EE2703 : Applied Programming Lab
Assignment 9
Spectra of non-periodic signals

Debojyoti Mazumdar
EE20B030

May 20, 2022

0.1 Abstract

Aim of this assignment is to know how to get the DFTs of non-periodic functions by windowing and also understand the concept of gibbs phenomenon. We will also learn how to plot a time-frequency plot of the DFT of a signal.

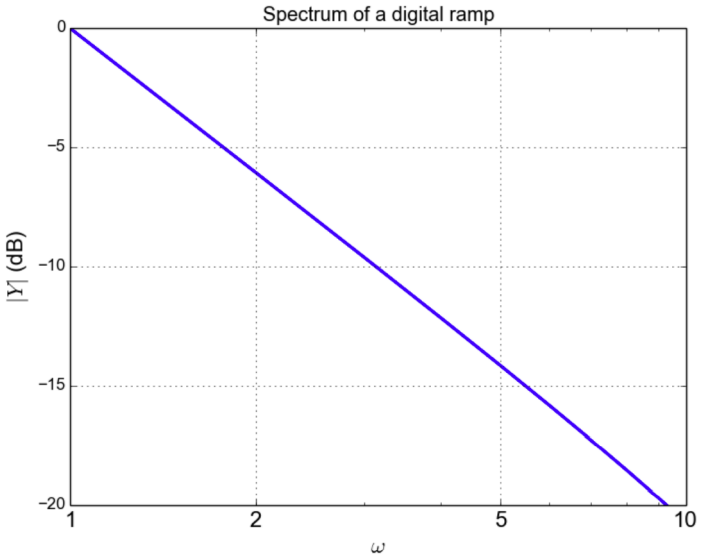
0.2 Introduction

Before we start we have to understand gibbs phenomenon and how can we increase the accuracy of the spectrum of non-periodic signals.

`Gibbs Phenomenon:`
Let us consider the ramp function which has big jumps at $n\pi$.

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

The spectrum of the digital ramp is shown below.



We can see that the spectrum of the ramp function decays slowly(20dB per decay), which corresponds to $1/\omega$. The big jumps at $n\pi$ force this slowly decaying spectrum. This is gibbs phenomenon.

To prevent this gibbs phenomenon from giving inaccurate results, we use a window function as follows.

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

so our new spectrum will be

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

We will be using the Hamming window where,

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

This window function helps in reducing the jump at the ends of the non-periodic signal.Thereby, reducing the inaccuracy due to gibbs phenomenon.

