MAT 465: Fourier Analysis

Project Title: Diffraction Patterns using Fourier Transformation

Group - 3

Members:

Enrollment Number	**Name**	**Programme**
AU2020126	Aum Raval	BS Physics
AU2120184	Freya Shah	BS CS
AU1940131	Vishwa Raval	BTech CSE

Background & Introduction:

The study of classical optics that utilizes Fourier transforms is known as Fourier optics. It considers the waveforms as a superposition of planes of waves. The theory is primarily based on the concepts of spatial correlation, convolution, as well as Fourier transformation. By using these concepts, it shows how the input of an optical field is transferred by the optical system to its corresponding output.

Fourier optics has very high computational potential and it deals with interesting questions. For example, suppose a beam of light is hitting a screen. The intensity at that particular point is known. Fourier optics asks the question, how the intensity of the light changes further down, away from the screen? In its path, the light can pass through multiple phenomena such as it can go through lenses, slits, etc.

The problem of this project deals only with the slits and the diffraction patterns. The basic understanding of slits and the diffraction pattern is given here. A slit is a little cut-out such that the light intensity is zero everywhere but non-zero at a particular place. Diffraction of light is defined as the bending of light around corners of the obstacles such that it spreads out and generates an illuminating pattern in the areas where a shadow is expected. For a diffraction pattern to actually occur, the slit has to be on the same order of magnitude as the wavelength of the light.

All the computation required here (to get the diffraction pattern further down the path), is related to Fourier transforms. If the initial intensity of light is known (for example 0 everywhere except 1 in the slit), then upon doing Fourier transforms, it can be computed what the diffraction pattern of the light looks like as the screen is moved farther away.

Problem Description:

Given the intensity of light at the source and the shape of the light source (slit), the aim of this project is to find the diffraction patterns down the line at some distance from the source, by utilizing a Fourier Analysis Technique. In addition to that, the slit shape can also be approximated given the diffraction patterns using the same technique.

The different types of slits used for the experiment include single slit, double slit, triangular, and square slits.

Fourier Analysis Techniques for the solution (Why and How):

In this project, specifically, the Fourier optics theory is utilized. The Fourier transform of a function of the intensity of light at the slit gives the diffraction pattern at various distances. And given the diffraction pattern, the inverse Fourier transform gives an approximated shape of the slit (light source).

The theory of scalar Fourier Optics deals with a function of intensity of light $U(x, y, z, t)$ that satisfies

$$\left(\nabla^2 - \frac{1}{c^2}\frac{\partial^2}{\partial t^2}\right) U(x, y, z) = 0 \qquad (1)$$

Where $|U|^2$ is the intensity of the light. U can interpreted as a dominant component of either of the magnetic or electric field.

For light of frequency $\omega = 2\pi f$, using the variable separable method $U = u(x, y, z)e^{-i\omega t}$ leads to the equation

$$\nabla^2 u + k^2 u = 0; \qquad k = \omega / c = \frac{2\pi f}{c} = \frac{2\pi}{\lambda} \qquad (2)$$

This is the Helmholtz Equation, where $|u|^2$ represents the intensity of light at a given point. To solve this equation, we will again use the variable separable method. Representing u_s (since u is separable) as a product of a function of ‘x’, a function of ‘y’, a function of ‘z’:

$$u_s(x, y, z) = f_x(x) \ X \ f_y(y) \ X \ f_z(z)$$

The general solution for the problem will not be separable but it can be written as the sum of all separable solutions

$$\nabla^2 u_s = \frac{\partial^2 u_s}{\partial x^2} + \frac{\partial^2 u_s}{\partial y^2} + \frac{\partial^2 u_s}{\partial z^2} \qquad (3)$$

$$f''_x(x) f_y(y) f_z(z) + f''_y(y) f_x(x) f_z(z) + f''_z(z) f_y(y) f_x(x) + k^2 f_x(x) f_y(y) f_z(z) = 0 \qquad (4)$$

On rearranging:

$$\frac{f''_x(x)}{f_x(x)} + \frac{f''_y(y)}{f_y(y)} + \frac{f''_z(z)}{f_z(z)} + k^2 = 0 \qquad (5)$$

