Project Report on

# Applications of Fourier Transform

Submitted by:

**Aneesh Panchal**

**2K20/A6/56**

Submitted to:

**Dr. Dinesh Udar**

Department of Applied Mathematics
Delhi Technological University

## Introduction:

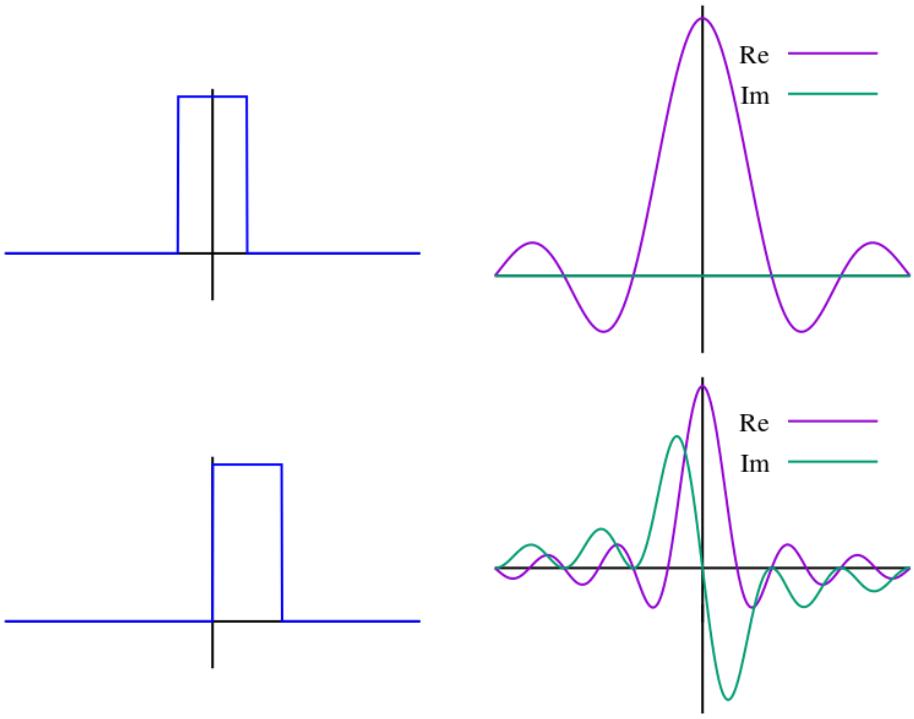
Fourier Transform is useful in the study of solution of partial differential equation to solve initial boundary value problems. A Fourier Transform when applied to partial differential equation reduces the number of independent variables by one. We mainly use Fourier Transform in signal &image processing. It is also useful in cell phones, LTI system & circuit analysis.

## Fourier Transform:

The Fourier Transform is a generalization of the Fourier series. It only applies to continuous & a periodic functions.

We defined Fourier Transform of a piecewise continuous & absolutely integrable function $f(x)$ by

$$F(\omega) = F\{f(x)\} = \frac{1}{\sqrt{2\Pi}} \int_{-\infty}^{\infty} f(x).e^{-i\omega x} dx$$




## Properties of Fourier Transform:

1. Linearity:
   The Fourier transform satisfies linearity & principle of superposition
   Consider 2 functions $f(x)\ and\ g(x)$
   $F[af(x) + bg(x)] = aF\big(f(x)\big) + bF(g(x))$

2. Shifting Property:
   $F[f(x - x_o)] = e^{-i\omega x_o}F(\omega)$

3. Differentiation:
   $F[f^n(x)] = (i\omega)^n F[f(x)]$

4. Modulation Property:
   $F\{f(x)cosat\} = \frac{1}{2}\{F_c(\omega + a) + F_c(\omega - a)\}$
   $F\{f(x)sinat\} = \frac{1}{2}\{F_s(\omega + a) - F_s(\omega - a)\}$

5. Convolution:
   Fourier transform makes the convolution of 2 signals into the product of their Fourier Transforms.

$$(f * g)(x) = \int_{-\infty}^{\infty} f(t)g(x-t)dt$$
$$F[f * g] = \sqrt{2\Pi}\, F(f).F(g)$$

## Fourier Sine and Cosine Transform:

Fourier transform can be written as,

$$F(\omega) = \frac{1}{\sqrt{2\Pi}} \int_{-\infty}^{\infty} f(x)[\cos(\omega x) - i\sin(\omega x)]dx$$

Fourier cosine transform is written as,

$$F_c(\omega) = \sqrt{\frac{2}{\Pi}} \int_0^{\infty} f(x)\cos(\omega x)dx$$

Fourier sine transform can be written as,

$$F_s(\omega) = \sqrt{\frac{2}{\Pi}} \int_0^{\infty} f(x)\sin(\omega x)dx$$

## Discrete Fourier Transform:

In mathematics, 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.

