EE2703:
APPLIED PROGRAMMING LAB

Assignment 8

The Digital Fourier Transform

Arjun R
EE20B016

May 12, 2022

Contents

1 Introduction **1**

2 Getting Started **1**

3 Problems **1**
3.1 Example Problems 1
3.1.1 Ex1: Spectrum of $sin5t$ 1
3.1.2 Ex2: Spectrum of $(1 + 0.1cost)cos10t$ 2
3.2 Assignment Problems 3
3.2.1 P1: Spectra of sin^3t and cos^3t 3
3.2.2 P2: Spectrum of $cos(20t + 5cos(t))$ 4
3.2.3 P3: Spectrum of Gaussian function $exp(-t^2/2)$ 4

4 Conclusion **6**

1 Introduction

In this assignment, we will be analysing signals using the Discrete Fourier Transform, by making use of *numpy's* `fft` module which makes use of the **Fast Fourier Transform** algorithm.

$$F(e^{j\theta}) = \sum_{n=-\infty}^{\infty} f[n]e^{-jn\theta}$$

$F(e^{j\theta})$ is called the Digital Spectrum of the samples f[n]. It is also called the DTFT of f[n].

2 Getting Started

```
x = rand(100)
X = fft(x)
y = ifft(X)
print(c_[x,y])
print(abs(x-y).max())
```

When we run this, the value we get is around the order of 10^{-15} (I got 3.5742822720378785e-16). This is ample evidence to the fact that the starting vector x and the vector got by running fft and ifft on x, are the same.

3 Problems

3.1 Example Problems

3.1.1 Ex1: Spectrum of $sin5t$

Before plotting the spectrum of $sin5t$, we must keep note of the following points:

- The DFT treats the position axis as another frequency axis. Our position vector started at 0 and went to 2π, which is correct. The fft gives an answer in the same range of values. So we need to shift the π to 2π portion to the left as it represents negative frequencies. This can be done with a command called `fftshift`.

- Also 0 and 2π represent the same frequency. Hence, we need to stop the frequency vector just before 2π to avoid ambiguity.

Code implementation:

```
N = 128
x = linspace(0, 2*pi, N+1); x = x[:-1] # Removing 2pi to avoid ambiguity
y = sin(5*x)
Y = fftshift(fft(y))/N
w = linspace(-N/2, N/2 -1, N)
```

The spectra plot for the same would look as follows:

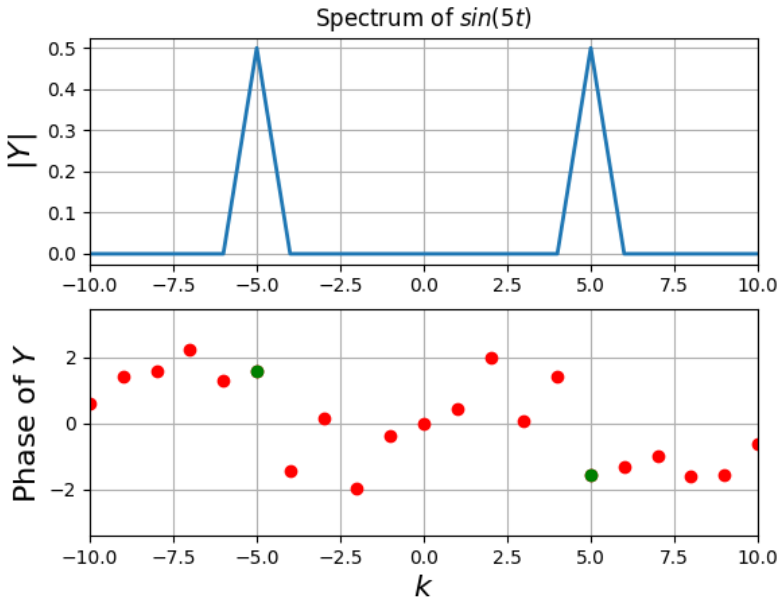

Figure 1: Spectra Plot for $sin5t$

We can make the following observations:

- The magnitude plot is even and the phase plot is odd, as expected for a real function.
- The magnitude plot has two peaks at $k = \pm 5$ with magnitudes of 0.5. This is expected as

$$sin5t = \frac{1}{2j}(e^{5jt} - e^{-5jt})$$

- The phase plot is $-\frac{\pi}{2}$ at k = 5 and $\frac{\pi}{2}$ at k = -5 as can be observed from the above relation.

