University College Dublin

An Coláiste Ollscoile, Baile Átha Cliath

PHYC30170 Physics Astro and Space Lab 1

Diffraction Pattern due to a Rectangular Aperture

20 September 2025

by Joana C.C. Adao (Student No. 23311051)

Contents

Abstract

The diffraction patterns formed through rectangular and circular apertures with the Huygens-Fresnel principle are considered and discussed through computational simulations of the intensity profiles and two-dimensional pattern simulations for a single slit, double slit, and circular aperture. The results obtained for the Fresnel (near-field) predictions were then compared to the Fraunhofer (far-field) approximations graphically. The computational approach reproduced the main qualitative features of diffraction that include a central maxima, destructive and constructive interference fringes, and an enclosing envelope that is wavelength-dependant. The experiment reveals that the far-field approximations by Fraunhofer do not account the Fresnel effects observed in the simulations conducted. The study goes on to show the validity of the Fraunhofer approximations in the approaching far-field regime while also verifying the framework provided by the Huygens-Fresnel principle in diffraction pattern formation.

1 Introduction

The wave nature of light allows for the diffraction or "spread out" of the electromagnetic waves when they encounter a barrier or obstacle, such as an aperture, of similar scale to the wavelength. The Fresnel (near-field) and Fraunhofer (far-field) diffraction regimes are the most common ways of describing the patterns observed. The Fraunhofer mathematical expressions for diffraction are typically simpler, thus often used to explain theory, yielding an intensity profile of the sinc^2 form for a single slit or the $\cos^2$-modulated form for the double slit. However, these approximations do not hold at finite close distances to the screen. [1–3]

In this report, the pattern and behaviour of diffraction is studied computationally with consideration for the Huygens-Fresnel principle. Simulations of the rectangular apertures for the single and double slits, as well as of a circular aperture, were performed and compared with the corresponding Fraunhofer approximations. This approach allowed fro the verification of the theoretical context and exploration of the Fresnel effects that arise with finite distances and wavelengths.

2 Theory

2.1 The Wave Nature of Light

Light is an electromagnetic wave. It is a transverse wave in such that it displaces the medium perpendicular to the direction of propagation. Although the wave moves through the medium, the medium itself is not displaced by the wave, and such the atoms themselves remain near their equilibrium positions. This is one of the significant properties that distinguishes waves from a stream of particles. [2]

2.1.1 Spherical Waves

An idealised point source of light emits radiation radially outwards and evenly in all directions, forming a spherical wavefront as shown in Figure (1). The source is then described as isotropic, producing wavefronts that expand as concentric spheres increasing in diameter with distance from the source. [2]

For a spherically symmetrical description of this wavefront the descriptive function must depend only on the radial component r such that [1, 2]:

$$\psi(\vec{\mathbf{r}}) = \psi(r, \theta, \phi) = \psi(r) \tag{1}$$

and for a harmonic spherical wave progressing radially outwards from the origin, the wave function is given by [2]:

$$\psi(r, t) = A\frac{\cos(kr - vt)}{r} \tag{2}$$

with a constant speed v and A as the source strength, converging toward the origin. At any given value for time, this function corresponds to a set of concentric spheres extending through all space. [2]

The electric field equation at position r for a spherical wave of monochromatic light originating from a point source at position r_1 is given by [4]:

$$E(r,t) = \sum_{i=1}^{N} A \frac{\cos(k|r - r_1| - \omega t)}{|r - r_1|} \tag{3}$$

where $|r - r_1|$ is the distance from the point source to the observation point, $k = \frac{2\pi}{\lambda}$ is the wave number, λ is the wavelength of light, $\omega = 2\pi f$ is the angular frequency and f is the frequency of light, with N wavelets superimposed. [4]

This function for the wave can also be described in its complex form [2]:

$$E(r,t) = \left(\frac{A}{|r - r_1|} \right) e^{i(k|r-r_1| \,\pm\, \omega t)} \tag{4}$$

The intensity observed corresponds to the time-averaged square of the electric field such that [4]:

$$I \propto |E|^2 \qquad \Longrightarrow \qquad \overline{E^2} = \frac{1}{T} \int_0^T E^2 dt \tag{5}$$

Equation (2) gives the general spherical wave solution, Equation (3) applying the same form to an electric field of light. Both show a dependency of amplitude A on $\frac{1}{r}$, a diminishing amplitude with increasing distance.

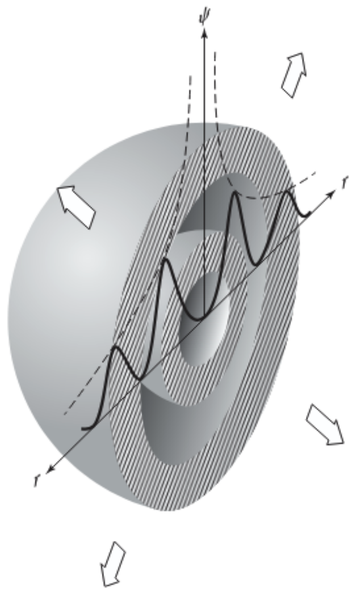


Figure 1: Visual representation of a spherical wavefront from a point source; graph illustrates the amplitude decreasing with distance as the wave propagates radially outward. [2]

2.2 Huygens-Fresnel Principle

Huygens Principle, or more commonly known as the "Huygens-Fresnel Principle", states that each unobstructed point on a wavefront can be though of as an individual source of secondary spherical wavelets, each oscillating at the same frequency as the original wave, as seen in Figure (2). [1, 2, 5, 6]

The resulting optical field at any later point is then obtained by summing all the wavelets and taking into account their relative phases and amplitudes, building upon the superposition principle idea. [1, 2]

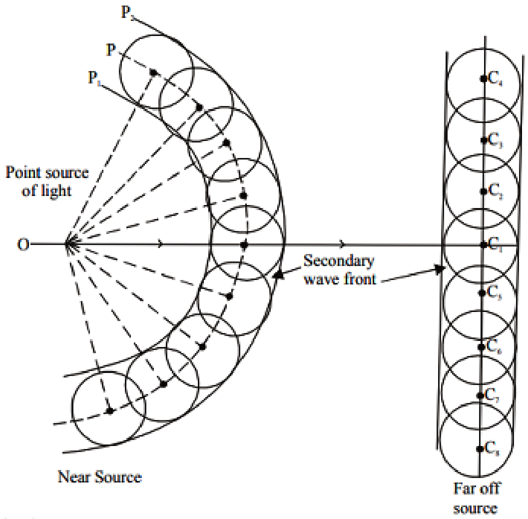


Figure 2: Diagram of Huygens' Principle for a near source (left) and far off source (right), showcasing the secondary wavelets at the secondary wavefront. [7]

Applying this idea qualitatively it can be understood that for large wavelengths relative to an aperture, the waves spread widely into the region past the obstruction through diffraction. Therefore, as the aperture size decreases, the diffracted waves take on an increasingly circular form and only then will the wavelets interfere constructively. [1, 2]

2.3 Diffraction

Diffraction is the phenomenon of a wave, such as light, sound, or matter, bends or spreads out as it encounters an obstacle. When part of a wavefront is blocked or altered in amplitude or phase, the waves that pass beyond that obstacle overlap and interfere with each other creating what is known as a **diffraction pattern**. [1, 2]

2.3.1 Fraunhofer vs. Fresnel Diffraction

Expanding on the concept of diffraction, when a light is shined through a single small aperture in a screen, the light will make an image of the hole on another screen placed closely behind the first one with very faint surrounding fringes. As the distance between the distance between both screens increases, the fringes become more visible, increasing in strength and detail, while the image of the hole is still prominent. This effect is known as **Fresnel** or **near-field** diffraction. [2, 5, 8]

If the second screen continues to be moved farther away, the fringe pattern observed will change gradually. At a very large distance, the pattern spreads out so much that it bears very little resemblance to the original aperture. This effect is known as **Fraunhofer** or **far-field** diffraction. [2, 8]

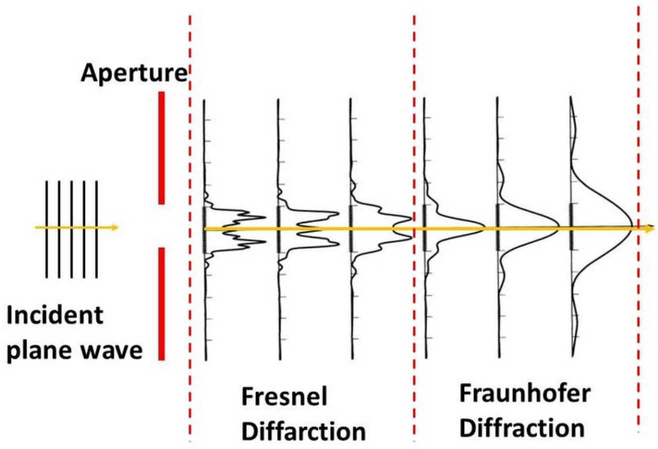


Figure 3: Intensity profiles in the near-field diffraction (middle) compared with the far-field diffraction (right) intensity profiles. [9]

Both Fresnel and Fraunhofer diffraction can be understood as direct consequences of the Huygens-Fresnel Principle (§2.2). In the near-field, the curvature of the spherical wavelets is considered, giving rise to the well-defined visible fringe patterns, while in the far-field the wavelets can be approximated to plane waves, leading to simpler and less defined diffraction patterns, see Figure (3). [2]

2.3.2 Diffraction for a Single Slit

When coherent monochromatic light passes through a narrow slit it spreads out to form fringes on a screen. The pattern consists of a bright central maximum that's significantly wider and more intense than the peripheral fringes. A series of alternating dark fringes (minima) and bright fringes (maxima) appear to either side. The peripheral maxima are only a small fraction of the intensity of the central maximum that decreases rapidly with increasing distance to the centre, see Figure (4). [1–3]

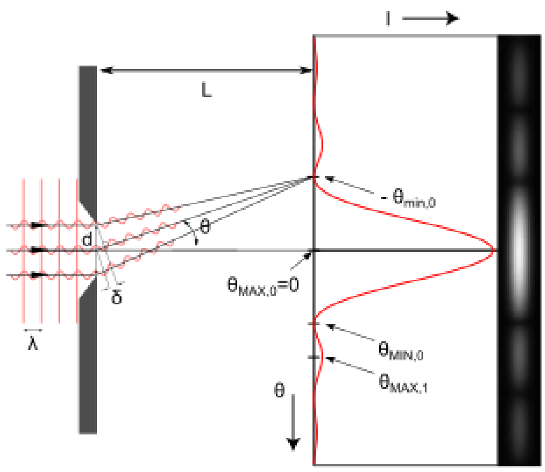


Figure 4: Single slit diffraction diagram; infinitely many points (three) shown along length d project phase contributions from the wavefront, producing a continuously varying intensity θ on the registering plate. [10]

