PHY407 Lab 02

Sang Bum Yi, 1004597714
Jianbang Lin, 1004970720

The workload was distributed as followings:

- Jianbang Lin did questions 1 and 3
- Sang Bum Yi did question 2

Question 1

c)

Forward difference scheme is a method used to approximate the integral of a function, the formula is:

$$\frac{\mathrm{d}f(x_i)}{\mathrm{d}x} = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} \qquad \text{[Eq 1]}$$

In math, the error of forward difference scheme will get smaller as we move x_{i+1} to x_i, however, in Python, the error will get larger if we move x_{i+1} too close to x_i. This is because Python has rounding error, it will round off any number that is too small.

To understand the relation between step and error, the following equation is used:

$$f(x) = e^{-x^2} \qquad \text{[Eq 2]}$$

The following graph shows the error with different step widths for equation 2:

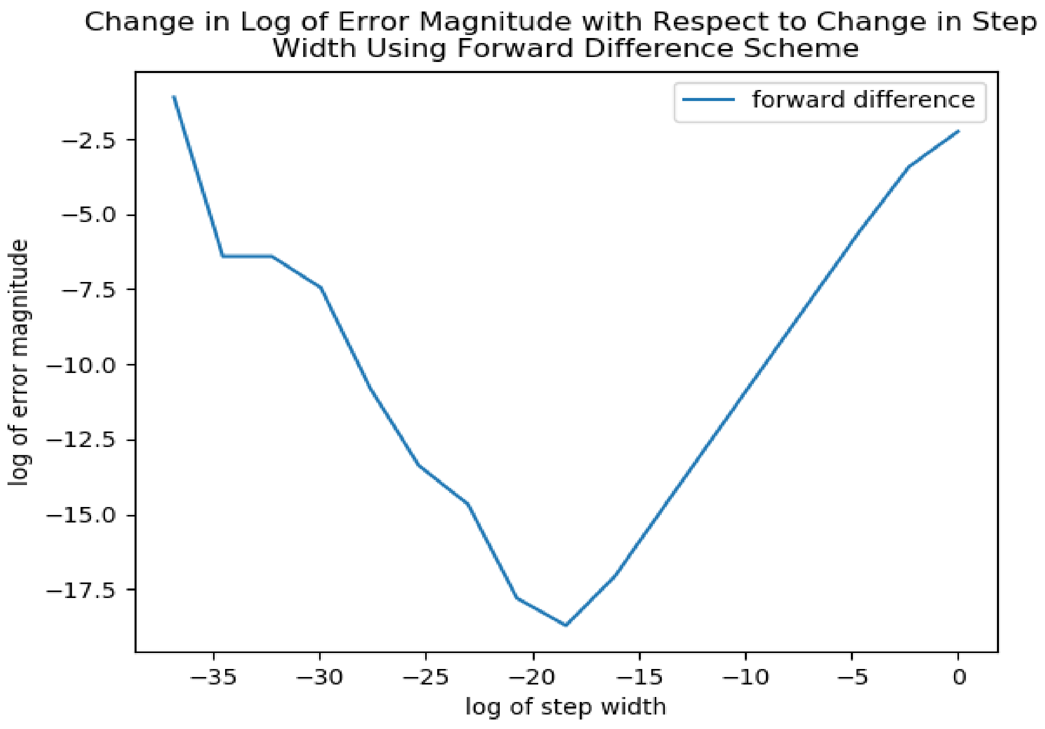


Figure 1: Change in log(error) with respect to change in log(step width) when using equation 2.

The error consists of truncation error and rounding error The error is increasing when the step is too large or too small, and it has a minimum value in between, when the step is 10^-8 wide. Truncation error will dominate when the step is large, and rounding error will dominate when the step is small. This graph shows that truncation error dominates when step is larger than 10^-8, otherwise, rounding error will dominate.

d)

Another way to approximate the integral is central difference scheme:

$$\frac{\mathrm{d}f(x_i)}{\mathrm{d}x} = \frac{f(x_{i+1}) - f(x_{i-1})}{x_{i+1} - x_{i-1}} \qquad \text{[Eq 3]}$$

The following graph compare the error made using central difference scheme and forward difference scheme:

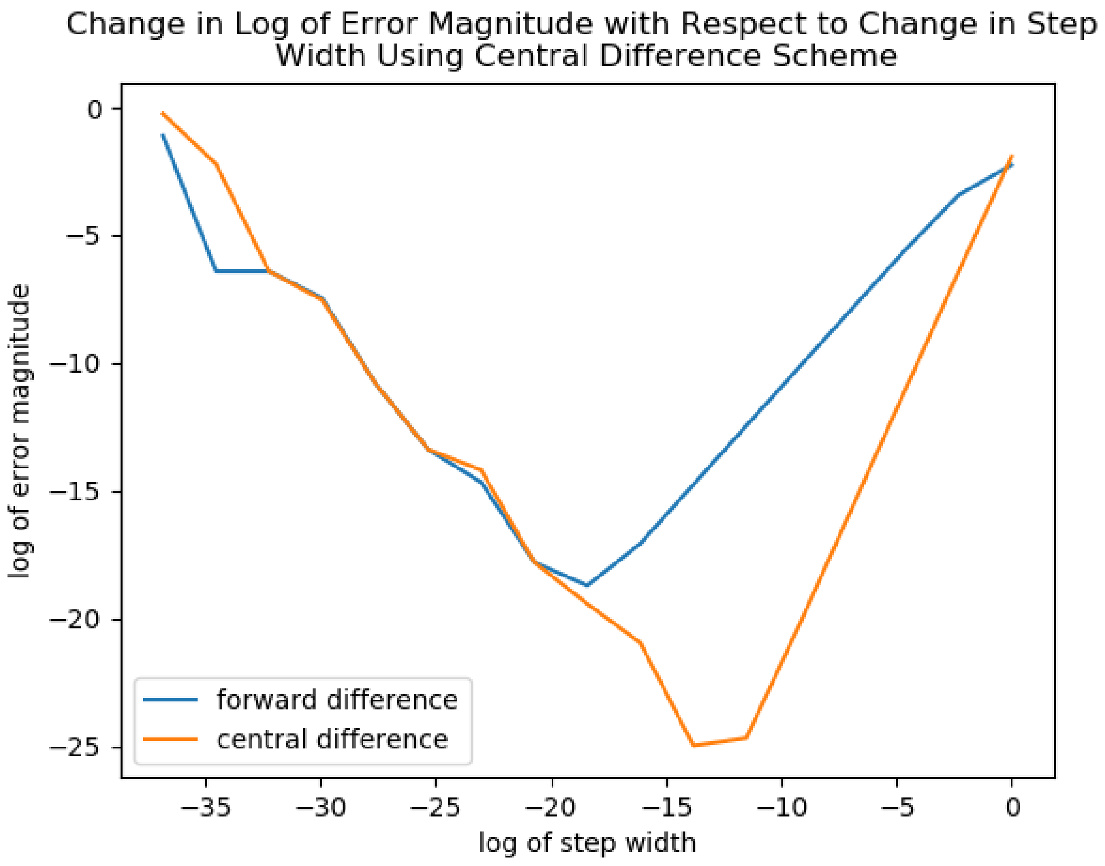


Figure 2: Comparison of log(error) made by forward difference scheme and central difference scheme.

Error for central difference has similar shape of error for forward difference scheme. However, their minimum error occurred at different step widths, the central difference scheme has minimum error when step width is 10^-6. In general, the central difference scheme produces a smaller error than the forward difference scheme, except when the step width is too small.

Question 2

a.i)

In order to evaluate an integral computationally, an approximation method is required. One of such approximation methods is called "Trapezoidal method", which sums the area of trapezoids created by connecting between x-axis and two points on a function. The trapezoids are created by slicing the interval from the starting point and the end point of the integral. It is therefore also referred to as a 'slice'. The more the trapezoids, the more accurate the approximation becomes. Given the function 'f(x)', the formula for the trapezoidal method is given as below. The detailed derivation was omitted as it can be found in the textbook.

$$I(a,b) \cong h[\tfrac{1}{2}f(a) + \tfrac{1}{2}f(b) + \textstyle\sum_{k=1}^{N-1} f(a+kh)] \qquad \text{[Eq 4]}$$

where 'a' and 'b' refer to the starting and end points of the integral respectively, 'h' refers to the width of each slice, and 'k' refers to the number of the slice.

There exists another method called "Simpson's Rule", which gives in general more precise approximation than the trapezoidal method. Instead of slicing the area under the curve into a set of trapezoids, the Simpson's rule slices the function into a set of quadratic functions and sums up the area of them. The formula for the Simpson's Rule is as follows below. Likewise, the derivation was omitted here because it can be found in the textbook.

$$I(a,b) = \tfrac{1}{3}[f(a) + f(b) + 4\sum_{\substack{k\ odd \\ 1\ldots N-1}} f(a+kh) + 2\sum_{\substack{k\ even \\ 2\ldots N-2}} f(a+kh)] \qquad \text{[Eq 5]}$$

where 'a' and 'b' refer to the starting and end points of the integral respectively, 'h' refers to the width of each slice, and 'k' refers to the number of the slice.

