Level 5 Laboratory: Computational Physics Exercise 3

The deadline for this exercise is **Monday 22nd February 2021** at 12:30 p.m. Your report and program (*.py) files should be uploaded into Blackboard at the appropriate point in the Second Year Laboratory (Physics DLM Year 2 2020) course.

S. Hanna

Objectives of the exercise

- To experiment with Simpson's rule for numerical integration.
- To gain experience of plotting data.

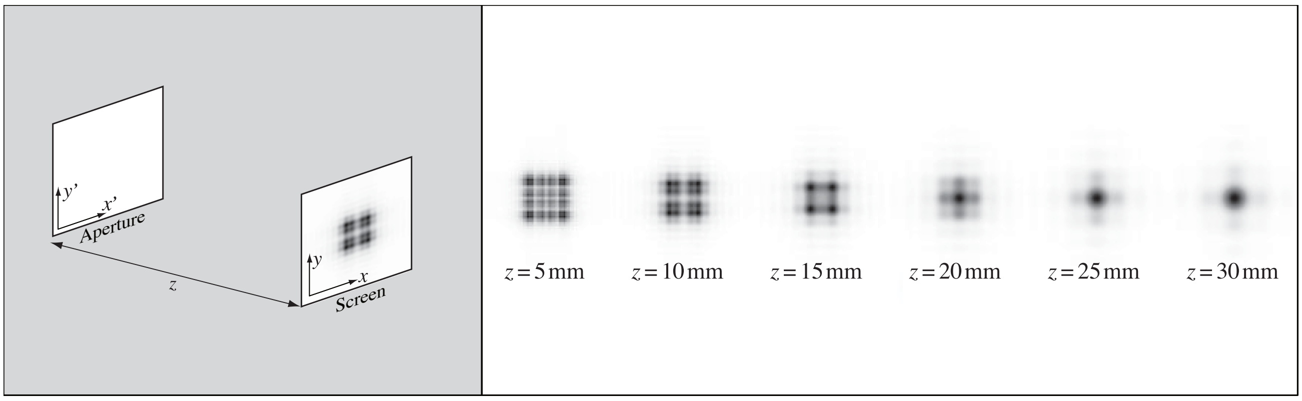
Notes:

- The exercises this term do not make use of Jupyter notebooks, although you are welcome to continue programming in a notebook if you find this easier, or if you do not have access to Python on your own machine. In general, it is expected that you will now have a copy of Anaconda Python on your personal computer, and that you will find it more convenient to develop and debug your code using Spyder. A brief run-through of how to use Spyder will be given in the week 13 live session.
- For this exercise, you will be required to submit a program **and** a short report describing your results.
- We expect that you will need help as you develop your programs. Please consult the demonstrators as frequently as you need during the drop-in sessions. They are there to help.

Problem: Fresnel diffraction from an aperture – Simpson's rule

In the computing lectures, you have been briefed about some basic types of numerical integration. Here we will apply one of them, Simpson's rule, to an interesting diffraction problem.

The general set-up for the diffraction experiment is shown below. Incident waves, travelling parallel to the z-axis, are diffracted by an aperture, and the scattering pattern observed on a screen. In the Fresnel approximation, the screen is near compared with the size of the aperture, and the diffraction pattern varies with the separation, z. Examples of this "near-field" diffraction are shown in the figure for a square aperture.



A general formula for Fresnel diffraction is given by:

$$E(x,y,z) = \frac{e^{ikz}}{i\lambda z}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} E(x',y')\exp\left\{\frac{ik}{2z}\left[(x-x')^2+(y-y')^2\right]\right\}\,\mathrm{d}x'\,\mathrm{d}y' \tag{1}$$

where the origin ($z = 0$) is taken at the aperture and $k = 2\pi/\lambda$. The integral is performed over the $x' - y'$ plane of the aperture; the electric field of the incident light is taken to be zero everywhere *except* in the aperture, where it is a constant, i.e.:

$$E(x',y') = \begin{cases} E_0 & x', y' \text{ in aperture} \\ 0 & \text{otherwise} \end{cases}$$

This is most simply achieved by choosing the limits of integration to fit around the aperture and ignoring the constant phase factor of e^{ikz}/i:

$$E(x, y, z) = \frac{kE_0}{2\pi z} \iint_{Aperture} \exp\left\{\frac{ik}{2z}\left[(x - x')^2 + (y - y')^2\right]\right\} \mathrm{d}x'\,\mathrm{d}y' \tag{2}$$

$E(x, y, z)$ is interpreted as the electric field of the diffracted light at coordinates (x, y) on a screen distance z from the aperture. The observed diffraction intensity will be proportional to $|E^2|$:

$$I(x, y, z) = \epsilon_0 c E(x, y, z) E^*(x, y, z) \tag{3}$$

where E^* is the complex conjugate of E, c is the speed of light and ϵ_0 is the permittivity of free space.