Since all the quotients are dependent on the variables that are mutually independent, each quotient in the equation above has to be constant. To prove this, let us assume that the first quotient is not a constant, and it is a function of ‘x’. As no other term in the equation has any dependence on the variable ‘x’, the first term should also be independent of ‘x’, i.e. it has to be a constant. This constant is denoted as k_x^2. Similar reasoning in terms of ‘y’ and ‘z’ leads to three ordinary differential equations for the f_x, f_y and f_z:

$$\frac{\mathrm{d}^2 f_x(x)}{\mathrm{d}x^2} + k_x^2 f_x(x) = 0 \tag{6}$$

$$\frac{\mathrm{d}^2 f_y(y)}{\mathrm{d}x^2} + k_y^2 f_y(y) = 0 \tag{7}$$

$$\frac{\mathrm{d}^2 f_z(z)}{\mathrm{d}z^2} + k_z^2 f_z(z) = 0 \tag{8}$$

Given the condition,

$$k_x^2 + k_y^2 + k_z^2 = k^2 \tag{9}$$

The solution of the above 3 differential equations in complex exponential form will be:

$$u_s(x, y, z) = Ae^{ik_x x}e^{ik_y y}e^{ik_z z} \tag{10}$$

$$u_s(x, y, z) = Ae^{i(k_x x + k_y y)}e^{\pm iz\sqrt{k^2 - k_x^2 - k_y^2}} \tag{11}$$

To get the general solution, we need to take a linear combination of all separable solutions (infinitely many of them so need an integral as a sum):

$$u(x, y, z) = \int\int_{-\infty}^{+\infty} u_s(x, y, z, k_x, k_y)\, dk_x\, dk_y \tag{12}$$

Since each separable solution can be represented by its value of k_x and k_y, we can denote the corresponding amplitude as $A(k_x,\ k_y)$:

$$u(x,y,z) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} A(k_x,k_y)e^{i(k_x x+k_y y)}e^{\pm iz\sqrt{k^2-k_x^2-k_y^2}}\,dk_x\,dk_y \tag{13}$$

At z=0,

$$u(x,y,0) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} A(k_x,k_y)e^{i(k_x x+k_y y)}\,dk_x\,dk_y \tag{14}$$

This implies that $A(k_x, k_y)$ is the Fourier transform of u at z=0.

$$A(k_x,k_y) = \mathcal{F}[u(x,y,0)] \tag{15}$$

The equation

$$u(x,y,z) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} A(k_x,k_y)e^{i(k_x x+k_y y)}e^{\pm iz\sqrt{k^2-k_x^2-k_y^2}}\,dk_x\,dk_y \tag{16}$$

Is the inverse Fourier transform of $A(k_x,k_y)e^{-iz\sqrt{k^2-k_x^2-k_y^2}}$

Therefore,

$$u(x,y,z) = \mathcal{F}^{-1}\left[A(k_x,k_y)e^{\pm iz\sqrt{k^2-k_x^2-k_y^2}}\right] \tag{18}$$

The negative sign is chosen as the wave is considered to be moving through the slit.

Using the equations 15 and 18, everything can be solved for given the information about

1) k, the wavelength of the light
2) $u(x, y, z = 0)$, which is given by the distribution of light as it exits the slit.

Tools used:

We have implemented the above theory of fourier transform and inverse fourier transform in python. For which, the following libraries are used:

fft2 - For computing 2 - dimensional Fourier transform

ifft2 - For Inverse fourier transform

fftfreq - When we have a Fourier transform of a function and we know what it is in a spatial domain. It will get us the corresponding frequencies in the fourier domain as well

fftshift - To shift the zero-frequency component to the center of the spectrum.

pint - To deal with units

matplotlib - To make some animations

Code:

```
# Importing libraries needed
import numpy as np
import scipy as sp
from scipy.fft import fft2
from scipy.fft import ifft2
from scipy.fft import fftfreq
from scipy.fft import fftshift
import imageio

import matplotlib.pyplot as plt
from matplotlib.colors import LinearSegmentedColormap
from matplotlib import animation
from matplotlib.animation import PillowWriter
import pint

u = pint.UnitRegistry()
```

```
#Defining a function that computes fourier transform and inverse fourier transform
def compute_U(U0, xv, yv, lam, z):
    A = fft2(U0) #fourier transform of U
    kx = 2*np.pi * fftfreq(len(x), np.diff(x)[0]) #angular frequency
    kxv, kyv = np.meshgrid(kx,kx)
    k = 2*np.pi/lam
    return ifft2(A*np.exp(1j*z*np.sqrt(k**2-kxv**2-kyv**2))) #taking inverse fourier
transform
```