Using the above two methods, the Dawson function was evaluated at x = 4 and with 8 slices. Since Python provided a built-in function (scipy.special.dawsn) that calculates the exact value of the Dawson function, the results from the trapezoidal and Simpson's rules were compared to it. The formula of the Dawson function and the values obtained by each method can be found below, respectively.

$$D(x) = e^{-x^2} \int_0^x e^{t^2} dt \quad \text{[Eq 6]}$$

Method	Value (4 sig. fig)
Trapezoidal Rule	2.622 * 10^{-1}
Simpson's Rule	1.827 * 10^{-1}
Scipy.special.dawsn	1.293 * 10^{-1}

Table 1: Dawson function evaluated at x = 4 and with 8 slices, using the trapezoidal rule, simpson's rule, and scipy.special.dawsn function in Python. The printed output can be found in Appendix B.

As noted above, the Simpson's rule gave a closer approximation to the exact value given by *scipy.special.dawsn* than the trapezoidal rule did.

a.ii)

With the exact value of the Dawson function given by the built-in function in Python, the errors of the trapezoidal and Simpson's rules could be obtained. Specifically, the number of slices required to approximate the integral with an error of O(10^{-9}) was determined for each method as found in the table [number].

	Trapezoidal Rule	Simpson's Rule
Number of slices	100,000	930
Error	1.067 * 10^{-9}	1.065 * 10^{-9}
Timing	1.106 * 10^{-1}	9.947 * 10^{-4}

Table 2: Comparison of trapezoidal rule and simpson's rule for the error estimate of O(10-^{9}). The error and timing are in 4 significant figures. The printed output can be found in Appendix B.

To get an error of O(10^{-9}), 100,000 and 930 slices were needed for the trapezoidal and Simpson's rule, respectively. Therefore, the Simpson's rule can be concluded to have shown better approximation as it required significantly fewer number of slices than the trapezoidal rule did for the similar error. In addition to the number of slices, the Simpson's rule took the shorter time for computation than trapezoidal rule.

a.iii)

In part a.ii), the error of each method was obtained by subtracting the exact value of the Dawson function from the approximated value. However, this method requires to know the exact value in order to get the error. Both trapezoidal and Simpson's rules offer a way to obtain the error without knowing the exact value of the integral. Equation 7 below is the formula for error on the second estimate of trapezoidal rule.

$$\epsilon_2 = \frac{1}{3}(I_2 - I_1) \qquad \text{[Eq 7]}$$

where I_1 refers to the first approximation of an integral and I_2 refers to the second approximation that uses twice more slices than the first approximation.

On the other hand, equation 8 below gives the formula for error on the second estimate of Simpson's rule.

$$\epsilon_2 = \frac{1}{15}(I_2 - I_1) \qquad \text{[Eq 8]}$$

In the instruction, the number of slices for the first approximation(I_1) was given to be 32. Therefore, the number of slices for the second approximation(I_2) is 64. As a result, the errors on the second estimate for each method were obtained and can be found in the table 3.

ϵ_2 - Trapezoidal Rule	**-2.547 * 10^{-3}**
ϵ_2 - Simpson's Rule	-1.754 * 10^{-3}

Table 3: The error on the second estimate for trapezoidal and Simpson's rules. The number of slices for the first and second approximate was 32 and 64, respectively. The printed output can be found in Appendix B.

In accordance with the observations made previously, Simpson's rule was shown to give a better approximation as it gave a smaller error than the trapezoidal rule for the same number of slices.

b)

When the light enters a telescope, it passes through the circular aperture and gets focused in the focal plane, forming a diffraction pattern in a circular shape. The intensity of such light in the circular diffraction pattern was given in the following formula.

$$I(r) = \left(\frac{J_1(kr)}{kr}\right)^2 \qquad \text{[Eq 9]}$$

where $J_1(kr)$ is the Bessel function with m=1, 'k' refers to $2\pi/\lambda$, and 'r' refers to the distance in the focal plane from the center of the diffraction pattern.

The Bessel function, which was used in the formula for the intensity of the light above, is expressed as below:

$$J_m(x) = \frac{1}{\pi}\int_0^{\pi} \cos(m\theta - x\ sin\theta)\, d\theta \qquad \text{[Eq 10]}$$

where ‘m’ is a non-negative integer and ‘x’ ≥ 0.

