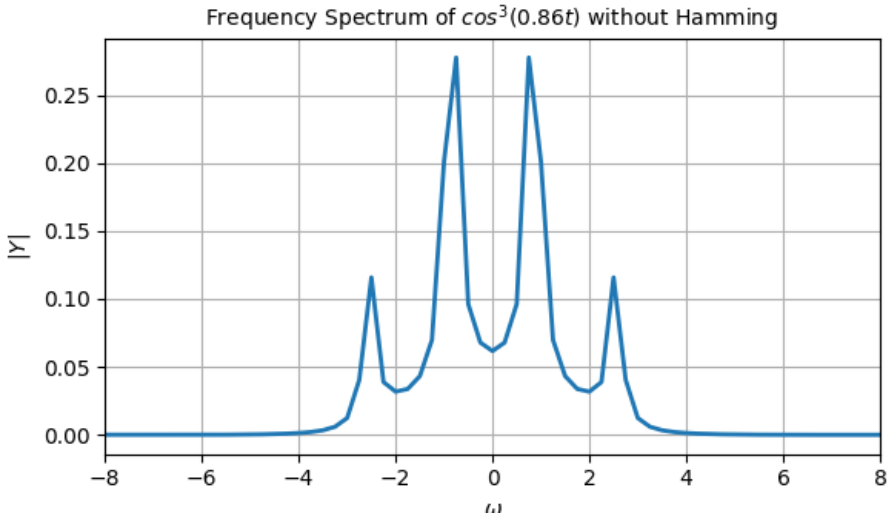


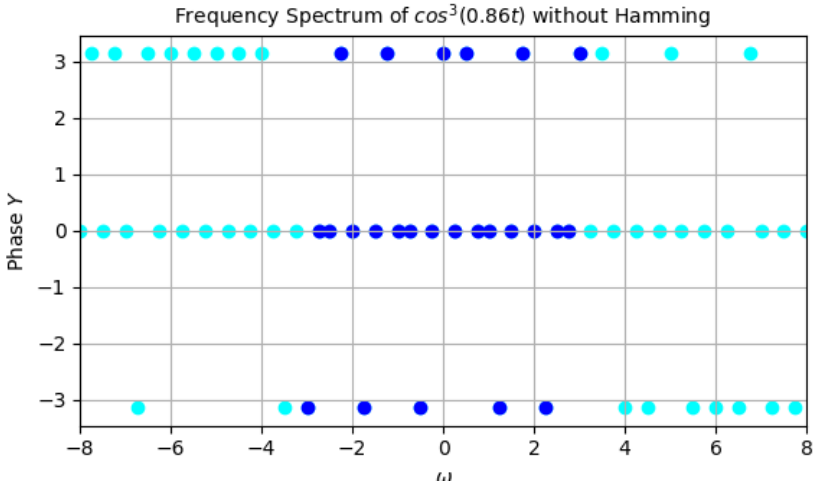


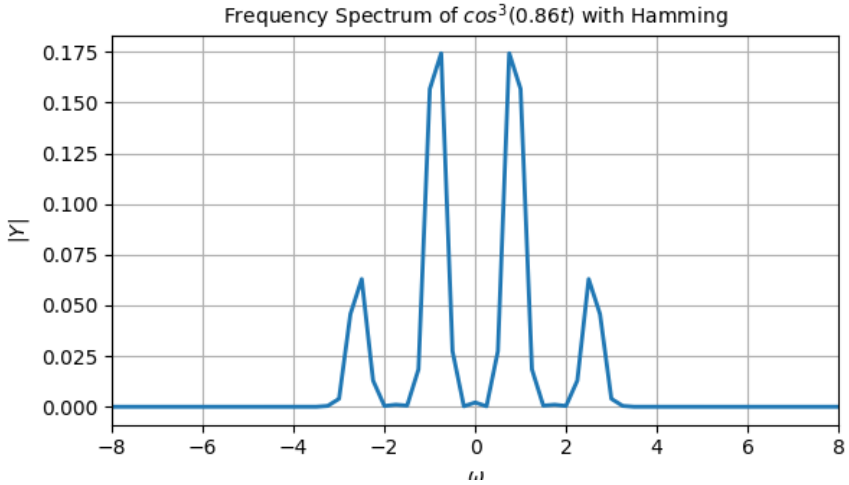
As expected the peaks are more well defined and the magnitude response falls more rapidly for higher values. But one can also observe that the width of the peaks have increased a bit with windowing for the same resolution. This is again due to windowing.