Single Slit Experiment

```
D = 0.1 * u.mm
lam = 660 * u.nm
```

```
x = np.linspace(-2,2,1600) * u.mm
xv, yv = np.meshgrid(x, x) #defining the grid

U0 = (np.abs(xv)< D/2) * (np.abs(yv)<0.5*u.mm) #defining the slit
U0 = U0.astype(float)
```

```
# Defining the slit
plt.figure(figsize=(5,5))
plt.pcolormesh(xv,yv,U0)
plt.xlabel('X-Position [mm]')
plt.ylabel('Y-Position [mm]')
plt.show()
```

```
#Diffraction Pattern
A = fft2(U0) #fourier coefficients of U
kx = fftfreq(len(x), np.diff(x)[0]) * 2 * np.pi
kxv, kyv = np.meshgrid(kx,kx)

plt.figure(figsize=(5,5))
plt.pcolormesh(fftshift(kxv.magnitude), fftshift(kyv.magnitude), np.abs(fftshift(A)))
plt.xlabel('$k_x$ [mm$^{-1}$]')
plt.ylabel('$k_y$ [mm$^{-1}$]')
plt.xlim(-100,100)
plt.ylim(-100,100)
plt.show()
```

```
k = 2*np.pi / (lam)
d = 3* u.cm #screen distance from the slit

U = compute_U(U0, xv, yv, lam, z=d)
```

```
# Inverse fourier transform to get back the original slit
plt.figure(figsize=(5,5))
plt.pcolormesh(xv,yv,np.abs(U), cmap='inferno')
plt.xlabel('$x$ [mm]')
plt.ylabel('$y$ [mm]')
plt.show()
```

```
# u vs x plot
```

```
m  = np.arange(1,5,1)
x_min = np.sqrt(m**2 * lam**2 * d**2 / (D**2 - m**2 * lam**2)).to('mm')
plt.plot(x, np.abs(U)[250])
[plt.axvline(x.magnitude, ls='--', color='r') for x in x_min]
[plt.axvline(-x.magnitude, ls='--', color='r') for x in x_min]
plt.xlabel('$x$ [mm]')
plt.ylabel('$u(x,y,z)$ [sqrt of intensity]')
plt.show()
```

Double Slit Experiment

```
S = 0.2*u.mm
D = 0.05*u.mm
x = np.linspace(-4,4,3200) * u.mm
xv, yv = np.meshgrid(x, x)

U0 = (np.abs(xv-S/2)< D/2) * (np.abs(yv)<2*u.mm) + (np.abs(xv+S/2)< D/2) *
(np.abs(yv)<2*u.mm)
U0 = U0.astype(float)
```

```
plt.figure(figsize=(5,5))
plt.pcolormesh(xv,yv,U0)
plt.xlabel('X-Position [mm]')
plt.ylabel('Y-Position [mm]')
plt.show()
```

```
#Diffraction Pattern
A = fft2(U0) #fourier coefficients of U
kx = fftfreq(len(x), np.diff(x)[0]) * 2 * np.pi
kxv, kyv = np.meshgrid(kx,kx)
```

```
plt.figure(figsize=(5,5))
plt.pcolormesh(fftshift(kxv.magnitude), fftshift(kyv.magnitude), np.abs(fftshift(A)))
plt.xlabel('$k_x$ [mm$^{-1}$]')
plt.ylabel('$k_y$ [mm$^{-1}$]')
plt.xlim(-100,100)
plt.ylim(-100,100)
plt.show()
```

```
U = compute_U(U0, xv, yv, lam, z=5*u.cm)
```

```
plt.figure(figsize=(5,5))
plt.pcolormesh(xv,yv,np.abs(U), cmap='inferno')
plt.xlabel('X-Position [mm]')
plt.ylabel('Y-Position [mm]')
plt.show()
```

```
# u vs x plot
central_line = np.abs(U)[250]

plt.plot(x, central_line)
plt.xlabel('$x$ [mm]')
plt.ylabel('$u(x,y,z)$ [sqrt of intensity]')
plt.grid()
```