Since the Bessel function includes an integral, Simpson’s rule with 1000 slices was used to evaluate the equation 10. In addition, a built-in function in Python that gives the exact value of the Bessel function, called *scipy.special.jv*, was used to compare the result. The following graphs represent the Bessel functions evaluated using the two methods, at three different ‘m’ values.

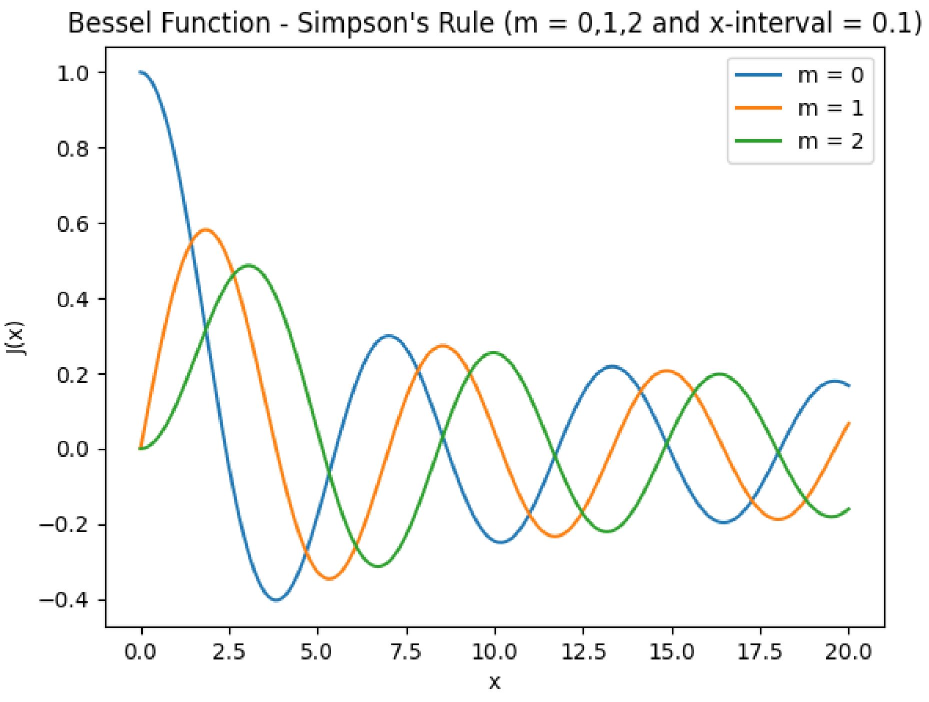


Figure 3: The Bessel functions produced with the Simpson’s rule, at three different ‘m’ values (m=0,1,2)

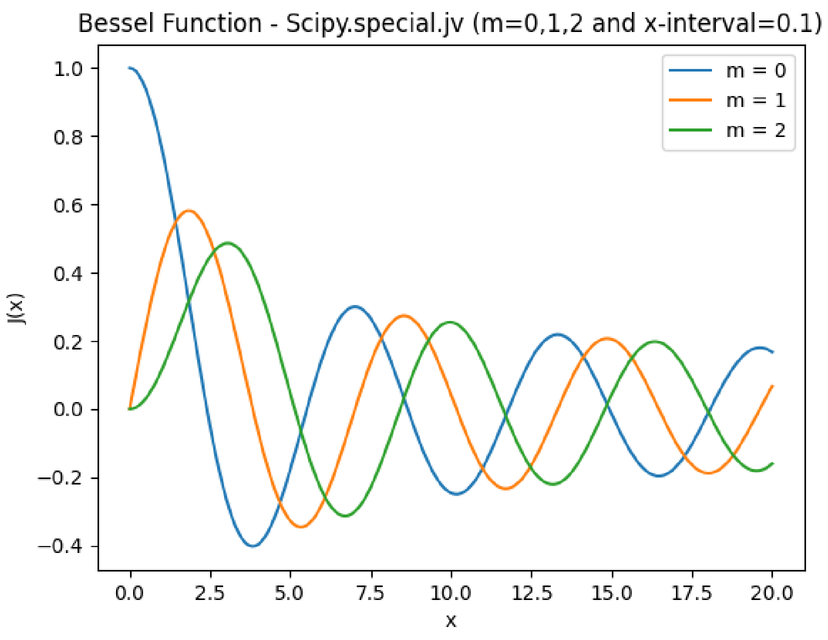


Figure 4: The Bessel functions produced with scipy.special.jv function in Python, at three different 'm' values (m=0,1,2)

Finally, a density plot of the intensity of the light was produced using equation 9. The wavelength of the light was assumed to be 500 nm. Python's built-in function *matplotlib.pyplot.imshow* was used to generate figure 5 below.

