Fresnel Diffraction

Imad Abzioud

Introduction

The aim of the exercise was to model the intensity of light due to the Fresnel diffraction. The diffraction occurs when light rays from a square aperture hit a screen of similar scale. The square aperture acts like a slit in the single slit experiment, causing the light to interfere constructively and destructively, creating a diffraction pattern. A python programme was written to map the intensity of the light diffracted against the axes of the projection screen in both one dimension and two dimensions. This was done by defining the function for the Fresnel diffraction and integrating it using Simpson's composite 1/3 rule:

(eq.1) $$\int_{x_0}^{x_N} f(x)dx \approx f_{N-2} + 4f_{N-1} + f_N \quad + O(h^5)$$

Simpson's rule of integration works much like the trapezium rule for integrating, only instead of approximating the integrand by a series of straight-line segments that form several trapeziums, the integrand is approximated by a series of quadratic functions defined by three adjacent points. This makes the error in finding the area much smaller, as less of the area is ignored when computed as a quadratic of similar shape to the integrand at a certain point.

Part A

The one-dimensional Fresnel equation two had to take values of the screen co-ordinates x, the number of integration steps N, distance to the screen z, the wavelength of light λ, and the lower and upper integration limits a and b respectively. N numbers in python will return a value of N + 1. This is problematic because Simpson's rule works best for an even number of integers, so it was necessary to ensure the number of intervals for integration was an even number.

(eq.2) $$E(x,z) = \frac{kE_0}{2\pi z}\int_{x_1'}^{x_2'} \exp\left[\frac{ik}{2z}(x-x')^2\right] \mathrm{d}x'$$

Once the function was defined and integrated, it was plotted with a fixed x value using Pylab to produce a line graph of intensity against position. For each x, it was necessary to generate a list of evenly spaced x' values between the limits a and b. This was done using NumPy arrays. The sum of the equation f(x') was then computed and multiplied with the correct coefficients for even, odd and end terms. Even values were multiplied by two, odd values were multiplied by four and the beginning and end values were simply multiplied by one. An example of the graph for one dimensional Fresnel diffraction can be seen in figure 1.

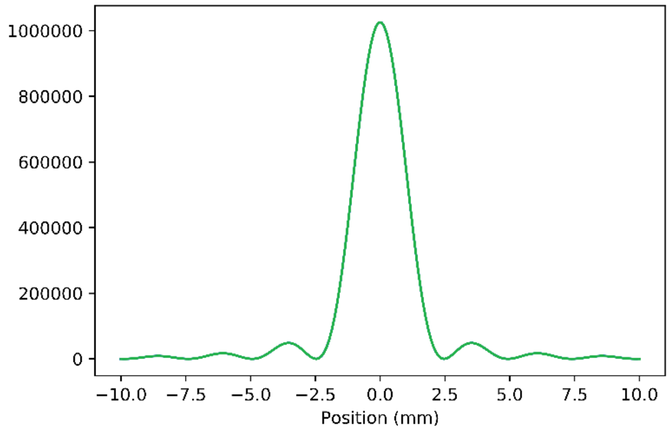

Figure 1, Graph of Fresnel diffraction in one dimension, at z = 2cm.

Figure 1 shows a gaussian curve similar to that of a Fourier transform, with a central peak and several smaller artefacts from the positive interference of light, occurring at integer intervals of λ or wavelength. These represent the fringes of light on the diffraction screen due to interference.

Part B

This part involved exploring the effect of changing the aperture width (x'_1 and x'_2) and the screen distance z on the one-dimensional plot. When the screen distance z was increased, the resulting graph showed a gaussian curve similar to figure one but much sharper, with one main peak and several smaller peaks besides. As the screen distance was decreased, the graph showed a much more intense and curved peak of greater than 10^{10} relative intensity, which is to be expected, as much more light was reaching the screen due to the inverse square law.

The screen size had to be adjusted for the smaller value of distance z, as the curve produced was much too large to fit on the original screen. This is due to most of the light scattering sooner than would have for a greater distance and providing less space for the light to diffract. This produced one large intense peak close to the aperture, and fewer, smaller much less intense peaks around it. Examples can be seen in figures two and three.

