EE2703: Applied Programming Lab
Assignment 9
Spectra of Non-Periodic Signals

Kaushik Ravibaskar
EE20B057

April 17, 2022

Contents

1 Aim

The assignment mainly deals with the following points:

- To obtain the DFT spectra of **non-periodic** signals by choosing the appropriate portion of the signal that will be replicated over time.
- To understand the use of **Hamming window**, which helps us to get rid of the **Gibbs phenomenon** in the signal, and some of its limitations.
- To retrieve certain features of the signal, such as frequency and phase, from the DFT of the signal.
- To plot the spectra obtained in various problems (in 2D and 3D) and explain the nature of the plots.

2 Introduction

In this assignment, we will deal with obtaining the **Digital Fourier Transform** of **non-periodic** signals. In the previous assignment, while finding the **DFT** of **periodic** signal, our approach was as follows:

- Sample the signal so that **Nyquist Criterion** is met, and so that Δf is small enough. Generate the frequency axis from $-\frac{f_{max}}{2}$ to $+\frac{f_{max}}{2}$ taking care to drop the last term.
- Ensure that the signal starts at $t = 0^+$ and ends at $t = 0^-$.
- Use 2^k samples and obtain the DFT. Rotate the samples so that they go from $f = -\frac{f_{max}}{2}$to $f = +\frac{f_{max}}{2} - \Delta f$.
- Plot the magnitude and phase of the spectrum and analyse the same.

In this assignment, for non-periodic signals, our approach would be as follows:

- Take a suitable interval of the signal and replicate it over the time axis i.e. make it periodic.
- Use an appropriate **Hamming Window** to negate the effect of **Gibbs Phenomenon** i.e. the effect of $\frac{1}{\omega}$ decay on the magnitude spectra.
- Follow all the procedure as was done for the periodic signals and obtain magnitude and phase plots. Analyse the same.

The above procedure will become more clear as we work on the problems in the upcoming sections.

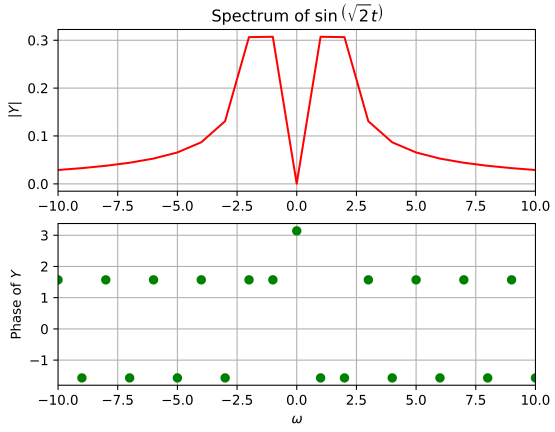
3 Assignment Problems

3.1 P1: Understanding the Spectrum of $\sin\sqrt{2}t$

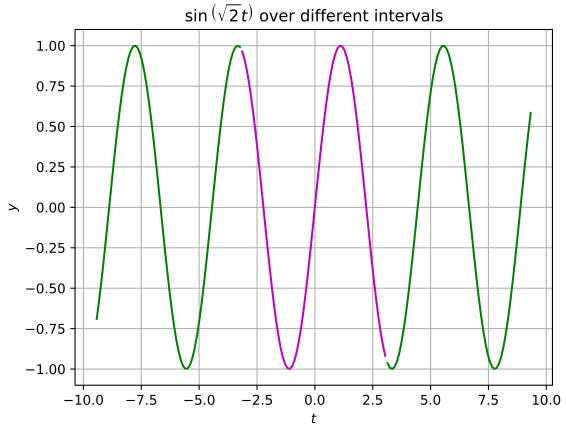
Let us analyse the given problems in the sheet. These problems will give us some idea on how to obtain an accurate DFT spectra of non-periodic signals.
Let us obtain the DFT spectra of $\sin\sqrt{2}t$ signal. This signal is non-periodic with respect to 2π time period. Taking the signal from $-\pi$ to π for sampling and computation, the piece of code that does the process is as follows:

```
t_1_1 =  py.linspace(-ma.pi, ma.pi, 65)[ :-1]
dt_1_1 = t_1_1[1] - t_1_1[0]
f_1_1 = 1/dt_1_1
y_1_1 = py.sin(ma.sqrt(2)*t_1_1)
y_1_1[0] = 0
y_1_1 = py.fftshift(y_1_1)
Y_1_1 = py.fftshift(py.fft(y_1_1))/64.0
w_1_1 = py.linspace(-ma.pi*f_1_1, ma.pi*f_1_1, 65)[ :-1]
```