Triangular Slit

```
S = 0.5*u.mm
D = 0.05*u.mm
x = np.linspace(-4,4,3200) * u.mm
xv, yv = np.meshgrid(x, x)

U0 =  (np.abs(xv+(yv)/2+S/2)< D/2) * (np.abs(yv)<1*u.mm) + (np.abs(xv-(yv)/2-S*1.4)<
D/2) * (np.abs(yv)<1*u.mm)+(np.abs(yv-2*S)< D/2) * (np.abs(xv-S/2)<1*u.mm)
U0 = U0.astype(float)
```

```
plt.figure(figsize=(5,5))
plt.pcolormesh(xv,yv,U0)
plt.xlabel('X-Position [mm]')
plt.ylabel('Y-Position [mm]')
plt.show()
```

```
A = fft2(U0)
kx = fftfreq(len(x), np.diff(x)[0]) * 2 * np.pi
kxv, kyv = np.meshgrid(kx,kx)
```

```
plt.figure(figsize=(5,5))
plt.pcolormesh(fftshift(kxv.magnitude), fftshift(kyv.magnitude), np.abs(fftshift(A)))
plt.xlabel('$k_x$ [mm$^{-1}$]')
plt.ylabel('$k_y$ [mm$^{-1}$]')
plt.xlim(-100,100)
plt.ylim(-100,100)
plt.show()
```

```
U = compute_U(U0, xv, yv, lam, z=5*u.cm)
```

```
plt.figure(figsize=(5,5))
plt.pcolormesh(xv,yv,np.abs(U), cmap='inferno')
plt.xlabel('X-Position [mm]')
plt.ylabel('Y-Position [mm]')
plt.show()
```

```
central_line = np.abs(U)[300]

plt.plot(x, central_line)
plt.xlabel('$x$ [mm]')
plt.ylabel('$u(x,y,z)$ [sqrt of intensity]')
plt.grid()
```

Square Slit:

```
S = 0.5*u.mm
D = 0.05*u.mm
x = np.linspace(-4,4,3200) * u.mm
xv, yv = np.meshgrid(x, x)

U0 =  ((np.abs(yv-5*S)< D/2) * (np.abs(xv-S/2)<1*u.mm)+(np.abs(yv-S)< D/2) *
(np.abs(xv-S/2)<1*u.mm)+(np.abs(xv-2.5*S)< D/2) *
(np.abs(yv-3*S)<1*u.mm)+(np.abs(xv+1.5*S)< D/2) * (np.abs(yv-3*S)<1*u.mm))
U0 = U0.astype(float)
```

```
plt.figure(figsize=(5,5))
plt.pcolormesh(xv,yv,U0)
plt.xlabel('X-Position [mm]')
plt.ylabel('Y-Position [mm]')
plt.show()
```

```
A = fft2(U0)
kx = fftfreq(len(x), np.diff(x)[0]) * 2 * np.pi
kxv, kyv = np.meshgrid(kx,kx)
```

```
plt.figure(figsize=(5,5))
plt.pcolormesh(fftshift(kxv.magnitude), fftshift(kyv.magnitude), np.abs(fftshift(A)))
plt.xlabel('$k_x$ [mm$^{-1}$]')
plt.ylabel('$k_y$ [mm$^{-1}$]')
plt.xlim(-100,100)
plt.ylim(-100,100)
plt.show()
```

```
U = compute_U(U0, xv, yv, lam, z=5*u.cm)
```