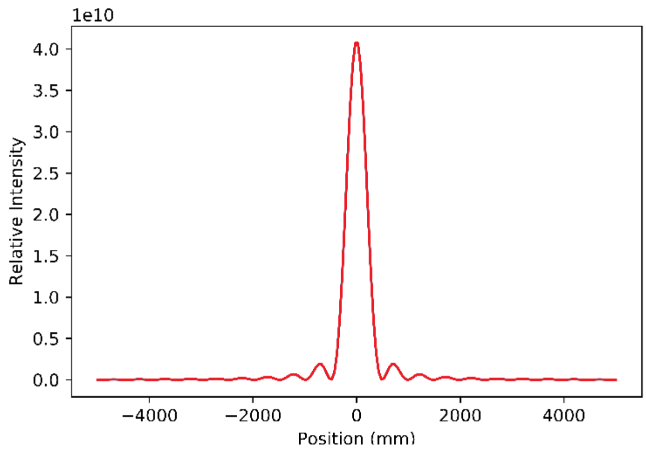

Figure 2, Graph of aperture/screen distance z = 0.1mm.

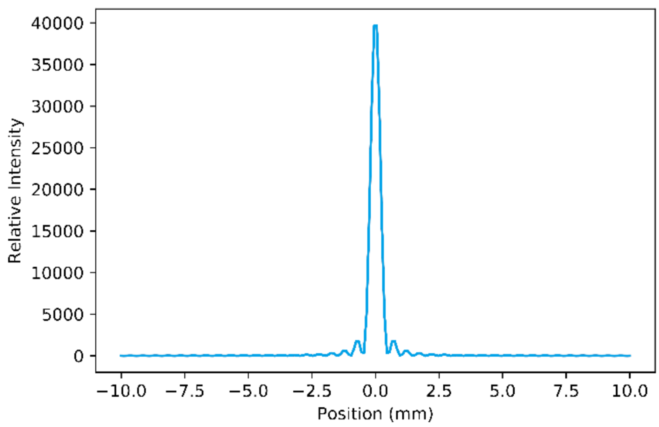

Figure 3, Graph of aperture/screen distance z = 10mm.

A smaller aperture width produces a lower relative intensity because less light is passing through the aperture, whereas for a wider aperture width the intensity peaks were much greater. The smaller aperture produced another gaussian graph as shown in figure four, similar to the previous ones seen in figures one to three, but a larger aperture produced a much more interesting graph, which can be found in figure five.

For the larger aperture, there were two central peaks instead of one, of much lower intensity at the centre between 0mm and 2.5mm, followed by less intense peaks either side of them. But going further out either side of 5mm can be seen much more intense peaks, the likes of which are of higher intensity than any of the inner ones put together. This appears to show an oscillatory motion of intensity with screen position, with the nadir being at the centre of the screen. It seems to suggest that the graph has been inverted, and the highest intensity of light can be found not at the point directly abreast the aperture but furthest away.

An improvement would be to zero the intensity so that the intensity at z = 2mm was of a unit intensity one, as although the graph displays the correct shape and x dimensions, intensity values of hundreds of thousands or more simply isn't ideal.

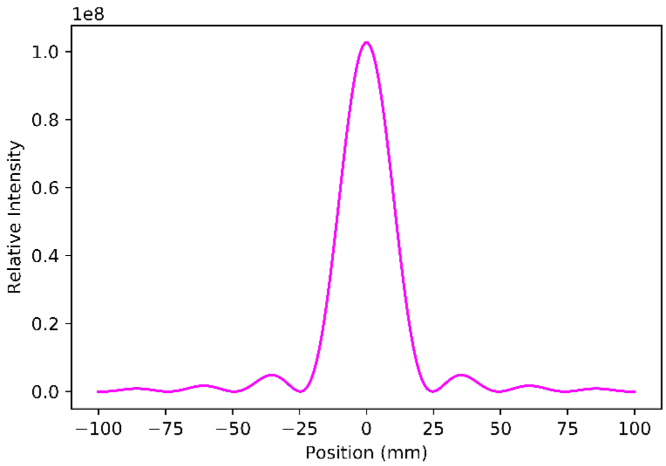

Figure 4, Graph of 0.02mm aperture width.