3.1.2 Ex2: Spectrum of $(1 + 0.1cost)cos10t$

This is an example of *amplitude modulation*. Spectrum with 128 points as the last case will give a single broad spike, not 3 spikes. So we need to increase the number of sampling points. The following code does that for us:

```
t=linspace(-4*pi,4*pi,513);t=t[:-1]
y=(1+0.1*cos(t))*cos(10*t)
Y=fftshift(fft(y))/512.0
w=linspace(-64,64, 513); w = w[:-1]
```

The spectrum will look as follows:

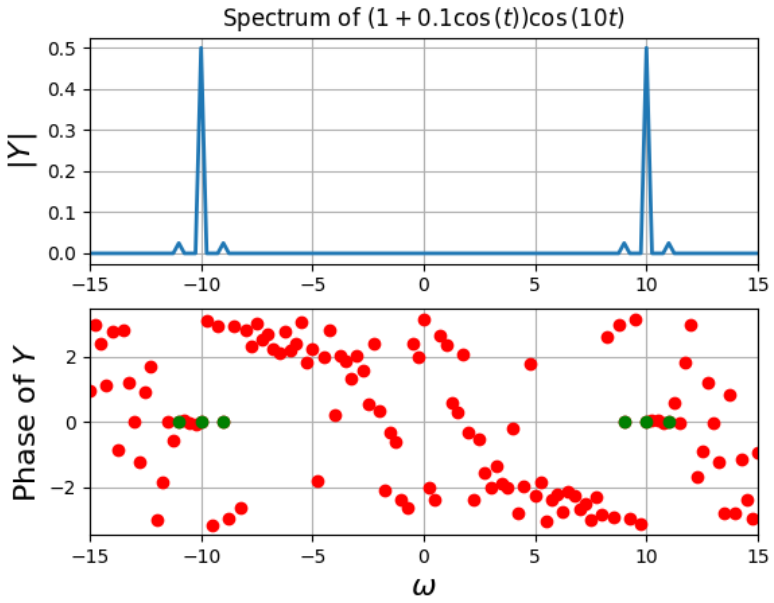

Figure 2: Spectra Plot for $(1 + 0.1cost)cos10t$

We can make the following observations:

- The magnitude plot has three peaks on each side at $|k| = 9, 10, 11$. This is expected as $(1 + 0.1cost)cos10t$ will have the frequency terms 10, 10-1, 10+1.
- The phase plot has 0 value at the above frequencies as is characteristic of cosine.

$$cosx = \frac{1}{2}(e^{jx} + e^{-jx})$$

3.2 Assignment Problems

3.2.1 P1: Spectra of sin^3t and cos^3t

Taking note of inferences and observations from the example problems on the sampling frequency, adequate number of sampling points, spacing criteria and proper shifting, the code for the finding spectra of sin^3t and cos^3t will be as follows:

```
t=linspace(0,2*pi,513);t=t[:-1]
y1 = sin(t)**3
y2 = cos(t)**3
Y1 = fftshift(fft(y1))/512.0
Y2 = fftshift(fft(y2))/512.0
w = linspace(-64, 64, 513); w = w[:-1]
```

Observations:

- Magnitude plot is even and phase plot odd as for real functions.
- Peaks at $k = \pm 1$ and $k = \pm 3$, with magnitude at $|k| = 1$ thrice of that at $|k| = 3$.