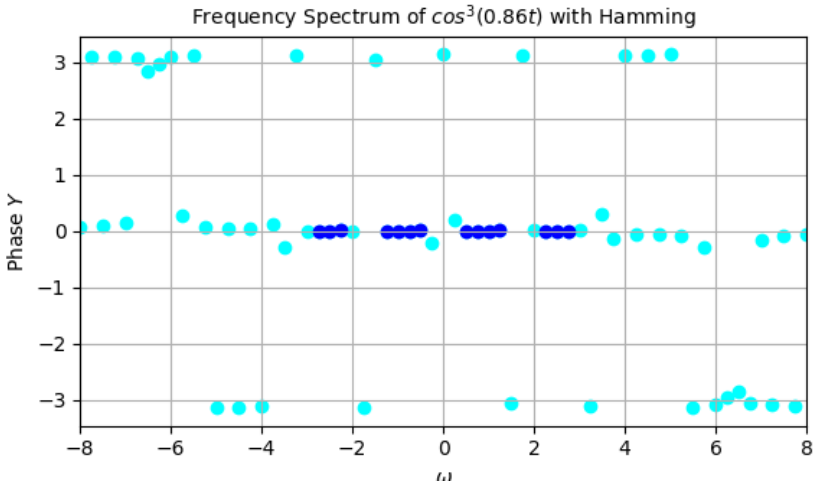
Analysis of Frequency Response of $\cos^3(0.86t)$

As before, the same procedure is applied and we observe the following for with and without hamming




Frequency Spectrum of $cos^3(0.86t)$ without Hamming
Phase Y
3
2
1
0
−1
−2
−3
−8
−6
−4
−2
0
2
4
6
8
ω


Frequency Spectrum of $cos^3(0.86t)$ with Hamming
|Y|
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
−8
−6
−4
−2
0
2
4
6
8
ω


Frequency Spectrum of $cos^3(0.86t)$ with Hamming
Phase Y
3
2
1
0
−1
−2
−3
−8
−6
−4
−2
0
2
4
6
8
ω

The graphs are expected as

$$cos^3(0.86t) = \frac{3}{4}cos(0.86t) + \frac{1}{4}cos(3(0.86t))$$

$$cos^3(t) = \frac{3}{8}e^{0.86jt} + \frac{3}{8}e^{-0.86jt} + \frac{1}{8}e^{3(0.86)jt} + \frac{1}{8}e^{-3(0.86)jt}$$

We expect peaks at $0.86, -0.86, 2.58, -2.58$ frequencies which is as observed with the peaks being at approximately correct spot. The phase are also as expected equal to zero

Analysis of Frequency Response of $cos(\omega t + \phi)$

Next we try to find the DFT of the signal $cos(\omega t + \phi)$ with arbitrary ω and ϕ values with $0.5 < \omega < 1.5$ and $-\pi < \phi < \pi$.

$$cos(\omega t + \phi) = \frac{1}{2}e^{j\phi}e^{j\omega t} + \frac{1}{2}e^{-j\phi}e^{-j\omega t}$$

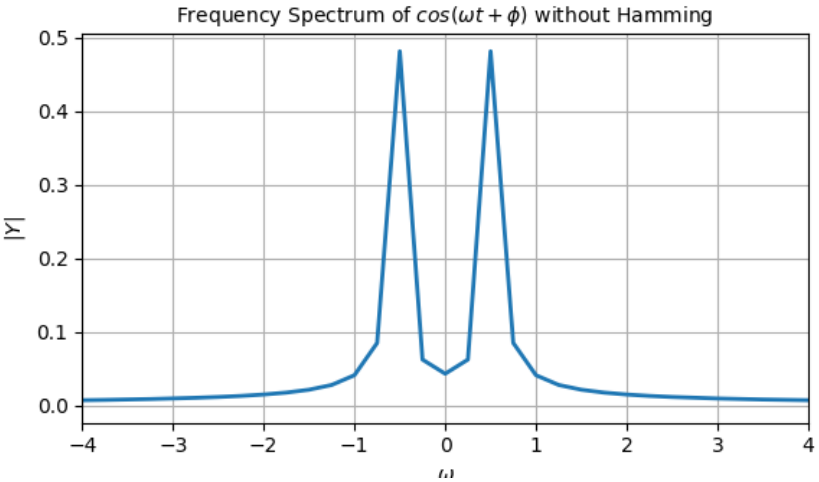
For this function, we observe that the peaks would lie at ω frequencies with the phase being equal to ϕ. The following code generates an arbitrary frequency and phase and computes the function.

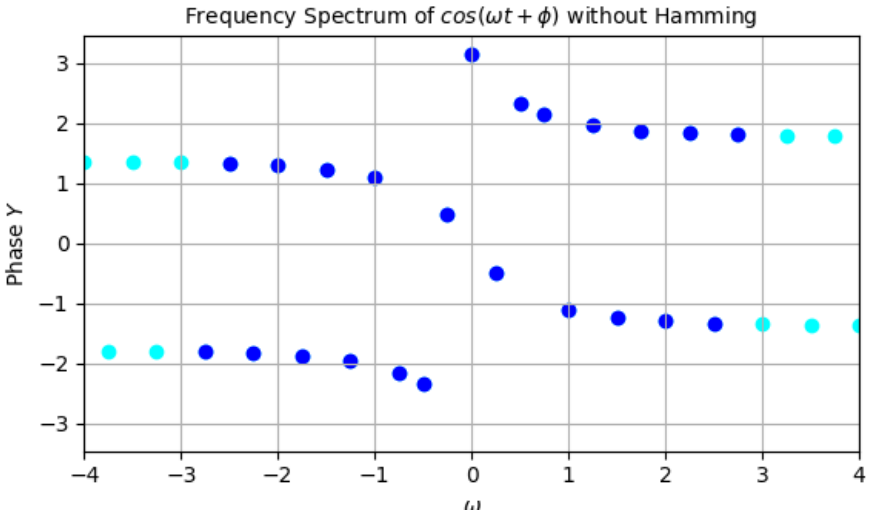
```
freq = random.uniform(0.5,1.5)
delta = random.uniform(-pi,pi)
def cosfunction(x):
    return cos(freq*x + delta)
```