2.3.3 Diffraction for Two Slits

When coherent monochromatic light passes through two closely spaced narrow slits the singular circular wavefront is split into two, each slit acting as a secondary source for the wavefronts that spread out and overlap. The new secondary wavefronts maintain coherency and a constant phase relationship as they are derived from the same initial wave. The superposition of these wavefronts produces a pattern of alternating dark and bright fringes when projected onto a screen. The central maximum remains the brightest and most intense, with the fringes decreasing in intensity with increasing distance from the centre (as with the single slit), see Figure (5). This diffraction pattern was first demonstrated in Thomas Young's 1801 experiment. [2, 3, 11]

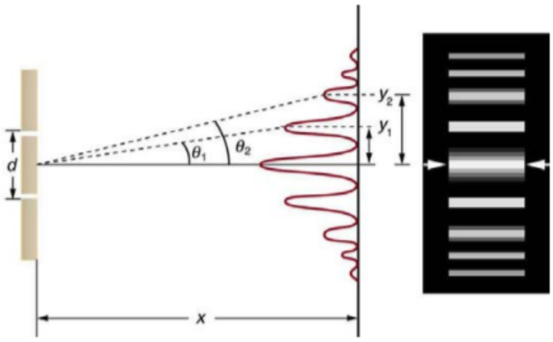


Figure 5: Double slit diffraction diagram; intensity falls off with angle to form an interference pattern consisting of a series of bright and dark fringes. [3]

2.3.4 Diffraction for a Circular Aperture

When coherent monochromatic light passes through a small circular aperture the pattern that appears is of a bright central spot maxima surrounded by fuzzy, airy concentric rings decreasing in intensity with increasing distance from the centre. The central disc is the most intense and the intensity of the surrounding rings is dependant on the size of the aperture. Rings become more regular and defined in the far-field (Fraunhofer) regime, notably more complex and less defined when in the near-field (Fresnel) regime, see Figure (6). Deviations from the usual pattern arise if coherence is not perfect, if the aperture size is comparable to the wavelength, or if the illumination is non-uniform. [3, 12–14]

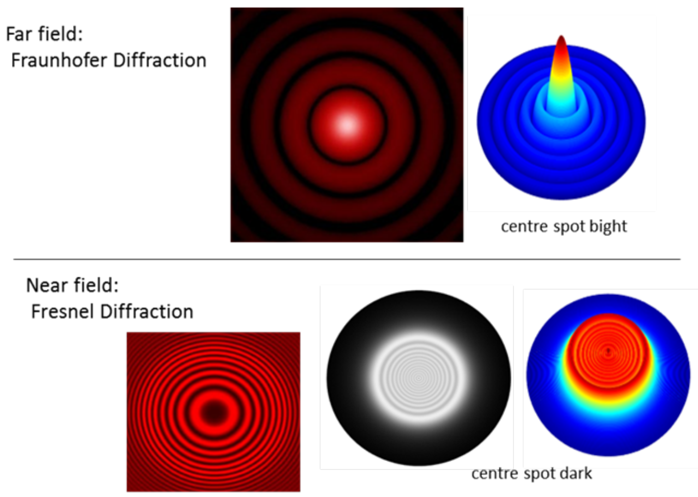


Figure 6: Circular aperture diffraction pattern; Fraunhofer (far-field) diffraction (top), Fresnel (near-field) diffraction (bottom) where a central black spot can be observed. [15]

3 Experimental Setup & Procedure

For this experiment, the standard numpy and matplotlib.pyplot packages were used, in addition to matplotlib.colors for aesthetic reasons, find_peaks imported from scipy.signal to indentify and plot peaks and troughs in resulting graphs, and the j1 function imported from scipy.special for the use of in theoretical plots for the circular aperture (see Appendix, lines 2–6).

The parameters for the experiment are set up, particularly the width (w) and height (h) of the slit, distance from the slit to the screen (L), the wavelength of light (lam) and the corresponding wave number/vector (k), calculated with $\frac{2\pi}{\text{lam}}$ (see Appendix, lines 21–25).

To define an aperture size a proper resolution (nx, ny) must be chosen, decided from the ratio between the width and height of the slit being used. For a 1mm x 3mm slit the ratio is 1:3, therefore the resolution would be nx and 3nx. The screen size is then defined spanning all width and height from $-\frac{w}{2}$ to $+\frac{w}{2}$ and $-\frac{h}{2}$ to $+\frac{h}{2}$ respectively, divided into the points defined with nx and ny. These points are overlaid and equally spaced to create an even array of data (see Appendix, lines 28–35).

3.1 Rectangular Aperture — Single Slit

For a rectangular aperture, a mask over which the code acts over is defined with the condition $|x| \leq \frac{w}{2}$ and $|y| \leq \frac{h}{2}$, and these values are flattened in the x- and y-direction and stacked to create a single symmetrical array for computing (see Appendix, lines 37–42, 218–232).

To calculate the intensity profile curve of the diffraction pattern Equation (4), omitting the time component as a monochromatic wave source does not change with time. The total electric field is summed and used to find intensity using the relationship in Equation (5), added into an empty array and normalised (see Appendix, lines 46–60, 236–250).

To plot the theoretical Fraunhofer plot lines for comparative analysis a new formula is introduced, with a as width of the slit (see Appendix, lines 62–72, 252–262) [3]:

$$I = I_0 \left(\frac{\sin\beta}{\beta} \right)^2 \qquad ; \qquad \beta = \frac{\pi a \sin\theta}{\lambda} \tag{6}$$

Additionally, the theoretical Fresnel plot lines can also be added with the following equation, but through comparative analysis it is deduced that the code matches theory (see Appendix, lines 74–91, 264–281) [16]:

$$E(x, y, z) = \frac{e^{ikz}}{i\lambda z} \iint_{-\infty}^{+\infty} E(x', y', 0) e^{\frac{ik}{2z}\left[(x-x')^2+(y-y')^2\right]} dx' dy' \tag{7}$$

To simulate the 2D diffraction pattern that would be observed on the screen the same Equation (4) and relationship Equation (5) is taken, redefining over a two-dimensional screen with n, m resolution points that are stacked to create a single two-dimensional grid of the simulated area over which the electrical field and intensity are calculated and summed over (see Appendix, lines 93–112, 283–302).

3.2 Rectangular Aperture — Double Slit

For the double slit computational experiment a similar setup to the single slit is followed, adding a separation variable (sep) and adjusting the values of the width (w), the ratio between a 0.2mm x 3mm slit with a 0.6mm separation (from left edge of left slit to right edge of right slit) is then a resolution of 1:1.9 and therefore nx and 1.9nx. The mask for the double slits is taken for each separate slit and then considered with the bitwise "OR" python component, i.e. one or both masks must agree on a position for it to be added to the array (see Appendix, lines 401–425).

The intensity profile curve and two-dimensional code setup remains the same as the single slit setup (see Appendix, lines 429–443, 477–496).

For theoretical Fraunhofer plot lines for comparative analysis a term of $\cos^2 \alpha$ is added to Equation (6) to become, with b as the slit separation (see Appendix, lines 445–456) [17]:

$$I = I_0 \left(\frac{\sin \beta}{\beta} \right)^2 \cos^2 \alpha \qquad ; \qquad \beta = \frac{\pi a \sin \theta}{\lambda} \quad , \quad \alpha = \frac{\pi b \sin \theta}{\lambda} \tag{8}$$

The same setup for the theoretical Fresnel approximation for a comparative analysis is used as for the single slit (see Appendix, lines 458–475).

3.3 Circular Aperture

For the circular aperture computational experiment a similar setup to the single and double slits is followed, adjusting the values of the width (w) and height (h) by adding the radius (R) such that $w = h = 2R$, the ratio between therefore becomes 1:1 so nx and ny are the same resolution value. The mask for the circular aperture is taken for the circle inequality ($x^2 + y^2 \leq R^2$) and proceeded with in the same fashion as the single slit (see Appendix, lines 595–617).

The intensity profile curve and two-dimensional code setup remains the same as the single slit setup (see Appendix, lines 621–635, 672–691).

To plot the theoretical Fraunhofer lines a different expression is used, using the Bessel function J_1 to calculate the airy disc pattern (see Appendix, lines 637–650) [18]:

$$I = I_0 \left(\frac{2J_1(kR\theta)}{kR\theta} \right)^2 \tag{9}$$

The same setup for the theoretical Fresnel approximation for a comparative analysis is used as for the single slit (see Appendix, lines 652–670).

3.4 Plots and Graphs

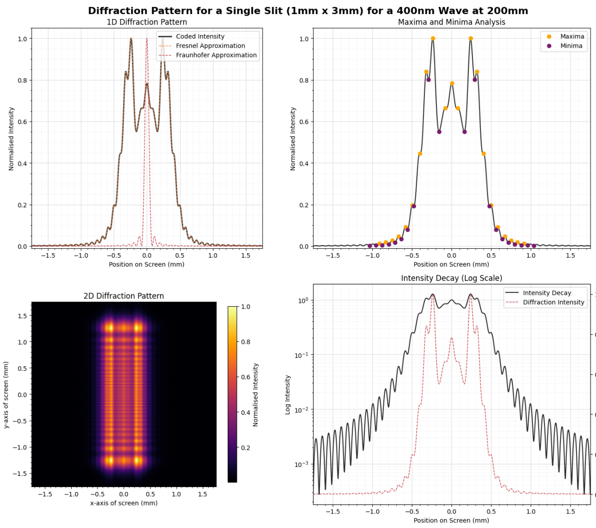
All graphs were plotted in a similar way: an intensity profile graph with theoretical Fraunhofer and Fresnel lines (top left), a two-dimensional simulation of the pattern produced (bottom left), marked and labelled peaks and throughs on the original intensity profile plot (top right), and a semi-log plot of the intensity decay of the diffraction patterns with the original intensity profile for comparison (bottom right).

4 Results

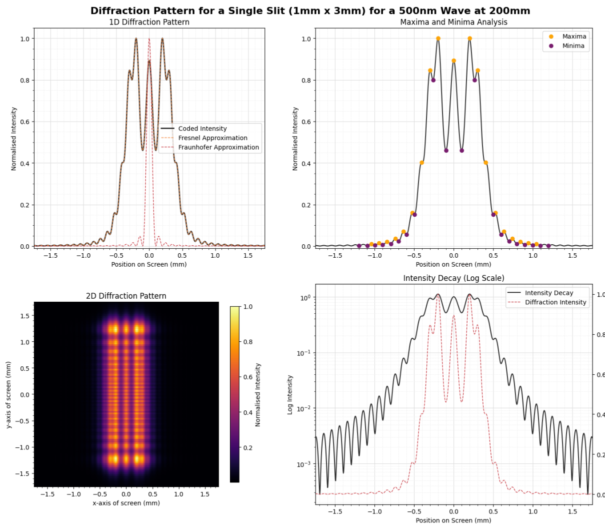
4.1 Rectangular Aperture — Single Slit

4.1.1 Vertical Aperture

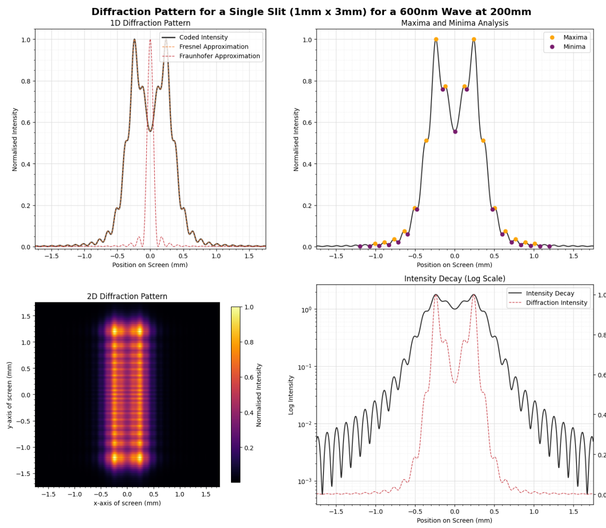
With a set screen distance of 200mm from a rectangular aperture 1mm by 3mm, each graph and two-dimensional simulation show a prominent bright maxima to either side of a dimmer central maxima that is completely absent with a wave of 600nm (Figure 7c). All three intensity decay plots show clearly dimmer fringes at the very edge that are otherwise hard to see in the intensity profile graphs and two-dimensional diffraction simulation.



