Assignment 8

K.R.Srinivas - EE18B136

April 22, 2020

Abstract

This week's Python assignment focuses on the following topics.

- Implementing DFT using Python's FFT module.
- Recovering Analog Fourier Transform of a signal by proper sampling of them.
- Attempt to approximate the CTFT of a Gaussian.

1 Introduction & Theory

The discrete Fourier transform (DFT) converts a finite sequence of equally-spaced samples of a function into a same-length sequence of equally-spaced samples of the discrete-time Fourier transform (DTFT), which is a complex-valued function of frequency.

Let suppose f[n] are the samples of some continous function $f(t)$ then we define the Z transform as

$$F(z) = \sum_{n=-\infty}^{n=\infty} f(n)z^{-n} \tag{1}$$

Replacing z with $e^{j\omega}$ we get DTFT of the sampled function

$$F(e^{j\omega}) = \sum_{n=-\infty}^{\infty} f(n)e^{-j\omega n} \tag{2}$$

$F(e^{j\omega})$ is continuous and periodic. $f[n]$ is discrete and aperiodic. Suppose now $f[n]$ is itself periodic with a period$N$, i.e.,

$$f[n+N] = f[n]$$

Then, it should have samples for its DTFT. This is true, and leads to the Discrete Fourier Transform or the DFT:
Suppose $f[n]$ is a periodic sequence of samples, with a period N.Then the

DTFT of the sequence is also a periodic sequence $F[k]$ with the same period N.

$$F[k] = \sum_{n=0}^{N-1} f[n] e^{-j\frac{2\pi nk}{N}} = \sum_{n=0}^{N-1} f[n] W^{nk} \tag{3}$$

$$f[n] = \frac{1}{N} \sum_{k=0}^{N-1} F[k] W^{-nk} \tag{4}$$

Here $W = e^{-j\frac{2\pi}{N}}$ is used simply to make the equations less cluttered. and k is sampled values of continous variable ω at multiples of $\frac{2\pi}{N}$

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

In this assignment, we want to explore how to obtain the DFT, and how to recover the analog Fourier Tranform for some known functions by the proper sampling of the function.

2 Assignment Tasks

2.1 Review of Assignment Examples

2.1.1 Error on Reconstruction

We find the Fourier transform and invert it back to the time domain for a random signal, find maximum error to test the reconstruction.

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

The maximum error obtained from the below code = 3.3941334505218686e-16

2.1.2 Spectrum of Pure Sinusoidal and Amplitude Modulated Signal

We use `fft()` and `fftshift()` commands available in Python to obtain the Spectrum of the signals $sin(5t)$ and $(1 + 0.1 * cos(t))(cos(10t))$.We choose appropriate sampling frequency and normalisation to obtain the required Spectrum.

```
x = linspace(0,2*pi,129); x=x[:-1]
y = sin(5*x)
Y = fftshift(fft(y))/128.0
w = linspace(-64,63,128)
```

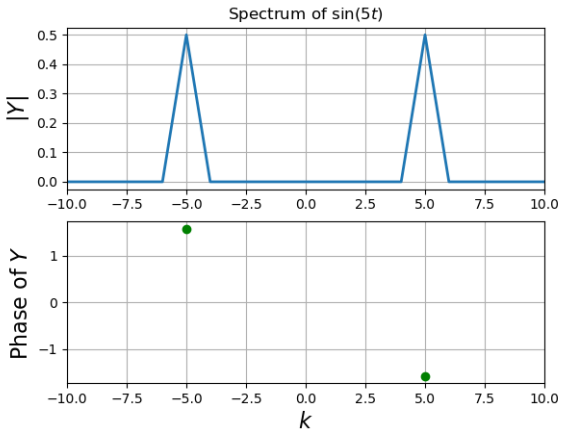

Figure 1: Fourier spectrum plot of $sin(5t)$

```
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]
```