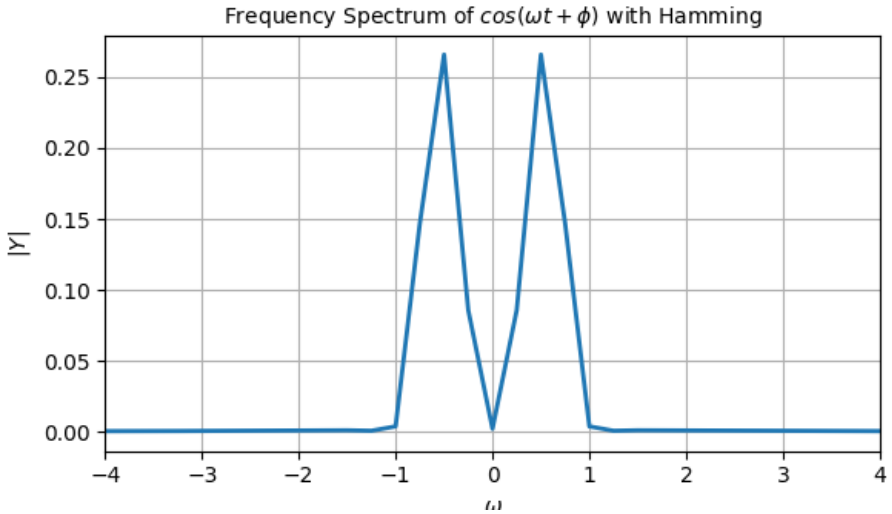
For the arbitrarily generated frequency and phase,

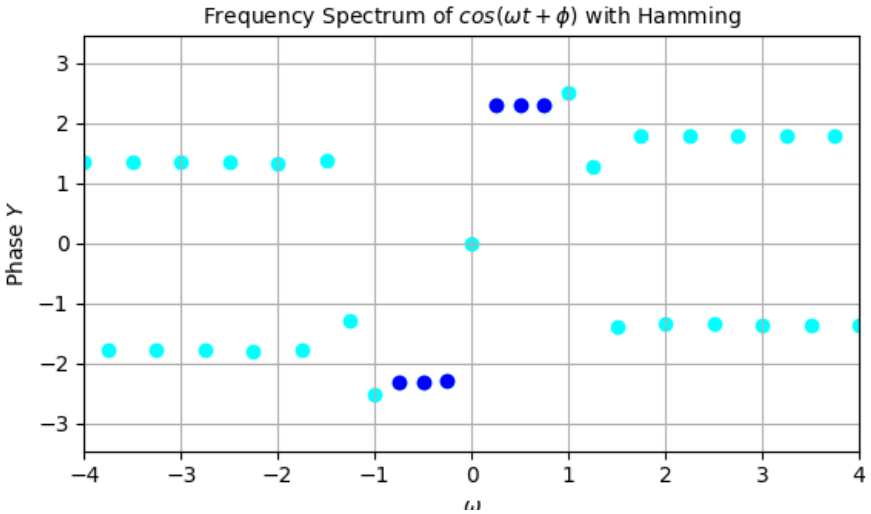
```
frequency =  0.5361703762348236
phase =  2.3007949240700682
```

the observed frequency response is




Frequency Spectrum of $cos(\omega t + \phi)$ without Hamming
Phase Y
3
2
1
0
−1
−2
−3
−4
−3
−2
−1
0
1
2
3
4


Frequency Spectrum of $cos(\omega t + \phi)$ with Hamming
|Y|
0.25
0.20
0.15
0.10
0.05
0.00
−4
−3
−2
−1
0
1
2
3
4


Frequency Spectrum of $cos(\omega t + \phi)$ with Hamming
Phase Y
3
2
1
0
−1
−2
−3
−4
−3
−2
−1
0
1
2
3
4

Suppose if we have to find the frequency ω and phase ϕ from the frequency response by computing a weighted average based on its magnitude. We reduce our region, by considering the magnitudes above a threshold point (0.1), after which weighted average is calculated.

```
def estimatingWo_and_delta(w,mag,phase):
    significantmag = where(mag > 0.1)
    w = w[significantmag]
    mag = mag[significantmag]
    phase = phase[significantmag]

    estimatedWo = sum((mag**2)*abs(w))/sum(mag**2)
    estimatedDelta = sum((mag**2)*abs(phase))/sum(mag**2)

    print("Estimated frequency:", estimatedWo)
    print("Estimated phase:", estimatedDelta)
```

