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

Devaganthan S S
EE19B018

July 7, 2021

0.1 Abstract

The assignment aims to understand how DFT can be done for non-periodic Signals, in python.

0.2 Introduction

For non-periodic Signals, using just the **fft()** function wouldn't get us the desired results. This is because the DFT Fourier analyses the replication of the -pi to pi part, which is not the same as the original function. This results in discontinuity, resulting in undesired results. Using Hamming's window, we correct this discontinuity to some extent.

0.3 Results and Implementation

0.3.1 Examples

The below function **spectrum()** plots the DFT of the given Signals.

```python
def spectrum(y, T, window, plot1, xlim1, label1):
    t = linspace(-T*pi, T*pi, len(y)+1)
    t = t[:-1]
    N = len(t)
    dt = t[1]-t[0]
    fmax = 1/dt
    n = arange(N)
    wnd = fftshift(0.54+0.46*cos(2*pi*n/N))
    if window:
        y = y*wnd
    y[0] = 0
    y = fftshift(y)
    Y = fftshift(fft(y))/N
    if plot1:
        w = linspace(-pi*fmax, pi*fmax, N+1)
        w = w[:-1]
        figure()
        subplot(2, 1, 1)
        plot(w, abs(Y), 'b', lw=2, label=label1)
        legend()
        xlim(-xlim1, xlim1)
        ylabel(r"$|Y|$", size=16)
        grid(True)
        subplot(2, 1, 2)
        plot(w, angle(Y), 'bo', lw=2, label=label1)
        ii = where(abs(Y) > 1e-3)
        plot(w[ii], angle(Y[ii]), 'ro', lw=1)
        xlim(-xlim1, xlim1)
        ylabel(r"Phase of $Y$", size=16)
        xlabel(r"$\omega$", size=16)
        grid(True)
        show()
    else:
        return Y[8:]


t = linspace(-4*pi, 4*pi, 257)
t = t[:-1]
y = sin(sqrt(2)*t)
spectrum(y, 4, True, True, 4,  'With Hamming Window')  # Plotting
    the Spectrum
```

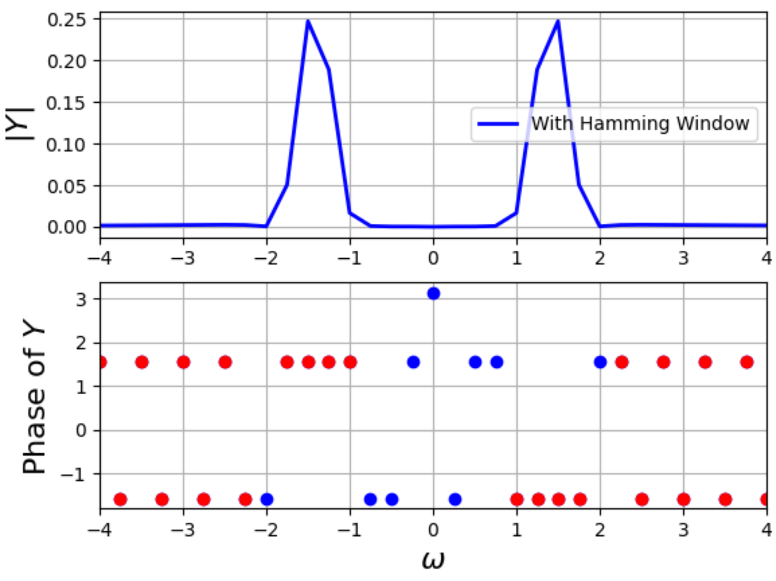
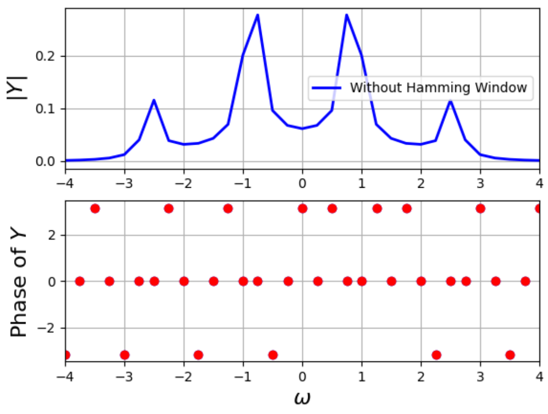


Figure 1: Spectrum of $\sin(\sqrt{2}t)$

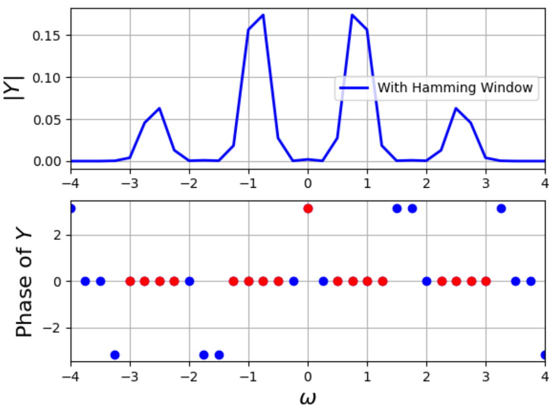
0.3.2 $\cos^3$(0.86t):

Using the function **spectrum()**, we plot the DFT of $\cos^3$(0.86t) with and without hamming window.

```
y = cos(0.86*t)**3
spectrum(y, 4, False, True, 4, 'Without Hamming Window')
spectrum(y, 4, True, True, 4,  'With Hamming Window')
```