(a) 400nm Wave.



(b) 500nm Wave.



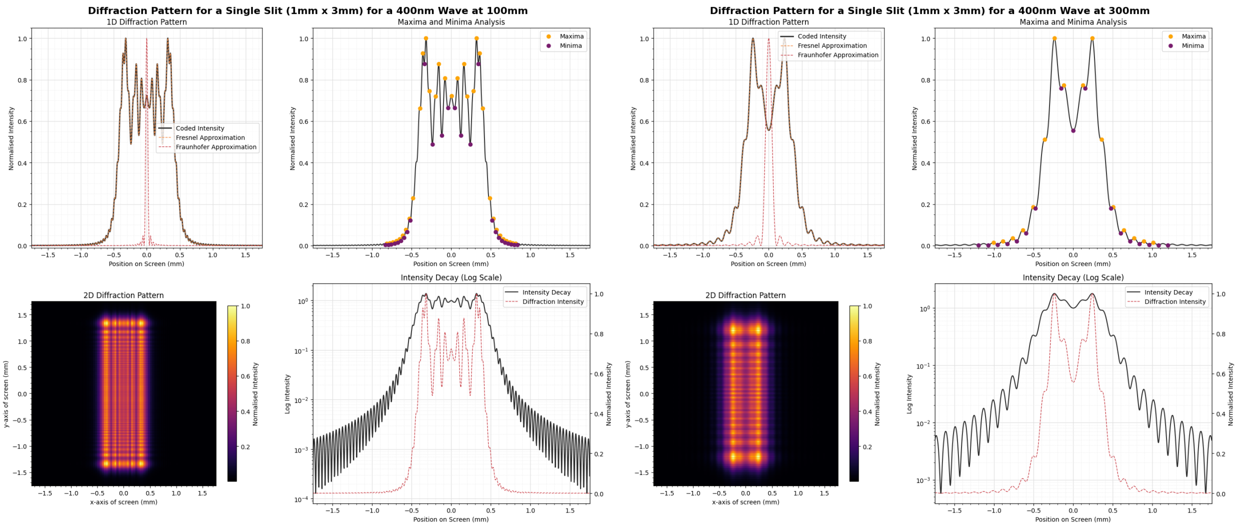
(c) 600nm Wave.

Figure 7: Diffraction Pattern for a Single Slit (1mm x 3mm) for a (a) 400nm; (b) 500nm; (c) 600nm Wave at 200mm.

The most visible and intense central maximum is seen with the 500nm wave (Figure 7b), present also with the 400nm wave but less intensely (Figure 7a). The two-dimensional renders of the diffraction patterns appear to have an airy look to them due to the fact that the fringes are closely spaced together.

4.1.2 Vertical Aperture — Varying Distance to Screen

Keeping the aperture dimensions as 1mm by 3mm and considering the same wavelengths as before, varying the distance +100mm and -100mm from the previous 200mm produces new diffraction patterns. The closer distance of 100mm generally created more fringes whilst the farther distance of 300mm divided the pattern into two more defined fringes as before with the 200mm.

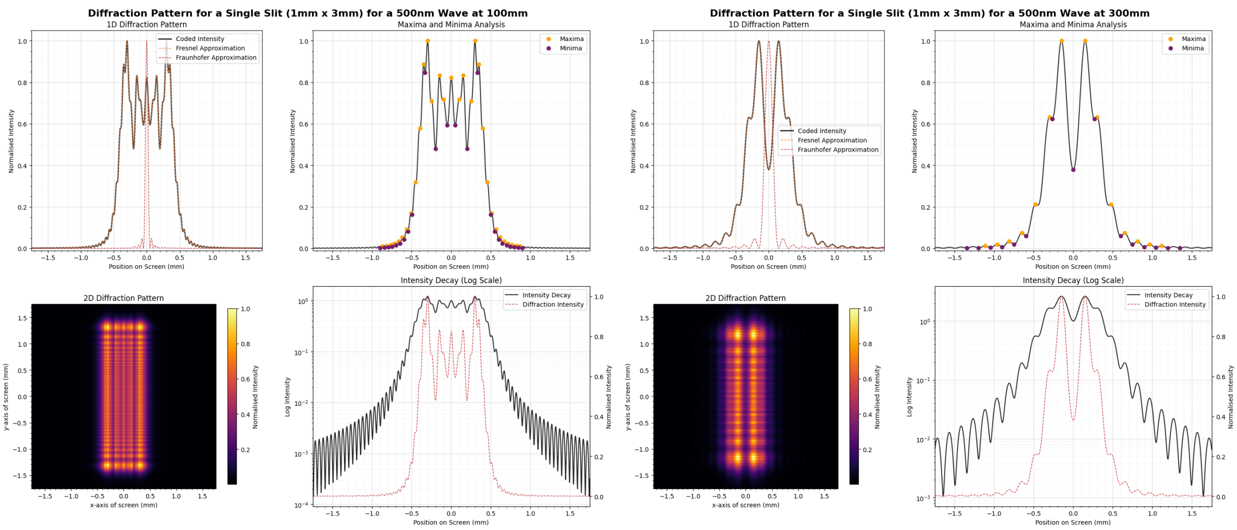


(a) Screen at 100mm.

(b) Screen at 300mm.

Figure 8: Diffraction Pattern for a Single Slit (1mm x 3mm) for a 400nm Wave at (a) 100mm and (b) 300mm.

At 100mm for a 400nm wave (Figure 8a) there appears to be more poorly-defined fringes closely packed together with the brightest fringes remaining in the outmost edges. In contrast, at 300mm (Figure 8b) the fringes appear as only two but much better and clearly defined despite the 'airy' surrounding fringes.

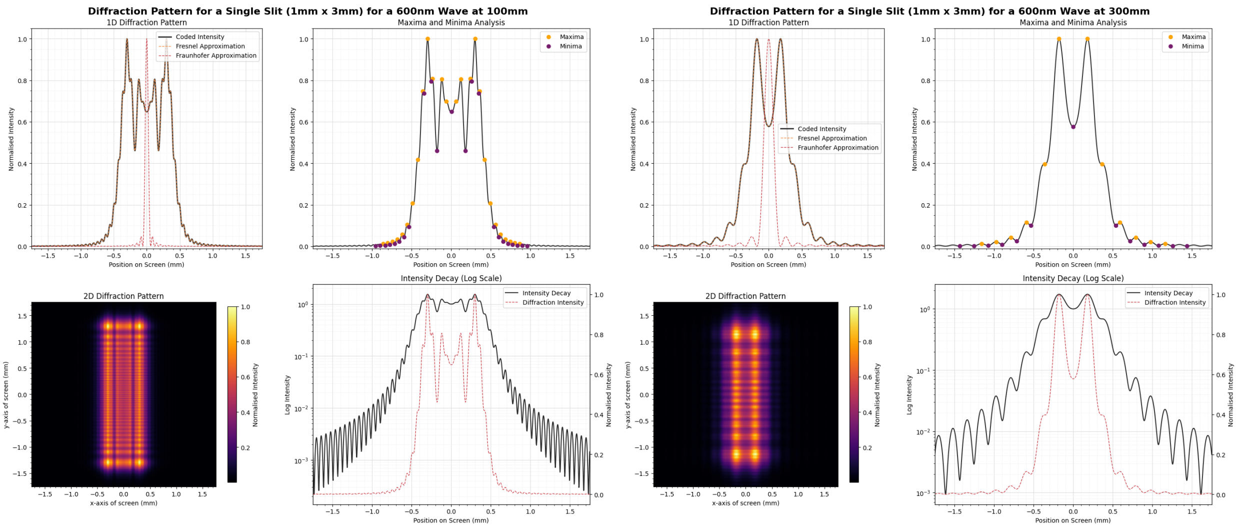


(a) Screen at 100mm.

(b) Screen at 300mm.

Figure 9: Diffraction Pattern for a Single Slit (1mm x 3mm) for a 500nm Wave at (a) 100mm and (b) 300mm.

The fringes at 100mm for a 500nm wave (Figure 9a) appear better than defined than those seen with the 400nm wave as less fringes appear, reducing the 'airy' look, the outermost fringes remaining the brightest. At 300mm (Figure 9b) the two well-defined fringes observed for the 400nm wave appear closer together, remaining just as bright and clear with the same 'airy' look to the sides.



(a) Screen at 100mm.

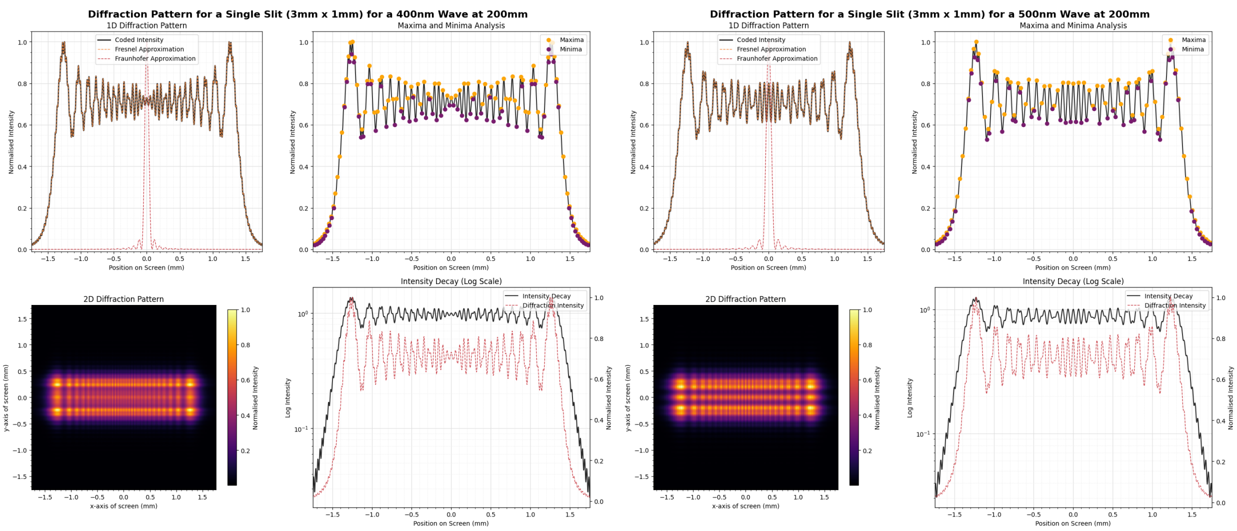
(b) Screen at 300mm.