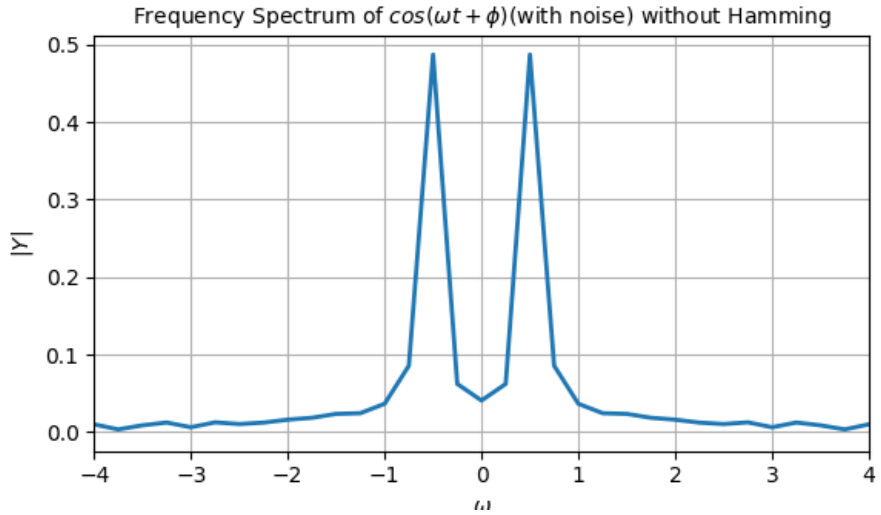
The estimated frequency and phase is

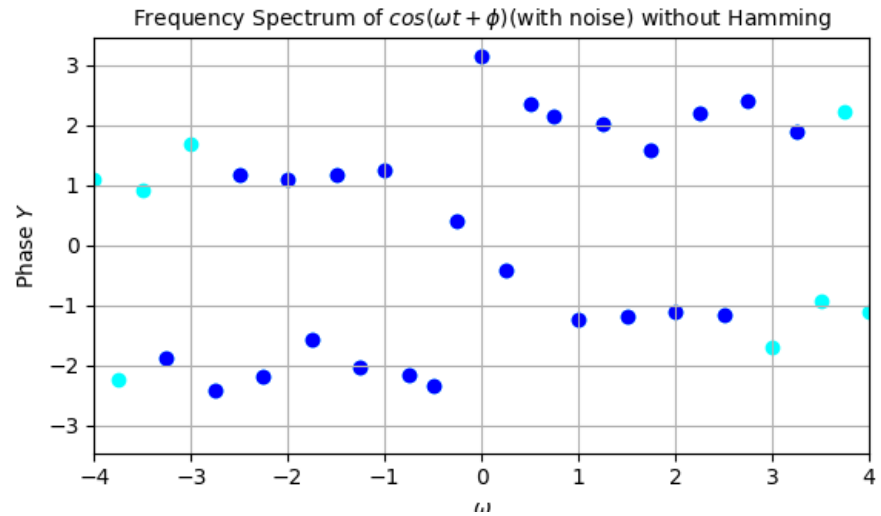
```
Estimations for cos(freq*t + phase) without hamming:
Estimated frequency: 0.5
Estimated phase: 2.3346428474376513

Estimations for cos(freq*t + phase) with hamming:
Estimated frequency: 0.5584304452185574
Estimated phase: 2.302385400626147
```

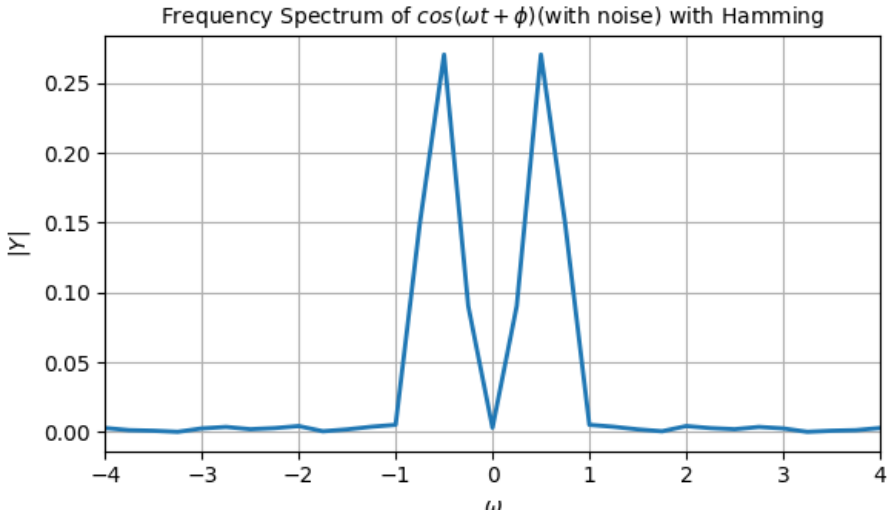
Analysis of Frequency Response of $cos(\omega t + \phi)$ with noise