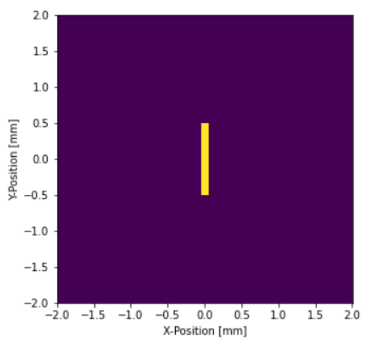
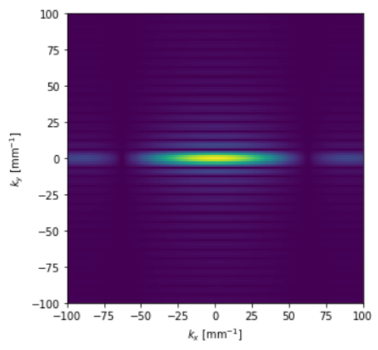
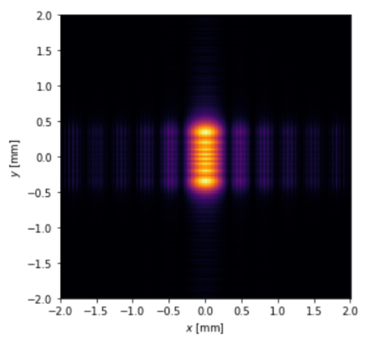
```
plt.figure(figsize=(5,5))
plt.pcolormesh(xv,yv,np.abs(U), cmap='inferno')
plt.xlabel('X-Position [mm]')
plt.ylabel('Y-Position [mm]')
plt.show()
```

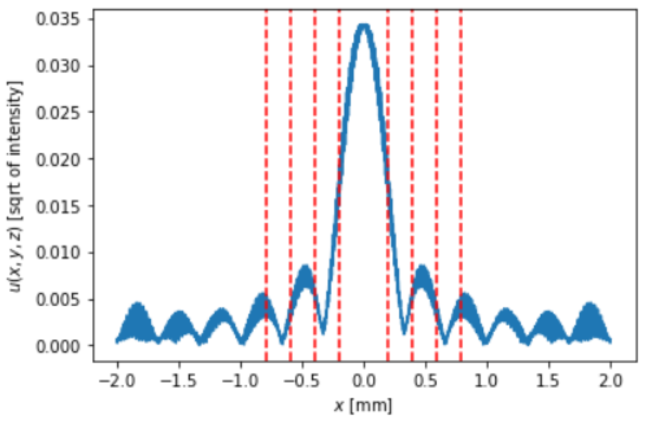
```
central_line = np.abs(U)[300]

plt.plot(x, central_line)
plt.xlabel('$x$ [mm]')
plt.ylabel('$u(x,y,z)$ [sqrt of intensity]')
plt.grid()
```

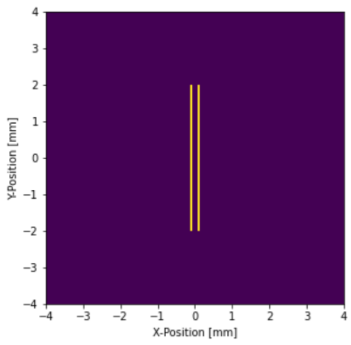
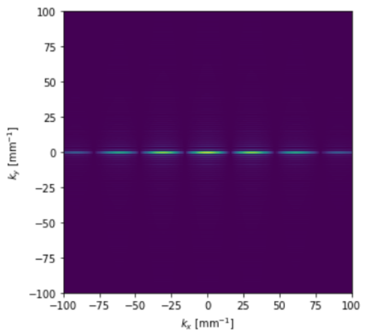
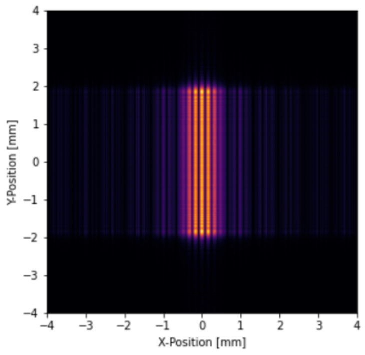
Results & Observations:

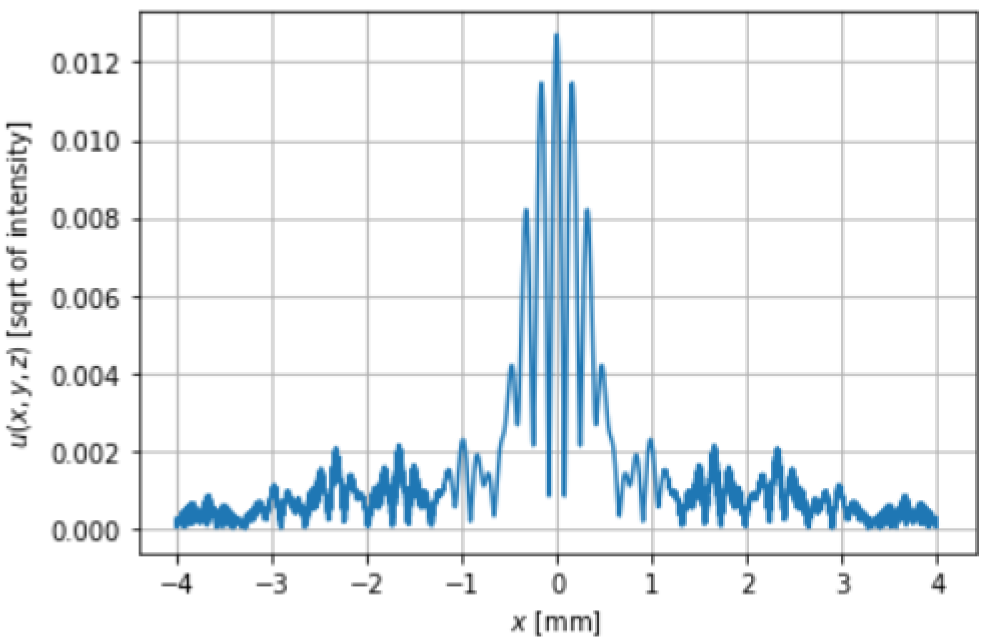
1) Single Slit

		
Original Slit	Diffraction Pattern	Inverse process (Getting the approximate slit shape from the diffraction pattern)

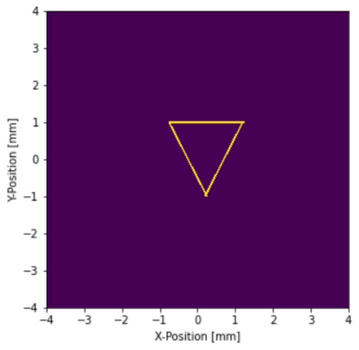
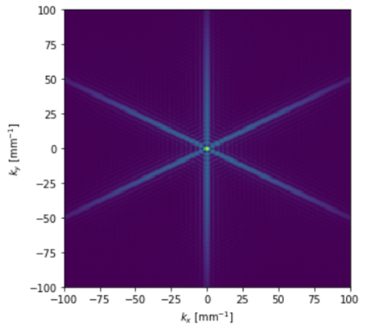
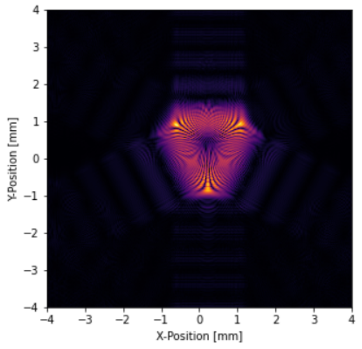