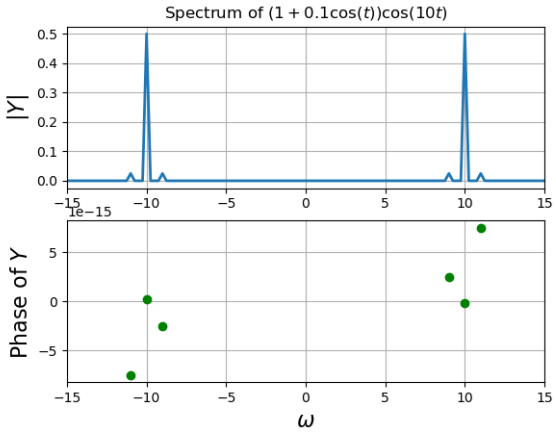

Figure 2: Correct Fourier spectrum plot $(1 + 0.1\cos(t))\cos(10t)$

Inorder to allow space for more frequencies it is essential to increase the number of samples taken, also only the phase at spikes are important for us. Hence, while plotting the angle,it is good to consider the magnitude as well.

2.2 Spectrum of $sin^3(t)$ and $cos^3(t)$

To compare the spectrum obtained for $\sin^3(t)$, we use

$$\sin^3(t) = \frac{3}{4}\sin(t) - \frac{1}{4}\sin(3t) \tag{5}$$

So the fourier transform of $\sin^3(t)$ using above relation is

$$F(\sin^3(t)) \rightarrow \frac{3}{8j}(\ \delta(\omega-1) - \delta(\omega+1)\ ) - \frac{1}{8j}(\ \delta(\omega-3) - \delta(\omega+3)\ ) \tag{6}$$

Similarly $\cos^3(t)$ is given by

$$\cos^3(t) = \frac{3}{4}\cos(t) + \frac{1}{4}\cos(3t) \tag{7}$$

So the fourier transform of $\cos^3(t)$ using above relation is

$$F(\cos^3(t)) \rightarrow \frac{3}{8}(\ \delta(\omega-1) + \delta(\omega+1)\ ) + \frac{1}{8}(\ \delta(\omega-3) + \delta(\omega+3)\ ) \tag{8}$$

CODE :

```
t=linspace(-4*pi,4*pi,513);t=t[:-1]
y= pow(sin(t),3)
Y=fftshift(fft(y))/512.0
w=linspace(-64,64,513);w=w[:-1]

t=linspace(-4*pi,4*pi,513);t=t[:-1]
y= pow(cos(t),3)
Y=fftshift(fft(y))/512.0
w=linspace(-64,64,513);w=w[:-1]
```

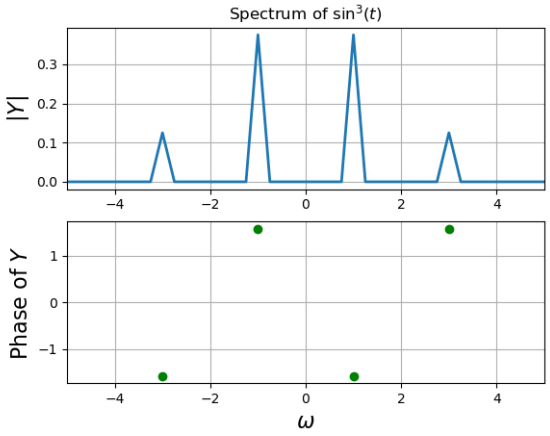

Figure 3: Spectrum of $sin^3(t)$

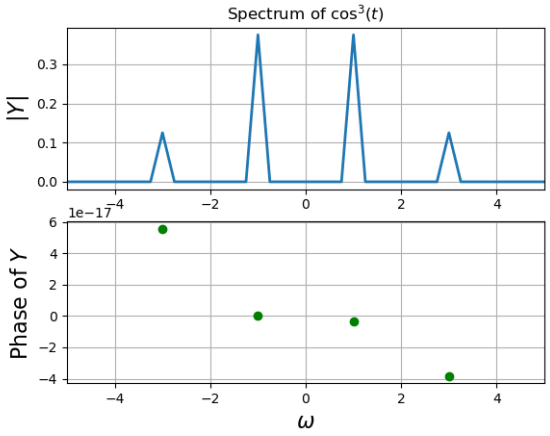

Figure 4: Spectrum of $cos^3(t)$

We observe the plot frequency contents are $\omega = 1, -1, 3, -3$ and with their amplitude in 3:1 ratio. For $cos^3(t)$,ideally the phase at all the spikes should be 0, but due to lack of infinite computing power it is only approximately 0. For $sin^3(t)$, the phases at the location of spike is evident from (6)