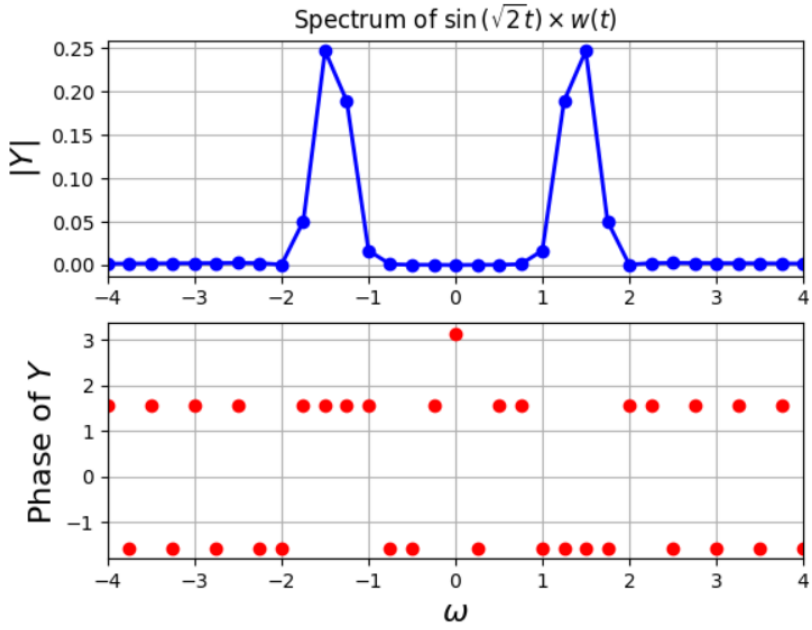
0.3 Assignment

0.3.1 Question 1

Here, we are going to plot the spectrum and the phase of the fourier transform of $sin(\sqrt{2}t)w(t)$. To do so we write the following code as given in the lab manual.

```
t = linspace(-4*pi, 4*pi, 257)
t = t[:-1]
dt = t[1]-t[0]
fmax = 1/dt
n = arange(256)
wnd = fftshift(0.54+0.46*cos(2*pi*n/256))
y = sin(sqrt(2)*t)
y = y*wnd
y[0] = 0  # the sample corresponding to -tmax should be set zeroo
y = fftshift(y)  # make y start with y(t=0)
Y = fftshift(fft(y))/256.0
w = linspace(-pi*fmax, pi*fmax, 257)
w = w[:-1]
figure()
subplot(2, 1, 1)
plot(w, abs(Y), 'b', w, abs(Y), 'bo', lw=2)
xlim([-4, 4])
ylabel(r"$|Y|$", size=16)
title(r"Spectrum of $\sin\left(\sqrt{2}t\right)\times w(t)$")
grid(True)
subplot(2, 1, 2)
plot(w, angle(Y), 'ro', lw=2)
xlim([-4, 4])
ylabel(r"Phase of $Y$", size=16)
xlabel(r"$\omega$", size=16)
grid(True)
show()
```

running the above code we get the following as the spectrum.

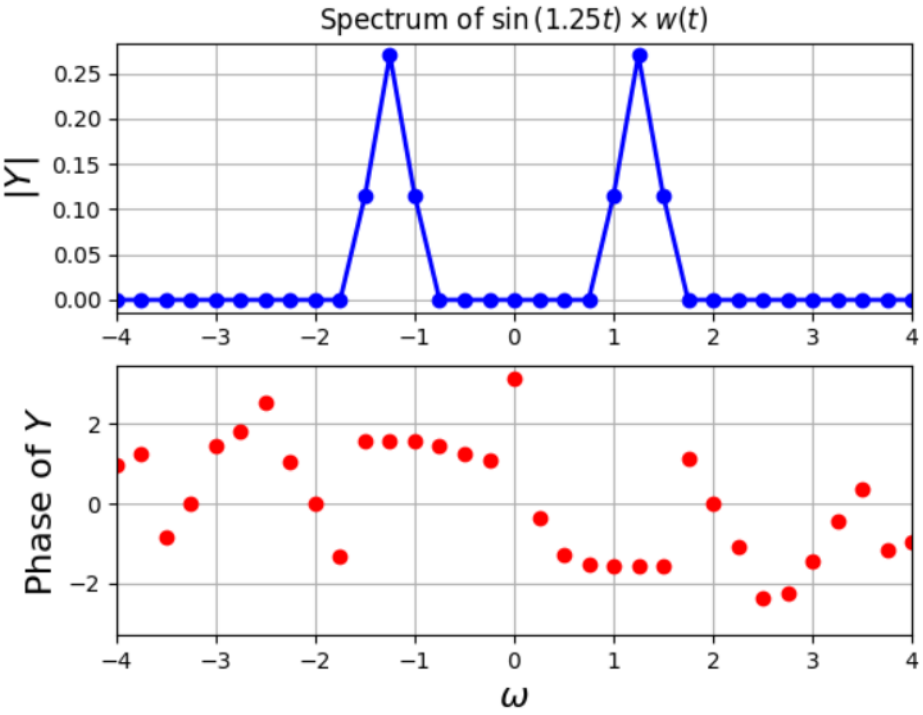


Now we plot the spectrum of $sin\,(1.25t)$.

To do so we just have to change one line in the above code as shown below.

```
y = sin(1.25*t)
```

running the code we get the following as the spectrum.



0.3.2 Question 2

Now we plot the spectrum of the fourier transform of $\cos^3(\omega_0 t)$ for $\omega_0 = 0.86$ with and without a Hamming window.

The below code written plots the spectrum without the hamming window.

```
t = linspace(-pi, pi, 65)
t = t[:-1]
dt = t[1]-t[0]
fmax = 1/dt
omega0 = 0.86
y = (cos(omega0*t))**3
y[0] = 0  # the sample corresponding to -tmax should be set zeroo
y = fftshift(y)  # make y start with y(t=0)
Y = fftshift(fft(y))/64.0
w = linspace(-pi*fmax, pi*fmax, 65)
w = w[:-1]
figure()
subplot(2, 1, 1)
plot(w, abs(Y), lw=2)
xlim([-10, 10])
ylabel(r"$|Y|$", size=16)
title(r"Spectrum of $\cos^{3}\left(\omega_{0}t\right)$")
grid(True)
subplot(2, 1, 2)
plot(w, angle(Y), 'ro', lw=2)
xlim([-10, 10])
ylabel(r"Phase of $Y$", size=16)
xlabel(r"$\omega$", size=16)
grid(True)
savefig("fig10-1.png")
show()
```

Running the above code generates the following plot.