The Discrete Fourier Transform transforms a sequence of N complex numbers $\{x_n\} = x_0,\ x_1, \dots\dots, x_{n-1}$ into another sequence of complex numbers, $\{X_k\} = X_0,\ X_1, \dots\dots, X_{n-1}$ which is defined by

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{i2\Pi kn}{N}} = \sum_{n=0}^{N-1} x_n \left[cos\left(\frac{2\Pi kn}{N}\right) - i.sin\left(\frac{2\Pi kn}{N}\right)\right]$$

Where, $k = 0, \dots\dots, N-1$

## Applications of Fourier Transform:

- **Application to Initial Boundary Value Problem (IBVP) (Heat Equation)**

The solution of IBVP consists of a partial differential equation together with boundary & initial conditions can be solved by Fourier Transform method. Here we solve the heat equation analytically by using boundary condition. In this case PDEs reduces to ODEs in Fourier Transform which is solved.

Heat equation in one spatial dimension

$\frac{\partial T}{\partial t} = p\frac{\partial^2 T}{\partial x^2}$

Where p is thermal diffusivity

Open boundaries – $T(x,t)$ defined on

$-\infty < x < \infty$ and $t \geq 0$

Also require that $T(x,t) \to 0$ as $x \to \pm\infty$

Initial value problem: $T(x, t=0) = \Phi(x)$

Apply Fourier Transform to heat equation (at constant t)

$F[T_t] = F[pT_{xx}]$

$\frac{\partial F[T]}{\partial t} = -k^2 pF[T]$

Let $r(k,t) = F[T(x,t)]$

$\frac{\partial r(k,t)}{\partial t} = -k^2 pr(k,t)$

Solution of this ODE is

$r(k,t) = e^{-k^2 pt} r(k,0)$

Using initial value problem,

$F[\Phi(x)] = r(k,0)$

Using above equations we get,

$T(x,t) = F^{-1}[e^{-k^2 pt} F[\Phi(x)]]$

According to convolution theorem,

$\frac{1}{\sqrt{2\Pi}}(f * g) = F^{-1}[F(f).F(g)]$

$g = \Phi(x)$

For f we need to use inverse Fourier Transform of a Gaussian:

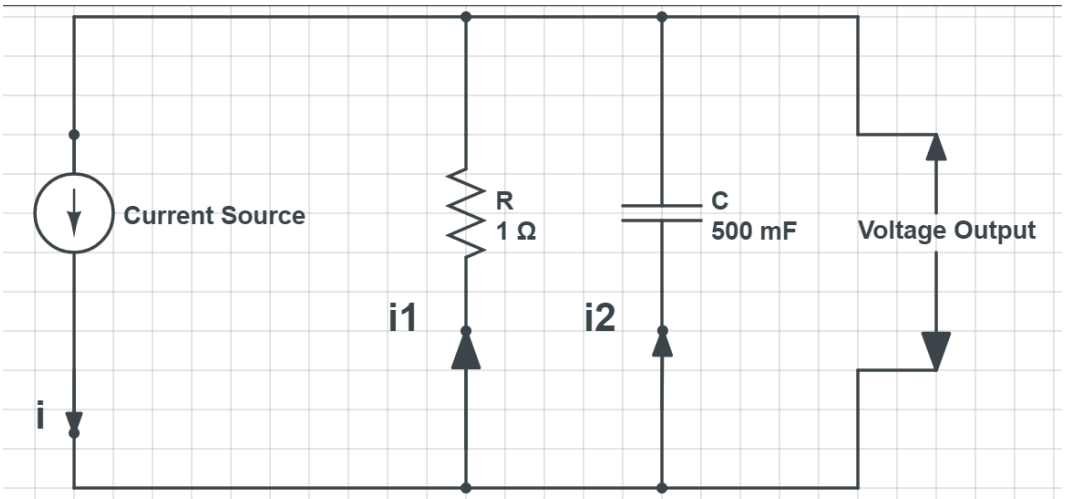
$$f = F^{-1}\left[e^{-k^2 pt}\right] = \frac{1}{\sqrt{2pt}} e^{-\frac{x^2}{4pt}}$$

$$T(x,t) = \frac{1}{\sqrt{2\Pi}}(f * g) = \frac{1}{\sqrt{4\Pi pt}} \int_{-\infty}^{\infty} e^{-\frac{(x-c)^2}{4pt}} . \Phi(c) dc$$

This is the fundamental solution of the heat equation with open boundaries for an initial condition $T(x, t = 0) = \Phi(x)$

## ➢ Circuit Analysis

There are many linear circuits used in Electronic engineering field .These circuits include various components like capacitor, inductor, resistor etc. Every Electronic circuit can be modelled using mathematical equations.