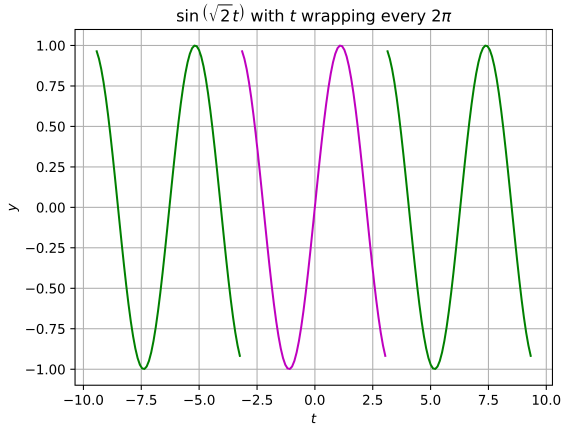
Note here that in the 5^{th} line of code, we have made $y_1 1[0] = 0$. This is done in order to ensure that **anti-symmetric** signals have a purely imaginary phase i.e. in order to avoid any phase ambiguity.
The plot of the DFT spectra is as follows:



We can observe from the plot that, instead of getting two spikes, we got two broad peaks with two finite values followed by a gradually decaying magnitude. In order to understand this ambiguity, let us look at the plot of $\sin\sqrt{2}t$ over different time intervals. The plot looks as shown below.
The **magenta** colored curve is the one whoch we are tryng to analyse and the **green** colored curve is the continuation of the signal. We can clearly



comment that, although the signal is periodic in general, it isnt periodic over $-\pi$ to π interval. The signal for which the DFT is getting computed by Python is as follows:

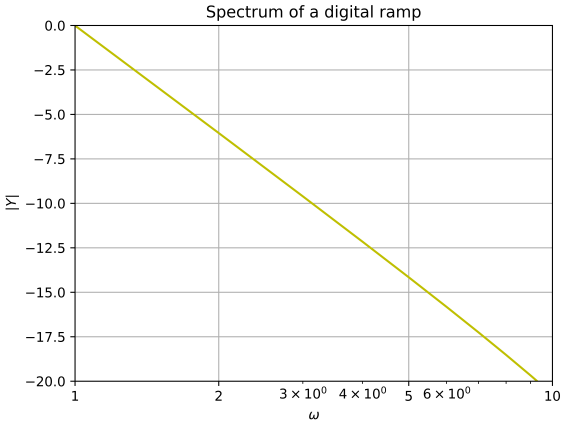


Clearly, this signal isnt $\sin\sqrt{2}t$ and hence we didn't get the expected spectrum plot. We can also observe that, there is an abrupt discontinuity in the signal at $t = n\pi$ points. This gives rise to **Gibbs Phenomenon**. Since our function is an odd function with a big jump, let us analyse the DFT of the ramp,

$$f(t) = t; -\pi < t < \pi$$

which is then repeated over the time axis.
The DFT spectra of the ramp signal looks as shown below.
This is expected as the fourier series of this forced-periodic ramp is as follows:

$$f(t) = 2(\frac{\sin t}{1} - \frac{\sin 2t}{2} + \frac{\sin 3t}{3} - \dots)$$



Hence the coefficients decay slowly in $\frac{1}{\omega}$ fashion. 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 $\frac{1}{\omega}$ as can be observed from the above plot. In order to nullify the effect of $\frac{1}{\omega}$ decay in the magnitude spectra, we can do the process of **Windowing**. What this means is that we damp the function near the periodic interval, where spike occurs by multiplying the sequence with a window sequence $w[n]$.