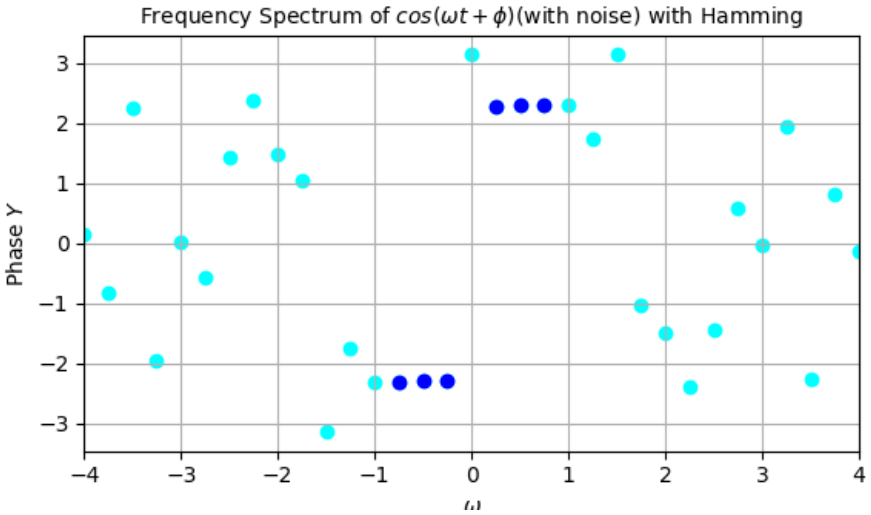
To the above signal $cos(\omega t + \phi)$, white gaussian noise is added - after which the frequency response is again observed.

```
def noisycos(x):
    return cos(freq*x + delta) + 0.1 * randn(len(x))
```


Frequency Spectrum of $cos(\omega t + \phi)$(with noise) without Hamming
|Y|
0.5
0.4
0.3
0.2
0.1
0.0
−4
−3
−2
−1
0
1
2
3
4


Frequency Spectrum of $cos(\omega t + \phi)$(with noise) without Hamming
Phase Y
3
2
1
0
−1
−2
−3
−4
−3
−2
−1
0
1
2
3
4


Frequency Spectrum of $cos(\omega t + \phi)$(with noise) with Hamming
|Y|
0.25
0.20
0.15
0.10
0.05
0.00
−4
−3
−2
−1
0
1
2
3
4



Similarly we try to estimate the frequency and phase by calculating weighted mean.

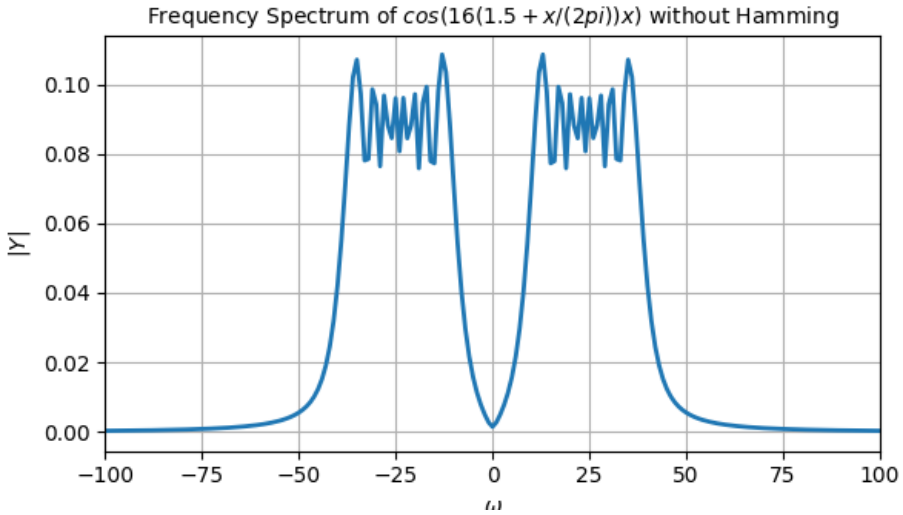
```
Estimations for cos(freq*t + phase) with noise without hamming:
Estimated frequency: 0.5
Estimated phase: 2.342789634255803

Estimations for cos(freq*t + phase) with noise with hamming:
Estimated frequency: 0.5585048423930653
Estimated phase: 2.294496634786052
```