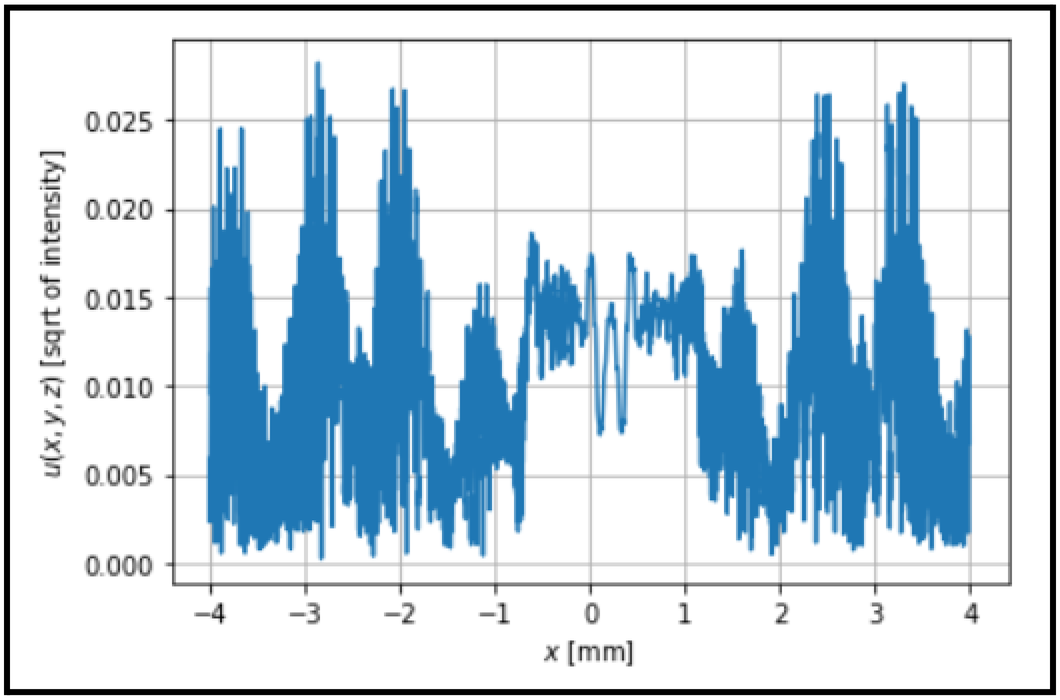
2) Double Slit

		
Original Slit	Diffraction Pattern	Inverse process (Getting the approximate slit shape from the diffraction pattern)

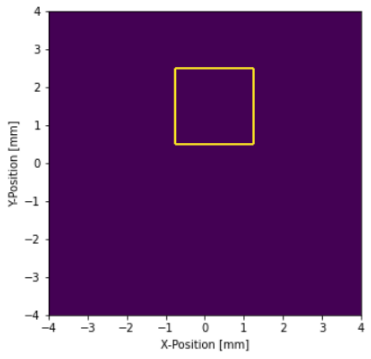
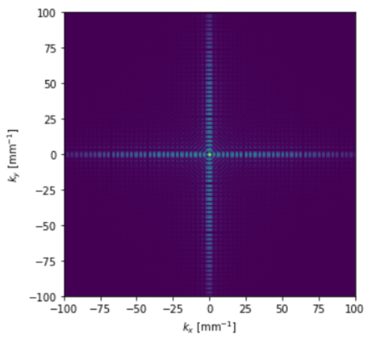
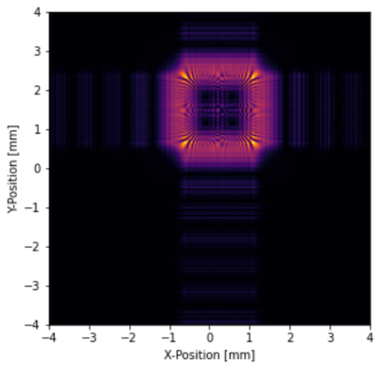