We will explore the solution to Eq. (2) in stages, first doing a 1-d integration using a self-written Simpson's rule method, and then evaluating the full 2-d integral using a built-in approach. In carrying out the following stages, you should write a single Python program, based around a similar menu to that used in previous exercises. Write a separate piece of code for each section that requires it, making sensible use of functions.

a) **1-d integration:** Write a function to perform a Simpson's rule integration of the 1-dimensional Fresnel integral:

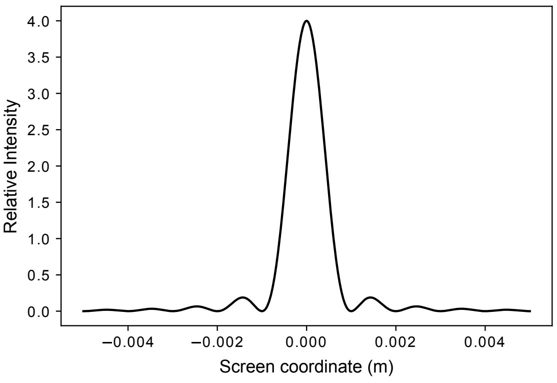
$$E(x, z) = \frac{kE_0}{2\pi z} \int_{x_1'}^{x_2'} \exp\left[\frac{ik}{2z}(x - x')^2\right] \mathrm{d}x' \tag{4}$$

The function should take the following arguments:

- the lower and upper limits of integration, x_1' and x_2'
- the wavenumber, k
- the distance to the screen, z
- the coordinate on the screen, x
- the number of intervals for the integration, N.

Remember, for N intervals there will be $N + 1$ points and, for the Simpson formula given in lectures to work properly, you must have N even.
Test your program by plotting the intensity against the screen coordinate x. To observe far-field diffraction for a single slit, as shown in the figure on the right, use the parameters listed in the table.
HINT: To generate the plot, you will need to create two arrays, one for the $|E(x)|^2$ values and the other for the x values. Plot your result using matplotlib as in the previous exercise.

Name	Variable	Value(s)
wavelength	λ	1×10^{-6} m
aperture width	$x_2' - x_1'$	2×10^{-5} m
screen distance	z	2 cm
screen coordinate	x	-5 mm to $+5$ mm
number of intervals	N	~ 100



b) When you are happy that your Simpson's rule code is working, explore the effect of changing the aperture width (x_1' and x_2') and the screen distance, z, on the appearance of the plot. Describe and discuss this in your report. Near-field effects will be observed by increasing the size of the aperture and/or reducing z. You may need to change the screen coordinate range to accommodate this. You should also examine the effect of varying N on the appearance of your plot. Generally, you will need larger N when you explore the near-field features. Can you suggest why?

c) **2-d integration:** Now you are going to evaluate the 2-d Fresnel integral given in Eq. (2). However, for square (or rectangular) apertures this can be rewritten as:

$$E(x, y, z) = \frac{kE_0}{2\pi z} \int_{x_1'}^{x_2'} \exp\left\{\frac{ik}{2z}(x - x')^2\right\} \mathrm{d}x' \int_{y_1'}^{y_2'} \exp\left\{\frac{ik}{2z}(y - y')^2\right\} \mathrm{d}y' \tag{5}$$

Since the x' and y' integrals for a square (or rectangular) aperture are separable, we can simply multiply together two 1-d integrals. i.e. for each screen coordinate (x, y) perform a 1-d Simpson's rule integration for x and another for y, and multiply the results together.

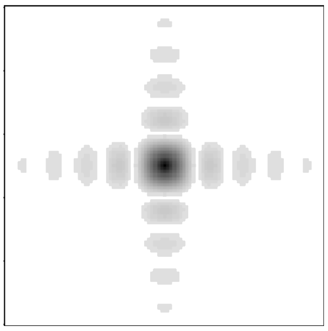
To test your approach, choose similar parameters to those in the table above, and calculate the diffraction intensity for a grid of (x, y) values on the screen. You will need to store these values in a 2-dimensional array and plot the array as an image. Here is an example of the use of a 2-d array to generate an image and, on the right, a typical far-field diffraction from a square aperture:

```
import numpy as np
import matplotlib.pyplot as plt

NumPoints = 200
delta = 4.0*np.pi / (NumPoints - 1)
intensity = np.zeros( (NumPoints,NumPoints) )

for i in range(NumPoints):
    x = i * delta
    for j in range(NumPoints):
        y = j * delta
        intensity[i,j] = np.sin(x) * np.cos(y)

plt.imshow(intensity)
plt.show()
```


Farfield diffraction from square aperture.

The 2-d diffraction calculations can be a little slow, so avoid taking too many points in the image; a 100×100 grid is reasonable. Repeat your calculation for different values of z and aperture size and shape (square or rectangular), adjusting N as needed if artefacts occur. Comment on any qualitative differences in your report.

Report

Your report should include, but not be restricted to, the following:

- A *brief* statement of the problem as you understand it;
- A *brief* description of the method used to solve the problem;
- Presentation of your results, in graphical format where applicable;
- A discussion of the results, which should include a critique of the method and ideas for improvement;
- Answers to any of the questions in the script, including appropriate discussion.

Generally, you should aim for conciseness, with most emphasis being placed on the presentation of your results and the discussion.

Submitting your work

You should submit the following to Blackboard:

- A concise report, in MS Word or pdf format;
- Your program for numerical integration using Simpson's rule;

Please note the following points:

- Please upload your "program.py" files.
- However, Turnitin won't accept a file ending ".py" so please rename your file with a ".txt" extension.
- Please also give your programs sensible distinguishing names, including your name or userid e.g. "my_userid_ex2.txt".

If you have any problems submitting your work, please contact Dr. Hanna (s.hanna@bristol.ac.uk) or ask a demonstrator.