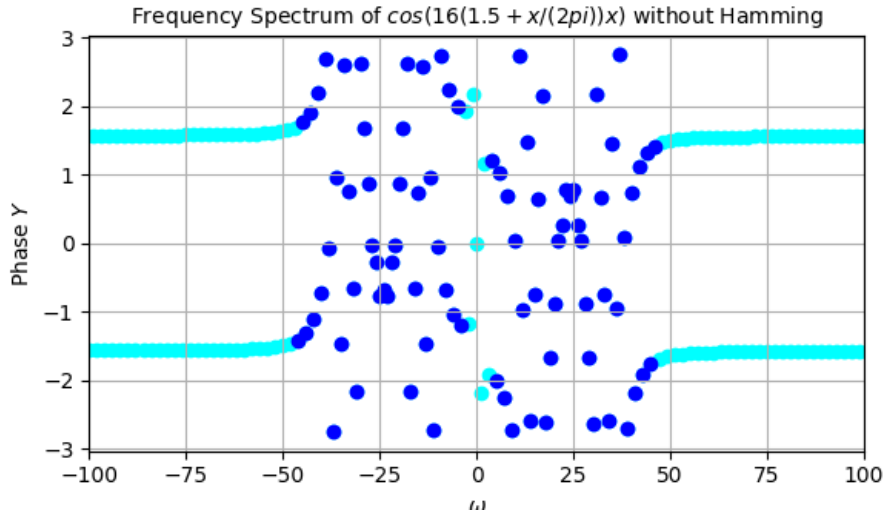
Frequency Spectrum of Chirped Signal $cos(16(1.5 + \frac{t}{2\pi})t)$

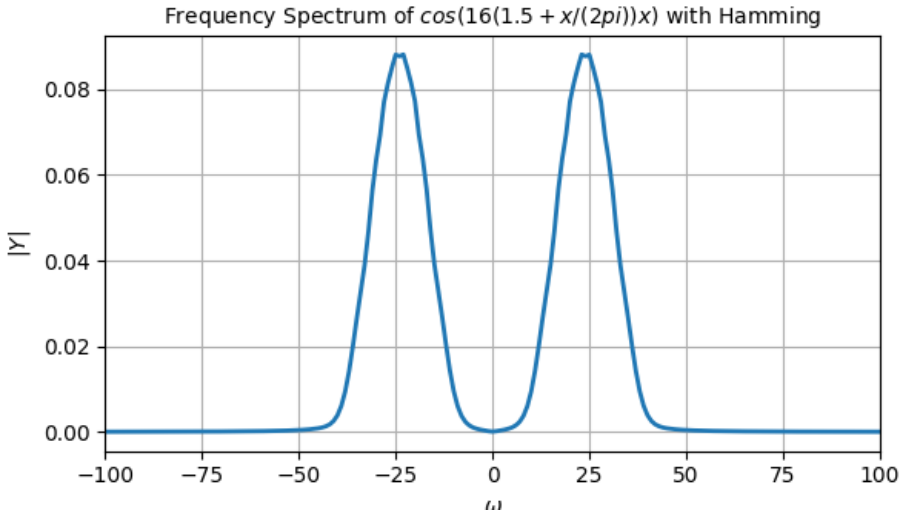
This signal $cos(16(1.5+\frac{t}{2\pi})t)$ is known as a **chirped signal**, since its frequency continuously changes from 16 to 32rad/s

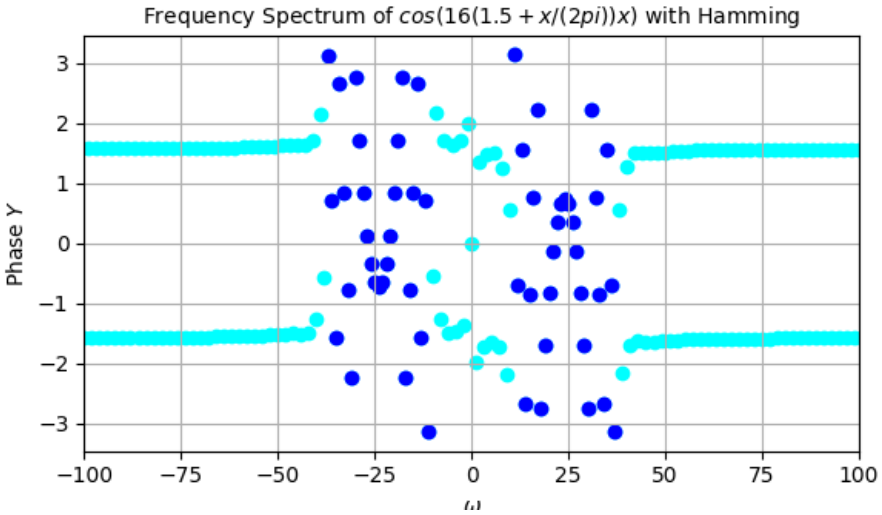
```
def chirp(x):
    return cos(16*(1.5 + x/(2*pi))*x)


def hammingchirp(x):
    return chirp(x)*hammingFunction(arange(len(x)))
```

we try to plot the DFT of this function from $t = -\pi$ to π in 1024 steps.