Figure 10: Diffraction Pattern for a Single Slit (1mm x 3mm) for a 600nm Wave at (a) 100mm and (b) 300mm.

For a 600nm wave projected onto a screen 100mm away (Figure 10a) the fringes are much less defined than previously, graphically showing there are four bright distinct fringes but visually appearing more like three due to the poorly-defined intensity difference between them, the two brightest fringes remaining as the outermost ones. At 300mm (Figure 10b) two distinct bright fringes remain the most well-defined with a less-defined diving minima between them, intensifying the 'airy' look.

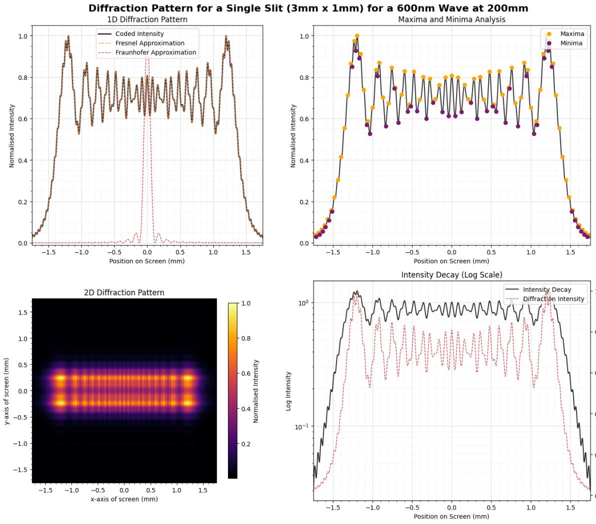
4.1.3 Horizontal Aperture

Considering the two-dimensional simulated render of the diffraction pattern, what is observed is incredibly similar to the pattern visualised when the slit was vertical. Graphically, it is where most changes occur.



(a) 400nm Wave.

(b) 500nm Wave.



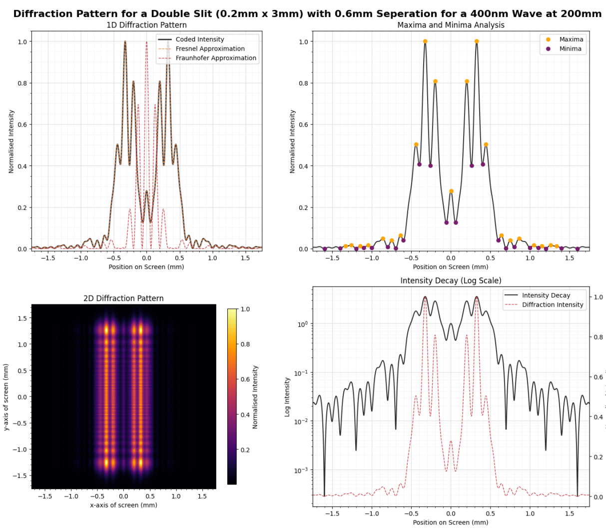
(c) 600nm Wave.

Figure 11: Diffraction Pattern for a (Horizontal) Single Slit (1mm x 3mm) for a (a) 400nm; (b) 500nm; (c) 600nm Wave at 200mm.

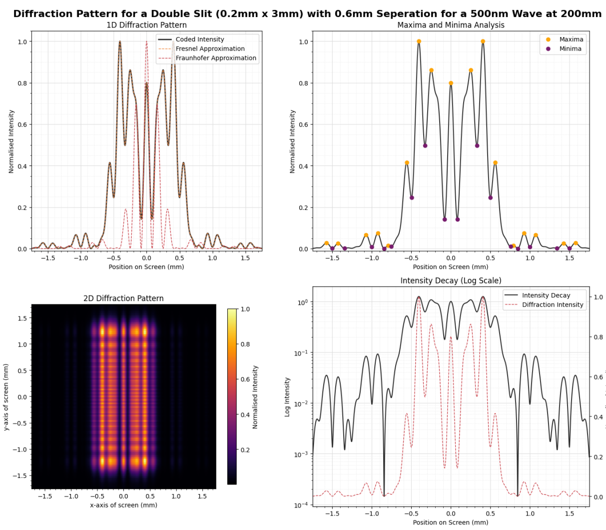
The graphs represented in Figure (11) above are all much wider than what was seen when the slit was vertical, the intensity profile representing the smaller fringes that make up the bigger, better-defined fringes that were observed in Figure (7). In all graphs the centre of the pattern is seen to be the most tightly-packed, the best-defined fringes at both ends of the pattern. The patterns continue to have the same 'airy' blurred look to them.

4.2 Rectangular Aperture — Double Slit

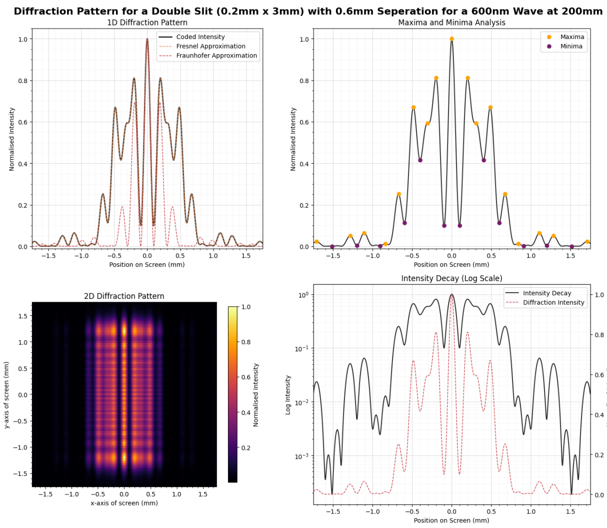
For the double slit diffraction pattern simulation renders two distinct bright fringes are always present, with a central fringe that is dimmest at 400nm (Figure 12a) and brightest at 600nm (Figure 12c). The two fringes to either side of the central fringe get broader and less-defined with increasing wavelength. Distinctly, two separate fringe groups (excluding central) can be observed for all three wavelengths.



(a) 400nm Wave.



(b) 500nm Wave.



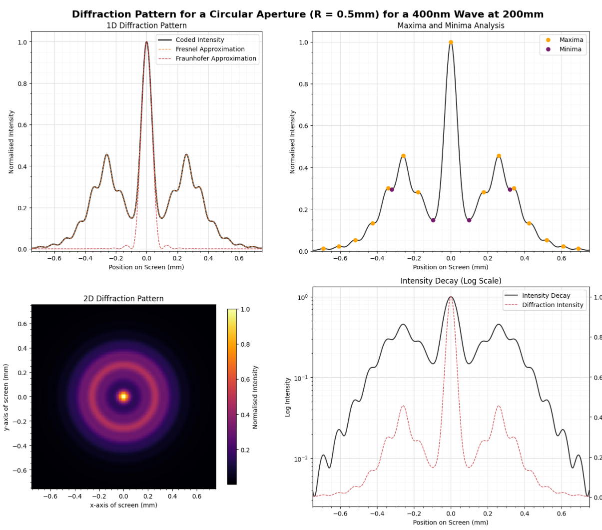
(c) 600nm Wave.

Figure 12: Diffraction Pattern for a Double Slit (0.2mm x 3mm) with 0.6mm Separation for a (a) 400nm; (b) 500nm; (c) 600nm Wave at 200mm.

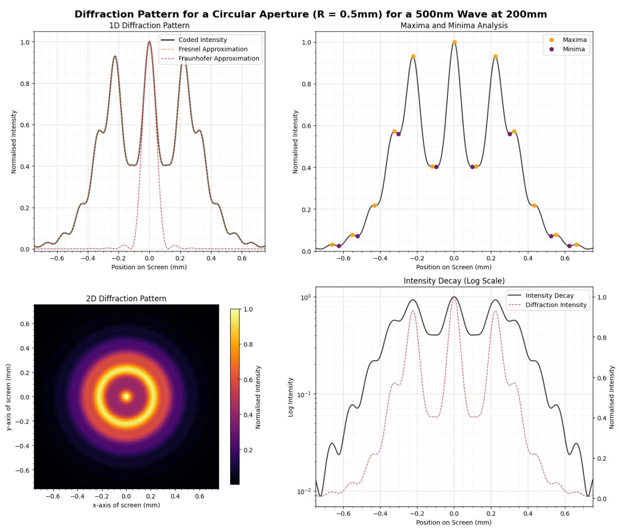
Graphically a trend can be seen in the increasing intensity of the central fringe with increasing wavelength, surrounding fringes gaining a more 'airy' look to them the brighter the central fringe is.

4.3 Circular Aperture

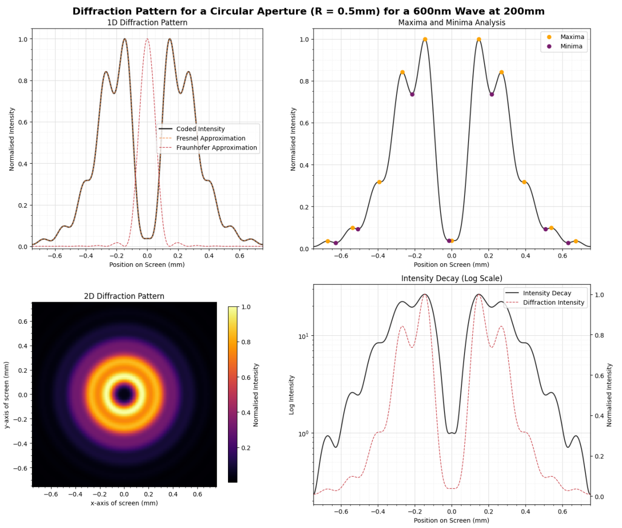
For a circular aperture the two-dimensional diffraction pattern render is very different for each, appearing to move outwards with increasing wavelengths. Wavelengths 400nm (Figure 13a) and 500nm (Figure 13b) have bright central maxima whilst for 600nm (Figure 13c) there is a central minimum instead. The 500nm and 600nm both are observed to have brighter surrounding ring fringes, the 400nm wave having quite weak fringes. All concentric fringes appear very well-defined with as 'airy' appearance for the outermost darkest ring.



(a) 400nm Wave.



(b) 500nm Wave.



(c) 600nm Wave.

Figure 13: Diffraction Pattern for a Circular Aperture (R = 0.5mm) for a (a) 400nm; (b) 500nm; (c) 600nm Wave at 200mm.

The graph for the pattern formed by the 400nm wave (Figure 13a) has a large and prominent central peak with a bright airy ring around it. The graph for the 500nm wave (Figure 13b) has a similar bright central maximum and the surrounding ring is of the same intensity as the central peak with 'airy' rings on the outside. The 600nm wave (Figure 13c) has a prominent, fully dark central minimum surrounded by a bright ring.