(a) Without Hamming's Window



(b) With Hamming's Window

0.3.3 Estimating ω_o and δ

To find the estimate of ω_o, we take the weighted average of ω, with weights $|\mathrm{Y}(\omega)|^2$. For finding δ, we use the lstsq() function. We form the matrices from the below equation, For $\omega_o = 1.2$ and $\delta = 1$, the estimate we get is,

$$\omega_o = 1.1749137503896578 \tag{1}$$

$$\delta = 1.0056655986981278 \tag{2}$$

The below code accomplishes the above,

```python
def estimate(y):
    N = len(y)
    vec = y.copy()
    y = fftshift(y)
    t = linspace(-N * pi / 64, N * pi / 64, N + 1)
    t = t[:-1]
    dt = t[1] - t[0]
    fmax = 1 / dt
    y[0] = 0
    Y = fftshift(fft(fftshift(y))) / float(N)
    w = linspace(-pi * fmax, pi * fmax, N + 1)
    w = w[:-1]
    i = intersect1d(where(w >= 0), where(abs(Y) > 0.1))
    freq = sum(w[i] * abs(Y[i]**2)) / sum(abs(Y[i]**2))
    A, B = lstsq(c_[cos(freq * t), sin(freq * t)], vec)[0]
    delta = arctan2(-B, A)
    while delta < 0:
        delta += pi
    while delta > pi:
        delta -= pi
    return freq, delta


t = linspace(-2 * pi, 2 * pi, 129)
t = t[:-1]
print('Estimate without Noise ', end='')
print(estimate(cos(1.2*t+1)))
```

0.3.4 With the addition of Noise

Undergoing the same process as before, the estimate ofω_o and δ, with added Noise to the signals is,

$$\omega_o = 1.09349537748562952 \tag{3}$$

$$\delta = 0.9903572478888653 \tag{4}$$

0.3.5 Chirped Signal

The signal $cos(16(1.5+\frac{t}{2\pi})t)$ is given as input in 1024 steps. Using the **spectrum()** function, we plot the spectrum of the signal.

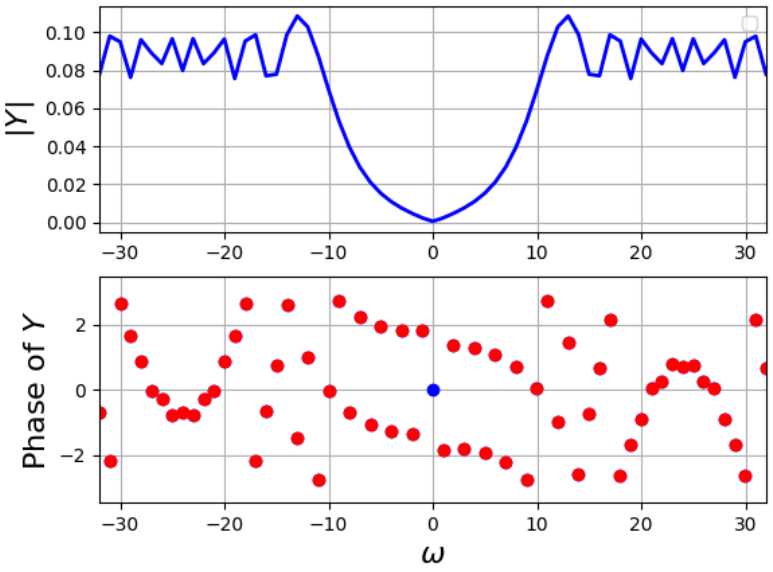


Figure 3: Spectrum of $cos(16(1.5+\frac{t}{2\pi})t)$

0.3.6 Time-Frequency plot for the Chirped signal

For the same chirped signal, we break the 1024 vector into pieces that are 64 samples wide. Extract the DFT of each and store it as a column in a 2D array. Then plot the array as a surface plot to show how the signal's frequency varies with time. The below function accomplishes the above task.

```python
w = linspace(0, pi * fmax, 9)[:-1]
Y, X = meshgrid(w, linspace(-pi, pi, 65)[:-1])

Freq = []
for x in array_split(t, 64):
    y = cos(24 * x + 8 * x * x / pi)
    Freq = Freq + [spectrum(y, 1, False, False, None, None)]

Freq = array(Freq)
ax = p3.Axes3D(figure(4))
surf = ax.plot_surface(Y, X, abs(Freq), rstride=1, cstride=1,
                       cmap=cm.jet, linewidth=0, antialiased=True)
show()
```

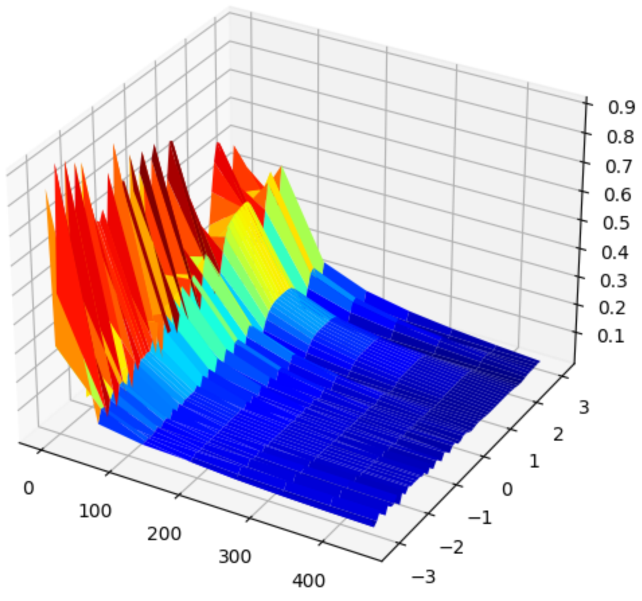


Figure 4: Time-Frequency Plot

0.4 Conclusion

DFTs of non-periodic signals can be computed in python with the help of a hamming window. Python