Frequency Spectrum of $cos(16(1.5 + x/(2pi))x)$ without Hamming
|Y|
0.10
0.08
0.06
0.04
0.02
0.00
−100
−75
−50
−25
0
25
50
75
100


Frequency Spectrum of $cos(16(1.5 + x/(2pi))x)$ without Hamming
Phase Y
3
2
1
0
−1
−2
−3
−100
−75
−50
−25
0
25
50
75
100


Frequency Spectrum of $cos(16(1.5 + x/(2pi))x)$ with Hamming
|Y|
0.08
0.06
0.04
0.02
0.00
−100
−75
−50
−25
0
25
50
75
100



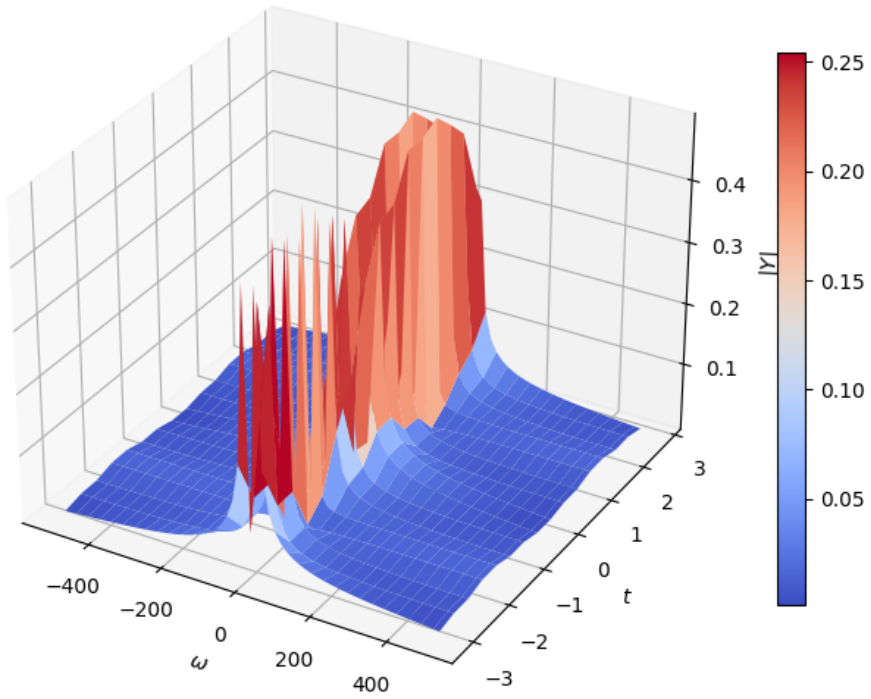
As expected the frequency of operation ranges from 16 rad/s to 32 rad/s. One can observe significant changes between the graphs with and without hamming. This is because the discontinuities have been removed to a great extend. The major frequencies are easily isolated and identifiable.

The same 1024 vector signal is broken into 16 contiguous pieces of 64 samples each to obtain the **Time Frequency Plot**. The DFT of each piece is obtained and stored in a 2D array. Each piece correspond to a different time interval. Plotting a surface plot with magnitude as a function of time and frequency using

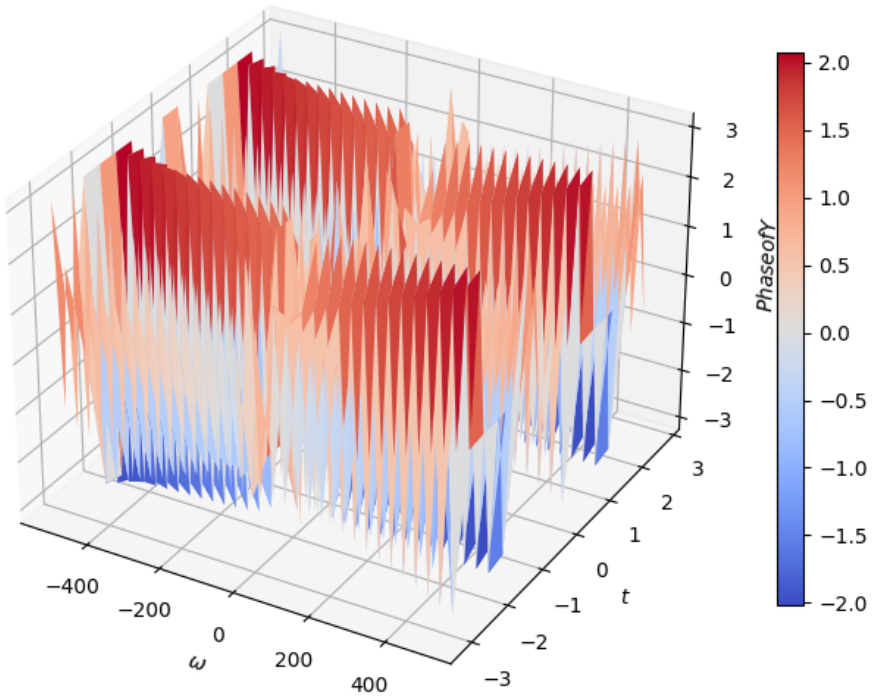
```
w = linspace(-512, 512, 65)[:-1]
t = linspace(-pi, pi, 1025)[:-1]
t = reshape(t, (16, 64))
mag = []
phase = []

for time in t:
    y = chirp(time)
    y[0] = 0
    Y = fftshift(fft(fftshift(y)))/64
    mag.append(abs(Y))
    phase.append(angle(Y))

mag = array(mag)
phase = array(phase)
```