5 Analysis & Discussion

The diffraction patterns obtained from computational simulation can be linked and interpreted through the theoretical context outlined in §2. By applying the Huygens-Fresnel Principle to each modelled apertures a collection of secondary point sources was computationally reproduced to emit spherical wavelets (§2.1–2.2). Theoretical framework focused mostly on the Fraunhofer (far-field) diffraction patterns for the different apertures (§2.3.2–2.3.4), but the electric field equation used in the coding implementation (§2.1.1, Eq. (4)) most closely relates to the Huygens-Fresnel integral (Eq. (7)) due to the fact that the parameters were set in the Fresnel (near-field) regime. As a result, the simulations most resemble the near-field behaviour which were then compared to the analytical Fraunhofer expressions (Eq. (6), (8), (9)) graphically.

5.1 Single Slit Diffraction

The results observed for the rectangular single slit (Figures 7–11) reproduce the characteristic intensity profile described in §2.3.2: a dominant central maximum with diminishing side fringes. The trend expected with the diffraction pattern presents itself in the plots with increasing wavelengths producing wider patterns consistent with the theoretical context. However, the computed profile of the intensity did not correspond with the sinc^2 form typically expected. This is due to the fact that this form is only present in the far-field regime. Instead, for all graphs, the side lobes appear broader and the minima less sharply defined, giving the appearance of a "doubled" sinc^2 form.

This discrepancy can then be said to arise due to the fact that the Fraunhofer expressions assume far-field, whereas the computational approach retained the full Fresnel description and

Huygens assumptions. At the simulated distances observed (100mm–300mm) the curvature of the spherical wavelets was still quite significant due to the proximity of the primary source to the screen, so the expected destructive interference at the first minima was incomplete and therefore not observed. In the Fraunhofer approximation the intensity at the first minima falls directly to zero, and for the computed curves the minima was found to be non-zero and smoother side fringes that were visually interpreted as an 'airy' look. This effect was more pronounced at the shorter distance (100mm) with the fringes becoming less distinct (Figures 8a–10a), while in the intensity profile graphs for the wavelengths at 300mm (Figures 8b–10b), the distinct double fringe pattern observed appears closer and more defined as it begins entering the Fraunhofer field.

5.2 Double Slit Diffraction

The results for the double slit (Figure 12) observed showcased the expected combination of diffraction and interference described in §2.3.3: a set of finer fringes contained within the broader diffraction envelope of the single slit. As with the single slit, the spacing and intensity of the fringes was dependant on the wavelength of the incident light, with the longer wavelengths producing broader envelopes and less distinct fringes. As was observed for the single slit, the patterns did not correspond to the ideal Fraunhofer form that was expected with Equation (8) as the minima were non-zero and the central peak diffraction was wider than the formula predicted.

The reasons for these discrepancies are parallel to the discrepancies found with the single slit. Due to the use of the full Huygens-Fresnel integral in the computed diffraction patterns the contributions of the Fresnel aspect of the formula over the finite screen distance meant that the destructive interference was incomplete and so the expected minima at zero that would be observed with the Fraunhofer approach were not present. In the intensity profile that was rendered the fringes appear broadened and 'blurred' though never fully cancelled. For the longest wavelength used in the computation (600nm) the intensity profile obtained appears most closely related to the Fraunhofer approximation (Figure 12c), showing a relationship between the distance to the screen and the wavelength.

5.3 Circular Aperture Diffraction

The computed circular aperture results (Figure 13) reproduced the expected airy-pattern of concentric circular fringes that was outlined in §2.3.4. A bright central maxima was observed for the pattern produced by the 400nm and 500nm wavelengths, surrounded by concentric rings of decreasing intensity consistent with the Bessel function dependance observed in Equation (9). At 600nm a central intensity minimum is observed instead surrounded by bright concentric rings formed by altering constructive and destructive interference to form a pattern.

The Fresnel features observed for the single and double slits are also present with the circular aperture, the outer rings appearing blurred with incomplete minima and reduced contrasting intensity that would be otherwise present with the Fraunhofer pattern. The semi-logarithmic plots of the intensity decay showcase the predicted rapid fall-off of the ring intensity that produces the 'airy' appearance. The results obtained show that the Fraunhofer airy disc pattern is an approximation and that, at finite distances, the Fresnel effects alter both the clarity and depth of the produced rings.

6 Conclusion

The computational study successfully showcased and modelled the diffraction patterns obtained from single and double rectangular apertures, as well as circular apertures through the use of the Huygens-Frensel Principle. The results reproduced the key features that were predicted and discussed in the theory section, including the central maxima, an encasing envelope that broadened with wavelength, and the characteristic interference fringes. The equations used to compare to the Fraunhofer approximation of the expected diffraction patterns in §3 did not align with the computed pattern as predicted, reflecting the fact that the computational renders retained the Fresnel properties of near-field diffraction. At greater distances from the screen, the intensity profiles increasingly resembled the Fraunhofer forms, confirming them as far-field approximations contained within the Huygens-Fresnel framework.

Overall, the study verified the theoretical foundations of wave optics regarding diffraction and interference patterns, while also displaying the distinction between the Fraunhofer (far-field) and Fresnel (near-field) regimes with consideration to the patterns produced. The main limitations came from the approach to modelling the aperture grid, intensity normalisation, and the finite distance to the screen. These factors collectively reduced the contrast and clarity of the diffraction patterns, blurring the resultant fringes and dark minima. For future renditions of this computational experiment, a finer aperture sampling size, modelling larger propagations distances for comparison, and exploring light that isn't coherent and monochromatic could be used.

References

[1] M. Born, E. Wolf, A. B. Bhatia, P. C. Clemmow, D. Gabor, A. R. Stokes, A. M. Taylor, P. A. Wayman, and W. L. Wilcock, *Principles of Optics: Electromagnetic Theory of Propagation, Interference and Diffraction of Light*, 7th ed. Cambridge University Press, 1999.

[2] E. Hecht, *Optics*. Pearson Education India, 2012.

[3] O. et al., *University Physics Volume 3*. OpenStax, 2016. [Online]. Available: "https://openstax.org/details/books/university-physics-volume-3"

[4] UCD, "7. Diffraction Pattern due to a Rectangular Aperture," *3rd Year Astro and Space Physics Laboratory Manual*, n.d.

[5] Wikipedia contributors, "Huygens–fresnel principle — Wikipedia, the free encyclopedia," 2025, [Accessed 27-September-2025]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Huygens%E2%80%93Fresnel_principle&oldid=1291861847

[6] K. K. Likharev, "Essential graduate physics," *EM: Classical Electrodynamics, part*, vol. 7, 2013.

[7] Amoli, "Huygens Principle for a Near Source and Far Off Source Diagram," 2019, [Accessed 27-September-2025]. [Online]. Available: "https://www.sarthaks.com/254292/verify-laws-of-reflection-or-laws-of-refraction-on-the-basis-of-huygens-wave-theory"

[8] J. M. Cowley, *Diffraction physics*. Elsevier, 1995.

[9] M. Norouzpour, "The Development of Self-interference of Split HOLZ (SIS-HOLZ) lines for Measuring z-dependent Atomic Displacement in Crystals," Ph.D. dissertation, University of Tehran, 04 2017.

[10] Wikipedia contributors, "Diffraction — Wikipedia, the free encyclopedia," 2025, [Accessed 27-September-2025]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Diffraction&oldid=1313573322

[11] J. I. Thomas, "The Classical Double Slit Interference Experiment: A New Geometrical Approach," *American Journal of Optics and Photonics*, vol. 7, no. 1, pp. 1–9, 2019. [Online]. Available: https://doi.org/10.11648/j.ajop.20190701.11

[12] C. Andrews, "Diffraction pattern of a circular aperture at short distances," *Physical Review*, vol. 71, no. 11, p. 777, 1947.

[13] D. Burch, "Fresnel diffraction by a circular aperture," *American Journal of Physics*, vol. 53, no. 3, pp. 255–260, 1985.

[14] E. Koushki and S. Alavi, "Diffraction of a partial temporal coherent beam from a single-slit and a circular aperture," *Optics Communications*, vol. 441, pp. 33–37, 2019.

[15] I. Cooper and D. Arora, "Single-slit diffraction — visual physics," n.d., [Accessed 27-September-2025]. [Online]. Available: "https://d-arora.github.io/VisualPhysics/mod31/m31_singleSlit.htm"

[16] Wikipedia contributors, "Fresnel diffraction — Wikipedia, the free encyclopedia," 2025, [Accessed 3-October-2025]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Fresnel_diffraction&oldid=1314266286

[17] S. J. Ling, “Physics Bootcamp,” 2025, [Accessed 3-October-2025]. [Online]. Available: ”https://www.physicsbootcamp.org/author-bio-TWJ.html”

[18] Anon., “13. Lecture, 14 October 1999,” 1999, [Accessed 3-October-2025]. [Online]. Available: ”https://www.pas.rochester.edu/~dmw/ast203/Lectures/Lect_13.pdf”