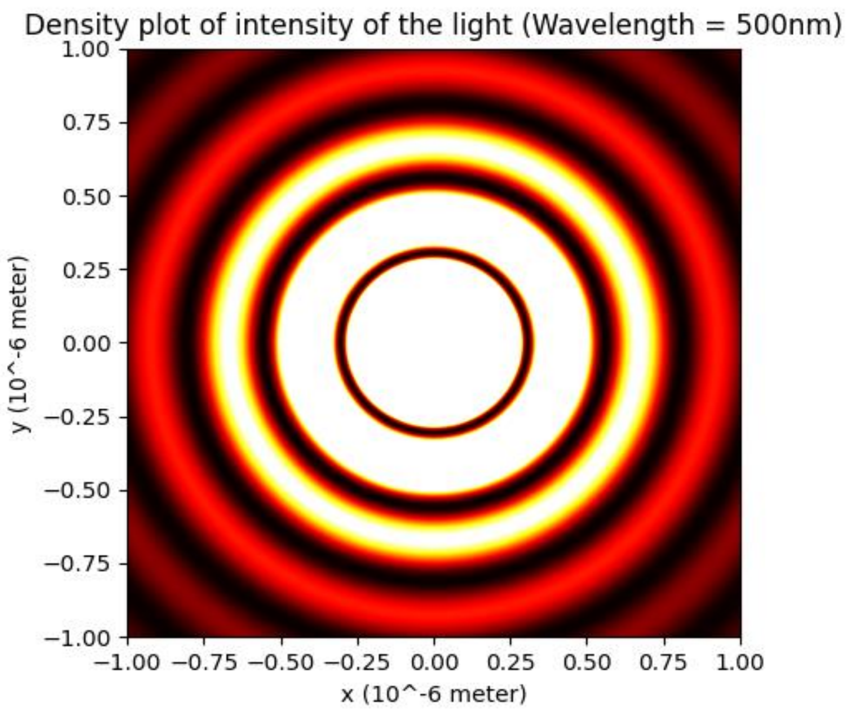


Figure 5: The density of the intensity of light was plotted in a range from r = 0m to r = 10^{-6}m. The interval of the distance from the focal center was $2*10^{-8}$m. The colormap used for the image was "hot".

In figure 5, the intensity of light was found to attenuate as it got farther away from the focal center.

Question 3

c)

The intensity Equation for diffraction grating is:

$$I(x) = \left| \int_{-\frac{\omega}{2}}^{\frac{\omega}{2}} \sqrt{q(u)}\, e^{\frac{2i\pi u}{\lambda f}} du \right|^2 \qquad \text{[Eq 11]}$$

Where x is position on the screen, $\omega$is the grating width, u is position on grating, ƛ is the wavelength, f is the focal length, and $q(u)$ is the transmission function, and it is expressed as:

$$q(u) = \sin^2(\alpha u) \qquad \text{[Eq 12]}$$

The value of **α** is dependent on the slit separation:

$$\alpha = \frac{\pi}{slit\ separation} \qquad \text{[Eq 13]}$$

Since the graph will be highly oscillatory, therefore the integral for intensity is done using the Trapezoidal rule, because the Trapezoidal rule makes better approximation than forward difference and central difference scheme, and Simpson's rule will not make an accurate approximation with highly oscillatory function. Therefore, Trapezoidal rule is used.

