Lecture Notes 2 Introduction to Fourier Optics

Benjamin Schreyer

University of Maryland, College Park, Maryland

May 26th 2024

1. WRITTEN HOMEWORK 1 SOLUTIONS

The homework problems prepare one to start dealing with more general formulations of diffraction besides the double slit, and using the correct approximations to make these calculations easier on paper and for a computer.

1. Fraunhofer Approximation Consider a two slit diffraction experiment as pictured in the diagram, where the slit separation a is much larger than the slit width b and a coherent laser beam with wavelength λ at the wall distance L from the slits. The law of cosines gives that $r_2 = (r_1^2 + a^2 - 2ar_1\cos(\frac{\pi}{2} - \theta))^{\frac{1}{2}} = r_1(1 + (\frac{a}{r_1})^2 - 2\frac{a}{r_1}\cos(\frac{\pi}{2} - \theta))^{\frac{1}{2}}$ [1].

Expand r_2 in powers of $\frac{a}{r_1}$ to third order about $a = 0$ (mind the square root). Derive a condition such that the second order term in $\frac{a}{r_1}$ is much smaller than one wavelength λ (negligible effect on the phase). This approximation proves very useful when the angle is small, the distance L is large compared to the parameter a.

Hint: Use a mathematical engine (ex: symbolab.com) to calculate the three derivatives with a, then evaluate, only then move to formulating in $\frac{a}{r_1}$. Do not do these derivatives by hand, but if you do, check units at each step.

It is given, or may be derived from geometry:

$$r_2 = (r_1^2 + a^2 - 2ar_1\cos(\frac{\pi}{2} - \theta))^{\frac{1}{2}} = r_1(1 + (\frac{a}{r_1})^2 - 2\frac{a}{r_1}\cos(\frac{\pi}{2} - \theta))^{\frac{1}{2}} \tag{1}$$

First change cos to sin purely for convenience.

$$r_2 = r_1(1 + (\frac{a}{r_1})^2 - 2\frac{a}{r_1}\sin(\theta))^{\frac{1}{2}} \tag{2}$$

Find the necessary derivatives for a third order Taylor expansion.

$$\frac{dr_2}{da} = \frac{\frac{a}{r_1} - \sin\theta}{\sqrt{1 + (\frac{a}{r_1})^2 - 2\frac{a}{r_1}\sin(\theta)}} \tag{3}$$

$$\frac{d^2r_2}{da^2} = \frac{1}{r_1\sqrt{1 + (\frac{a}{r_1})^2 - 2\frac{a}{r_1}\sin(\theta)}} - \frac{(\frac{a}{r_1} - \sin\theta)(\frac{a}{r_1^2} - \frac{\sin\theta}{r_1})}{(1 + (\frac{a}{r_1})^2 - 2\frac{a}{r_1}\sin(\theta))^{\frac{3}{2}}} \tag{4}$$

$$\frac{d^3r_2}{da^3} = -\frac{\frac{a}{r_1} - \sin\theta}{r_1^2(1 + (\frac{a}{r_1})^2 - 2\frac{a}{r_1}\sin(\theta))^{\frac{3}{2}}} + \frac{3(\frac{a}{r_1} - \sin\theta)(\frac{a}{r_1^2} - \frac{\sin\theta}{r_1})^2}{(1 + (\frac{a}{r_1})^2 - 2\frac{a}{r_1}\sin(\theta))^{\frac{5}{2}}} \tag{5}$$

$$-\frac{(\frac{a}{r_1^2} - \frac{\sin\theta}{r_1})}{r_1(1 + (\frac{a}{r_1})^2 - 2\frac{a}{r_1}\sin(\theta))^{\frac{3}{2}}} - \frac{(\frac{a}{r_1} - \sin\theta)}{r_1^2(1 + (\frac{a}{r_1})^2 - 2\frac{a}{r_1}\sin(\theta))^{\frac{3}{2}}} \tag{6}$$