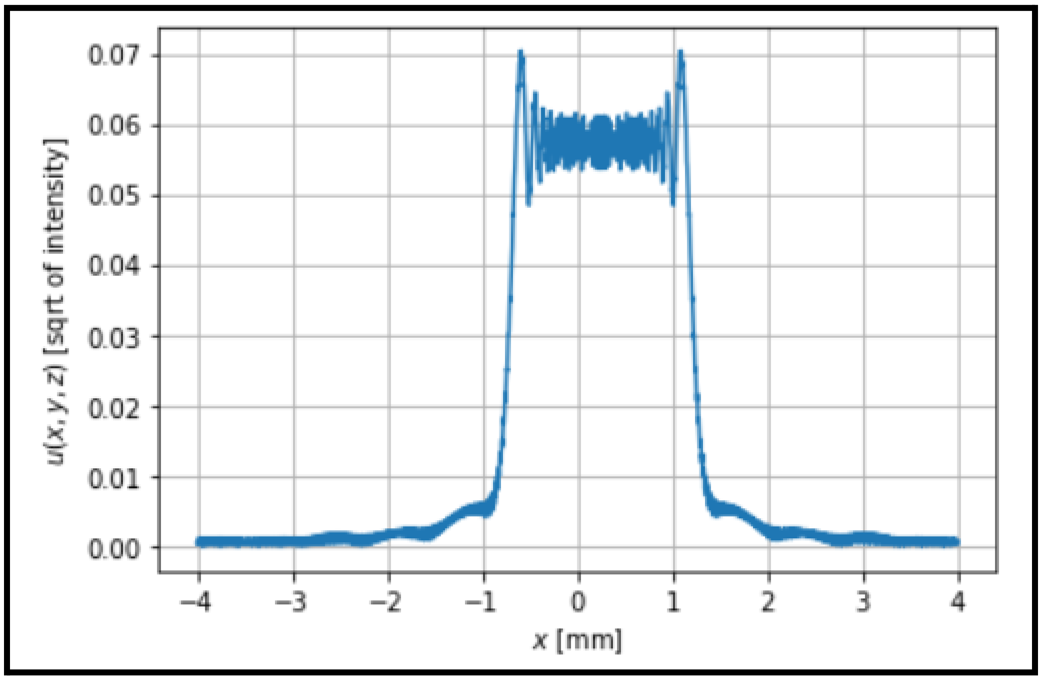
3) Triangular Slit

		
Original Slit	Diffraction Pattern	Inverse process (Getting the approximate slit shape from the diffraction pattern)

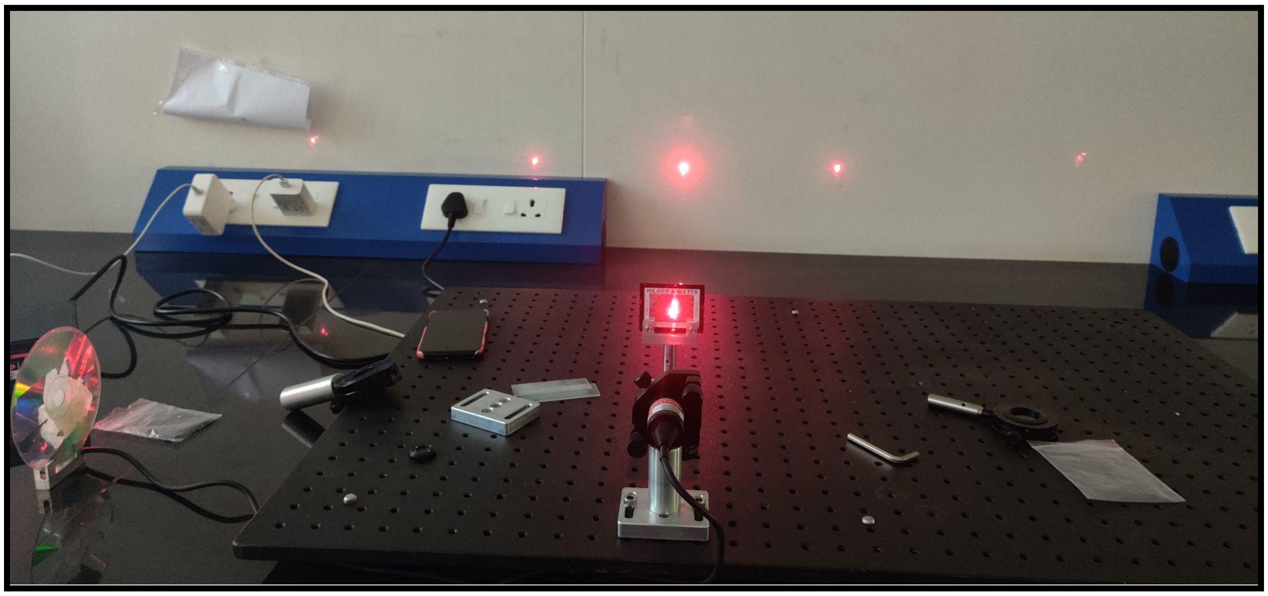


4) Square Slit

		
Original Slit	Diffraction Pattern	Inverse process (Getting the approximate slit shape from the diffraction pattern)



5) Experiment in the Lab (Diffraction pattern using diffraction grating):



Conclusion:

We were able to calculate the approximate diffraction pattern we would obtain when we pass the light through a slit using fourier analysis and then plotting the graph using the python code. Along with that we also plotted the intensity distribution with respect to the distance as shown in the above section. We performed the actual experiment in the lab and it was wonderful to see that our analytical, numerical and lab results match each other.

References:

[1] Abramowitz, Milton; Stegun, Irene, eds. (1964). Handbook of Mathematical functions with Formulas, Graphs and Mathematical Tables. New York: Dover Publications. ISBN 978-0-486-61272-0.

[2] Goodman, Joseph (2005). Introduction to Fourier Optics (3 ed.). Roberts & Company Publishers. ISBN 0-9747077-2-4. Retrieved 2017-10-28.