If one closely observes, the gap between peaks increases with time

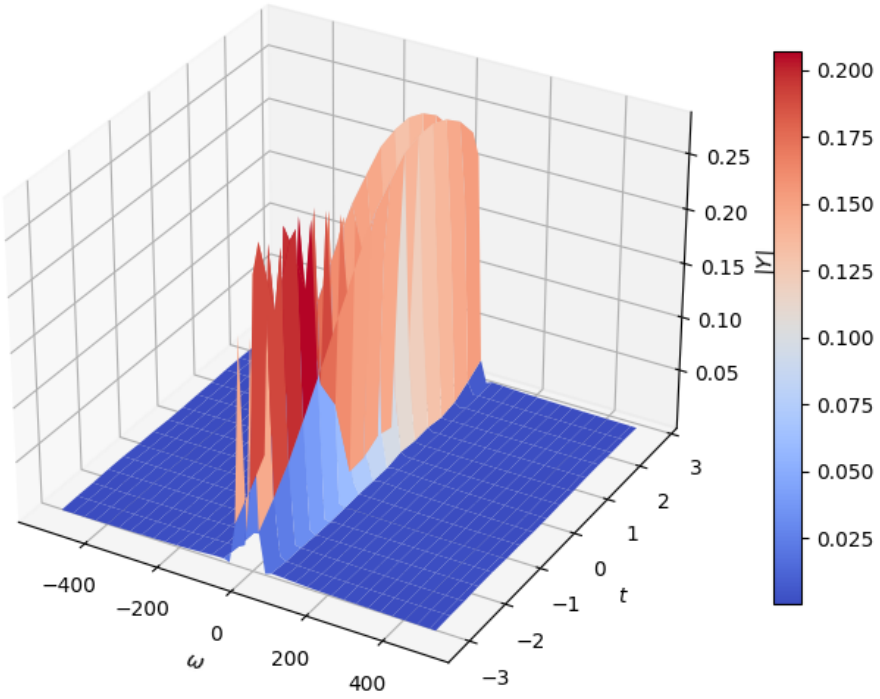
Surface plot of Magnitude response vs. frequency and time



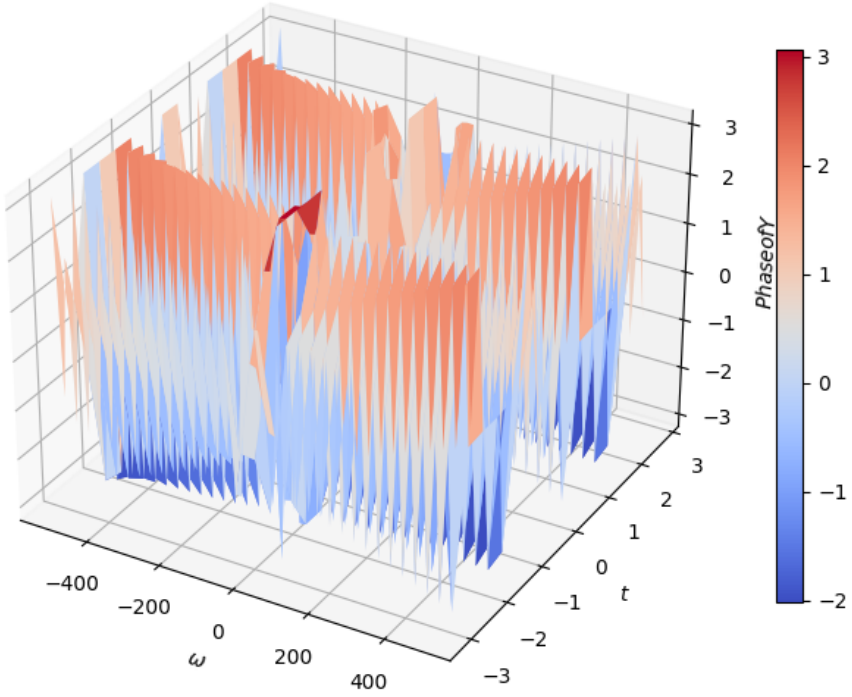
Surface plot of Phase response vs. frequency and time



Surface plot of Magnitude response vs. frequency and time with hamming



Surface plot of Phase response vs. frequency and time with hamming



Conclusion

Thus the frequency spectrum of non-periodic signals were analyzed and plotted. We observed how the discontinuities cause Gibbs Phenomenon and how it affects our DFT. By Windowing using Hamming technique, the effects of the discontinuities were reduced and well defined peaks were generated. We extracted the frequency and phase present in the signal by obtaining the weighted mean of the magnitudes. We also observed the time-frequency plot of a chirped signal to understand the time variation of DFT of a signal whose frequency continuously changes