ASSIGNMENT 8: THE DIGITAL FOURIER TRANSFORM

NITHIN BABU [EE18B021]

April 19, 2020

Abstract

The goal of this assignment is the following:

- To learn about DFT, FFT and how it is implemented in python.
- Obtaining the DFT of various functions.
- To see how the DFT can be used to approximate the CTFT.
- To plot graphs to understand this.

Digital Fourier Transform

- We analyse and use DFT to find the Fourier transform of periodic signals and non periodic ones using fast fourier transform algorithms which are implemented in python using fft and $fftshift$ which is used to center the fourier spectra of a discrete signal.
- 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

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

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

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

- $F(e^{j\omega})$ 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.

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

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

- 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

DFT of simple functions

- We need to find Discrete Fourier Transform DFT of $\sin(5t)$ and (AM) Amplitude Modulated signal given by $(1 + 0.1\cos(t))\cos(10t)$.
- Plot and analyse the spectrum obtained for both the functions given above.
- Cross validate the spectrum obtained with what is expected.
- To compare the spectrum obtained for $\sin(5t)$, we use

$$\sin(5t) = \frac{1}{2j}e^{j5} - \frac{1}{2j}e^{-j5}$$

- So the fourier transform of $\sin(5t)$ using above relation is

$$\mathscr{F}(\sin(5t)) \rightarrow \frac{1}{2j}(\ \delta(\omega - 5) - \delta(\omega + 5)\ )$$

- Similarly for finding Fourier Transform AM signal following relations are used

$$(1 + 0.1\cos(t))\cos(10t) \rightarrow \cos(10t) + 0.1\cos(10t)\cos(t)$$
$$0.1\cos(10t)\cos(t) \rightarrow 0.05(\ \cos(11t) + cos(9t)\ )$$
$$(1 + 0.1\cos(t))\cos(10t) \rightarrow 0.025(e^{j11t} + e^{j9t} + e^{j11t} + e^{-j9t})$$

- So we can find fourier transform from above relation
- So using this we compare the plots of Magnitude and phase spectrum obtained using DFT and analyse them.

The python code to calculate the DFT of the above functions is as follows:

```
# Declaring the necessary variables and calculating the DFT of sin(5t)
N1 = 128.0
t1 = linspace(0,2*pi,N1+1); t1 = t1[:-1]
y1 = sin(5*t1)
Y1 = fftshift(fft(y1))/N1
w1 = linspace(-64,63,N1)

# Magnitude plot for the DFT of sin(5t)
figure(0)
plot(w1,abs(Y1))
title(r"Spectrum of $\sin(5t)$")
```

```
ylabel(r"$|Y(\omega)|\rightarrow$")
xlabel(r"$\omega\rightarrow$")
xlim([-10,10])
grid(True)

# Phase plot for the DFT of sin(5t)
figure(1)
plot(w1,angle(Y1),'ro')
ii = where(abs(Y1)>1e-3)
plot(w1[ii],angle(Y1[ii]),'go')
title(r"Phase of $\sin(5t)$")
ylabel(r"$\angle Y(\omega)\rightarrow$")
xlabel(r"$\omega\rightarrow$")
xlim([-10,10])
grid(True)

# Declaring the necessary variables and calculating the DFT of (1 + 0.1cos(t))cos(1
N2 = 512.0
t2 = linspace(-4*pi,4*pi,N2+1); t2 = t2[:-1]
y2 = (1 + 0.1*cos(t2))*cos(10*t2)
Y2 = fftshift(fft(y2))/N2
w2 = linspace(-64,64,N2+1); w2 = w2[:-1]

# Magnitude plot for the DFT of (1 + 0.1cos(t))cos(10t)
figure(2)
plot(w2,abs(Y2))
title(r"Spectrum of $(1 + 0.1*cos(t))*cos(10*t)$")
ylabel(r"$|Y(\omega)|\rightarrow$")
xlabel(r"$\omega\rightarrow$")
xlim([-15,15])
grid(True)

# Phase plot for the DFT of (1 + 0.1cos(t))cos(10t)
figure(3)
plot(w2,angle(Y2),'ro')
title(r"Phase of $(1 + 0.1*cos(t))*cos(10*t)$")
ylabel(r"$\angle Y(\omega)\rightarrow$")
xlabel(r"$\omega\rightarrow$")
xlim([-15,15])
grid(True)
show()
```