2.3 Spectrum of Phase Modulated Wave

Consider the signal $\cos(20t + 5\cos(t))$.Using the same helper function as before, we get the following output:
CODE:

```
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]
```

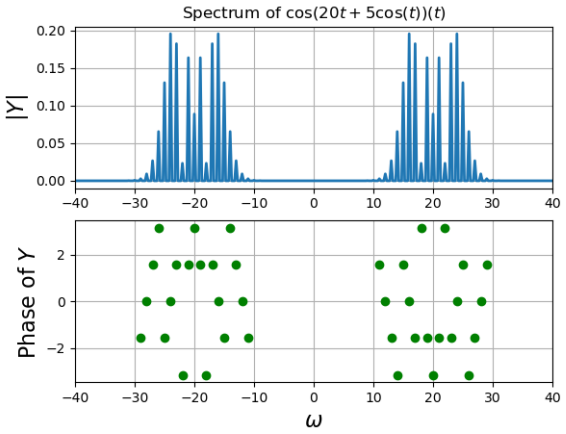

Figure 5: Spectrum of $\cos(20t + 5\cos(t))$

We observe the plot that its a Phase modulation since phase of the signal is varying proportional to amplitude of the message signal being $\omega = 20$ and infinite side band frequencies which are produced by $5\cos t$.Phase spectra is a mix of different phases from $[-\pi, \pi]$ because of phase modulation, i.e since the phase is changed continuously with respect to time.

2.4 Continuous time Fourier Transform of a Gaussian

We generate the spectrum of the Gaussian $e^{-\frac{t^2}{2}}$ which is not *bandlimited* in frequency and aperiodic in time domain find Fourier transform of it using DFT and to recover the analog fourier transform from it.

$$F(\ e^{-\frac{t^2}{2}}\ ) \rightarrow \frac{1}{\sqrt{2\pi}} e^{\frac{-\omega^2}{2}} \tag{9}$$

To find the normalising constant for DFT obtained we use following steps to derive it :

- window the signal $e^{-\frac{t^2}{2}}$ by rectangular function with gain 1 and window_size 'T' which is equivalent to convolving with $Tsinc(\omega T)$ in frequency domain. So As T is very large the $sinc(\omega T)$ shrinks , we can approximate that as $\delta(\omega)$. So convolving with that we get same thing.
- Windowing done because finite computing power and so we cant represent infinetly wide signal .

- Now we sample the signal with sampling rate N,which is equivalent to convolving impulse train in frequency domain
- And finally for DFT we create periodic copies of the windowed sampled signal and make it periodic and then take one period of its Fourier transform i.e is DFT of gaussian.
- Following these steps we get normalising factor of **Window_size/(**2π **Sampling_rate)**

$$exp(-\frac{t^2}{2}) \longleftrightarrow \frac{1}{\sqrt{2\pi}} exp(-\frac{\omega^2}{2}) \tag{10}$$

$$rect(\frac{t}{\tau}) = \begin{cases} 1 \ for \ |t| < \tau \\ 0 \ otherwise \end{cases} \tag{11}$$

- For windowing the signal, we will multiply with the rectangular function,

$$y(t) = gaussian(t) \times rect(\frac{t}{\tau}) \tag{12}$$

- In fourier domain, its convolution (since multiplication is convolution in fourier domain)

$$Y(\omega) = \frac{1}{2\pi}(\frac{1}{\sqrt{(2\pi)}} e^{-\omega^2/2} * \frac{sin(\tau\omega)}{\omega}) \tag{13}$$

$$\lim_{\tau\to\infty} Y(\omega) = \frac{\tau}{2\pi}(\frac{1}{\sqrt{(2\pi)}} e^{-\omega^2/2} * \delta(\omega)) \tag{14}$$

$$\lim_{\tau\to\infty} Y(\omega) = \frac{\tau}{2\pi}(\frac{1}{\sqrt{(2\pi)}} e^{-\omega^2/2}) \tag{15}$$

- Now, sampling this signal with a period of $\frac{2\pi}{T_s}$, we will get (multiplication by an impulse train in fourier domain),

$$Y_{sampled} = \frac{\tau}{2\pi T_s} \frac{1}{\sqrt{(2\pi)}} e^{-\omega^2/2} \sum_{k=-\infty}^{\infty} \delta(\omega - \frac{k2\pi}{T_s}) \tag{16}$$

