EE2703: Applied Programming Lab
Assignment 8
The Digital Fourier Transform

Kaushik Ravibaskar
EE20B057

April 16, 2022

Contents

1 Aim

The assignment mainly deals with the following points:

- To obtain Digital Fourier Transform (DFT) using **pylab** module in python of some periodic and aperiodic (Gaussian) functions of time.
- To recover the analog **Fourier Transform** of the signals by proper sampling of the signals.
- To decide upon the range of time axis and digital **k** axis needed to obtain the exact plot.
- To plot the spectra obtained in various problems and explain the nature of the plots.

2 Introduction

In this assignment, we are going to use the **Digital Fourier Transform** (DFT) approach to obtain the spectra of some analog based signals. We already know that, any signal $f(t)u_o(t)$ which decays slower than $e^{\alpha t}$ for some real α can be expressed in laplace domain as follows:

$$F(s) = \int_{0^-}^{\infty} f(t)e^{-st}\,dt$$

where $s = \alpha + j\omega$.

If the signal $f(t)$ has a finite energy, then it can be expressed in its Fourier Transform as,

$$F(j\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}\,dt$$

If $f(t)$ is periodic with period 2π, then the Fourier Transform collapses into the **Fourier Series** as follows,

$$f(t) = \sum_{n=-\infty}^{\infty} a_n e^{jnt}$$

where,

$$a_n = \frac{1}{2\pi}\int_{t_o}^{t_o+2\pi} f(t)e^{-jnt}\,dt$$

where t_o can be any value, we can choose $t_o = 0$ for simplicity.

In discrete time case, for the signals having finite energy, say $f[n]$, we can define **Discrete Time Fourier Transform** (DTFT) as follows,

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

where $F(e^{j\theta})$ is also called the **Digital Spectrum** of the sample $f[n]$ sequence.

Now suppose $f[n]$ is itself periodic with period N, i.e. $f[n] = f[n+N]$, then the DTFT of the sequence is also periodic with same period N, it can be expressed as follows,

$$F[k] = \sum_{n=0}^{N-1} f[n] e^{\frac{-2\pi kn j}{N}}$$

$$f[n] = \frac{1}{N} \sum_{n=0}^{N-1} F[k] e^{\frac{2\pi kn j}{N}}$$

What this means is that the DFT is a sampled version of the DTFT, which is the digital version of the analog Fourier Transform.

3 Assignment Problems

3.1 P1: Analysing Given Problems in the Sheet

Firstly, let us analyse the given problems in the sheet. These problems will give us some confidence in using the inbuilt functions in python, namely **fft** and **ifft**, and other functions related to them. They are all housed under the **pylab** module.

3.1.1 Ex 1: Checking the accuracy of DFT using Python

In order to check the accuracy of Python computed DFT, let us take a randomly generated function (vector), x, and get back x by first doing `fft` and then `ifft` on the same. The code to perform this is as follows,

```
x_0 = py.rand(256)
X_0 = py.fft(x_0)
y_0 = py.ifft(X_0)
```

In one case, we get the maximum absolute difference between the two sets of data to be $3.3486903561235222e-16$. Thus we can observe that, the DFT operations are fairly accurate enough to get back the real vector.

3.1.2 Ex 2: Spectrum of $\sin 5t$

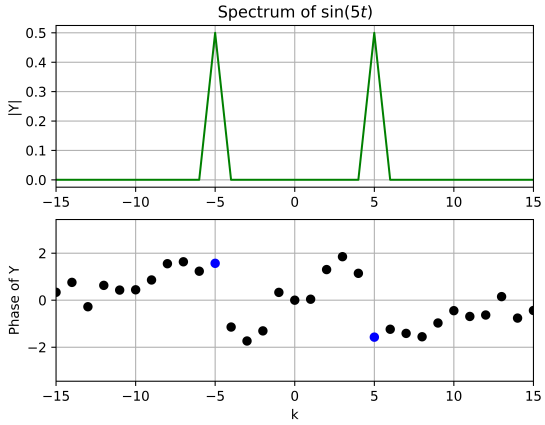
We will now compute the spectrum of $\sin 5t$ function. Before computing the spectrum, we must take care of the following points as highlighted in the assignment,

- The DFT in Python treats position axis as another frequency axis. Our position vector $t11$ went from 0 to 2π, which is correct. The `fft` gave the answer in the same range of values. Hence, we need to shift the π to 2π portion to the left as it represents negative frequency. We can do this using **fftshift** function.
- Also, 0 and 2π represents the same frequency. Hence, we need to stop the $t11$ vector just before 2π to avoid ambiguity.