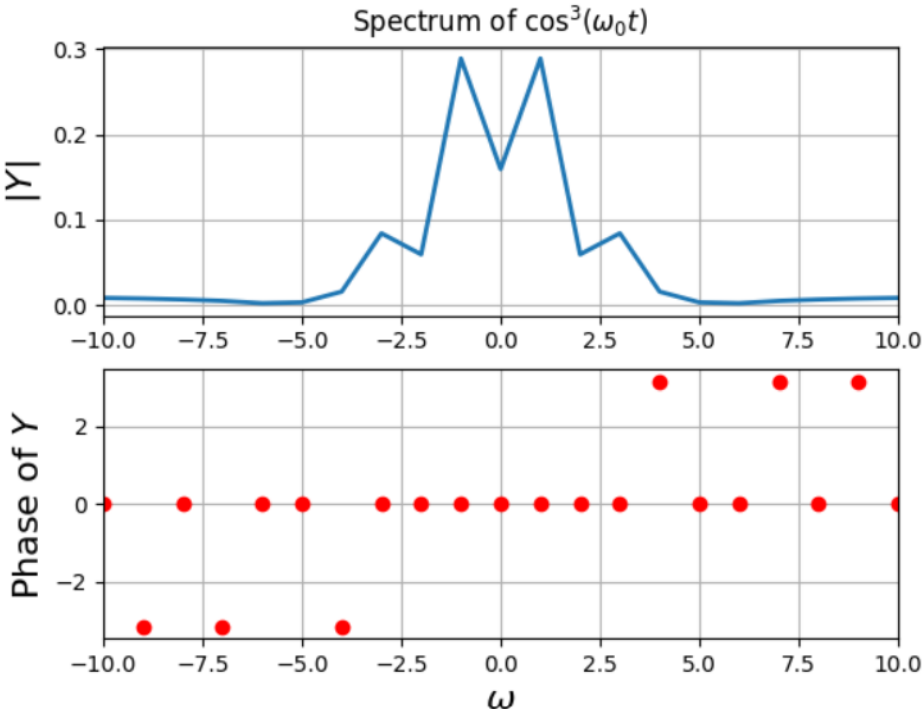


Figure 1: Spectrum of $\cos^3(\omega_0 t)$ without hamming window

To generate the hamming window , we add the following piece of code.

```
n = arange(64)
wnd = fftshift(0.54+0.46*cos(2*pi*n/63))
y = ((cos(omega0*t))**3)*wnd
```

Making the required changes and running the code we get the following as the spectrum.