Example,

$i(t) = current\ souce$

$i_1(t) = current\ flowing\ through\ resistance$

$i_2(t) = current\ flowing\ through\ capacitor$

$v_o(t) = output\ voltage$

$i(t) = e^{-t}u(t)$

$i(t) = i_1(t) + i_2(t)$

$v_o(t) = i_2(t)$

$$e^{-t}u(t) = v_o(t) + \frac{1}{2}v_o(t)$$

By taking Fourier Transform

$$\frac{1}{j\omega+1} = v_o(j\omega) + \frac{1}{2}(j\omega)v_o(j\omega)$$

$$v_o(j\omega) = 2\frac{1}{(j\omega+1)(2+j\omega)}$$

$$v_o(j\omega) = 2[\frac{1}{1+j\omega} + \frac{-1}{2+j\omega}]$$

$$F^{-1}[v_o(j\omega)] = 2F^{-1}\{[\frac{1}{1+j\omega} + \frac{-1}{2+j\omega}]\}$$

$$v_o(t) = 2[e^{-t}u(t) - e^{-2t}u(t)]$$

- ## Signal Processing

The Fourier Transform is extensively used in the field of Signal Processing. In fact, the Fourier Transform is probably the most important tool for analyzing signals in that entire field.

A signal is any waveform (function of time). This could be anything in the real world - an electromagnetic wave, the voltage across a resistor versus time, the air pressure variance due to your speech (i.e. a sound wave), or the value of Apple Stock versus time. Signal Processing then, is the act of processing a signal to obtain more useful information, or to make the signal more useful.

- ## Image processing

Fourier transform is used in a wide range of applications such as image analysis, image filtering, image reconstruction and image compression.

The Fourier Transform is an important image processing tool which is used to decompose an image into its sine and cosine components.

The Fourier Transform is used if we want to access the geometric characteristics of a spatial domain image. Because the image in the Fourier domain is decomposed into its sinusoidal components, it is easy to examine or process certain frequencies of the image, thus influencing the geometric structure in the spatial domain.

## Fast Fourier Transform (FFT):

A Fast Fourier Transform (FFT) is an algorithm that computes the Discrete Fourier Transform (DFT) of a sequence.

Let $x_o, \dots\dots, x_{N-1}$ be complex numbers. The DFT is defined by the formula

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{i2\Pi kn}{N}}$$

$k = 0, \dots\dots, N-1$

Where $e^{i2\Pi/N}$ is a primitive Nth root of 1

Evaluating this definition directly requires $O(N^2)$ operations:

There are N outputs $X_k$, and each output requires a sum of N terms.

An FFT is any method to compute the same results in $O(N\ logN)$ operations.

All known FFT algorithms require $O(N\ logN)$ operations.

## Cooley – Tukey FFT Algorithm

The Cooley – Tukey algorithm is the most common fast Fourier transform (FFT) algorithm. It re - express the DFT of an arbitrary composite size $N = N_1N_2$ in terms of $N_1$ smaller DFTs of size $N_2$, recursively to reduce the computation time to O(N logN) for highly composite N.

### Working of Cooley – Tukey FFT Algorithm

In this algorithm 1 dimensional DFT can be viewed as 2 dimensional DFT or simply a matrix. The 1 dimensional array of length N is reinterpreted as 2 dimensional $N_1XN_2$ matrix stored in column major order i.e. 1st column 1st then 2nd column and so on.

1. Perform $N_1$ DFTs of size $N_2$.
2. Multiply by complex roots of unity. (Twiddle factors)
3. Perform $N_2$ DFTs of size $N_1$.