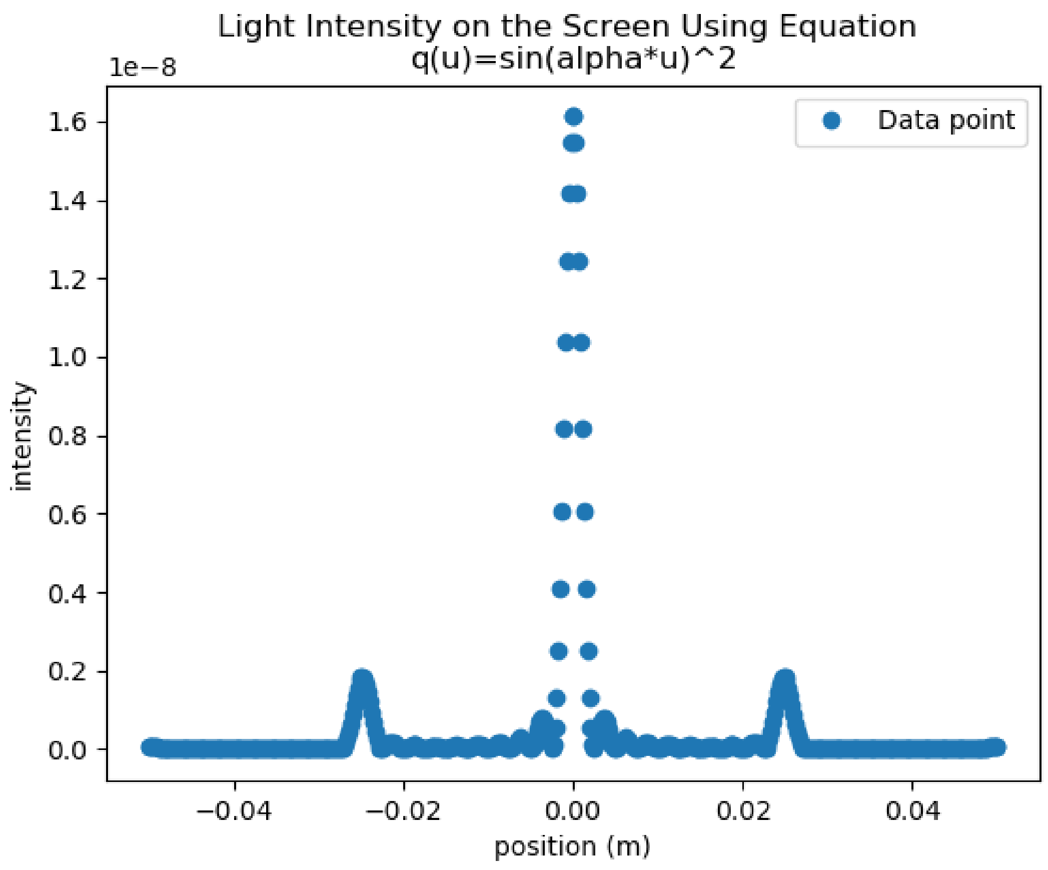


Figure 6: Light intensity vs position of diffraction grating of 10 slits and using transmission equation q(u) =sin^2(au)

The center of the screen has the highest light intensity, intensity decreases from the center, and it has another peak as shown. This graph is highly oscillatory as predicted.

d)

Using the result obtained above, a density plot can be made:

