**Fresnel Diffraction and the Simpsons rule**

**Computing exercise 2**

For exercise 2 I was tasked with creating a program that could predict the diffraction pattern created by light passing through an aperture of some size and shape. The structure of this report will be to first give some background on the problem, then the computational methods I needed to apply to solve the problem and briefly outline why they work and finally to present the results and any problems.

**Background**

There are 2 classifications for diffraction patterns, Fresnel and Fraunhofer diffraction. The underlying physics of diffraction is always the same, the intensity of light at a point some distance away from an aperture is given by the superposition of the light rays from every point inside the aperture. The differences between Fresnel and Fraunhofer diffraction patterns arises from the natural limits that can be applied when trying to model the behaviour of light moving through an aperture. The Fraunhofer model approximates the behaviour of light in the far field limit, where the size of the aperture is much smaller than the distance an observer is from the aperture. Whereas the Fresnel model approximates the behaviour of light in the near field limit. The intensity of light in the near field is highly dependent on the phase difference between light rays traveling from points along the aperture to the observer. Because of this the diffraction pattern produced in the near field vary greatly as you change the separation between an observer and the aperture. Fresnel diffraction occurs when where d is the distance between the observer and the aperture, k is the wavenumber of the light and a is the aperture length. For this exercise I was given a simplified formula for Fresnel diffraction to model the behaviour of diffraction patterns.

Equation 1 Fresnel diffraction formula

Where is the complex unit, k is the wavenumber of the light, z is the separation of the aperture and a screen which the diffraction pattern is projected on, E0 is the electric field strength at the aperture, (x,y) is the coordinate system of the screen and (x’,y’) is the coordinate system of the aperture. The x’ and y’ integrals can be separated to get two one dimensional integrals

Equation 2

Equation 3

This transform a 2D integral into two 1D integrals which are easier to solve numerically. Finally, the intensity at any point on the screen can be calculated by

Equation

Where c is the speed of light, ε0 is the permittivity of free space and E\* is the complex conjugate of E.

**Computational methods**

To evaluate the integral in equation 2 and 3 I had to create a function in python to numerically calculate the values of the 2 integrals. The function I created used the composite Simpson’s rule to calculate an approximate value for an integral. The Simpson’s rule works by approximating the integrand using a series of quadratics, each quadratic fit to three points along the integrand. By working out the area under each quadratic the value of a definite integral can be approximated. The general equation for the composite Simpson’s rule is:

Equation Composite Simpson’s rule formula

Where N is the number of iterations, b and a are the upper and lower bound of the integral respectively and f(xn) is the function evaluated at the nth position. Some things to note about the Simpson’s rule are that N must be even, all the odd n terms in the series have a coefficient of 4 and all the even n terms have a coefficient of 2 apart from the first and the last terms which have a coefficient of 1. The point xn is given by

Equation

I used the following code to program my Simpsons rule integral:

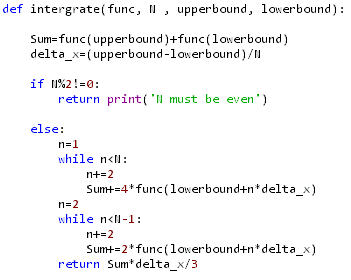
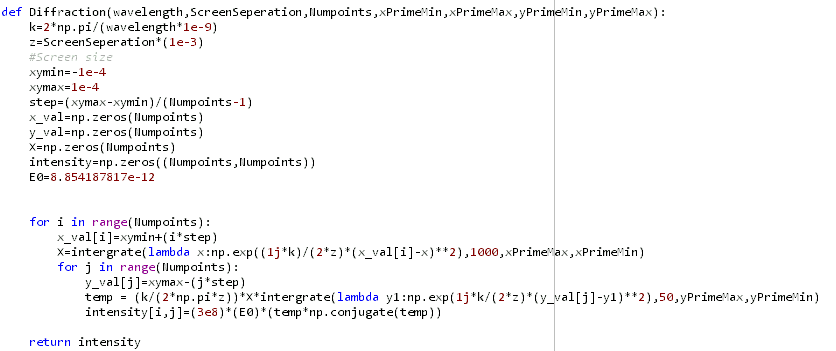