The magnitude and phase plots of $sin(5t)$ and $(1 + 0.1cos(t))cos(10t)$ are as shown below:

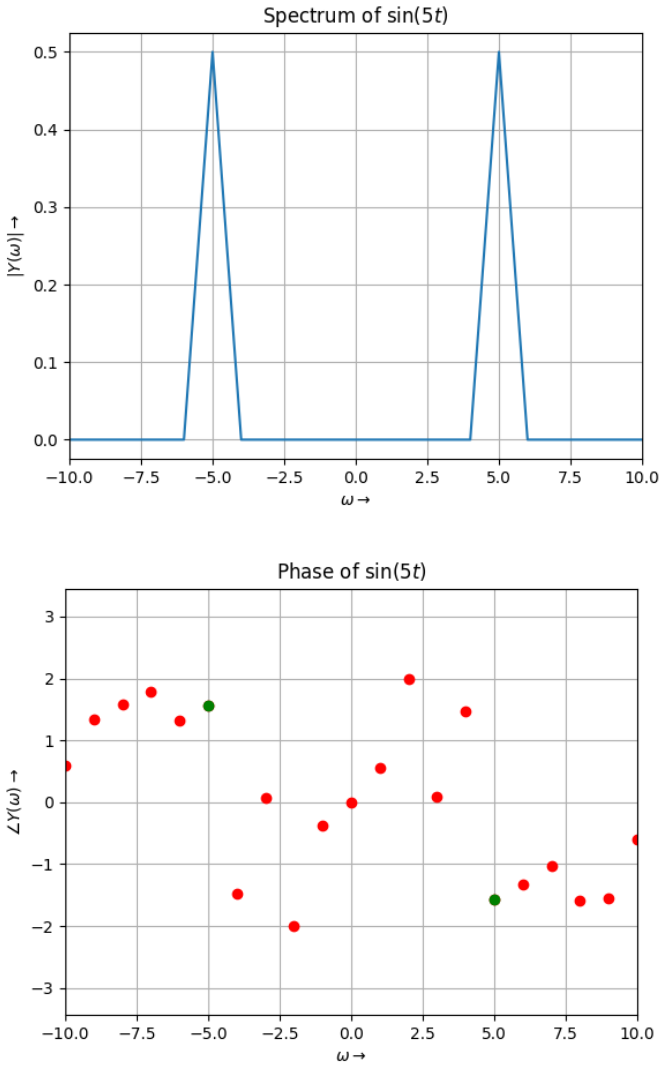

Figure 1: Spectrum and phase plots of $sin(5t)$

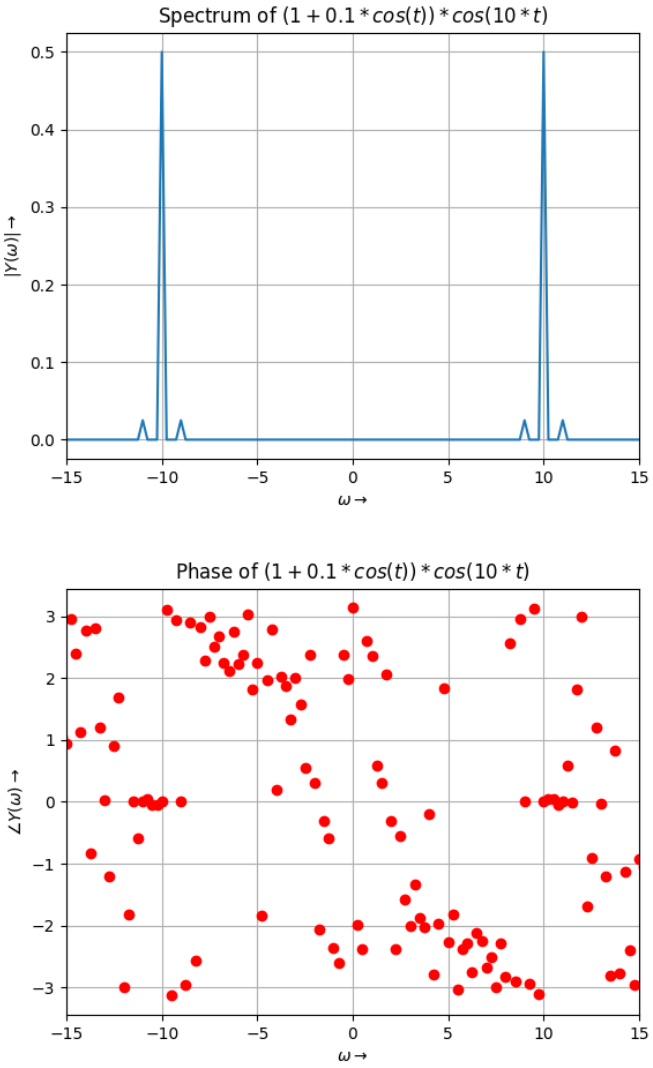

Figure 2: Spectrum and phase plots of $(1 + 0.1\cos(t))\cos(10t)$

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

- We need to find Discrete Fourier Transform DFT of $\sin^3(t)$ and $\cos^3(t)$
- Plot and analyse the spectrum obtained for both the functions given above.
- Cross validate the spectrum obtained with what is expected.
- To compare the spectrum obtained for $\sin^3(t)$, we use

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

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

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

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

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

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

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

- So using this we compare the plots of Magnitude and phase spectrum obtained using DFT and analyse them.

The python code to find the spectrum of the above functions is as shown below:

```
# Declaring the necessary variables and calculating the DFT of sin^3(t)
N3 = 512.0
t3 = linspace(-4*pi,4*pi,N3+1); t3 = t3[:-1]
y3 = (3*sin(t3) - sin(3*t3))/4
Y3 = fftshift(fft(y3))/N3
w3 = linspace(-64,64,N3+1); w3 = w3[:-1]

# Magnitude plot for the DFT of sin^3(t)
```

```
figure(4)
plot(w3,abs(Y3))
title(r"Spectrum of $sin^{3}(t)$")
ylabel(r"$|Y(\omega)|\rightarrow$")
xlabel(r"$\omega\rightarrow$")
xlim([-15,15])
grid(True)

# Calculating the DFT of cos^3(t)
y4 = (3*cos(t3) + cos(3*t3))/4
Y4 = fftshift(fft(y4))/N3

# Magnitude plot for the DFT of cos^3(t)
figure(5)
plot(w3,abs(Y4))
title(r"Spectrum of $cos^{3}(t)$")
ylabel(r"$|Y(\omega)|\rightarrow$")
xlabel(r"$\omega\rightarrow$")
xlim([-15,15])
grid(True)
show()
```

The spectrum plots of $\sin^3(t)$ and $\cos^3(t)$ are:

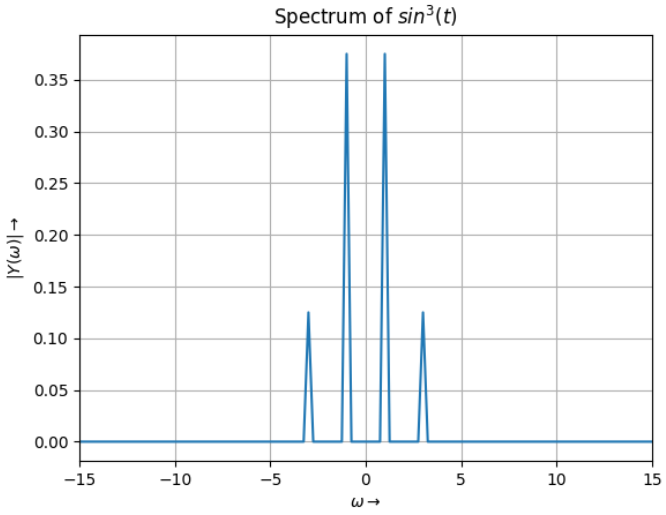

Figure 3: Spectrum of $sin^3(t)$

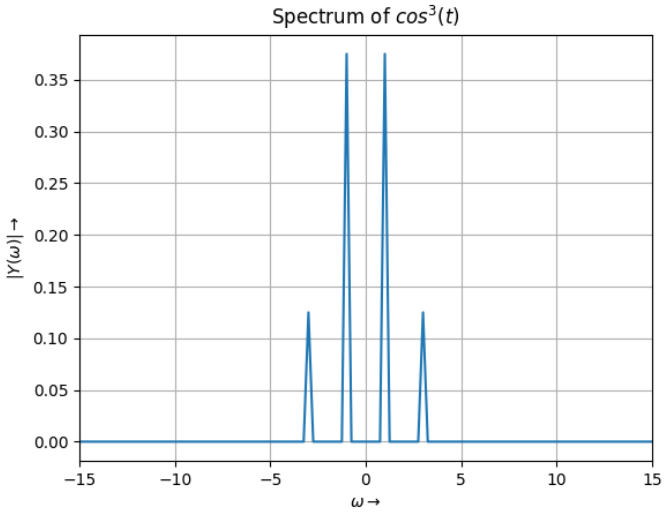

Figure 4: Spectrum of $cos^3(t)$

Sprectrum of $\cos(20t + 5\cos(t))$

- We need to generate the spectrum of $\cos(20t + 5\cos(t))$.
- Plot phase points only where the magnitude is significant ($> 10^{-3}$).
- Analyse the spectrums obtained.

The python code to find the spectrum of $\cos(20t + 5\cos(t))$ is as shown below:

```
 # Calculating the DFT of cos(20t + 5cos(t))
y5 = cos(20*t3 + 5*cos(t3))
Y5 = fftshift(fft(y5))/N3

# Magnitude plot for the DFT of cos(20t + 5cos(t))
figure(6)
plot(w3,abs(Y5))
title(r"Spectrum of $cos(20t + 5cos(t))$")
ylabel(r"$|Y(\omega)|\rightarrow$")
xlabel(r"$\omega\rightarrow$")
xlim([-40,40])
grid(True)

# Phase plot for the DFT of cos(20t + 5cos(t)) where magnitude is greater than 1e-3
```

```
figure(7)
ii = where(abs(Y5)>=1e-3)
plot(w3[ii],angle(Y5[ii]),'go')
title(r"Phase of $cos(20t + 5cos(t))$")
ylabel(r"$\angle Y(\omega)\rightarrow$")