List of Figures

Appendix — Code

```python
# %%
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.colors
from scipy.signal import find_peaks
from scipy.special import j1

# %%
lam400 = '#8200b5' # Hex color code for 400nm wavelength (violet)
lam500 = '#00ff92'  # Hex color code for 500nm wavelength (greenish)
lam600 = '#ffbe00' # Hex color code for 600nm wavelength (yellow)

# https://academo.org/demos/wavelength-to-colour-relationship/

# %% [markdown]
# # Vertical Single Slit

# %%
# Wave Equation and Properties

w, h = 1e-3, 3e-3
L = 200e-3 # Distance from aperture to screen
lam = 500e-9 # Wavelengths of light

k = 2 * np.pi / lam  # Wave number/vector

# Defining Aperture
nx, ny = 300, 900
x_ap = np.linspace(-w/2, w/2, nx)
y_ap = np.linspace(-h/2, h/2, ny)
xx_ap, yy_ap = np.meshgrid(x_ap, y_ap)

dx = x_ap[1] - x_ap[0]
dy = y_ap[1] - y_ap[0] # Spacing between points in the aperture grid
area = dx * dy # Area of each point source in the aperture (Riemann sum element)

mask = (np.abs(xx_ap) <= w/2) & (np.abs(yy_ap) <= h/2)

xx_ap_srect = xx_ap[mask].ravel()
yy_ap_srect = yy_ap[mask].ravel() # Flattening the arrays to 1D for easier computation

aperture_points = np.stack((xx_ap_srect, yy_ap_srect), axis=-1)  # Creating a list of (x,y) points in the aperture

# Defining the Wave Equation Functions for 1D and 2D

def srect_1d(screen=(-1.75e-3, 1.75e-3), points=1000):

    screen = np.linspace(screen[0], screen[1], points)  # Screen

    I = [] # List to hold the resultant field values at each point on the screen (intensity)
    for x in screen:
        r = np.sqrt((x - aperture_points[:,0])**2 + (aperture_points[:,1])**2 + L**2)
        E = (np.exp(1j * k * r) / r) * area  # Electric field at each point on the screen due to each point in the aperture
        E_total = E.sum()  # Total electric field
        I.append(np.abs(E_total)**2)

    I = np.array(I)  # Convert list to array for easier plotting
    I /= I.max()  # Normalize intensity for plotting

    return screen, I

def fraun_srect_1d(screen=(-1.75e-3, 1.75e-3), points=1000):

    screen = np.linspace(screen[0], screen[1], points)

    theta = np.arctan(screen / L)
    beta = (np.pi * w / lam) * np.sin(theta)

    I_fraun = (np.sinc(beta/np.pi)**2)
    I_fraun /= I_fraun.max()

    return screen, I_fraun
```

```

def fresnel_srect_1d(screen=(-1.75e-3, 1.75e-3), points=1000):

    screen = np.linspace(screen[0], screen[1], points)
    factor = (np.exp(1j * k * L)) / (1j * lam * L) # Fresnel Approximation

    xp = aperture_points[:,0]
    yp = aperture_points[:,1]

    E_theory = np.zeros_like(screen, dtype=complex)
    for i, x in enumerate(screen):
        phase = np.exp((1j * k)/(2 * L) * ((x - xp)**2 + yp**2)) # Fresnel Approximation (Integral Part)
        E = phase * area
        E_theory[i] = factor * E.sum()

    I_fresnel = (np.abs(E_theory)**2)
    I_fresnel /= I_fresnel.max()

    return screen, I_fresnel

def srect_2d(screen=(-1.75e-3, 1.75e-3), screen_res=1.25e-5):

    screen_x1, screen_x2 = screen
    screen_y1, screen_y2 = screen
    n = int((screen_x2 - screen_x1) / screen_res)
    m = int((screen_y2 - screen_y1) / screen_res)

    X = np.linspace(screen_x1, screen_x2, n)
    Y = np.linspace(screen_y1, screen_y2, m)
    X, Y = np.meshgrid(X, Y)  # 2D grid of screen points

    E_total = np.zeros((m, n), dtype=complex)
    for ap in aperture_points:
        r = np.sqrt((X - ap[0])**2 + (Y - ap[1])**2 + L**2) # ap[0] and ap[1] are the x and y coordinates of the aperture point
        E_total += (np.exp(1j * k * r) / r) * area  # Sum contributions from each point source

    I = np.abs(E_total)**2
    I /= I.max()

    return X, Y, I

# Patterns
screen, I1D = srect_1d()
screen, I1DF = fresnel_srect_1d()
screen, I1DFh = fraun_srect_1d()
X, Y, I2D = srect_2d()

# Plots
fig, axes = plt.subplots(2, 2, figsize=(14, 12), gridspec_kw={'width_ratios':[1, 1.2]})

fig.suptitle('Diffraction Pattern for a Single Slit (1mm x 3mm) for a 500nm Wave at 200mm', fontsize=16, fontweight='bold')

# 1D Plot; top left
axes[0,0].plot(screen * 1e3, I1D, color='#2A2A2A', linewidth=2, label='Coded Intensity')
axes[0,0].plot(screen * 1e3, I1DF, '--', color='#f68d45', linewidth=1, label='Fresnel Approximation')
axes[0,0].plot(screen * 1e3, I1DFh, '--', color='#cb4149', linewidth=1, label='Fraunhofer Approximation')

axes[0,0].set_title('1D Diffraction Pattern')
axes[0,0].set_xlabel('Position on Screen (mm)')
axes[0,0].set_ylabel('Normalised Intensity')
axes[0,0].set_ylim([-0.01, 1.05])
axes[0,0].set_xlim([-1.75, 1.75])

axes[0,0].minorticks_on()
axes[0,0].grid(True, which='major', linewidth=0.8, color='#DDDDDD', zorder=2)
axes[0,0].grid(True, which='minor', linewidth=0.5, color='#EEEEEE', linestyle='--', zorder=1)
axes[0,0].legend()

# 2D Plotl; bottom left
graph = axes[1,0].pcolormesh(X*1e3, Y*1e3, I2D, shading='auto', cmap='inferno')
axes[1,0].set_aspect('equal')
```

```
axes[1,0].set_title('2D Diffraction Pattern')
axes[1,0].set_xlabel('x-axis of screen (mm)')
axes[1,0].set_ylabel('y-axis of screen (mm)')
fig.colorbar(graph, ax=axes[1,0], orientation='vertical', label='Normalised Intensity', shrink
    =0.8)
axes[1,0].minorticks_on()

# Peak and Valley Slices; top right
# https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html#scipy.
    signal.find_peaks
peaks, _ = find_peaks(I1D, height=0.01)
valleys, _ = find_peaks(-I1D, prominence=0.005)

axes[0,1].plot(screen * 1e3, I1D, color='#2A2A2A', linewidth=1.5)
axes[0,1].plot(screen[peaks] * 1e3, I1D[peaks], 'o', color='#fca50a', label='Maxima')
axes[0,1].plot(screen[valleys] * 1e3, I1D[valleys], 'o', color='#781c6d', label='Minima')
axes[0,1].set_title('Maxima and Minima Analysis')
axes[0,1].set_xlabel('Position on Screen (mm)')
axes[0,1].set_ylabel('Normalised Intensity')
axes[0,1].set_ylim([-0.01, 1.05])
axes[0,1].set_xlim([-1.75, 1.75])

axes[0,1].grid(True, which='major', linewidth=0.8, color='#DDDDDD', zorder=2)
axes[0,1].grid(True, which='minor', linewidth=0.5, color='#EEEEEE', linestyle='--', zorder=1)
axes[0,1].minorticks_on()
axes[0,1].legend(loc='upper right')

# Intensity Decay Log Plot; bottom right
centre = np.argmin(np.abs(screen))
centre_peak = I1D[centre]

I1D_norm = I1D / centre_peak

mask = I1D_norm > 1e-12

ax_log = axes[1,1]

ax_log.semilogy(screen * 1e3, I1D_norm[mask], color='#2A2A2A', linewidth=1.5, label='Intensity
    Decay')
ax_log.set_ylabel('Log Intensity')
ax_log.tick_params(axis='y')

ax_lin = ax_log.twinx()
ax_lin.plot(screen * 1e3, I1D, '--', color='#cb4149', linewidth=1, label='Diffraction
    Intensity')
ax_lin.set_ylabel('Normalised Intensity')
ax_lin.tick_params(axis='y')

ax_log.set_xlabel('Position on Screen (mm)')
ax_log.set_title('Intensity Decay (Log Scale)')
ax_log.set_xlim([-1.75,1.75])

ax_log.grid(True, which='major', linewidth=0.8, color='#DDDDDD', zorder=2)
ax_log.grid(True, which='minor', linewidth=0.5, color='#EEEEEE', linestyle='--', zorder=1)
ax_log.minorticks_on()

lines1, labels1 = ax_log.get_legend_handles_labels()
lines2, labels2 = ax_lin.get_legend_handles_labels()
ax_log.legend(lines1 + lines2, labels1 + labels2, loc='upper right')

plt.tight_layout()
plt.subplots_adjust(wspace=0.2, top=0.94)

plt.show()

# %% [markdown]
# # Horizontal Single Slit

# %%
# Wave Equation and Properties

w, h = 3e-3, 1e-3
L = 200e-3 # Distance from aperture to screen
lam = 500e-9 # Wavelengths of light

k = 2 * np.pi / lam  # Wave number/vector
```

```python

# Defining Aperture
nx, ny = 300, 900
x_ap = np.linspace(-w/2, w/2, nx)
y_ap = np.linspace(-h/2, h/2, ny)
xx_ap, yy_ap = np.meshgrid(x_ap, y_ap)

dx = x_ap[1] - x_ap[0]
dy = y_ap[1] - y_ap[0] # Spacing between points in the aperture grid
area = dx * dy # Area of each point source in the aperture (Riemann sum element)

mask = (np.abs(xx_ap) <= w/2) & (np.abs(yy_ap) <= h/2)

xx_ap_srect = xx_ap[mask].ravel()
yy_ap_srect = yy_ap[mask].ravel() # Flattening the arrays to 1D for easier computation

aperture_points = np.stack((xx_ap_srect, yy_ap_srect), axis=-1)  # Creating a list of (x,y) points in the aperture

# Defining the Wave Equation Functions for 1D and 2D

def srect_1d(screen=(-1.75e-3, 1.75e-3), points=1000):

    screen = np.linspace(screen[0], screen[1], points)  # Screen

    I = [] # List to hold the resultant field values at each point on the screen (intensity)
    for x in screen:
        r = np.sqrt((x - aperture_points[:,0])**2 + (aperture_points[:,1])**2 + L**2)
        E = (np.exp(1j * k * r) / r) * area  # Electric field at each point on the screen due to each point in the aperture
        E_total = E.sum()  # Total electric field
        I.append(np.abs(E_total)**2)

    I = np.array(I)  # Convert list to array for easier plotting
    I /= I.max()  # Normalize intensity for plotting

    return screen, I

def fraun_srect_1d(screen=(-1.75e-3, 1.75e-3), points=1000):

    screen = np.linspace(screen[0], screen[1], points)

    theta = np.arctan(screen / L)
    beta = (np.pi * h / lam) * np.sin(theta)

    I_fraun = (np.sinc(beta/np.pi)**2)
    I_fraun /= I_fraun.max()

    return screen, I_fraun

def fresnel_srect_1d(screen=(-1.75e-3, 1.75e-3), points=1000):

    screen = np.linspace(screen[0], screen[1], points)
    factor = (np.exp(1j * k * L)) / (1j * lam * L) # Fresnel Approximation

    xp = aperture_points[:,0]
    yp = aperture_points[:,1]

    E_theory = np.zeros_like(screen, dtype=complex)
    for i, x in enumerate(screen):
        phase = np.exp((1j * k)/(2 * L) * ((x - xp)**2 + yp**2)) # Fresnel Approximation (Integral Part)
        E = phase * area
        E_theory[i] = factor * E.sum()

    I_fresnel = (np.abs(E_theory)**2)
    I_fresnel /= I_fresnel.max()

    return screen, I_fresnel

def srect_2d(screen=(-1.75e-3, 1.75e-3), screen_res=1.25e-5):

    screen_x1, screen_x2 = screen
    screen_y1, screen_y2 = screen
    n = int((screen_x2 - screen_x1) / screen_res)
    m = int((screen_y2 - screen_y1) / screen_res)
```

```python

    X = np.linspace(screen_x1, screen_x2, n)
    Y = np.linspace(screen_y1, screen_y2, m)
    X, Y = np.meshgrid(X, Y)  # 2D grid of screen points

    E_total = np.zeros((m, n), dtype=complex)
    for ap in aperture_points:
        r = np.sqrt((X - ap[0])**2 + (Y - ap[1])**2 + L**2) # ap[0] and ap[1] are the x and y coordinates of the aperture point
        E_total += (np.exp(1j * k * r) / r) * area  # Sum contributions from each point source

    I = np.abs(E_total)**2
    I /= I.max()

    return X, Y, I

# Patterns
screen, I1D = srect_1d()
screen, I1DF = fresnel_srect_1d()
screen, I1DFh = fraun_srect_1d()
X, Y, I2D = srect_2d()

# Plots
fig, axes = plt.subplots(2, 2, figsize=(14, 12), gridspec_kw={'width_ratios':[1, 1.2]})

fig.suptitle('Diffraction Pattern for a Single Slit (3mm x 1mm) for a 500nm Wave at 200mm', fontsize=16, fontweight='bold')

# 1D Plot; top left
axes[0,0].plot(screen * 1e3, I1D, color='#2A2A2A', linewidth=2, label='Coded Intensity')
axes[0,0].plot(screen * 1e3, I1DF, '--', color='#f68d45', linewidth=1, label='Fresnel Approximation')
axes[0,0].plot(screen * 1e3, I1DFh, '--', color='#cb4149', linewidth=1, label='Fraunhofer Approximation')

axes[0,0].set_title('1D Diffraction Pattern')
axes[0,0].set_xlabel('Position on Screen (mm)')
axes[0,0].set_ylabel('Normalised Intensity')
axes[0,0].set_ylim([-0.01, 1.05])
axes[0,0].set_xlim([-1.75, 1.75])

axes[0,0].minorticks_on()
axes[0,0].grid(True, which='major', linewidth=0.8, color='#DDDDDD', zorder=2)
axes[0,0].grid(True, which='minor', linewidth=0.5, color='#EEEEEE', linestyle='--', zorder=1)
axes[0,0].legend(loc='upper center')

# 2D Plotl; bottom left
graph = axes[1,0].pcolormesh(X*1e3, Y*1e3, I2D, shading='auto', cmap='inferno')
axes[1,0].set_aspect('equal')
axes[1,0].set_title('2D Diffraction Pattern')
axes[1,0].set_xlabel('x-axis of screen (mm)')
axes[1,0].set_ylabel('y-axis of screen (mm)')
fig.colorbar(graph, ax=axes[1,0], orientation='vertical', label='Normalised Intensity', shrink=0.8)
axes[1,0].minorticks_on()

# Peak and Valley Slices; top right
# https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html#scipy.signal.find_peaks
peaks, _ = find_peaks(I1D, height=0.01)
valleys, _ = find_peaks(-I1D, prominence=0.005)

axes[0,1].plot(screen * 1e3, I1D, color='#2A2A2A', linewidth=1.5)
axes[0,1].plot(screen[peaks] * 1e3, I1D[peaks], 'o', color='#fca50a', label='Maxima')
axes[0,1].plot(screen[valleys] * 1e3, I1D[valleys], 'o', color='#781c6d', label='Minima')
axes[0,1].set_title('Maxima and Minima Analysis')
axes[0,1].set_xlabel('Position on Screen (mm)')
axes[0,1].set_ylabel('Normalised Intensity')
axes[0,1].set_ylim([-0.01, 1.05])
axes[0,1].set_xlim([-1.75, 1.75])

axes[0,1].grid(True, which='major', linewidth=0.8, color='#DDDDDD', zorder=2)
axes[0,1].grid(True, which='minor', linewidth=0.5, color='#EEEEEE', linestyle='--', zorder=1)
axes[0,1].minorticks_on()
axes[0,1].legend(loc='upper right')