In general Cooley – Tukey algorithm indices are rewritten as

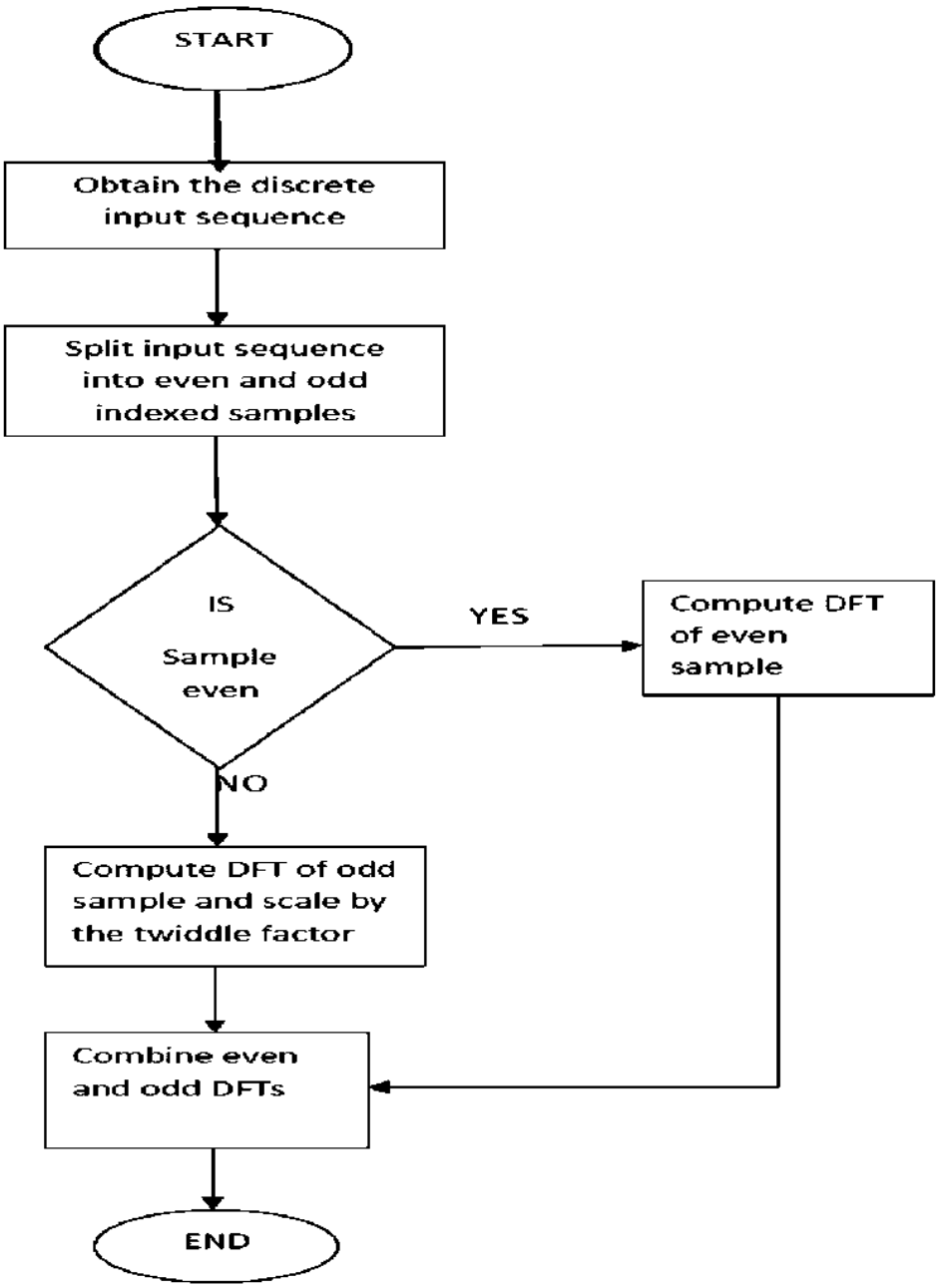
$k = N_2k_1 + k_2$

$n = N_1n_2 + n_1$

$$X_{N_2k_1+k_2} = \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} x_{N_1n_2+n_1}\, e^{-\frac{2\Pi i}{N_1N_2}.(N_1n_2+n_1)(N_2k_1+k_2)}$$

$$= \sum_{n_1=0}^{N_1-1}\left(\sum_{n_2=0}^{N_2-1} x_{N_1n_2+n_1} e^{-\frac{2\Pi i}{N_2}.n_2k_2}\right) e^{-\frac{2\Pi i}{N_1N_2}.n_1(N_2k_1+k_2)}$$

Where each inner sum is a DFT of size $N_2$, each outer sum is a DFT of size $N_1$.



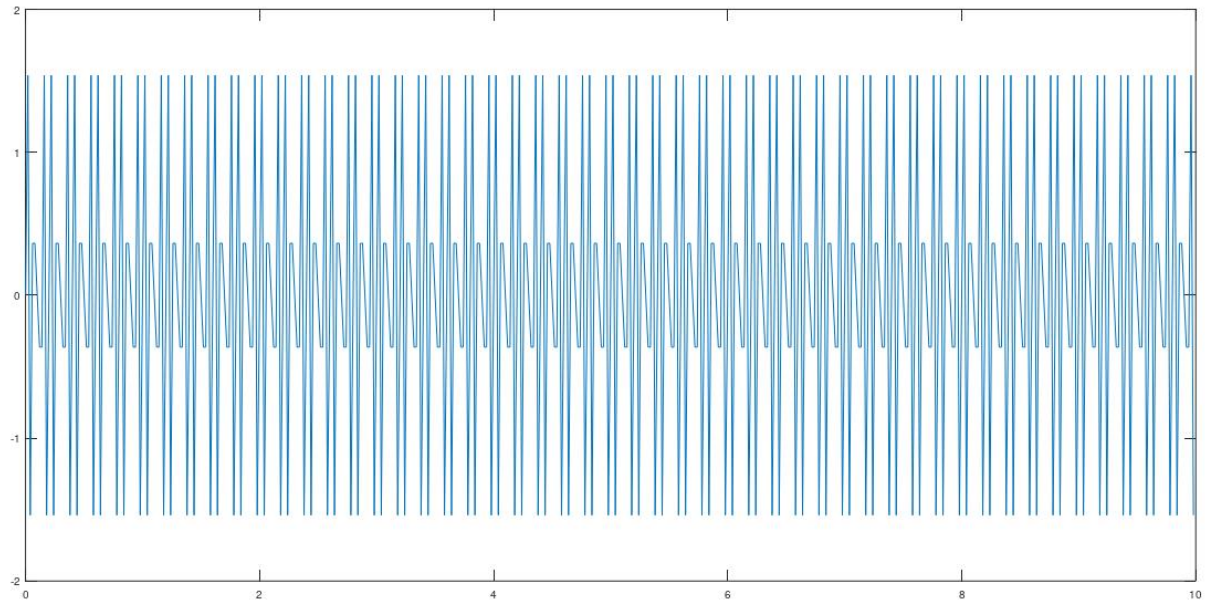
## Image Processing and Fourier Transform Simulation:

Fourier Transform breaks a function (signal) into an alternate representation (using sine and cosine).