xlabel(r"$\omega\rightarrow$")
xlim([-40,40])
grid(True)
show()
```

The spectrum and phase plots of $\cos(20t + 5\cos(t))$ are:

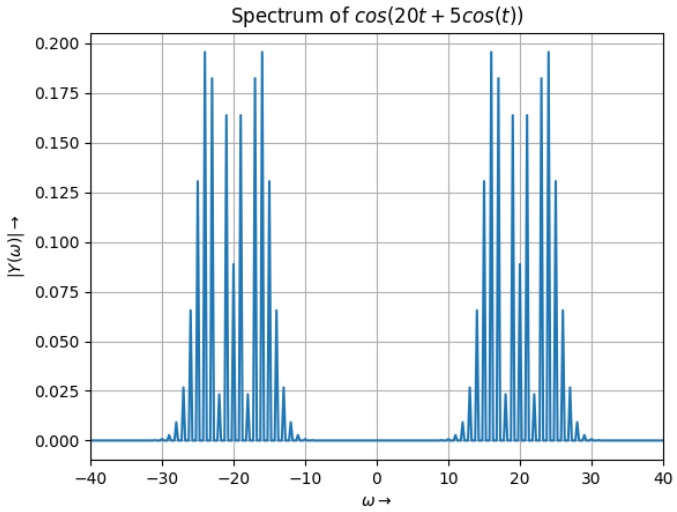

Figure 5: Spectrum of $\cos(20t + 5\cos(t))$

- It is observed that the plot is Phase modulated since phase of the signal, varying proportionally to amplitude of the message signal, is $\omega = 20$ and infinite side band frequencies which are produced by $5\cos t$, since $\cos(t)$ is infinitely long signal. But the strength of the side band frequencies either decays or is very small as they are away from the center frequency or carrier frequency component as we observe from the plot.

- 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, the carrier signal can represent either a *sine* or *cosine* depending on the phase contribution from $cos(t)$.

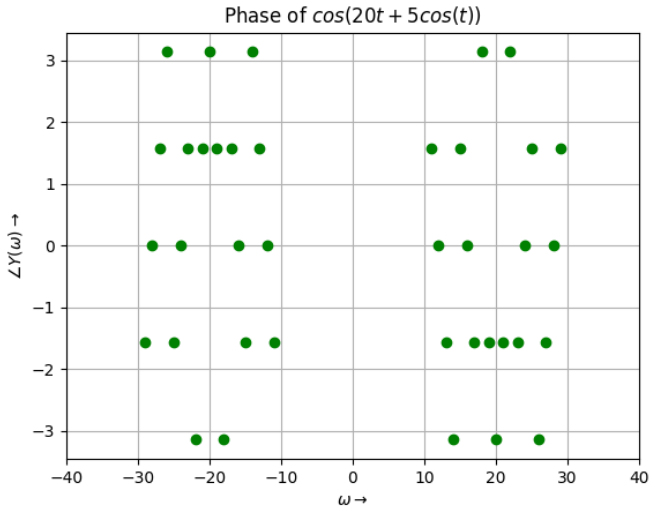

Figure 6: Phase plot of $\cos(20t + 5\cos(t))$

DFT of a Gaussian

- To 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.