```

```python
# Intensity Decay Log Plot; bottom right
centre = np.argmin(np.abs(screen))
centre_peak = I1D[centre]

I1D_norm = I1D / centre_peak

mask = I1D_norm > 1e-12

ax_log = axes[1,1]

ax_log.semilogy(screen * 1e3, I1D_norm[mask], color='#2A2A2A', linewidth=1.5, label='Intensity Decay')
ax_log.set_ylabel('Log Intensity')
ax_log.tick_params(axis='y')

ax_lin = ax_log.twinx()
ax_lin.plot(screen * 1e3, I1D, '--', color='#cb4149', linewidth=1, label='Diffraction Intensity')
ax_lin.set_ylabel('Normalised Intensity')
ax_lin.tick_params(axis='y')

ax_log.set_xlabel('Position on Screen (mm)')
ax_log.set_title('Intensity Decay (Log Scale)')
ax_log.set_xlim([-1.75,1.75])

ax_log.grid(True, which='major', linewidth=0.8, color='#DDDDDD', zorder=2)
ax_log.grid(True, which='minor', linewidth=0.5, color='#EEEEEE', linestyle='--', zorder=1)
ax_log.minorticks_on()

lines1, labels1 = ax_log.get_legend_handles_labels()
lines2, labels2 = ax_lin.get_legend_handles_labels()
ax_log.legend(lines1 + lines2, labels1 + labels2, loc='upper right')

plt.tight_layout()
plt.subplots_adjust(wspace=0.2, top=0.94)

plt.show()

# %% [markdown]
# # Double Slit

# %%
# Wave Equation and Properties

w, h = 0.2e-3, 3e-3
sep = 0.6e-3 # Distance between slits
L = 300e-3 # Distance from aperture to screen
lam = 500e-9 # Wavelengths of light

k = 2 * np.pi / lam  # Wave number/vector

# Defining Aperture
nx, ny = 550, 1045 # Ratio ny = 1.9 * nx
x_ap = np.linspace(-(sep + w), (sep + w), nx)
y_ap = np.linspace(-h/2, h/2, ny)
xx_ap, yy_ap = np.meshgrid(x_ap, y_ap)

mask_s1 = (np.abs(xx_ap + sep/2) <= w/2) & (np.abs(yy_ap) <= h/2)
mask_s2 = (np.abs(xx_ap - sep/2) <= w/2) & (np.abs(yy_ap) <= h/2)
dmask = mask_s1 | mask_s2

xx_ap_drect = xx_ap[dmask].ravel()
yy_ap_drect = yy_ap[dmask].ravel()

dx = x_ap[1] - x_ap[0]
dy = y_ap[1] - y_ap[0] # Spacing between points in the aperture grid
area = dx * dy # Area of each point source in the aperture (Riemann sum element)

aperture_points = np.stack((xx_ap_drect, yy_ap_drect), axis=-1)  # Creating a list of (x,y) points in the aperture

# Defining the Wave Equation Functions for 1D and 2D

def drect_1d(screen=(-1.75e-3, 1.75e-3), points=1500):

    screen = np.linspace(screen[0], screen[1], points)  # Screen
```

```python

    I = [] # List to hold the resultant field values at each point on the screen (intensity)
    for x in screen:
        r = np.sqrt((x - aperture_points[:,0])**2 + (aperture_points[:,1])**2 + L**2)
        E = (np.exp(1j * k * r) / r) * area  # Electric field at each point on the screen due
            to each point in the aperture
        E_total = E.sum()  # Total electric field
        I.append(np.abs(E_total)**2)

    I = np.array(I)  # Convert list to array for easier plotting
    I /= I.max()  # Normalize intensity for plotting

    return screen, I

def fraun_drect_1d(screen=(-1.75e-3, 1.75e-3), points=1500):

    screen = np.linspace(screen[0], screen[1], points)

    theta = np.arctan(screen / L)
    beta = (np.pi * w / lam) * np.sin(theta)
    alpha = (np.pi * sep / lam) * np.sin(theta)

    I_fraun = (np.sinc(beta/np.pi)**2) * (np.cos(alpha)**2)
    I_fraun /= I_fraun.max()

    return screen, I_fraun

def fresnel_drect_1d(screen=(-1.75e-3, 1.75e-3), points=1500):

    screen = np.linspace(screen[0], screen[1], points)
    factor = (np.exp(1j * k * L)) / (1j * lam * L) # Fresnel Approximation

    xp = aperture_points[:,0]
    yp = aperture_points[:,1]

    E_theory = np.zeros_like(screen, dtype=complex)
    for i, x in enumerate(screen):
        phase = np.exp((1j * k)/(2 * L) * ((x - xp)**2 + yp**2)) # Fresnel Approximation (
            Integral Part)
        E = phase * area
        E_theory[i] = factor * E.sum()

    I_fresnel = (np.abs(E_theory)**2)
    I_fresnel /= I_fresnel.max()

    return screen, I_fresnel

def drect_2d(screen=(-1.75e-3, 1.75e-3), screen_res=1.25e-5):

    screen_x1, screen_x2 = screen
    screen_y1, screen_y2 = screen
    n = int((screen_x2 - screen_x1) / screen_res)
    m = int((screen_y2 - screen_y1) / screen_res)

    X = np.linspace(screen_x1, screen_x2, n)
    Y = np.linspace(screen_y1, screen_y2, m)
    X, Y = np.meshgrid(X, Y)  # 2D grid of screen points

    E_total = np.zeros((m, n), dtype=complex)
    for ap in aperture_points:
        r = np.sqrt((X - ap[0])**2 + (Y - ap[1])**2 + L**2) # ap[0] and ap[1] are the x and y
            coordinates of the aperture point
        E_total += (np.exp(1j * k * r) / r) * area  # Sum contributions from each point source

    I = np.abs(E_total)**2
    I /= I.max()

    return X, Y, I

# Patterns
screen, I1D = drect_1d()
screen, I1DF = fresnel_drect_1d()
screen, I1DFh = fraun_drect_1d()
X, Y, I2D = drect_2d()

# Plots
```

```
fig, axes = plt.subplots(2, 2, figsize=(14, 12), gridspec_kw={'width_ratios':[1, 1.2]})

fig.suptitle('Diffraction Pattern for a Double Slit (0.2mm x 3mm) with 0.6mm Seperation for a 500nm Wave at 300mm', fontsize=16, fontweight='bold')

# 1D Plot; top left
axes[0,0].plot(screen * 1e3, I1D, color='#2A2A2A', linewidth=2, label='Coded Intensity')
axes[0,0].plot(screen * 1e3, I1DF, '--', color='#f68d45', linewidth=1, label='Fresnel Approximation')
axes[0,0].plot(screen * 1e3, I1DFh, '--', color='#cb4149', linewidth=1, label='Fraunhofer Approximation')

axes[0,0].set_title('1D Diffraction Pattern')
axes[0,0].set_xlabel('Position on Screen (mm)')
axes[0,0].set_ylabel('Normalised Intensity')
axes[0,0].set_ylim([-0.01, 1.05])
axes[0,0].set_xlim([-1.75, 1.75])

axes[0,0].minorticks_on()
axes[0,0].grid(True, which='major', linewidth=0.8, color='#DDDDDD', zorder=2)
axes[0,0].grid(True, which='minor', linewidth=0.5, color='#EEEEEE', linestyle='--', zorder=1)
axes[0,0].legend()

# 2D Plotl; bottom left
graph = axes[1,0].pcolormesh(X*1e3, Y*1e3, I2D, shading='auto', cmap='inferno')
axes[1,0].set_aspect('equal')
axes[1,0].set_title('2D Diffraction Pattern')
axes[1,0].set_xlabel('x-axis of screen (mm)')
axes[1,0].set_ylabel('y-axis of screen (mm)')
fig.colorbar(graph, ax=axes[1,0], orientation='vertical', label='Normalised Intensity', shrink=0.8)
axes[1,0].minorticks_on()

# Peak and Valley Slices; top right
# https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html#scipy.signal.find_peaks
peaks, _ = find_peaks(I1D, height=0.01)
valleys, _ = find_peaks(-I1D, prominence=0.005)

axes[0,1].plot(screen * 1e3, I1D, color='#2A2A2A', linewidth=1.5)
axes[0,1].plot(screen[peaks] * 1e3, I1D[peaks], 'o', color='#fca50a', label='Maxima')
axes[0,1].plot(screen[valleys] * 1e3, I1D[valleys], 'o', color='#781c6d', label='Minima')
axes[0,1].set_title('Maxima and Minima Analysis')
axes[0,1].set_xlabel('Position on Screen (mm)')
axes[0,1].set_ylabel('Normalised Intensity')
axes[0,1].set_ylim([-0.01, 1.05])
axes[0,1].set_xlim([-1.75, 1.75])

axes[0,1].grid(True, which='major', linewidth=0.8, color='#DDDDDD', zorder=2)
axes[0,1].grid(True, which='minor', linewidth=0.5, color='#EEEEEE', linestyle='--', zorder=1)
axes[0,1].minorticks_on()
axes[0,1].legend(loc='upper right')

# Intensity Decay Log Plot; bottom right
centre = np.argmin(np.abs(screen))
centre_peak = I1D[centre]

I1D_norm = I1D / centre_peak

mask = I1D_norm > 1e-12

ax_log = axes[1,1]

ax_log.semilogy(screen * 1e3, I1D_norm[mask], color='#2A2A2A', linewidth=1.5, label='Intensity Decay')
ax_log.set_ylabel('Log Intensity')
ax_log.tick_params(axis='y')

ax_lin = ax_log.twinx()
ax_lin.plot(screen * 1e3, I1D, '--', color='#cb4149', linewidth=1, label='Diffraction Intensity')
ax_lin.set_ylabel('Normalised Intensity')
ax_lin.tick_params(axis='y')