Simply, Fourier Transform shows that any signal can be reconstructed by summing up individual sine waves of different frequencies.

### Fourier Transform using GNU Octave software

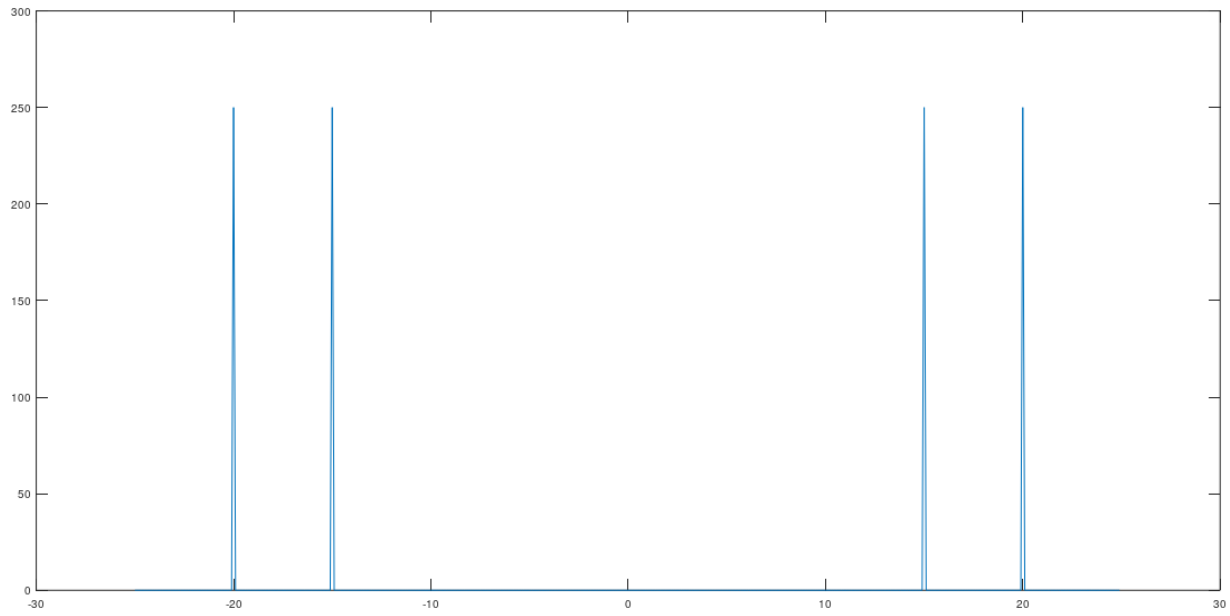
```
Ts = 1/50;
t = 0:Ts:10-Ts;
a = sin(2*pi*15*t) + sin(2*pi*20*t);
plot(t,a)
```




x axis is time (t) in sec

y axis is amplitude (a)

```
Ts = 1/50;
t = 0:Ts:10-Ts;
a = sin(2*pi*15*t) + sin(2*pi*20*t);

y = fft(a);
fs = 1/Ts;
f = (0:length(y)-1)*fs/length(y);

n = length(a);
fshift = (-n/2:n/2-1)*(fs/n);
yshift = fftshift(y);
plot(fshift,abs(yshift))
```




The transform also produces a mirror copy of the spikes, which correspond to the signal's negative frequencies.

Fourier Transform takes place using Fast Fourier transform (FFT).

We use fshift to move centre and visualize more clearly.

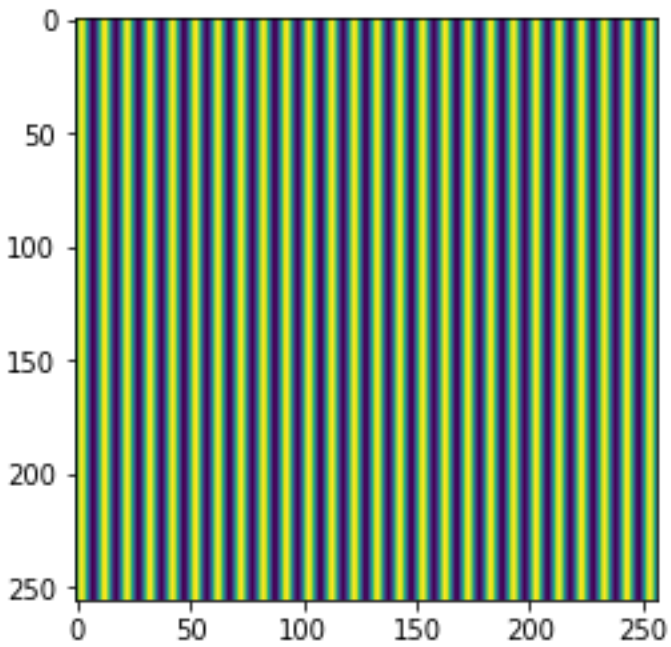
We clearly see 2 spikes in this graph due to 2 components of sine.