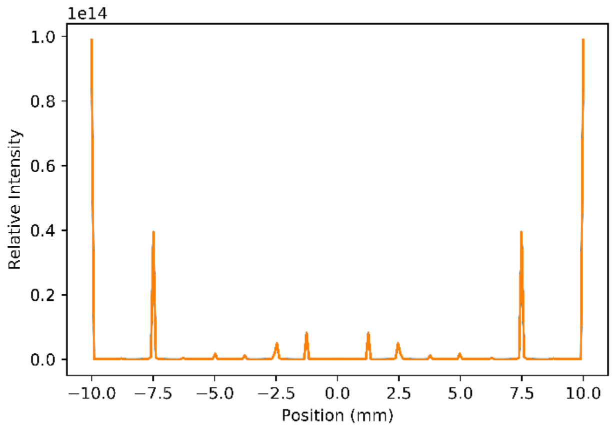

Figure 5, Graph for 2mm aperture width.

Part C

Next, the Fresnel equation was evaluated in 2D space using the 2D Fresnel equation for square apertures:

(eq.3) $$E(x,y,z) = \frac{kE_0}{2\pi z}\int_{x'_1}^{x'_2} \exp\left\{\frac{ik}{2z}(x-x')^2\right\} \mathrm{d}x' \int_{y'_1}^{y'_2} \exp\left\{\frac{ik}{2z}(y-y')^2\right\} \mathrm{d}y'$$

Because the values of y' and x' for a square are separable, calculating the double integral simply requires multiplying the previous one-dimension x' integration with a new one-dimension y' integration. This time, the integrations were performed using the SciPy integration operation, which required generating two NumPy.linspace arrays of equal length. The values of the diffraction intensity were stored in the two-dimensional array, and integrated against screen position x and y so that SciPy could compute the values. The array was then plotted as a 100x100 image, using a simple loop to multiply the integrated arrays p and q, and plot them against displacement using Matplotlib.

The image produced resembles a rounded square with smaller rectangles extending from all sides of the square, decreasing in size and luminosity. Essentially forming a cross or four headed star formation.

The effect of changing the aperture width (x'_1 and x'_2) and the screen distance z on the two-dimensional image was then explored. It was found that when the screen distance was increased, the diffraction pattern looked more and more like a homogenous dot. When the screen distance was decreased the patter appeared to generate a square, and when decreased further produced more and more square interference patterns in the shape of a cross. Increase the distance further and more and more bright primary squares joined the main square at the centre. The number of square interference patterns produced seemed to follow an inverse square law, which is what is to be expected from light waves.

Unfortunately, getting part c to work in the code proved difficult. The only image that was produced for the code, regardless of changing any values, can be seen in figure 6.

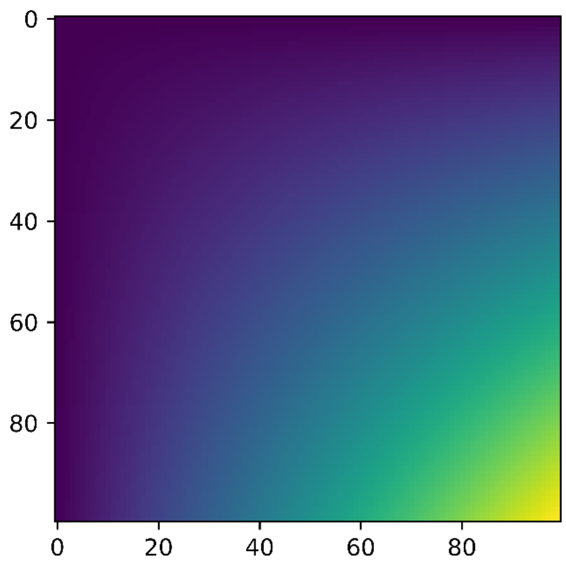

Figure 6, 2D Fresnel diffraction image.

This might suggest that the plotting function wasn't dependant on the Fresnel equation, and wasn't processing the intensity array that the diffraction had generated. An improvement would be to look over parts c and d again and ensure that they function properly, by taking in the correct x axis array and intensity arrays.

Part D

This required changing the values of x' and y' so that they were not equal, essentially creating a rectangular aperture. By changing the aspect ratio of the aperture, this produced a similar rectangular aspect ratio on the two-dimensional image, surrounded by squares instead of rectangles, extending in a cross formation around the central rectangle.

Further Improvements

The most basic improvement to all of the calculations would be simply to increase the number of intervals of integration from 100 to 500. This would take a little more time but would produce a much more accurate graph of the integration. Another method would be to increase the size of the arrays in both one dimension and two dimension to allow for a more precise graph to be plotted with higher definition.