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

- 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)**

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

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

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

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

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

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

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

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

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

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

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

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

- Hence, we need 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}

The python code to calculate the exact DFT of a Gaussian is as shown:

```
# The below piece piece of code is to find out the most accurate DFT of a Gaussian
# exp(-0.5t^2) when compared to its CTFT.
# Declaring all the necessary variables.
T = 2*pi
N = 128
iter = 0
tolerance = 1e-15
error = tolerance + 1

# This loop will calculate the DFT and also the error between the calculated and ac
# Only when the error is less than a tolerance value will the loop be terminated.
while True:

        t = linspace(-T/2,T/2,N+1)[:-1]
        w = N/T * linspace(-pi,pi,N+1)[:-1]
        y = exp(-0.5*t**2)
        iter = iter + 1

        Y = fftshift(fft(y))*T/(2*pi*N)
        Y_actual = (1/sqrt(2*pi))*exp(-0.5*w**2)
        error = mean(abs(abs(Y)-Y_actual))

        if error < tolerance:
                break

        T = T*2
        N = N*2

print(" Error: %g \n Interation: %g" % (error,iter))
print(" Best value for T: %g*pi \n Best value for N: %g"%(T/pi,N))

# Magnitude plot for the most accurate DFT of the Gaussian.
figure(8)
plot(w,abs(Y))
title(r"Spectrum of a Gaussian function")
ylabel(r"$|Y(\omega)|\rightarrow$")
xlabel(r"$\omega\rightarrow$")
grid(True)
xlim([-10,10])

show()
```

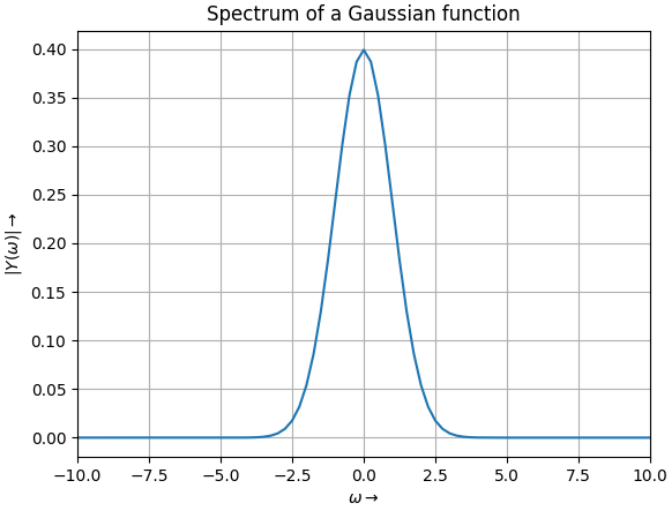

Figure 7: Spectrum of a Gaussian function

Results and Discussion :

- Accuracy of the DFT is: 2.91515e-17 and Iterations took: 3
  Best Window_size: 25.1327 (8π), Sampling_rate: 512
- As we observe the magnitude spectrum of $e^{-\frac{t^2}{2}}$ we see that it almost coincides with exact Fourier Transform plotted below with accuracy of $2.91515e^{-17}$
- To find the correct Window size and sampling rate, a While loop is used to minimize the error by increasing both window_size and sampling rate as we made assumption that when Window_size is large the sinc(wT) acts like impulse $\delta(\omega)$
- so we increase window_size, similarly sampling rate is increased to overcome aliasing problems when sampling the signal in time domain.

Conclusion

- Hence we analysed on 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 with tolerance upto 10^{-15}.
- We used fast Fourier transform method to compute DFT as it improves the computation time from $\mathcal{O}n^2 \rightarrow \mathcal{O}n\log_2(n)$.
- FFT works well for signals with samples in 2^k , as it divides the samples into even and odd and goes dividing further to compute the DFT.
- That's why we use no of samples in the problems above taken as powers of 2.