The following piece of code does the computation,

```
t_1_1 = py.linspace(0, 2*ma.pi, 257)[: -1]
y_1_1 = py.sin(5*t_1_1)
Y_1_1 = py.fftshift(py.fft(y_1_1))/256
w_1_1 = py.linspace(-128, 127, 256)
```

The spectra plot for the same looks as follows,



We can make the following observations,

- The magnitude plot is even and phase plot is odd, as it should be for a real function.
- The magnitude plot has two peaks at $k = \pm 5$. This is expected as

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

  hence, two peaks at ± 5 with magnitude of 0.5.
- 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.

3.1.3 Ex 3: Spectrum of $(1 + 0.1\cos t)\cos 10t$

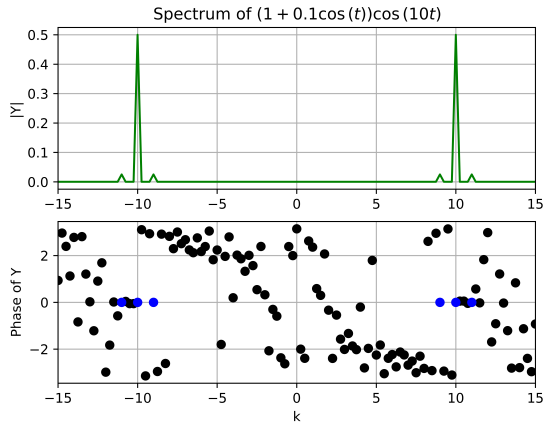
We will now compute the spectrum of $(1 + 0.1\cos t)\cos 10t$ function by taking care of the points mentioned in the above example along with these additional points,

- For this function, it is observed that we need to take in more frequency under consideration. Hence, we need to stretch the time axis along with increasing the number of sampling points.
- While doing this, we should ensure that although the spacing between the frequencies has been tightened, the sampling frequency remains unaffected.

The following piece of code does the computation,

```
t_1_2 = py.linspace(-4*ma.pi, 4*ma.pi, 1025)[: -1]
y_1_2 = (1 + 0.1*py.cos(t_1_2))*py.cos(10*t_1_2)
Y_1_2 = py.fftshift(py.fft(y_1_2))/1024
w_1_2 = py.linspace(-128, 128, 1025)[ :-1]
```

The spectra plot for the same looks as follows,



We can make the following observations,

- The magnitude plot is even and phase plot is odd, as it should be for a real function.
- The magnitude plot has three peaks on each side at $|k| = 9, 10, 11$. This is expected as $(1 + 0.1\cos t)\cos 10t$ would have frequency terms of 10, $10-1$, $10+1$, hence would reflect impulses in magnitude domain.
- The phase plot is 0 at all the peaks as cos doesn't have phase at the peaks i.e. $\cos x = \frac{1}{2}(e^{jx} + e^{-jx})$.

3.2 P2: Spectrum of $\sin^3(t)$ and $\cos^3(t)$

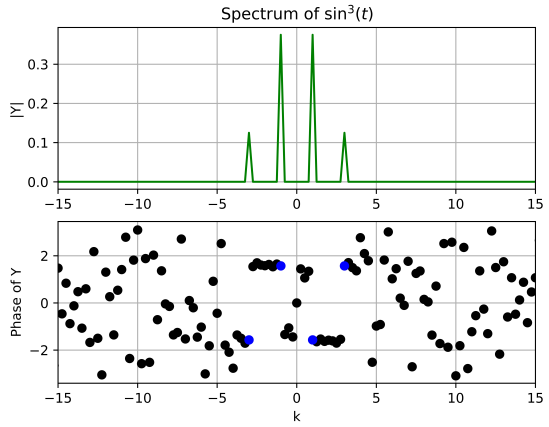
In this section, we will analyse the spectrum of $\sin^3(t)$ and $\cos^3(t)$ functions.