- Solving it further we get the multiplication factor to be,

$$const = \frac{\tau}{T_s 2\pi} \tag{17}$$

- To find the Discrete Fourier transform equivalent for Continous Fourier transform of Gaussian function by finding absolute error between the DFT obtained using the normalising factor obtained with exact Fourier transform and find the parameters such as Window_size and sampling rate by minimising the error obtained with tolerance of 10^{-15}

CODE:

```
# initial window_size and sampling rate defined
window_size = 2*pi
sampling_rate = 128

# tolerance for error
tolerance = 1e-15

# normalisation factor derived
norm_factor = (window_size)/(2*pi*(sampling_rate))

# In order to decrease the error in IFFT we increase the
  Window-Size iteratively.Also to overcome the issue of aliasing it i

for i in range(1, 10):

 t = linspace(-window_size/2, window_size/2,sampling_rate+1)[:-1]
 y = exp(-pow(t, 2)/2)
 N = sampling_rate
 Y = fftshift((fft(ifftshift(y)))*norm_factor)

 w_lim = (2*pi*N/((window_size)))
 w = linspace(-(w_lim/2), (w_lim/2), (sampling_rate+1))[:-1]

 actual_Y = (1/sqrt(2*pi))*exp(-pow(w, 2)/2)
 error = (np.mean(np.abs(np.abs(Y)-actual_Y)))
 print("Absolute error at Iteration - %g is : %g" % ((i,
       error)))

if(error < tolerance):
    print("\nAccuracy of the DFT is: %g and Iterations
         took: %g" %((error, i)))
    print("Best Window_size: %g , Sampling_rate: %g"
         %((window_size, sampling_rate)))
    break
else:
    window_size = window_size*2
    sampling_rate = (sampling_rate)*2
```

```
norm_factor = (window_size)/(2*pi*(sampling_rate))
```

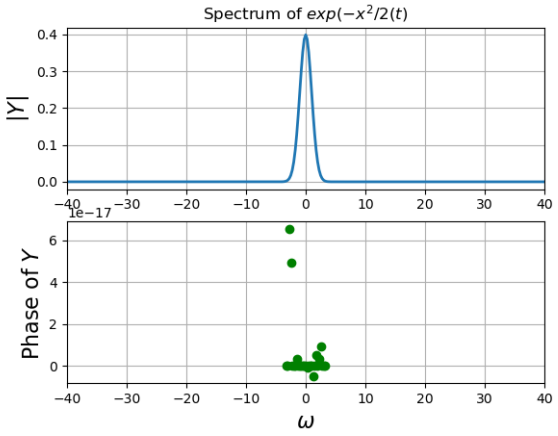

Figure 6: Estimated CTFT of Gaussian

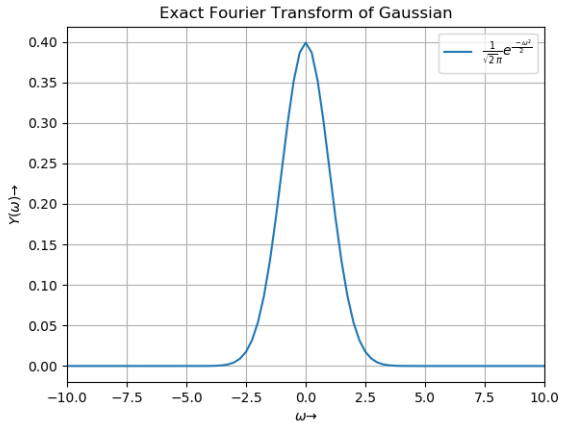

Figure 7: Expected CTFT of Gaussian

We observe the magnitude spectrum of $e^{-\frac{t^2}{2}}$ and that it almost coincides with exact Fourier Transform plotted with accuracy of $1.32645e^{-17}$ in 3 iterations at the sampling rate of 512.

The iteration used minimizes the error by increasing the window size and sampling rate. The optimal window size turns out to be 25.1327 s.

3 Conclusion

Hence we analysed the how to find DFT for various types of signals and how to estimate normalising factors for Gaussian functions and hence recover the analog Fourier transform using DFT ,also to find parameters like window_size and sampling rate by minimizing the error.Also, FFT works well for signals with samples in 2^k , as it dividesthe samples into even and odd and goes dividing further to compute the DFT.