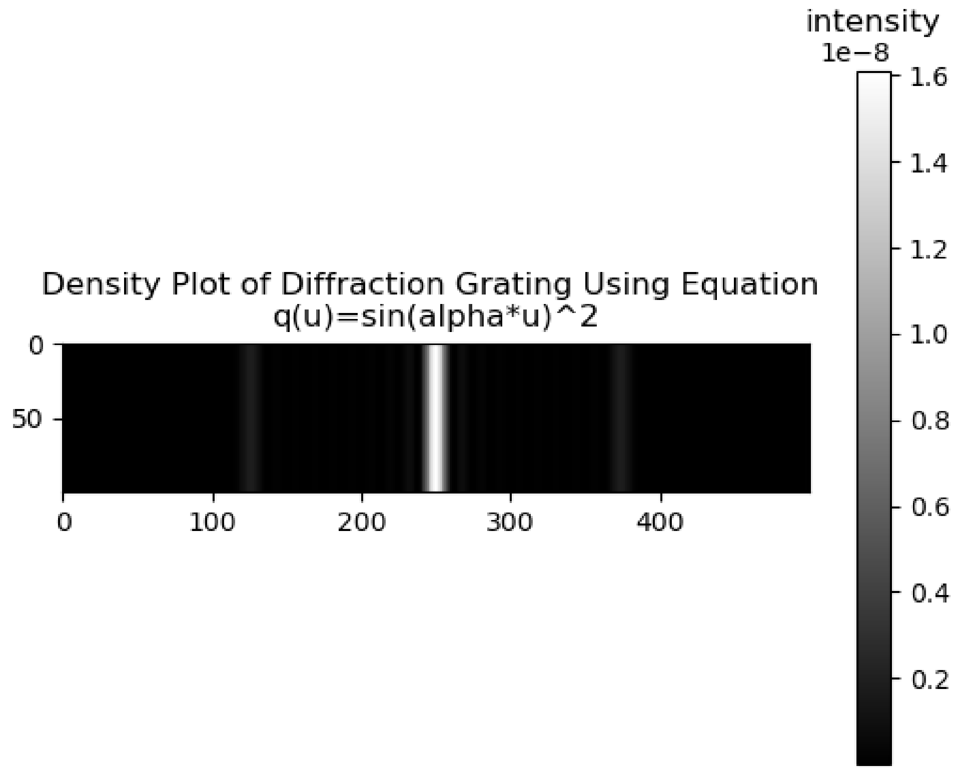


Figure 7: Density plot of figure 3

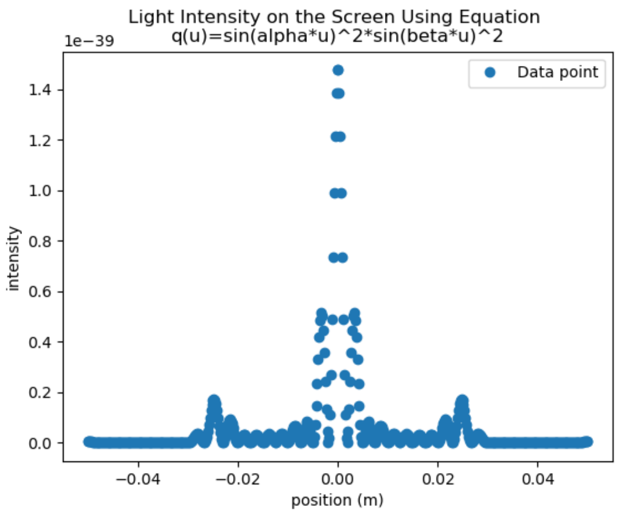
Density plot has a better visualization, it is more obvious from this plot that the light intensity is the highest at the center. And it decreases when it is away from the center, dimmer peaks are around the center and same gaps between them. It shows that the pattern is oscillatory.

e)

If the transmission function $q(u)$ is modified to the following:

$$q(u) = \sin^2(\alpha u)\sin^2(\beta u) \qquad \text{[Eq 14]}$$

The position vs intensity graph will change:



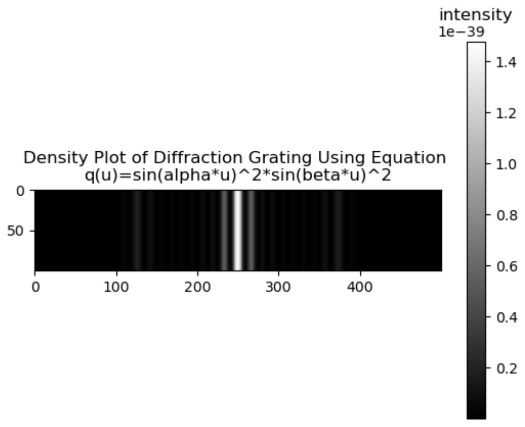


Figure 8: Light intensity vs position of diffraction grating of 10 slits and using transmission equation q(u) =sin^2(au)*sin^2(βu)

Compared with the graph with unmodified $q(u)$, the light intensity for this graph decreases less in each oscillation. The amplitude for each oscillation is greater, but they have the same period for each oscillation. The only visible difference compared with the unmodified $q(u)$graph is that the peaks around the center are brighter.

If we modify the grating, making it to have 2 slits of different size, then the light intensity graph will look like:

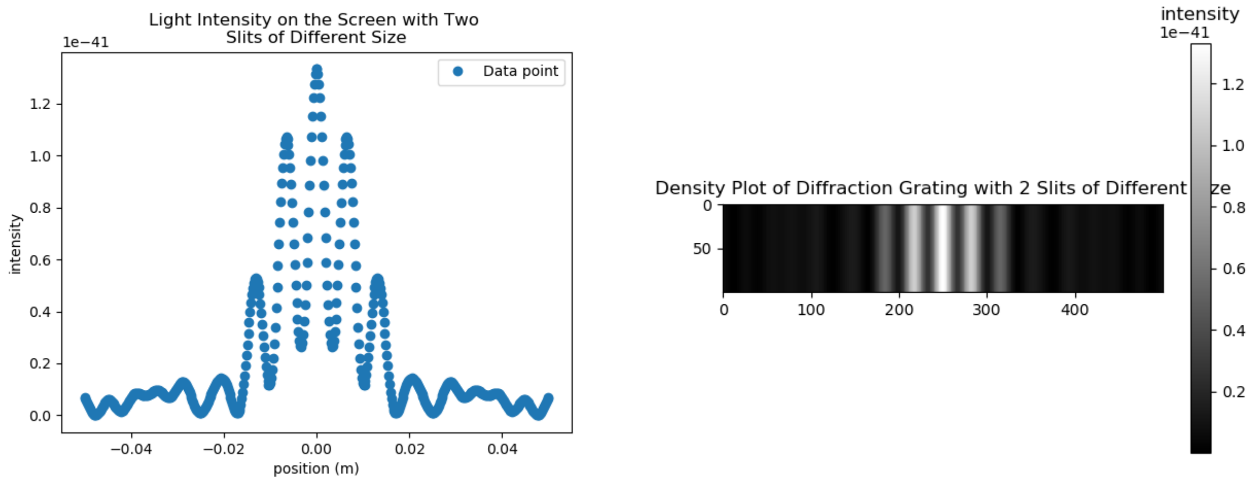


Figure 9: Light intensity vs position of diffraction grating of 2 slits and using transmission equation q(u) =$\sin^2(au)*\sin^2(\beta u)$