x axis is frequency in Hz

y axis is magnitude of frequency

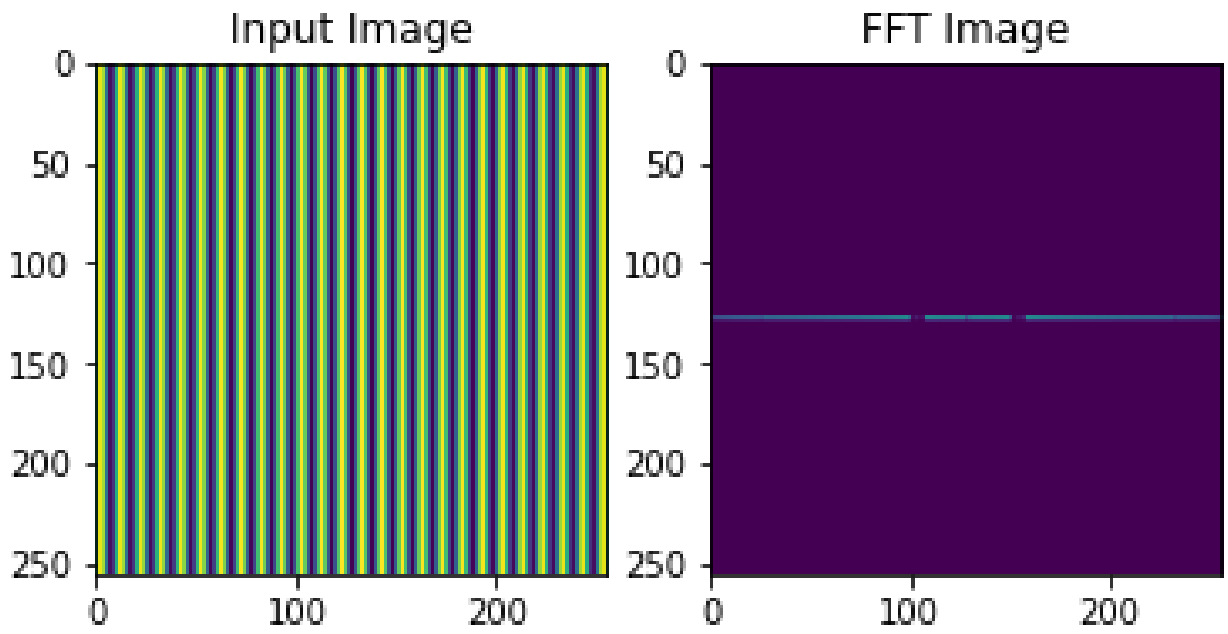
## Fourier Transform using Python CV2

```
import cv2
from matplotlib import pyplot as plt
import numpy as np

x = np.arange(256)
y = np.sin(2*np.pi*x/10)

y += max(y)

img = np.array([[y[j]*127 for j in range(256)]for i in range(256)],
                                    dtype = np.uint8)
plt.imshow(img)
```




First of all we simply create a function y which have sine function in it.
This is representation of y with respect to time.
Range of x and amplitude we take is 256.

```
f = np.fft.fft2(img)
fshift = np.fft.fftshift(f)
magnitude_spectrum = 20*np.log(np.abs(fshift))

plt.subplot(121)
plt.imshow(img.astype('uint8'))
plt.title('Input Image')

plt.subplot(122)
plt.imshow(magnitude_spectrum.astype('uint8'))
plt.title('FFT Image')

plt.show
```