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

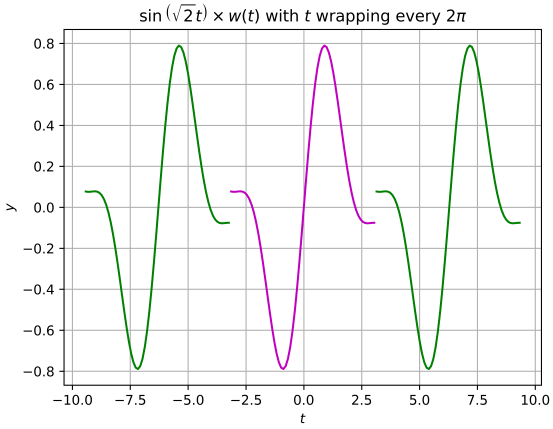
The new spectrum is got by convolving the two fourier transforms:

$$G_k = \sum_{n=0}^{N-1} F_n W_{k-n}$$

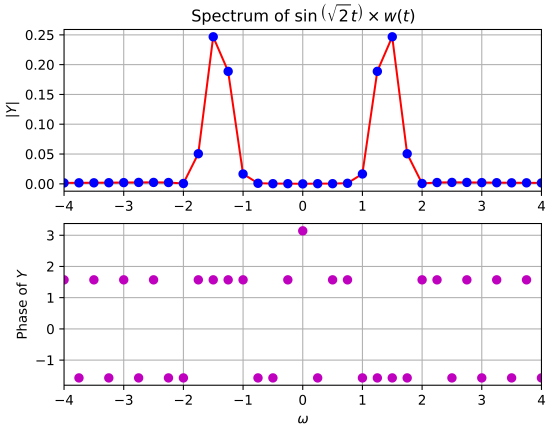
Suppose f_n is a sinusoid as we have taken. Then F_k has two spikes but the two spikes are now smeared out by W_k. So we expect to get broader peaks. But what this also does is to suppress the jump at the edge of the window. The window we will use is called the **Hamming Window**.

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

Let us now look at our time sequence of $\sin \sqrt{2}t$ windowed using the Hamming Window. The plot looks as follows:



The jump is still there, but it is much reduced. There is a small trick in keeping some of the jump, **it gives us an extra 10 db of suppression!**. Now let us take the DFT of this sequence (we have increased the time stretch and the number of samples to get a clearer and better plot). The plot of the DFT spectra looks as follows:



The reason for not getting a single spike on both sides, even after doing windowing, is because of $w[n]$. Multiplication in time is convolution in frequency and vice versa. So by multiplying with $w(t)$, we got rid of the $\frac{1}{\omega}$ 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. **Note that it is not because $\sqrt{2}$ is between** 1.25 **and** 1.5.

3.2 P2: Analysing the Spectrum of $\cos^3 \omega_o t$

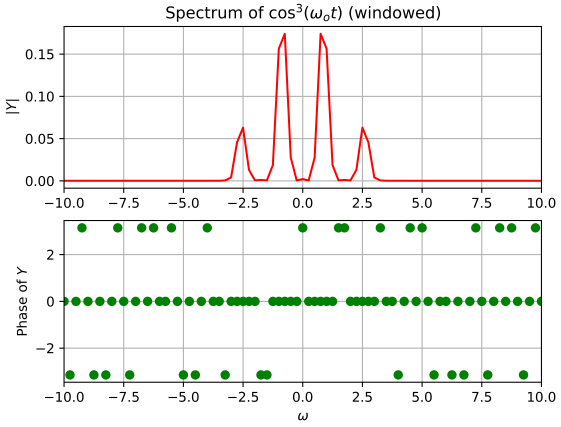
Let us now analyse the spectrum of $\cos^3 \omega_o t$ for $\omega_o = 0.86$. We will analyse the spectra in two cases, one with the **Hamming Window** and other without the same.