3.2.1 Spectrum of $\sin^3(t)$

Taking care of all the points mentioned in the previous problems regarding sampling frequency, number of adequate sampling points, spacing criteria and proper shifting, the code that gives the spectra of $\sin^3(t)$ is as follows,

```
t_2 = py.linspace(-4*ma.pi, 4*ma.pi, 513)[: -1]
w_2 = py.linspace(-64, 64, 513)[ :-1]
y_2_1 = (py.sin(t_2))**3
Y_2_1 = py.fftshift(py.fft(y_2_1))/512
```

The plot for the same is as follows,



We can make the following observations,

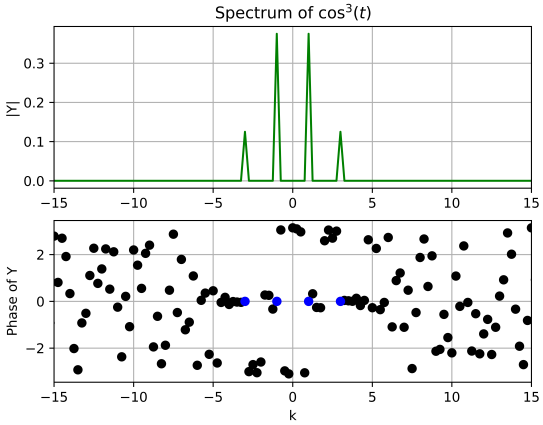
- The magnitude plot is even and phase plot is odd, as it should be for a real function.
- We know that $\sin 3x = 3\sin x - 4\sin^3 x$, thus $\sin^3 x = (3\sin x - \sin 3x)/4$, hence we should expect peaks at $k = \pm 1$ and $k = \pm 3$. Their magnitudes can also be verified by writing the sin terms in exponential forms and computing the coefficients.
- As $\sin^3 x = (3\sin x - \sin 3x)/4$, we should expect phase of $\frac{-\pi}{2}$ at $k = 1, -3$ and phase of $\frac{\pi}{2}$ at $k = -1, 3$,which can be analysed by writing it in complex exponential form. This is truly observed in the plot.

3.2.2 Spectrum of $\cos^3(t)$

Again, taking care of all the points mentioned in the previous problems, the code that gives the spectra of $\cos^3(t)$ is as follows,

```
t_2 = py.linspace(-4*ma.pi, 4*ma.pi, 513)[: -1]
w_2 = py.linspace(-64, 64, 513)[ :-1]
y_2_2 = (py.cos(t_2))**3
Y_2_2 = py.fftshift(py.fft(y_2_2))/512
```

The plot for the same is as follows,



We can make the following observations,

- The magnitude plot is even and phase plot is odd, as it should be for a real function.
- We know that $\cos 3x = 4\cos^3 x - 3\cos x$, thus $\cos^3 x = (3\cos x + \cos 3x)/4$, hence we should once again expect peaks at $k = \pm 1$ and $k = \pm 3$. Their magnitudes can also be verified by writing the cos terms in exponential forms and computing the coefficients.
- As $\cos^3 x = (3\cos x + \cos 3x)/4$, we should expect phase of 0 at all their peaks, which can be analysed by writing it in complex exponential form. This is truly observed in the plot.