Figure Simpson’s rule code

I created to loops to evaluate the terms in the odd series and the terms in the even series for an arbitrary function. Each series term was multiplied by it respective coefficient (4 for an odd term 2 for an even term) and the terms where added to the variable sum as well as the arbitrary function evaluated at a and b. I tested my integration function by evaluate the integral of sin(x) from 0 to pi to test the accuracy of my Simpsons function. Once I had tested my Simpson I used it to evaluate equation 3, I wanted to plot a graph of |X|2 against x as this would give me the diffraction pattern for single slit diffraction. Here is the code I used:



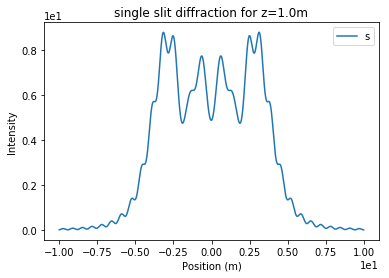
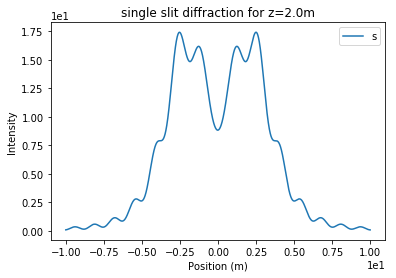
To plot |X|2 against x I had to use a for loop where each iteration the loop would calculate a value for |X|2 (using the integrate function) and x and then store the values as new elements in the lists X and x\_domain. I then used the matplotlib.pyplot module to plot the X list against the x\_domain list. Next, I set out to evaluate equation 2 for a square aperture, by evaluating equation 2 and using equation 4 I produced the 2D diffraction pattern for light moving through a square aperture. I defined the following function achieve this:

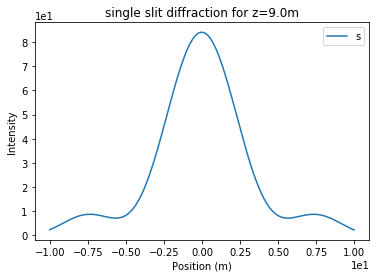
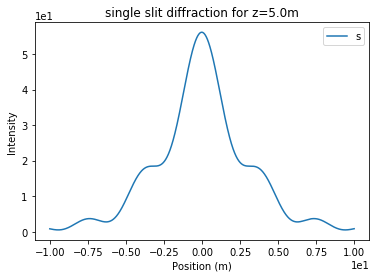
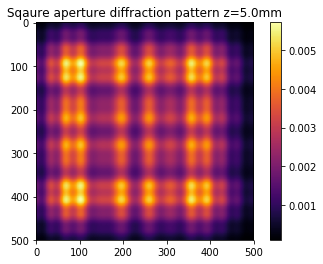


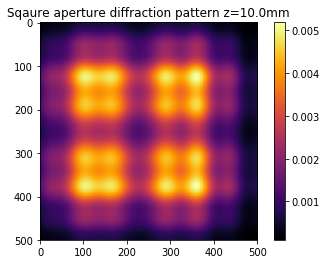
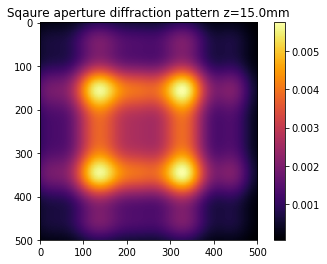
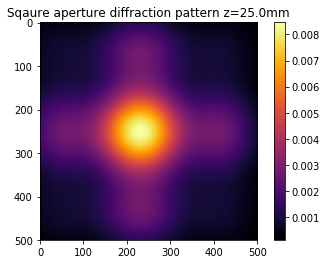
I generated an empty 2D array which represents the screen the diffraction pattern is projected on. I then created a nested loop structure to move through each element in my 2d array. The outer loop would calculate an x value for each row and then calculate an X(x,y’,z) value which corresponded to the x value. The inner loop would calculate a y value for each column. The x and y values for each element was then used as well as the X(x,y’,z) value to calculate the electric field strength for each element in my 2D array. I then used equation 4 to find the intensity for each element in my 2D array and saved the value to the corresponding element. I then used the matplotlib module to plot a heat map of my 2d array.

**Results**

My integrate function evaluated the integral of sin(x) between 0 and pi to be 1.99999999996 for 1000000 iterations the actual value of sin(x) between 0 and pi is 2 exactly. For the single slit diffraction pattern my program generated the following plots:



For each of these plots k=1m-1 and a=10m. For the square aperture diffraction pattern my program generated the following images.



For each of these plots the square aperture is 0.2mm by 0.2mm across and λ=500nm.

**Discussion**