ax_log.set_xlabel('Position on Screen (mm)')
ax_log.set_title('Intensity Decay (Log Scale)')
```

```
ax_log.set_xlim([-1.75,1.75])

ax_log.grid(True, which='major', linewidth=0.8, color='#DDDDDD', zorder=2)
ax_log.grid(True, which='minor', linewidth=0.5, color='#EEEEEE', linestyle='--', zorder=1)
ax_log.minorticks_on()

lines1, labels1 = ax_log.get_legend_handles_labels()
lines2, labels2 = ax_lin.get_legend_handles_labels()
ax_log.legend(lines1 + lines2, labels1 + labels2, loc='upper right')

plt.tight_layout()
plt.subplots_adjust(wspace=0.2, top=0.94)

plt.show()

# %% [markdown]
# # Circular Aperture

# %%
# Wave Equation and Properties

R = 0.5e-3 # Radius of circular aperture
w = h = 2 * R
L = 200e-3 # Distance from aperture to screen
lam = 500e-9 # Wavelengths of light

k = 2 * np.pi / lam  # Wave number/vector

# Defining Aperture
nx, ny = 1000, 1000
x_ap = np.linspace(-R, R, nx)
y_ap = np.linspace(-R, R, ny)
xx_ap, yy_ap = np.meshgrid(x_ap, y_ap)

dx = x_ap[1] - x_ap[0]
dy = y_ap[1] - y_ap[0] # Spacing between points in the aperture grid
area = dx * dy # Area of each point source in the aperture (Riemann sum element)

mask = xx_ap**2 + yy_ap**2 <= R**2 # Circular aperture mask

xx_ap_circ = xx_ap[mask].ravel()
yy_ap_circ = yy_ap[mask].ravel() # Flattening the arrays to 1D for easier computation

aperture_points = np.stack((xx_ap_circ, yy_ap_circ), axis=-1)  # Creating a list of (x,y)
    points in the aperture

# Defining the Wave Equation Functions for 1D and 2D

def circ_1d(screen=(-0.75e-3, 0.75e-3), points=1000):

    screen = np.linspace(screen[0], screen[1], points)  # Screen

    I = [] # List to hold the resultant field values at each point on the screen (intensity)
    for x in screen:
        r = np.sqrt((x - aperture_points[:,0])**2 + (aperture_points[:,1])**2 + L**2)
        E = (np.exp(1j * k * r) / r) * area  # Electric field at each point on the screen due
            to each point in the aperture
        E_total = E.sum()  # Total electric field
        I.append(np.abs(E_total)**2)

    I = np.array(I)  # Convert list to array for easier plotting
    I /= I.max()  # Normalize intensity for plotting

    return screen, I

def fraun_circ_1d(screen=(-0.75e-3, 0.75e-3), points=1000):

    screen = np.linspace(screen[0], screen[1], points)

    theta = screen / L # Small angle approximation
    q = k * R * theta

    with np.errstate(divide='ignore', invalid='ignore'):
        I_fraun = ((2 * j1(q)) / q)**2

    I_fraun = np.where(np.abs(q) < 1e-12, 1, I_fraun)
```

```
    I_fraun /= I_fraun.max()

    return screen, I_fraun

def fresnel_circ_1d(screen=(-0.75e-3, 0.75e-3), points=1000):

    screen = np.linspace(screen[0], screen[1], points)

    factor = (np.exp(1j * k * L) / (1j * lam * L)) # Fresnel Approximation

    xp = aperture_points[:,0]
    yp = aperture_points[:,1]

    E_theory = np.zeros_like(screen, dtype=complex)
    for i, x in enumerate(screen):
        phase = np.exp((1j * k)/(2 * L) * ((x - xp)**2 + yp**2)) # Fresnel Approximation (
            Integral Part)
        E = phase * area
        E_theory[i] = factor * E.sum()

    I_fresnel = (np.abs(E_theory)**2)
    I_fresnel /= I_fresnel.max()

    return screen, I_fresnel

def circ_2d(screen=(-0.75e-3, 0.75e-3), screen_res=1.25e-5):

    screen_x1, screen_x2 = -0.75e-3, 0.75e-3
    screen_y1, screen_y2 = -0.75e-3, 0.75e-3
    n = int((screen_x2 - screen_x1) / screen_res)
    m = int((screen_y2 - screen_y1) / screen_res)

    X = np.linspace(screen_x1, screen_x2, n)
    Y = np.linspace(screen_y1, screen_y2, m)
    X, Y = np.meshgrid(X, Y)  # 2D grid of screen points

    E_total = np.zeros((m, n), dtype=complex)
    for ap in aperture_points:
        r = np.sqrt((X - ap[0])**2 + (Y - ap[1])**2 + L**2) # ap[0] and ap[1] are the x and y
            coordinates of the aperture point
        E_total += (np.exp(1j * k * r) / r) * area  # Sum contributions from each point source

    I = np.abs(E_total)**2
    I /= I.max()

    return X, Y, I

# Patterns
screen, I1D = circ_1d()
screen, I1DF = fresnel_circ_1d()
screen, I1DFh = fraun_circ_1d()
X, Y, I2D = circ_2d(screen_res=1.25e-5)

# Plots
fig, axes = plt.subplots(2, 2, figsize=(14, 12), gridspec_kw={'width_ratios':[1, 1.2]})

fig.suptitle('Diffraction Pattern for a Circular Aperture (R = 0.5mm) for a 500nm Wave at 200
    mm', fontsize=16, fontweight='bold')

# 1D Plot; top left
axes[0,0].plot(screen * 1e3, I1D, color='#2A2A2A', linewidth=2, label='Coded Intensity')
axes[0,0].plot(screen * 1e3, I1DF, '--', color='#f68d45', linewidth=1, label='Fresnel
    Approximation')
axes[0,0].plot(screen * 1e3, I1DFh, '--', color='#cb4149', linewidth=1, label='Fraunhofer
    Approximation')

axes[0,0].set_title('1D Diffraction Pattern')
axes[0,0].set_xlabel('Position on Screen (mm)')
axes[0,0].set_ylabel('Normalised Intensity')
axes[0,0].set_ylim([-0.01, 1.05])
axes[0,0].set_xlim([-0.75, 0.75])

axes[0,0].minorticks_on()
axes[0,0].grid(True, which='major', linewidth=0.8, color='#DDDDDD', zorder=2)
axes[0,0].grid(True, which='minor', linewidth=0.5, color='#EEEEEE', linestyle='--', zorder=1)
axes[0,0].legend()
```

```
# 2D Plotl; bottom left
graph = axes[1,0].pcolormesh(X*1e3, Y*1e3, I2D, shading='auto', cmap='inferno')

axes[1,0].set_aspect('equal')
axes[1,0].set_title('2D Diffraction Pattern')
axes[1,0].set_xlabel('x-axis of screen (mm)')
axes[1,0].set_ylabel('y-axis of screen (mm)')

fig.colorbar(graph, ax=axes[1,0], orientation='vertical', label='Normalised Intensity', shrink
    =0.8)
axes[1,0].minorticks_on()

# Peak and Valley Slices; top right
# https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html#scipy.
    signal.find_peaks
peaks, _ = find_peaks(I1D, height=0.01)
valleys, _ = find_peaks(-I1D, prominence=0.005)

axes[0,1].plot(screen * 1e3, I1D, color='#2A2A2A', linewidth=1.5)
axes[0,1].plot(screen[peaks] * 1e3, I1D[peaks], 'o', color='#fca50a', label='Maxima')
axes[0,1].plot(screen[valleys] * 1e3, I1D[valleys], 'o', color='#781c6d', label='Minima')
axes[0,1].set_title('Maxima and Minima Analysis')
axes[0,1].set_xlabel('Position on Screen (mm)')
axes[0,1].set_ylabel('Normalised Intensity')
axes[0,1].set_ylim([0, 1.05])
axes[0,1].set_xlim([-0.75, 0.75])

axes[0,1].grid(True, which='major', linewidth=0.8, color='#DDDDDD', zorder=2)
axes[0,1].grid(True, which='minor', linewidth=0.5, color='#EEEEEE', linestyle='--', zorder=1)
axes[0,1].minorticks_on()
axes[0,1].legend(loc='upper right')

# Intensity Decay Log Plot; bottom right
centre = np.argmin(np.abs(screen))
centre_peak = I1D[centre]

I1D_norm = I1D / centre_peak

mask = I1D_norm > 1e-12

ax_log = axes[1,1]

ax_log.semilogy(screen * 1e3, I1D_norm[mask], color='#2A2A2A', linewidth=1.5, label='Intensity
     Decay')
ax_log.set_ylabel('Log Intensity')
ax_log.tick_params(axis='y')

ax_lin = ax_log.twinx()
ax_lin.plot(screen * 1e3, I1D, '--', color='#cb4149', linewidth=1, label='Diffraction
    Intensity')
ax_lin.set_ylabel('Normalised Intensity')
ax_lin.tick_params(axis='y')

ax_log.set_xlabel('Position on Screen (mm)')
ax_log.set_title('Intensity Decay (Log Scale)')
ax_log.set_xlim([-0.75, 0.75])

ax_log.grid(True, which='major', linewidth=0.8, color='#DDDDDD', zorder=2)
ax_log.grid(True, which='minor', linewidth=0.5, color='#EEEEEE', linestyle='--', zorder=1)
ax_log.minorticks_on()

lines1, labels1 = ax_log.get_legend_handles_labels()
lines2, labels2 = ax_lin.get_legend_handles_labels()
ax_log.legend(lines1 + lines2, labels1 + labels2, loc='upper right')


plt.tight_layout()
plt.subplots_adjust(wspace=0.2, top=0.94)

plt.show()
```