EE2703: Applied Programming Lab
Assignment 8
The Digital Fourier Transform

Devaganthan S S
EE19B018

June 21, 2021

0.1 Abstract

The assignment aims to understand how DFT can be visualized in python using the fast Fourier algorithm

0.2 Introduction

In this assignment, we compute the DFT of Various Discrete-Time Signals using the function **fft()** from NumPy. We then visualize the DFTs after making certain modifications to the output of the **fft()** function.

0.3 Results and Implementation

0.3.1 Examples

The range for the time axis is from -4π to 4π and the frequency range is from -64 to 64 with N = 512. The function **fftshift()** is used to get the desired spectrum. The below function plots the DFT spectrum of an array passed through it.

```python
def dftPlotter(y, xLim, Title):
    Y = fftshift(fft(y))/512
    w = linspace(-64, 64, 513)
    w = w[:-1]
    subplot(2, 1, 1)
    plot(w, abs(Y), lw=2)
    ylabel(r"$|Y|$", size=16)
    title(Title)
    xlim(-xLim, xLim)
    grid(True)
    subplot(2, 1, 2)
    plot(w, angle(Y), 'bo', lw=1, markersize='4')
    ii = where(abs(Y) > 1e-3)
    plot(w[ii], angle(Y[ii]), 'ro', lw=1)
    xlim([-xLim, xLim])
    ylabel(r"Phase of $Y$", size=16)
    xlabel(r"$\omega$", size=16)
    grid(True)
    show()


dftPlotter(sin(5*t), 10, 'Spectrum of sin(5t)')  # Example
dftPlotter((1+0.1*cos(t))*cos(10*t), 15,
           'Spectrum of 1+0.1*cos(t))*cos(10t) ')  # Example
```