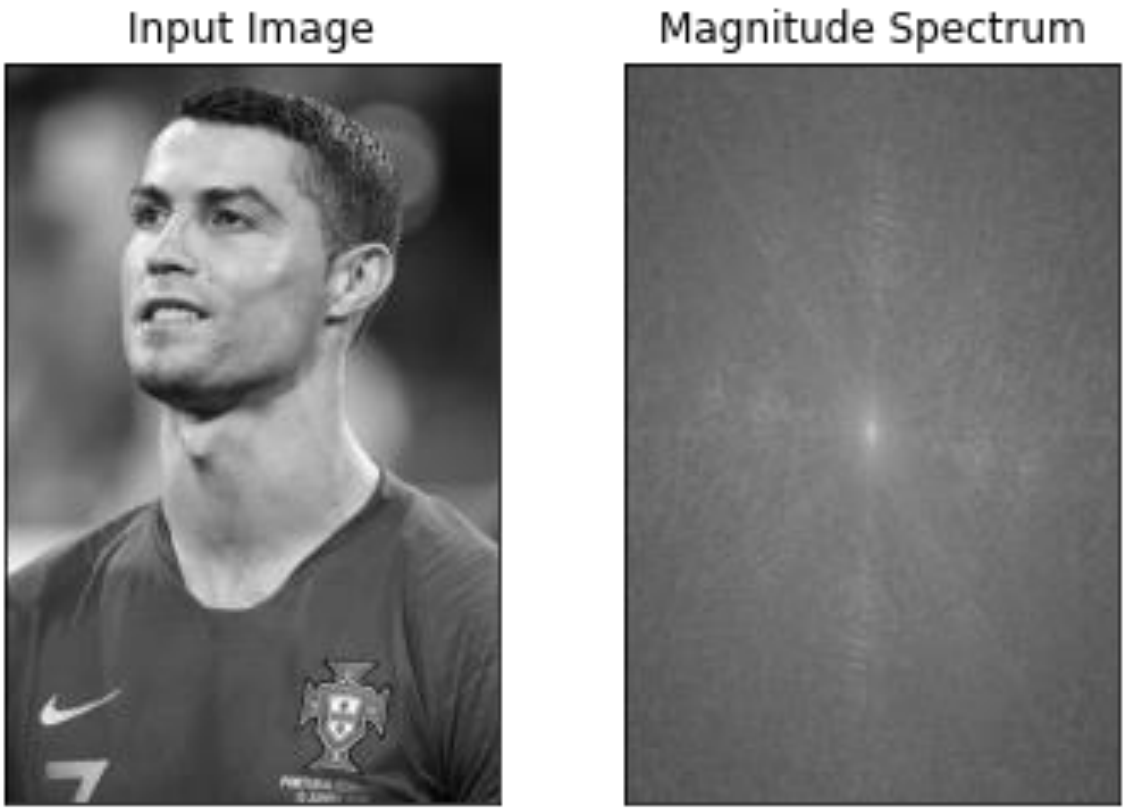

First of all we take fast Fourier transform of function y,
Then shift the centre from (0,0) to the middle of the output image (for better visualization).
fshift returns complex form and can not be displayed directly that's why we take absolute value.
Then, the logarithm compresses the range of values – larger peaks are scaled down more than smaller peaks for better visualization.

## Fourier Transform using Python CV2 (Practical Application)


**Cristiano Ronaldo, Portuguese Footballer**

**=>> Creating FFT image (frequency image of the given image)**

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

img = cv2.imread('/Cristiano_Ronaldo_2018.jpg',0)

dft = cv2.dft(np.float32(img),flags = cv2.DFT_COMPLEX_OUTPUT)
dft_shift = np.fft.fftshift(dft)

magnitude_spectrum = 20*np.log(cv2.magnitude(dft_shift[:,:,0],dft_shift[:,:,1]))