$$sin^3x = \frac{1}{4}(3sinx - sin3x)$$

$$cos^3x = \frac{1}{4}(3cosx + cos3x)$$

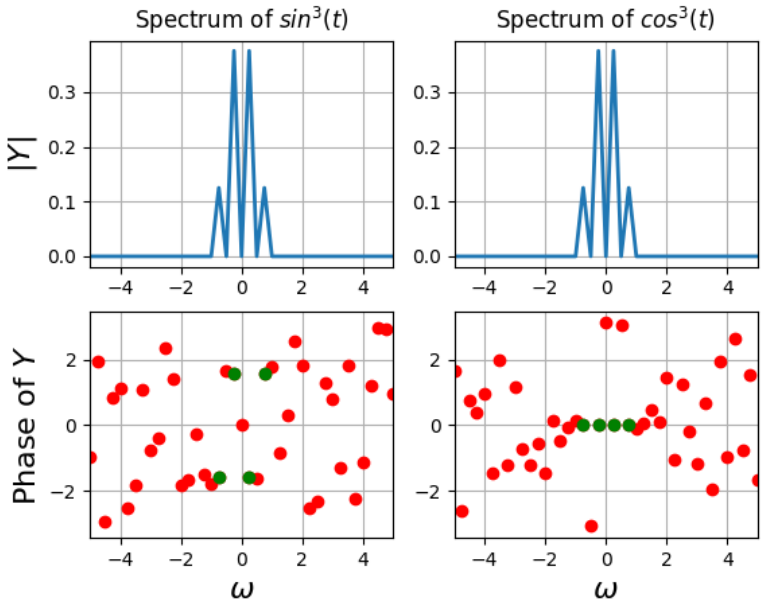

Figure 3: Spectra Plot for sin^3t and cos^3t

- Writing the sine and cosine terms in their Euler's forms helps us in analyzing the phase plot.

3.2.2 P2: Spectrum of $cos(20t + 5cos(t))$

This is a *frequency modulated* signal.
Code implementation:

```
t=linspace(-4*pi,4*pi,513);t=t[:-1]
y=cos(20*t+5*cos(t))
Y=fftshift(fft(y))/512.0
w=linspace(-64,64,513); w = w[:-1]
```

Observations:

- As it is an FM signal, the side frequencies also have comparable energy as of main frequency (k=20). This can be analysed using the power series of cosine:

$$cost = 1 + \frac{t^2}{2} + \frac{t^4}{4} + \dots$$

- The phase plot also can be similarly commented upon using the power series.

3.2.3 P3: Spectrum of Gaussian function $exp(-t^2/2)$

Given Gaussian function:

$$f(t) = e^{\frac{-t^2}{2}}$$

We must compute the spectrum of this Gaussian function accurate to 6 digits. For that we will calculate DFTs for the function increasing the number of samples N by a factor of 2, iterating until the maximum error between current DFT and previous DFT drops below the specified tolerance value (1e-6). This will provide us the required number of samples and

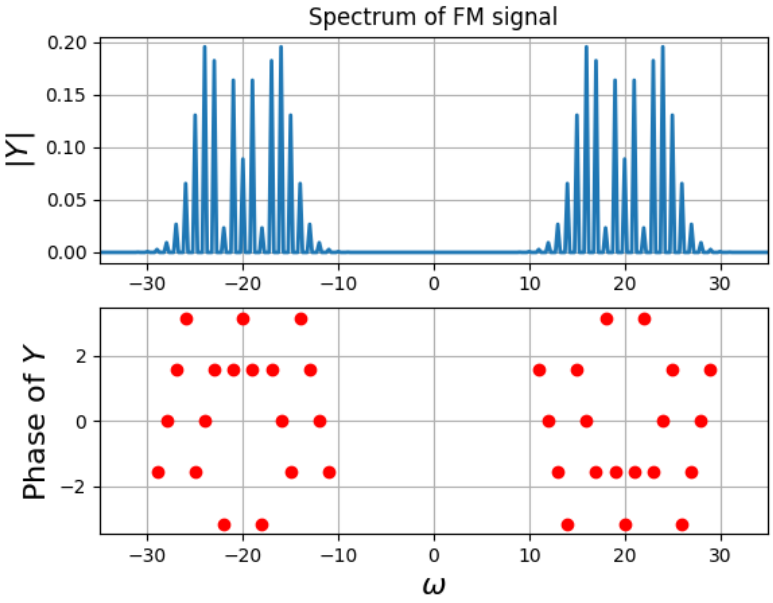

Figure 4: Spectra Plot for $cos(20t + 5cos(t))$

the time stretch to compute the actual spectrum using the exact function with the required accuracy. The CTFT of the Gaussian is given by:

$$F(j\omega) = \frac{1}{\sqrt{2\pi}} e^{\frac{-\omega^2}{2}}$$

Code implementation:

```
def dft_acc(tolerance = 1e-6, N = 128, T = 8*pi):

    err = tolerance + 1 # to enter the loop
    Y_old = 0

    while err > tolerance:
        x = linspace(-T/2, T/2, N+1)[:-1]
        w = linspace(-pi*N/T, pi*N/T, N+1)[:-1]
        y = Gaussian_exp(x)
        Y = fftshift(fft(ifftshift(y)))*T/(2*pi*N)
        err = max(abs(Y[::2]-Y_old))
        #print(err)
        Y_old = Y
        T *= 2
        N *= 2

    true_final_error = max(abs(Y - Gauss_fft(w)))

    print("True error: ", true_final_error)
    print("samples = ",N//2, ", time period = pi*", (T/pi)//2)
```