3.3 P3: Spectrum of Frequency Modulated signal

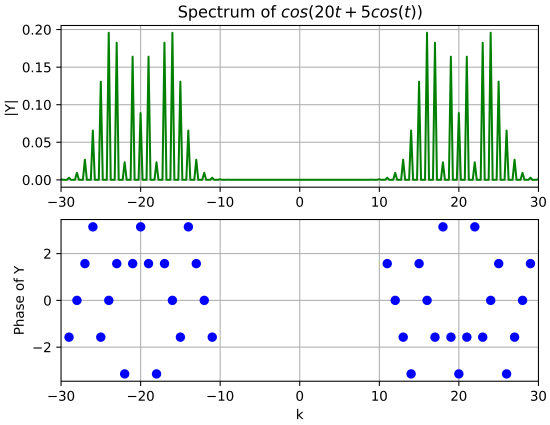
We are given the signal,

$$f(t) = \cos(20t + 5\cos(t))$$

Once again, following all the points mentioned above, the code that gives the spectra is as follows,

```
t_3 = py.linspace(-4*ma.pi, 4*ma.pi, 513)[: -1]
w_3 = py.linspace(-64, 64, 513)[ :-1]
y_3 = py.cos(20*t_3 + 5*py.cos(t_3))
Y_3 = py.fftshift(py.fft(y_3))/512
```

The plot for the same is as follows,



We can make the following observations,

- The magnitude plot is even and phase plot is odd, as it should be for a real function.
- As its a frequency modulated signal, the side frequencies also have comparable energy as of main frequency i.e. corresponding to $k = \pm 20$. This can be analysed by writing the power series of cos as,

  $$\cos x = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \dots$$

  We can note that, apart from $\cos 20t$ term, many other frequency would also surface up hence contributing to many peaks in the plot.
- The phase plot can be similarly commented upon using the power series.

3.4 P4: Spectrum of Gaussian $\exp(-t^2/2)$

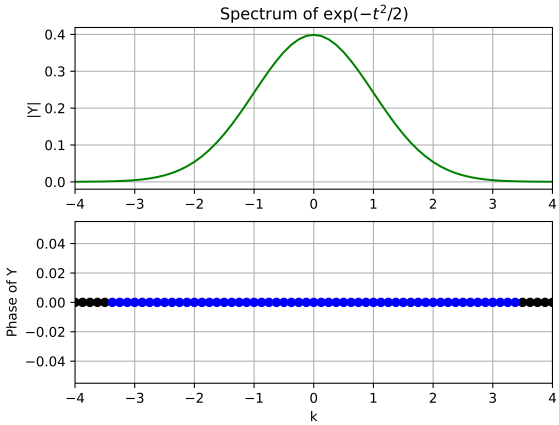
We have been given a Gaussian function $\exp(-t^2/2)$. We are asked to compute its spectrum with accuracy of 6 digits. We can compute the error-sum between the DFT vector **scaled down by** ω_s (sampling frequency) and the older DFT vector computed one iteration before in a **while** loop. The **While** loop will run until error becomes less than $1e-6$ (tolerance value). This will provide us the required number of samples and the time stretch

that meets the required accuracy. We must remember that the Gaussian is an aperiodic signal, hence we are dividing the **fft** computation by the sampling frequency instead of number of samples. The code that does all this is depicted below,

```
T_4 = 8*ma.pi
N_4 = 512
Y_4_0 = 0
tolerance = 1e-6
error = tolerance + 1

#bringing accuracy to 6 digits
while error > tolerance:
    t_4 = py.linspace(-T_4/2, T_4/2, N_4 + 1)[:-1]
    w_4 = py.linspace(-N_4*ma.pi/T_4, N_4*ma.pi/T_4, N_4 +
    1)[:-1]
    y_4 = py.exp(-(t_4*t_4)/2)
    Y_4 = py.fftshift(py.fft(py.ifftshift(y_4)))*T_4/(2*ma.pi*N_4)
    error = py.sum(abs(Y_4[::2] - Y_4_0))
    Y_4_0 = Y_4
    T_4 *= 2
    N_4 *= 2
```

The plot of the spectra (estimated plot) is as follows,



4 Conclusion

We can conclude that the **Digital Fourier Transform** does provide a good approximation to the actual analog Fourier Transform. However, we must

be careful while sampling the signal and deciding upon the time stretch to compute the spectrum. Lastly, we can conclude that the DFT handsomely agrees to all the algebraic proofs that the function's graph must satisfy, as was shown in the case of sinusoidal inputs.