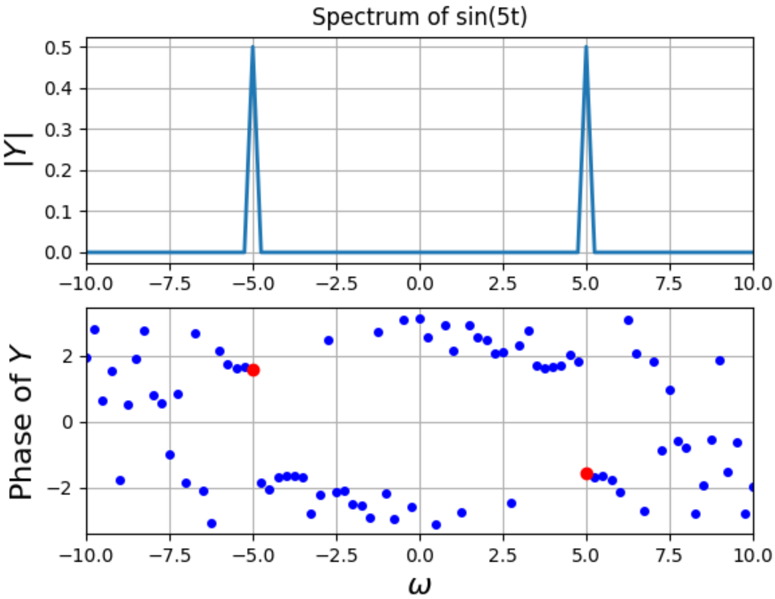

Figure 1

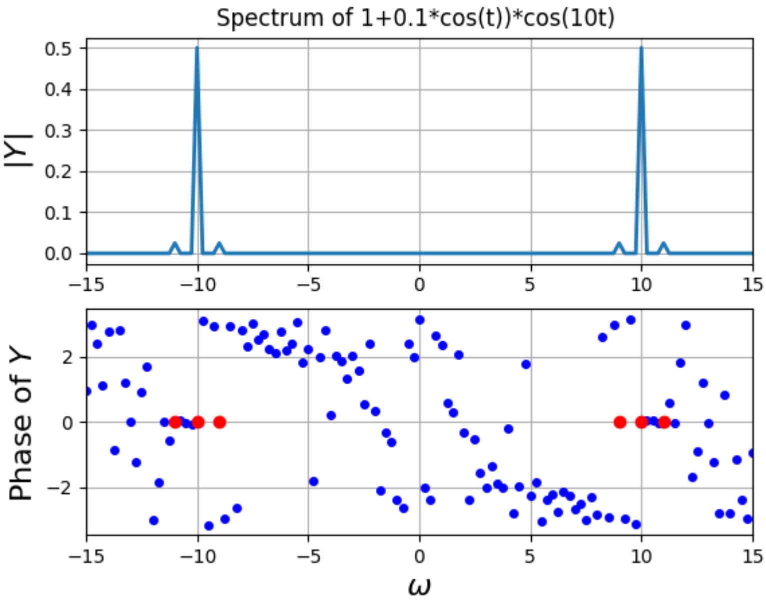

Figure 2

0.3.2 The spectrum of $\sin^3 t$ and $\cos^3 t$:

Using the same function as before, we get the below plot

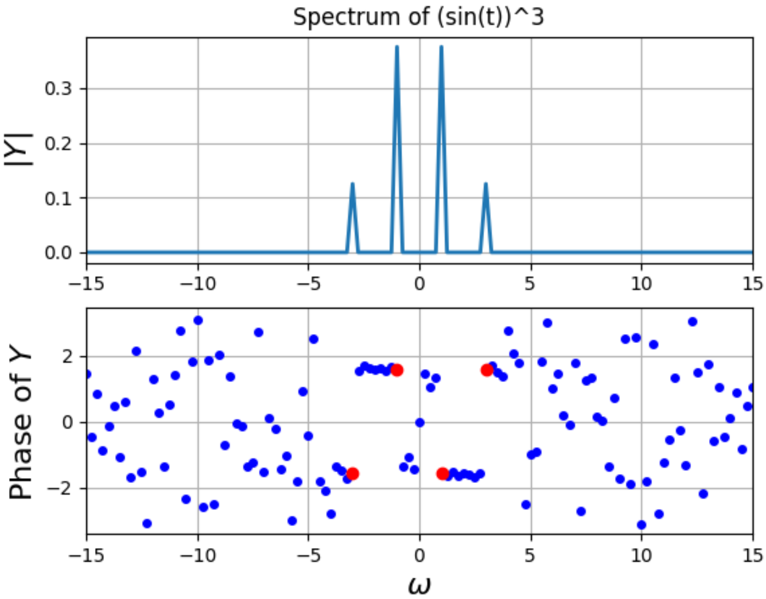

Figure 3

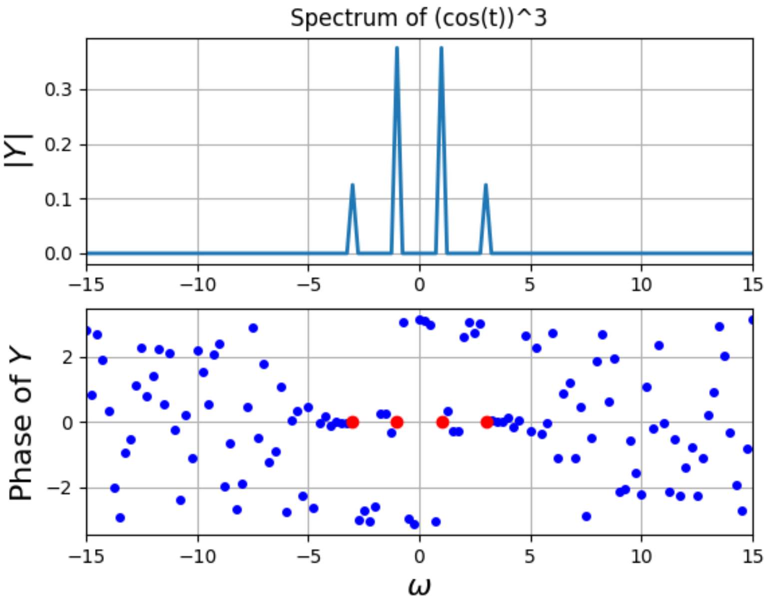

Figure 4

0.3.3 The spectrum of cos(20t+5 cos(t))

Using the same function as before, we get the below plot

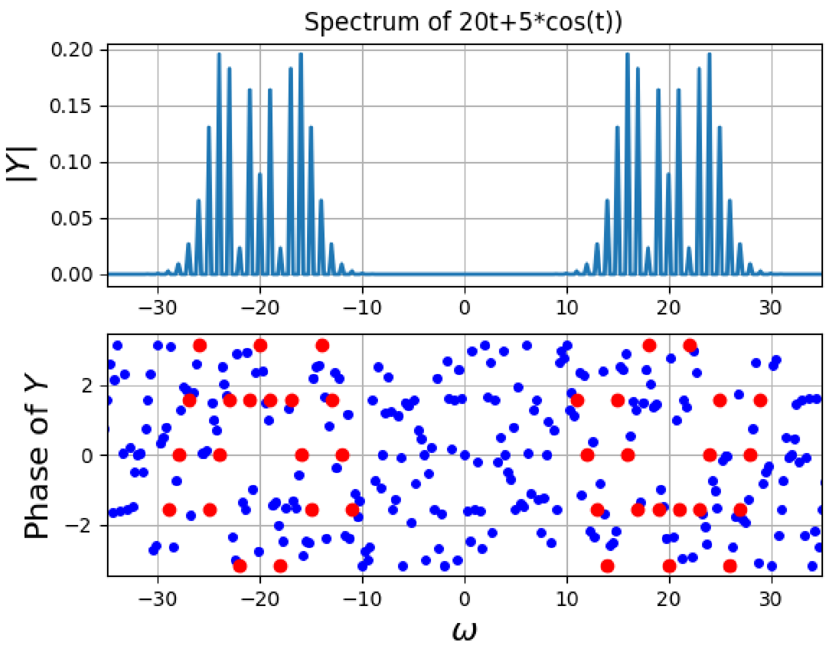

Figure 5

0.3.4 The spectrum of the Gaussian

To get an error of less than 10^{-6}, the range required is from -4π to 4π. The below code finds the desired range for the gaussian and plots the spectrum for the same. The code finds the least square Error, from using the DFTs found using **fft()** function and the computed DFT.

```
def estimate(N, T):
    t = linspace(- T / 2, T / 2, N + 1)[:-1]
    w = linspace(- N * pi / T, N * pi / T, N + 1)[:-1]
    y = exp(-0.5 * t**2)