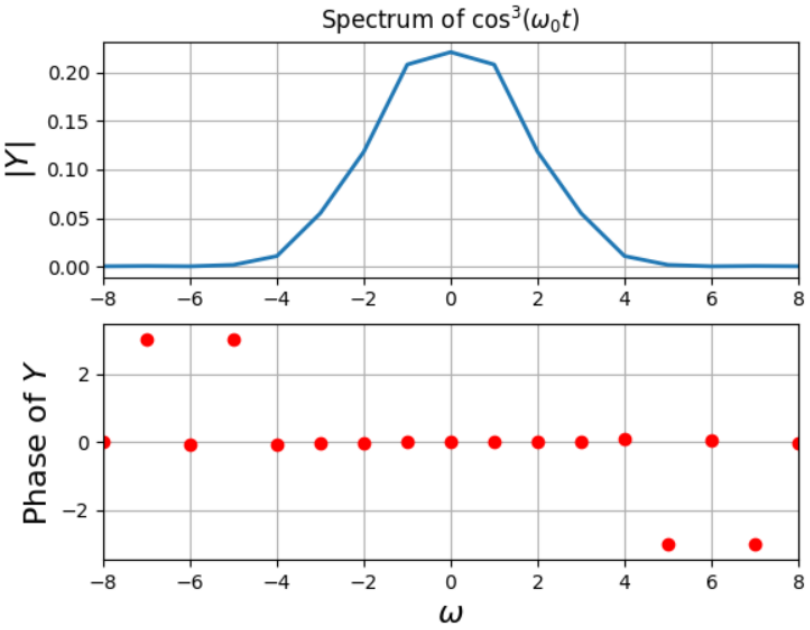


Figure 2: Spectrum of $\cos^3(\omega_0 t)$ with hamming window

0.3.3 Question 3

Now we will try to get the value of ω_0 and δ in a 128 element vector containing $\cos(\omega_0 t + \delta)$ using the digital spectrum of the signal.
We know that

$$\cos(\omega_0 + \delta) = \frac{e^{j(\omega_0 t+\delta)}}{2} + \frac{e^{-j(\omega_0 t+\delta)}}{2} = \frac{e^{j\delta}}{2}e^{j\omega_0 t} + \frac{e^{-j\delta}}{2}e^{-j\omega_0 t}$$

For estimating frequency,we find the ω which equals average of all by magnitude. Since the plot is symmetric about the origin, it is sufficient to consider only one half.

But due to inaccuracies of the DFT from windowing and a non-integral frequency, the tail of the magnitude plot does not fall fast enough and the resultant estimate is poor. Hence, we consider only the first 5 frequencies.

For finding the phase, we know the input frequency is between 0.5 and 1.5, the resolution of our spectrum is 1. The phase at ± 1 will give us δ. Since there can be inaccuracies due to non-integral frequency, we can take average of phases at each frequency weighted by magnitude.

So $\angle DFT(f, \omega_0) = \delta$.

To estimate the value of ω_0 we have the following code.

```
ii = where(w > 0)
omega = (sum(abs(Y[ii])**2*w[ii])/sum(abs(Y[ii])**2))  # weighted average
print("omega = ", omega)
```

Here, w is the 129 elements vector of ω from $-\pi f_{max}$ to πf_{max} .

To estimate the value of δ we have the following code.

```
ii_1 = np.where(np.logical_and(np.abs(Y) > 1e-4, w > 0))[0]
np.sort(ii_1)
points = ii_1[1:3]
# weighted average for first 2 points
print("delta = ", np.sum(np.angle(Y[points]))/len(points))
```

Here, Y is the fft of $y = \cos(\omega_0 t + \delta)$ signal.
The full code to plot as well as to estimate the values of ω_0 and δ is given below.

```
# generates cos(omega0t + delta)
def gen_cos(omega0, delta):
    t = linspace(-pi, pi, 129)
    t = t[:-1]
    y = cos((omega0*t) + delta)
    y[0] = 0

    return y
```

```
def omega0_and_delta_estimator(y, plotting=False):
    t = linspace(-pi, pi, 129)
    t = t[:-1]
    dt = t[1]-t[0]
    fmax = 1/dt

    # generating the window function
    n = arange(128)
    wnd = fftshift(0.54+0.46*cos(2*pi*n/127))
    y = y*wnd

    # generating Y
    y = fftshift(y)  # make y start with y(t=0)
    Y = fftshift(fft(y))/128.0
    w = linspace(-pi*fmax, pi*fmax, 129)
    w = w[:-1]

    # Plotting spectrum of y
    if plotting == True:
        figure()
        subplot(2, 1, 1)
        plot(w, abs(Y), lw=2)
        xlim([-8, 8])
        ylabel(r"$|Y|$", size=16)
        title(r"Spectrum of $\cos\left(\omega_{0}t + \delta\right)$")
        grid(True)
        subplot(2, 1, 2)
        plot(w, angle(Y), 'ro', lw=2)
        xlim([-8, 8])
        ylabel(r"Phase of $Y$", size=16)
        xlabel(r"$\omega$", size=16)
        grid(True)
        show()

    # Estimating omega
    ii = where(w > 0)
    omega = (sum(abs(Y[ii])**2*w[ii])/sum(abs(Y[ii])**2))  # weighted average
    print("omega = ", omega)

    # Estimating delta
    ii_1 = np.where(np.logical_and(np.abs(Y) > 1e-4, w > 0))[0]
    np.sort(ii_1)
    points = ii_1[1:3]
    # weighted average for first 2 points
    print("delta = ", np.sum(np.angle(Y[points]))/len(points))

def Q3(omega0, delta, plotting=False):
    y = gen_cos(omega0, delta)
```

```
print("omega and delta value without noise :\n")
omega0_and_delta_estimator(y, plotting)
```

The spectrum obtained for $\omega_0 = 1$ and $\delta = 0.5$ is shown below.