3.2.1 With the Hamming Window

The piece of code that gives the DFT spectra of the above function which is windowed with the Hamming Window is as follows:

```
w_o_2 = 0.86
t_2_2 = py.linspace(-4*ma.pi, 4*ma.pi, 257)[ :-1]
dt_2_2 = t_2_2[1] - t_2_2[0]
f_2_2 = 1/dt_2_2
y_2_2 = (py.cos(w_o_2*t_2_2))**3
n_2_2 = py.arange(256)
wnd_2_2 = py.fftshift(0.54+0.46*py.cos(2*ma.pi*n_2_2/256))
y_2_2 = y_2_2*wnd_2_2
y_2_2[0] = 0
y_2_2 = py.fftshift(y_2_2)
Y_2_2 = py.fftshift(py.fft(y_2_2))/256.0
w_2_2 = py.linspace(-ma.pi*f_2_2, ma.pi*f_2_2, 257)[ :-1]
```

The plot for the spectra is as follows:

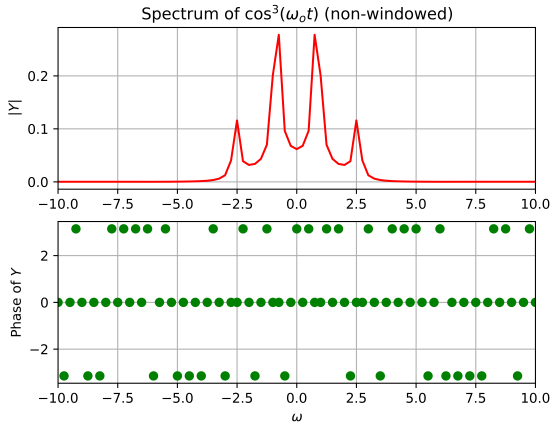


3.2.2 Without the Hamming Window

The piece of code that gives the DFT spectra of the above function which isn't windowed with the Hamming Window is as follows:

```
w_o_2 = 0.86
t_2_1 = py.linspace(-4*ma.pi, 4*ma.pi, 257)[ :-1]
dt_2_1 = t_2_1[1] - t_2_1[0]
f_2_1 = 1/dt_2_1
y_2_1 = (py.cos(w_o_2*t_2_1))**3
y_2_1[0] = 0
y_2_1 = py.fftshift(y_2_1)
Y_2_1 = py.fftshift(py.fft(y_2_1))/256.0
w_2_1 = py.linspace(-ma.pi*f_2_1, ma.pi*f_2_1, 257)[ :-1]
```

The plot for the spectra is as follows:



Thus we can clearly observe that, in the magnitude plot, for the windowed version, the spike looks sharper than the non-windowed case.

3.3 P3: Analysing the Spectrum of $\cos(\omega_o t + \delta)$

Let us now look at the signal $\cos(\omega_o t + \delta)$. We want to write a program that will take a 128 element vector known to contain the signal for $0.5 < \omega_o < 1.5$ and for some arbitrary δ. The values of t go from $-\pi$ to π. We want to obtain its DFT spectra (with windowing) and estimate ω_o and δ from the spectra.
Firstly, the piece of code that does the computation is as follows:

```
w_o_3 = rand.uniform(0.51, 1.5)
delta_3 = rand.uniform(-ma.pi, ma.pi)

t_3 = py.linspace(-ma.pi, ma.pi, 129)[ :-1]
dt_3 = t_3[1] - t_3[0]
```

```
f_3 = 1/dt_3
y_3 = py.cos(w_o_3*t_3 + delta_3)
n_3 = py.arange(128)
wnd_3 = py.fftshift(0.54+0.46*py.cos(2*ma.pi*n_3/128))
y_3 = y_3*wnd_3
y_3[0] = 0
y_3 = py.fftshift(y_3)
Y_3 = py.fftshift(py.fft(y_3))/128.0
w_3 = py.linspace(-ma.pi*f_3, ma.pi*f_3, 129)[ :-1]
```