plt.subplot(121)
plt.imshow(img, cmap = 'gray')
plt.title('Input Image')
plt.subplot(122)
plt.imshow(magnitude_spectrum, cmap = 'gray')
plt.title('Magnitude Spectrum')
plt.show()
```




Magnitude spectrum is the frequency image of the corresponding input image.
We used gray scale here for better visualization.
We used 2D arrays for making magnitude spectrum.
Because [:,:,0] contains real part of image and [:,:,1] contains imaginary part.

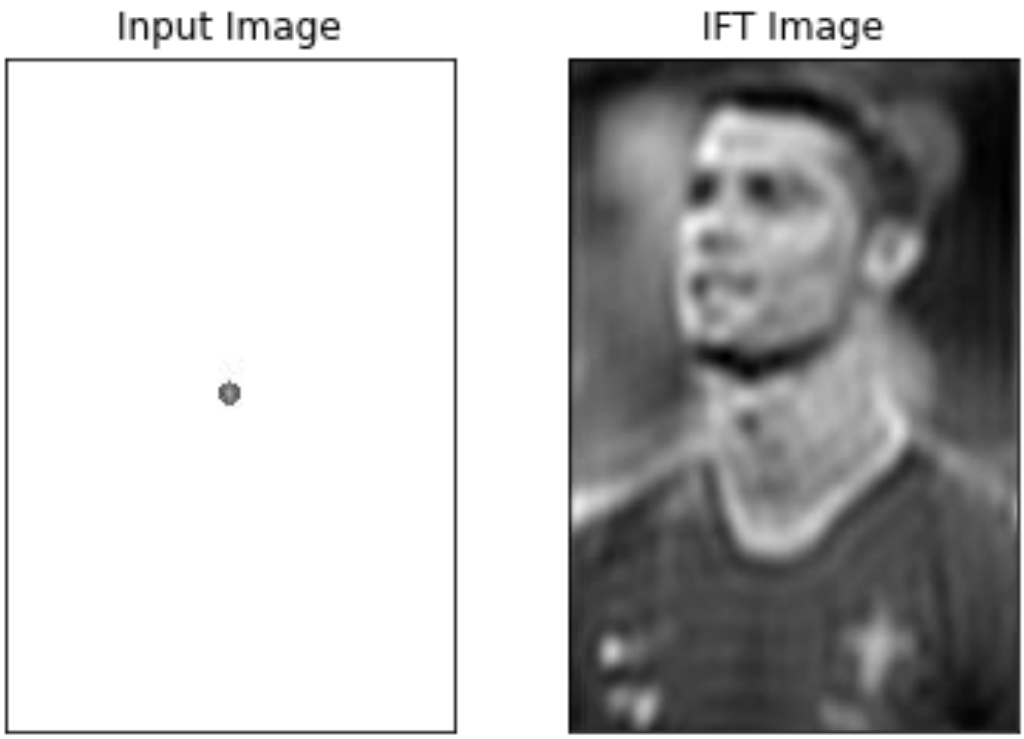
Central white spot represents the highest frequencies and the magnitude of frequencies decreases as we go away from the center point or from white spot.

**=>> Creating a circular mask for the Magnitude spectrum**

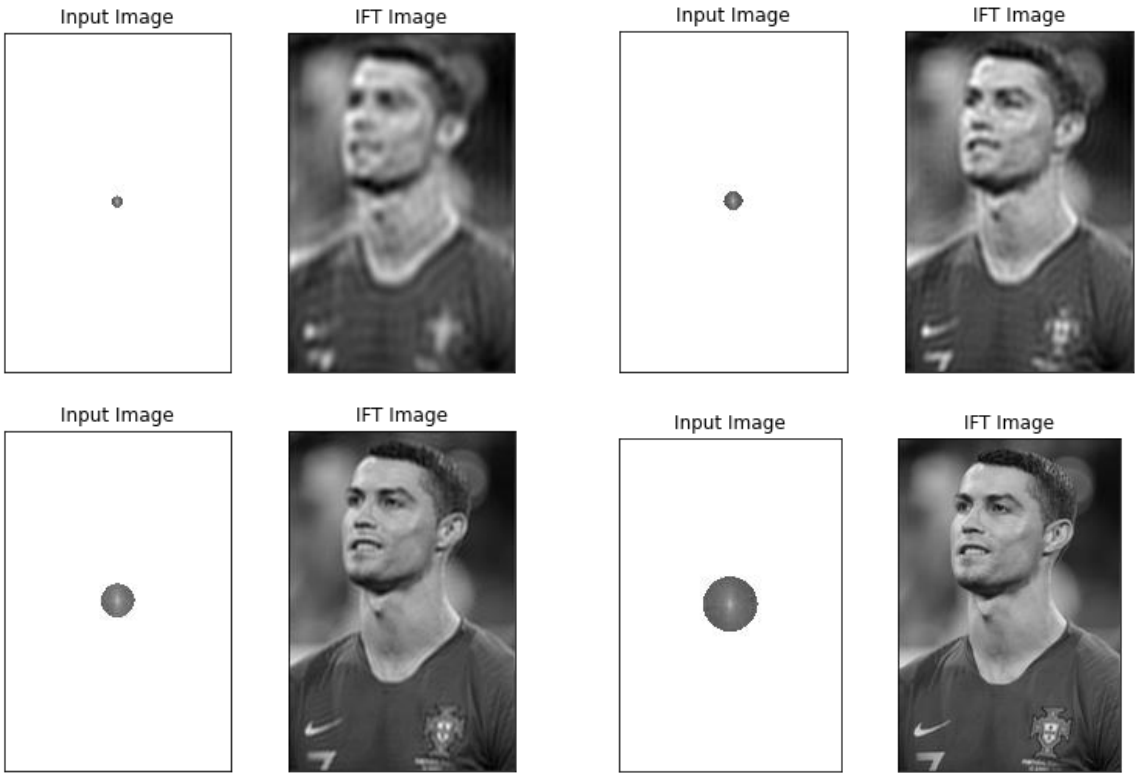
```
rows, cols = img.shape
crow,ccol = rows/2 , cols/2

mask = np.zeros((rows,cols,2),np.uint8)
r = 30
center = [crow,ccol]
x, y = np.ogrid[:rows,:cols]
mask_area = (x-center[0])**2 + (y-center[1])**2 <= r*r
mask[mask_area] = 1
```

**=>> Using Inverse FFT to get back the original image with the filter of mask**

```
fshift = dft_shift*mask

fshiftmask = 20*np.log(cv2.magnitude(fshift[:,:,0],fshift[:,:,1]))

f_ishift = np.fft.ifftshift(fshift)
img_back = cv2.idft(f_ishift)

img_back = cv2.magnitude(img_back[:,:,0],img_back[:,:,1])

plt.subplot(121),plt.imshow(fshiftmask, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(img_back, cmap = 'gray')
plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
plt.show()
```




If we only allow very few high frequency which lies inside the circular mask the corresponding image (using inverse Fourier Transforms) will looks blurry.
Here the image is blurry because we allow very few high frequencies in our Inverse DFT image.




The more we allow the high frequencies to pass the clearer image we get.

## Conclusion:

Fourier transforms is a generalization of Fourier series. Fourier transforms only applies to continues and periodic functions. Fourier transform applications include Solution of IBVP, Circuit analysis, Signal analysis and many more. Image processing is one of the most used applications of Fourier transforms in the field of Computational engineering. Fourier transforms is an important image processing tool which is used to decompose an image into its sine and cosine components. It allows us to operate and work with images in an efficient way.

## References:


- Wikipedia.org
- In.mathswork.com
- Github.com
- Stackoverflow.com
- Ijirset.com
- Youtube.com