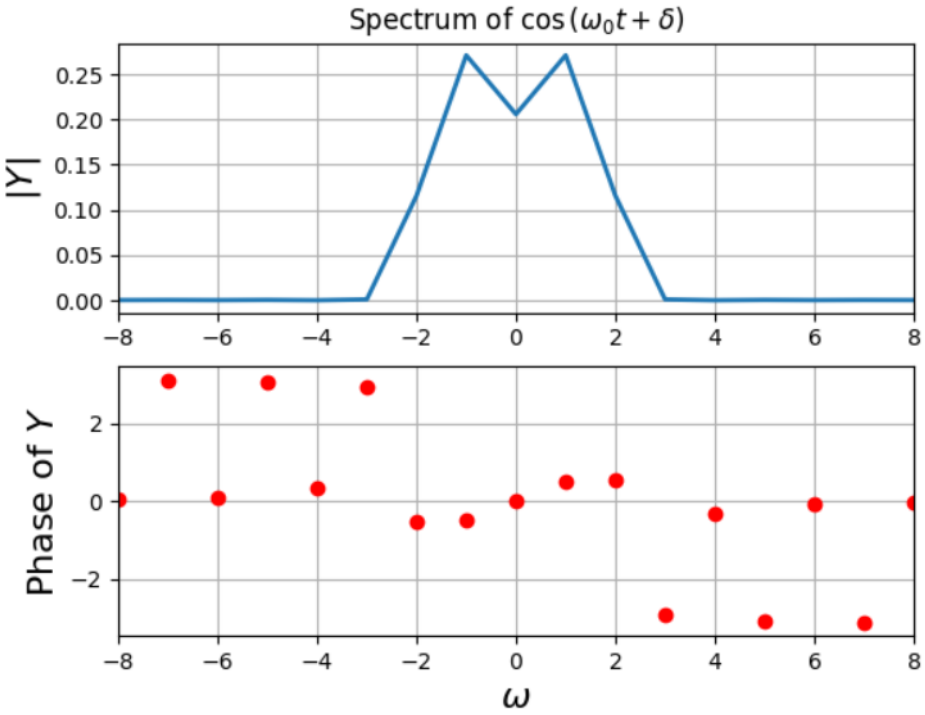


Figure 3: Spectrum of $\cos(t + 0.5)$ without noise

The values of ω_0 and δ obtained are shown below.

$$\omega_0 = 1.1611750390052884$$
$$\delta = -1.1960912944778668$$

0.3.4 Question 4

Now we add a white gaussian noise to $\cos(\omega_0 t + \delta)$.

To generate the white gaussian noise, we add the following code to the code already written for Question-3.

```
n = 0.1*randn(128)
y+=n
```

Running the above code we get the following spectrum.