Now to estimate the value of ω_o from the spectra, we can do the **Weighted Average** approach since the ω_o will correspond to the maximally generated value of the DFT. The piece of code that does the computation is as follows:

```
temp_3 = py.where(w_3 > 0)
w_est_3 = py.sum(abs(Y_3[temp_3])**3*w_3[temp_3])/py.sum(abs(Y_3[temp_3])**3)
print('The estimated value of w_o is: {}'.format(w_est_3))
```

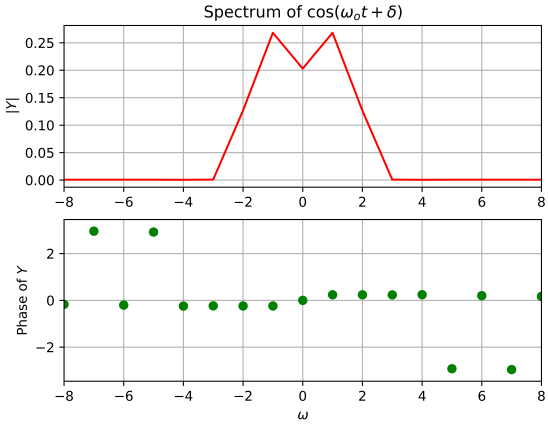
To obtain the value of δ, we can first find the indices of the DFT for which its magnitude is greater than $3e-3$ (can be taken as any value in that order). Then we can take the average of the angles of the first two samples which satisfies above condition. The piece of code that does the computation is as follows:

```
temp_3 = py.where(py.logical_and(abs(Y_3)>3e-3, w_3>0))[0]
delta = (py.angle(Y_3[temp_3[0]]) + py.angle(Y_3[temp_3[1]]))/2
print('The estimated value of delta is: {}'.format(delta))
```

The results that were obtained for some randomly generated ω_o and δ satisfying above conditions is as follows:

```
The value of true w_o is: 1.0518598558379746
The value of true delta is: 0.23847726729033836
The estimated value of w_o is: 1.0951385886729235
The estimated value of delta is: 0.2392406037955216
```

Hence we can conclude that our approach is a decent one as we are able to get handsome estimates. The plot for the above spectra is as follows:



3.4 P4: Analysing the Spectrum of $\cos(\omega_o t + \delta)$ with White Gaussian Noise

Let us now analyse the $\cos(\omega_o t + \delta)$ signal which has **White Gaussian Noise** added to it. White Gaussian noise consists of randomly generated **normalised** values which are scaled up by some constant amplitude. We can generate it as $0.1 \times randn(N)$, where 0.1 is amplitude and N denotes the number of samples. This noise is added to the above cosine signal.
The piece of code that does the computation is as follows:

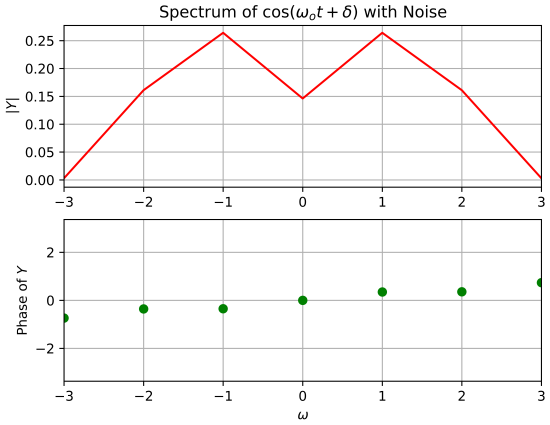
```
w_o_4 = rand.uniform(0.51, 1.5)
delta_4 = rand.uniform(-ma.pi, ma.pi)


t_4 = py.linspace(-ma.pi, ma.pi, 129)[ :-1]
dt_4 = t_4[1] - t_4[0]
f_4 = 1/dt_4
n_4 = py.arange(128)
y_4 = py.cos(w_o_4*t_4 + delta_4) + 0.1*np.random.randn(128)
wnd_4 = py.fftshift(0.54+0.46*py.cos(2*ma.pi*n_4/128))
y_4 = y_4*wnd_4
y_4[0] = 0
y_4 = py.fftshift(y_4)
Y_4 = py.fftshift(py.fft(y_4))/128.0
w_4 = py.linspace(-ma.pi*f_4, ma.pi*f_4, 129)[ :-1]
```