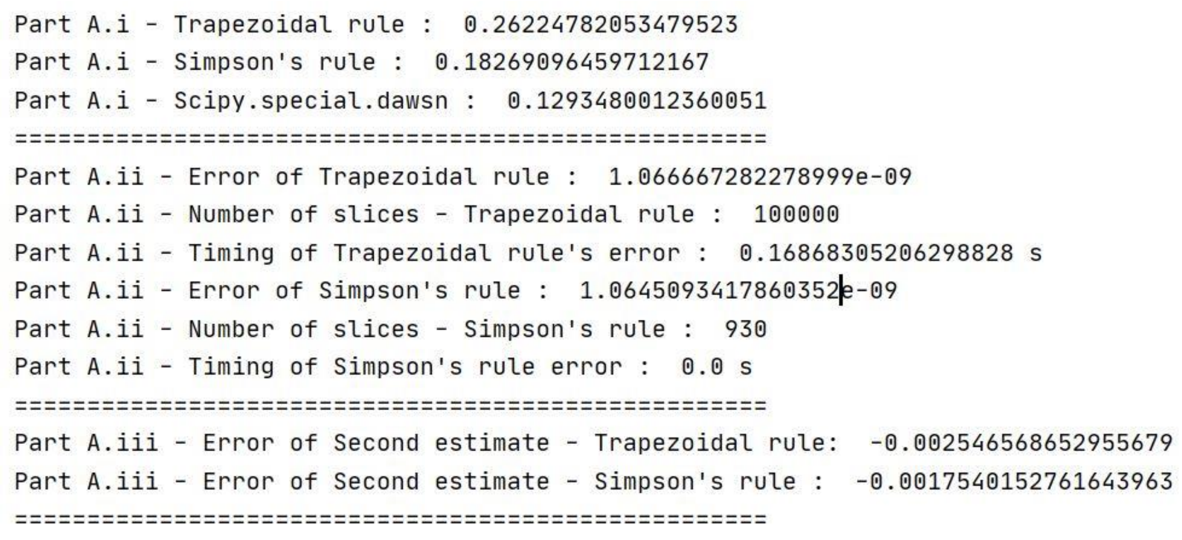
Compared with grating with 10 slits, the light intensity decreases much slower when it has only 2 slits. Also, amplitude decreases much slower. However, the light intensity at the center is weaker.

Appendix A – Step and Error for Q3.b

Step/error for the forward difference scheme (Q1.b):

step	error
1e-16	0.33142224155375166
1e-15	0.0016446658337954112
1e-14	0.0016446658337953002
1e-13	0.0005757802154550129
1e-12	2.066870314243463e-05
1e-11	1.5357573500685007e-06
1e-10	4.2553432544334413e-07
1e-09	1.8554884406718486e-08
1e-08	7.45265416046692e-09
1e-07	3.8538898849971304e-08
1e-06	3.8936937463152077e-07
9.999999999999999e-06	3.893932680965051e-06
9.999999999999999e-05	3.8933549444619686e-05
0.001	0.00038875135587198521
0.01	0.00382907405754207
0.09999999999999999	0.03244378693233374
1.0	0.10539922456186435

Appendix B – Printed Output for Question 2



```
Part A.i - Trapezoidal rule :  0.26224782053479523
Part A.i - Simpson's rule :  0.18269096459712167
Part A.i - Scipy.special.dawsn :  0.1293480012360051
====================================================
Part A.ii - Error of Trapezoidal rule :  1.066667282278999e-09
Part A.ii - Number of slices - Trapezoidal rule :  100000
Part A.ii - Timing of Trapezoidal rule's error :  0.16868305206298828 s
Part A.ii - Error of Simpson's rule :  1.0645093417860352e-09
Part A.ii - Number of slices - Simpson's rule :  930
Part A.ii - Timing of Simpson's rule error :  0.0 s
====================================================
Part A.iii - Error of Second estimate - Trapezoidal rule:  -0.002546568652955679
Part A.iii - Error of Second estimate - Simpson's rule :  -0.0017540152761643963
====================================================
```

Figure 10: Printed Output for Question 2