    Y_true = exp(-0.5 * w**2) / sqrt(2 * pi)
    Y = fftshift(fft(ifftshift(y))) * T / (2 * pi * N)
    return sum(abs(Y - Y_true)), w, Y, Y_true


i = 1
while estimate(N=512, T=i * pi)[0] > 1e-6:
```

```
    i += 1

print('Time range for accurate spectrum : ' + str(i) + 'pi')
print('Error : ' + str(estimate(N=512, T=i * pi)[0]))

w, Y, Y_true = estimate(N=512, T=i * pi)[1:]

xLim = 5
subplot(2, 1, 1)
plot(w, abs(Y), lw=2)
title('Spectrum of exp(-(t^2)/2)')
ylabel(r"$|Y|$", size=16)
xlim([-xLim, xLim])
grid(True)
subplot(2, 1, 2)
plot(w, angle(Y), 'bo', lw=1, markersize='4')
ylabel(r"Phase of $Y$", size=16)
xlabel(r"$\omega$", size=16)
ii = where(abs(Y) > 1e-3)
plot(w[ii], angle(Y[ii]), 'ro', lw=2)
xlim([-xLim, xLim])
grid(True)
show()
```

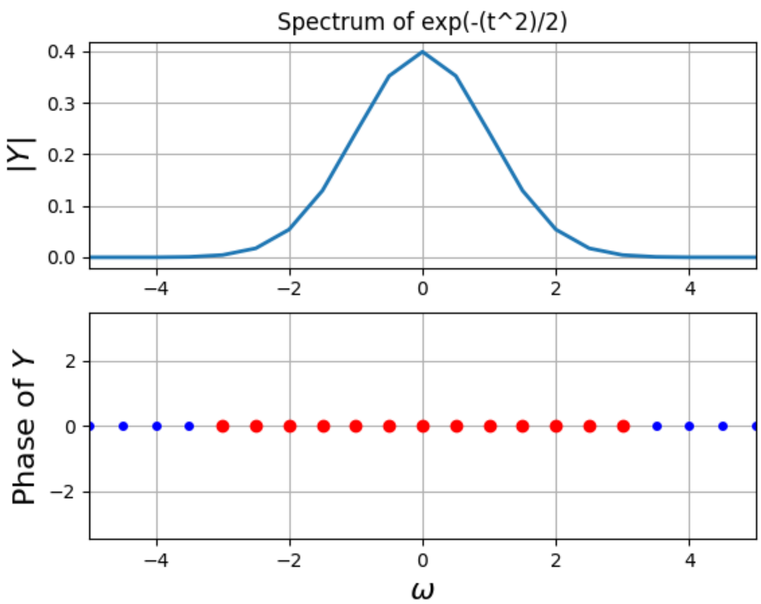

Figure 6

0.4 Conclusion

With the help of the functions **fft()** and **fftshift()**, DFTs of time signals can be visualized in Python