By following the exact approach to find the estimated values of ω_o and δ done in the previous problem, we get the following results:

```
The value of true w_o is: 1.2091454321393296
The value of true delta is: 0.33066255081581986
```

```
The estimated value of w_o is: 1.2066053254076983
The estimated value of delta is: 0.35209478299038766
```

The plot for the above computed spectra is as follows:



3.5 P5: Analysing the Spectrum of Chirped Signal

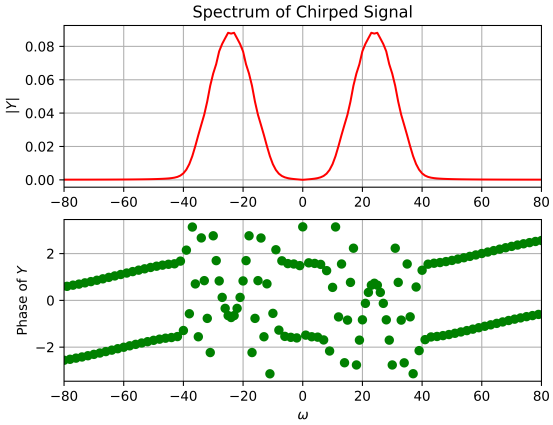
We are given a **Chirped Signal** as follows:

$$f(t) = \cos\left(16t(1.5 + \frac{t}{2\pi})\right)$$

Its frequency continuously changes from 16 to 32 radians per second as me move from $-\pi$ to π. The piece of code that computes its spectra (windowed) is as follows:

```
t_5 = py.linspace(-ma.pi, ma.pi, 1025)[ :-1]
dt_5 = t_5[1] - t_5[0]
f_5 = 1/dt_5
y_5 = py.cos(16*t_5*(1.5 + t_5/(2*ma.pi)))
n_5 = py.arange(1024)
wnd_5 = py.fftshift(0.54+0.46*py.cos(2*ma.pi*n_5/1024))
y_5 = y_5*wnd_5
y_5[0] = 0
y_5 = py.fftshift(y_5)
Y_5 = py.fftshift(py.fft(y_5))/1024.0
w_5 = py.linspace(-ma.pi*f_5, ma.pi*f_5, 1025)[ :-1]
```

The plot for the same is as follows:



3.6 P6: Obtaining Time-Frequency Plot of the Chirped Signal

Let us now obtain the **Time-Frequency** plot of the above signal. Following protocol must be followed to obtain the 2D array, which can be used to obtain the 3D plots:

- Break the 1024-size t vector into pieces that are 64 samples wide.
- Extract the DFT corresponding to each piece and store it as a column in a 2D array.
- Plot the array as a surface plot to show how the frequency of the signal varies with time.

The piece of code that does the computation to obtain 2D array corresponding to magnitude and phase of the DFT spectra is as follows:

```
t_6 = py.linspace(-ma.pi, ma.pi, 1025)[ :-1]
t_6_array = py.split(t_6, 16)

#initialising the 2D magnitude and phase arrays
Y_6_mag = py.zeros((16,64))
Y_6_angles = py.zeros((16,64))

#computing the DFT and stroing it in the 2D array
for i in range(len(t_6_array)):
    t_6_temp = t_6_array[i]
    dt_6 = t_6_temp[1] - t_6_temp[0]
    f_6 = 1/dt_6
    y_6 = py.cos(16*t_6_temp*(1.5 + t_6_temp/(2*ma.pi)))
```

```
n_6 = py.arange(64)
wnd_6 = py.fftshift(0.54+0.46*py.cos(2*ma.pi*n_6/64))
y_6 = y_6*wnd_6
y_6[0] = 0
y_6 = py.fftshift(y_6)
Y_6 = py.fftshift(py.fft(y_6))/64.0
Y_6_mag[i] =  abs(Y_6)
Y_6_angles[i] =  py.angle(Y_6)
```