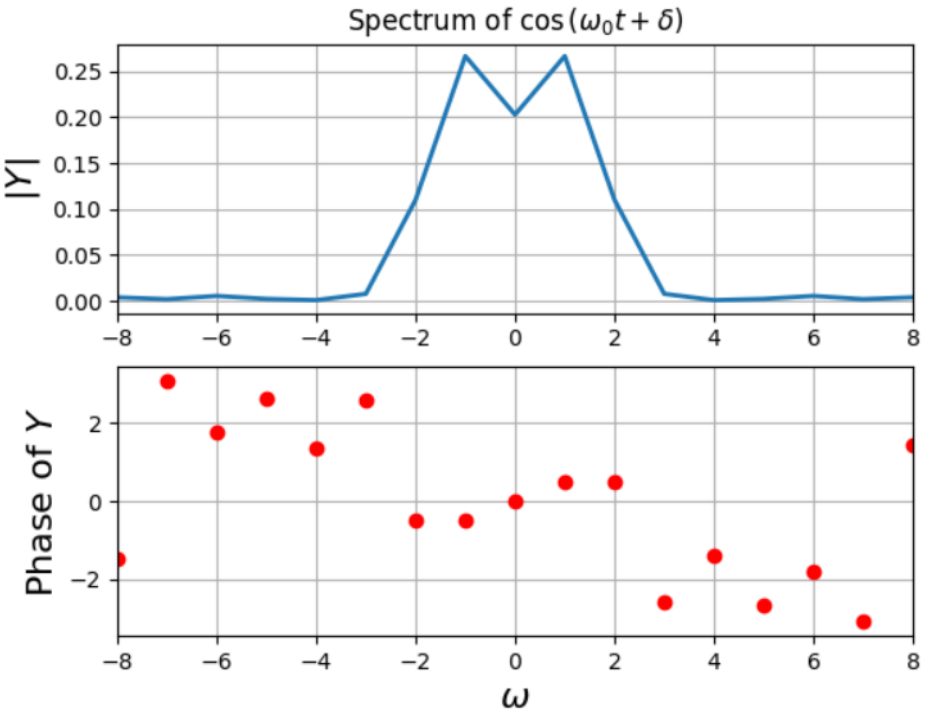


Figure 4: Spectrum of $\cos(t + 0.5)$ with noise

And the values of ω_0 and δ we get are:

$$\omega_0 = 1.845806189928219$$
$$\delta = -0.8945460695604764$$

0.3.5 Question 5

Now we plot the DFT of the chirped signal.

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

We write the following code to plot the DFT of the chirped signal.

```
def plotting_spectrum_of_chirp_signal(y, N):

    t = linspace(-pi, pi, N+1)
    t = t[:-1]
    dt = t[1]-t[0]
    fmax = 1/dt

    # generating the window function
    n = arange(N)
    wnd = fftshift(0.54+0.46*cos(2*pi*n/(N-1)))
    y = y*wnd

    # generating Y
```

```
    y = fftshift(y)  # make y start with y(t=0)
    Y = fftshift(fft(y))/N
    w = linspace(-pi*fmax, pi*fmax, N+1)
    w = w[:-1]

    # Plotting spectrum of y
    figure()
    subplot(2, 1, 1)
    plot(w, abs(Y), lw=2)
    xlim([-60, 60])
    ylabel(r"$|Y|$", size=16)
    title(r"Spectrum of $\cos\left(16t\left(1.5+\frac{t}{2\pi}\right)\right)$")
    grid(True)
    subplot(2, 1, 2)
    plot(w, angle(Y), 'ro', lw=2)
    xlim([-8, 8])
    ylabel(r"Phase of $Y$", size=16)
    xlabel(r"$\omega$", size=16)
    grid(True)
    show()

def Q5():
    N = 1024
    plotting_spectrum_of_chirp_signal(gen_chirp_signal(N), N)
```

Running the above code we get the following as the spectrum.

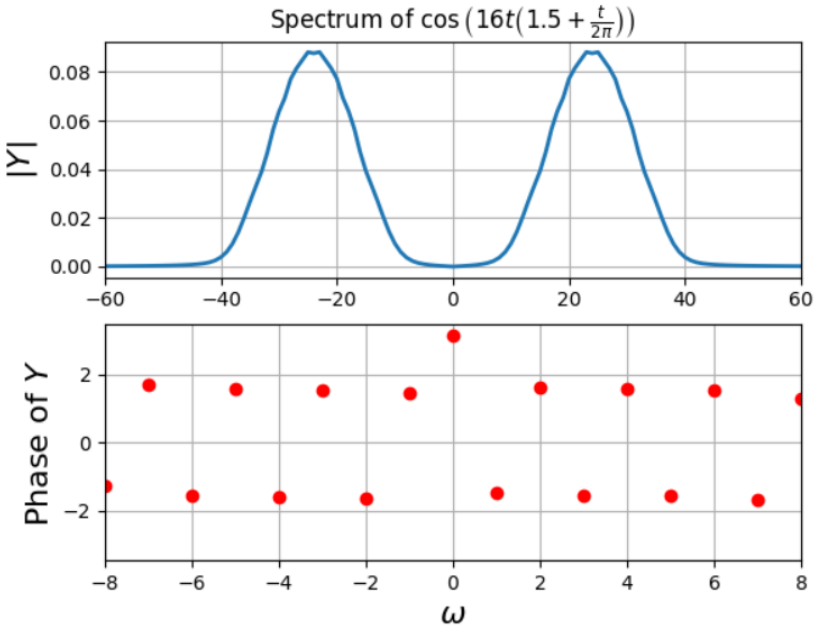


Figure 5: Spectrum of chirped signal

0.3.6 Question 6

Now we plot the time-frequency plot of the DFT of the chirped signal.