The Spectra plot will look as follows:

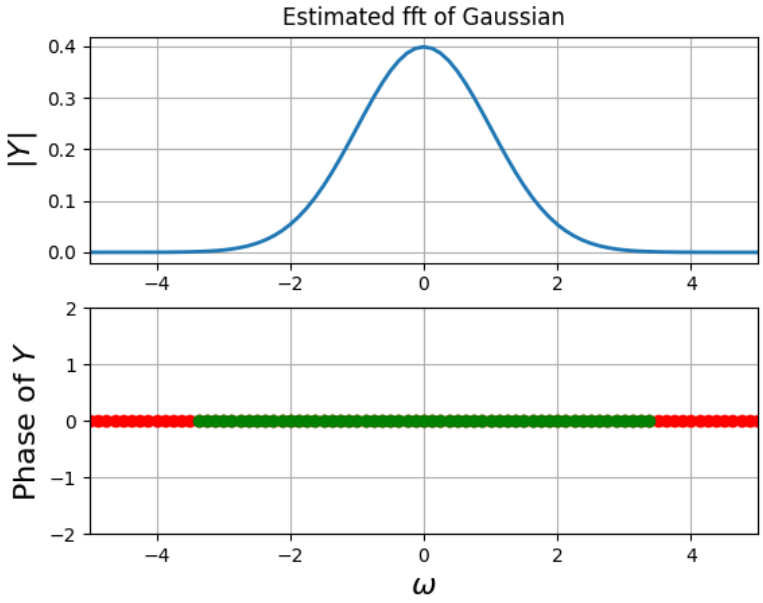

Figure 5: Estimated Spectra Plot for Gaussian

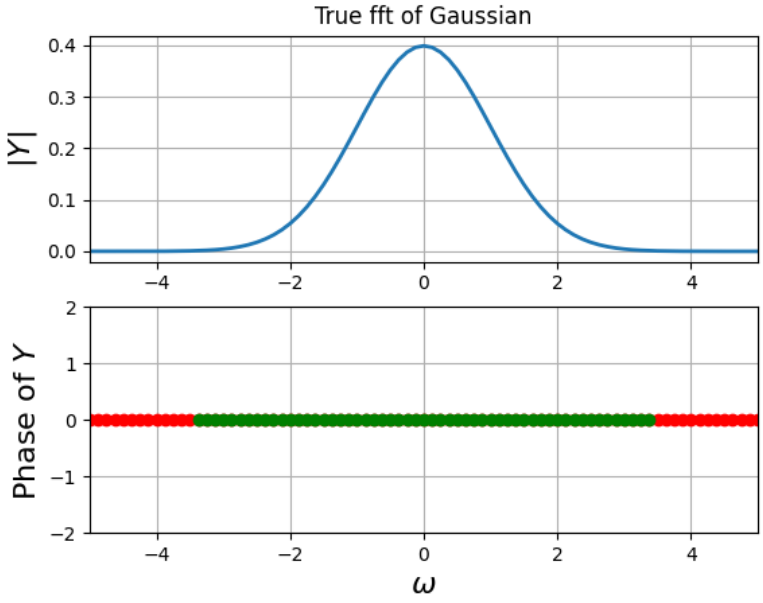

Figure 6: True Spectra Plot for Gaussian

4 Conclusion

We can conclude that the **Digital Fourier Transform** does approximate the *continuous time Fourier Transform* to a great extent. However, we must keep note of the right number of samples and the right time stretch to compute the spectrum. We should find these for aperiodic signals using the *error-tolerance method* covered in problem 4. Finally, we can also conclude that the computed DFTs perfectly agrees to the algebraic observations, as was shown in the case of sinusoidal functions.