The piece of code that does the surface plots is as follows:

```
fig12 = py.figure(12)
t_axis = py.linspace(-np.pi,np.pi,1025)[ :-1]
f_6 = 1/(t_axis[1] - t_axis[0])
t_axis = t_axis[::64]
w_axis = py.linspace(-f_6*ma.pi, f_6*np.pi, 65)[ :-1]
(T_axis, W_axis) = py.meshgrid(t_axis, w_axis)
ax = p3.Axes3D(fig12)
surf_6_1 = ax.plot_surface(W_axis, T_axis, Y_6_mag.T,
rstride=1, cstride=1, cmap = py.cm.jet)
fig12.colorbar(surf_6_1, shrink=0.5, aspect=5)
py.ylabel("Time")
py.xlabel("Frequency")
py.title('Magnitude Surface Plot (Chirped Signal)')
py.savefig("12.png", dpi = 1000)

fig13 = py.figure(13)
ax = p3.Axes3D(fig13)
surf_6_2 = ax.plot_surface(W_axis, T_axis, Y_6_angles.T)
fig13.colorbar(surf_6_2, shrink=0.5, aspect=5)
py.ylabel("Time")
py.xlabel("Frequency")
py.title('Phase Surface Plot (Chirped Signal)')
py.savefig("13.png", dpi = 1000)
```

The 3D surface plots look as follows:

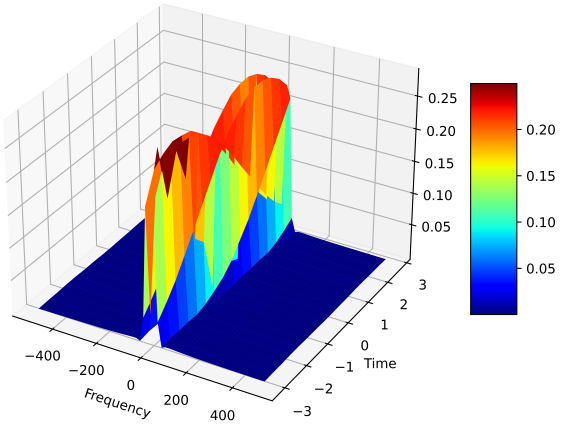


Figure 1: Magnitude Surface Plot

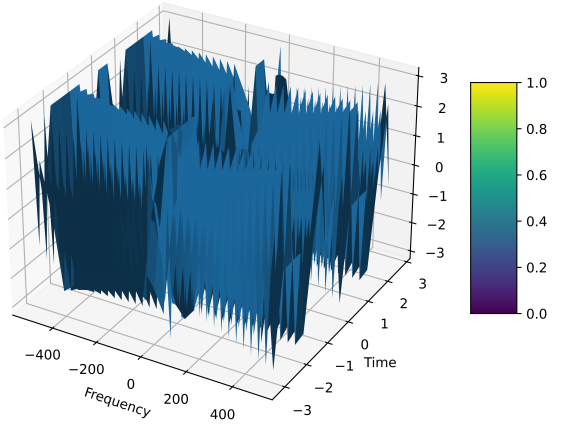


Figure 2: Phase Surface Plot

4 Conclusion

We can conclude that it is possible to obtain the DFT spectra for non-periodic signal by using suitable windowing signals like the **Hamming Window**. Using the window, we get sharper and clearer DFT spectrum. Also by using approaches like **Weighted Average**, we can obtain certain features of the signal such as its frequency and phase. Lastly, we can obtain surface plots, by which we can analyse localised DFTs and can show how the spectra evolves with time.