To plot the time-frequency plot we write the following code.

```
N = 1024
t = linspace(-pi, pi, N+1)
t = t[:-1]
dt = t[1]-t[0]
fmax = 1/dt

# splitting t into 64 samples wide
N_array = 64
t_array = split(t, (N/N_array))

# finding Y for each t_array
Y = zeros((16, 64), dtype=complex)
for i in range(len(t_array)):
    Y[i] = gen_Y_from_t(t_array[i])

# plotting "time-frequency" plot of Y
t = t[::N_array]
w = linspace(-pi*fmax, pi*fmax, N_array+1)
w = w[:-1]
t, w = meshgrid(t, w)

fig1 = figure()
ax = fig1.add_subplot(111, projection='3d')
surf = ax.plot_surface(w, t, abs(Y).T, cmap='viridis',
                       linewidth=0, antialiased=False)
fig1.colorbar(surf, shrink=0.5, aspect=5)
ax.set_title('Magnitude plot')
ylabel(r"$\omega\rightarrow$")
xlabel(r"$t\rightarrow$")
show()
```

Running the above code we get the following plots.

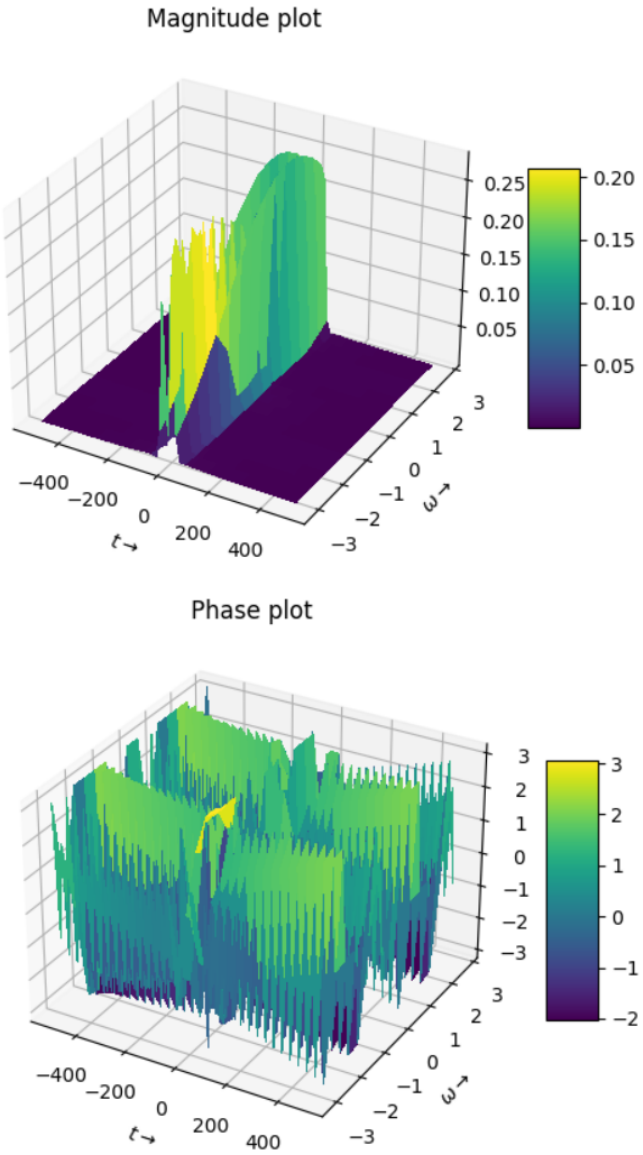


Figure 6: time-frequency plot of the DFT of chirped signal

0.4 Conclusions

Through this assignment we

- Understood how to generate the spectra of non-periodic signals.
- Understood the concept of Gibbs phenomenon.
- Understood how to reduce error due to gibbs phenomenon by using a window function.
- Understood how to estimate the value of ω_0 and δ of $\cos(\omega_0 t + \delta)$ using its spectrum.
- Understood how to plot a time-frequency plot of the DFT of a signal.