Correspondence E-mail: bschrey1@terpmail.umd.edu

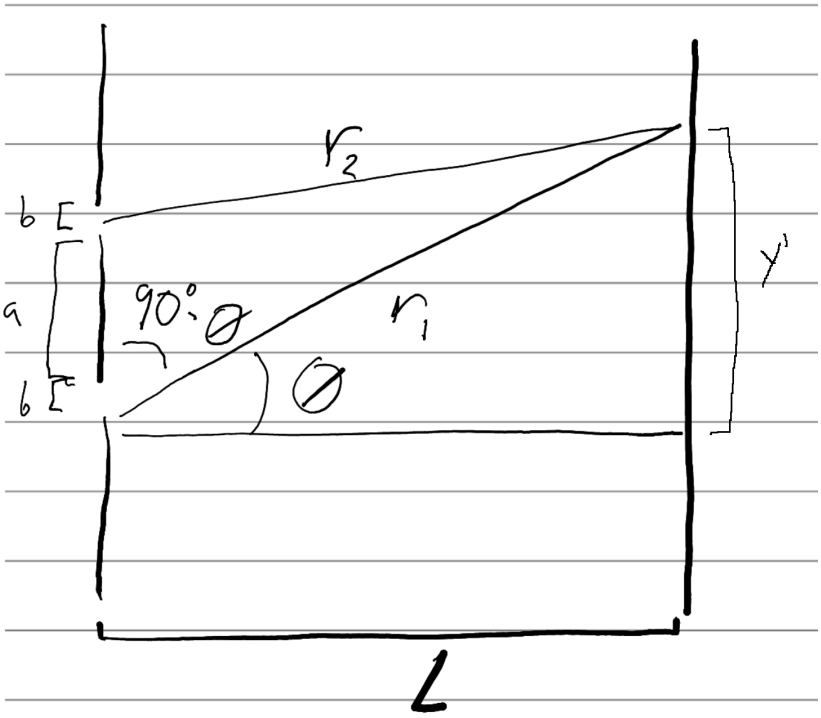

Figure 1. Diffraction over a length L from two slits.

Plug in $a = 0$.

$$\frac{dr_2}{da}|_{a=0} = -\sin\theta \tag{7}$$

$$\frac{d^2r_2}{da^2}|_{a=0} = \frac{1}{r_1} - 2\frac{\sin^2\theta}{r_1} \tag{8}$$

$$\frac{d^3r_2}{da^3}|_{a=0} = 3\frac{\sin\theta}{r_1^2} - 3\frac{\sin^3\theta}{r_1^2} \tag{9}$$

Assemble the zero centered Taylor series.

$$r_2 = \frac{r_1}{0!} - \frac{a}{1!}\sin\theta + \frac{a^2}{2!}(\frac{1}{r_1} - 2\frac{\sin^2\theta}{r_1}) + \frac{a^3}{3!}(3\frac{\sin\theta}{r_1^2} - 3\frac{\sin^3\theta}{r_1^2}) \tag{10}$$

Finally investigate the second order term in comparison to the wavelength.

$$|\frac{a^2}{2!}(\frac{1}{r_1} - 2\frac{\sin^2\theta}{r_1})| << \lambda \tag{11}$$

Let $\sin\theta = 1$ and get rid of constant factors ($\frac{1}{2!}$). This does not bias the inequality.

$$\frac{a^2}{r_1} << \lambda \tag{12}$$

If r_1 is assumed to be minimally L, then this is the Fraunhofer condition for diffraction.

2. Arbitrary 1d Grating The steady state phase for the spherical wave is $e^{i\frac{2\pi}{\lambda}r_{1/2}}$. Recall under the Fraunhofer condition for two slits $r_2 = r_1 - asin\theta$. Consider now instead of slits at 2 determined positions that an arbitrary slit pattern $S(x)$ of some spatial extent $2y$. $S(x)$ takes on only the values 0 and 1 (fully blocked or fully exposed). The source is still fully coherent. In the case of two slits, you added up two complex phasors, $A_{wall} = B(e^{i\frac{2\pi}{\lambda}r_2} + e^{i\frac{2\pi}{\lambda}r_1}) = Be^{i\phi_{global}}(1 + e^{i\Delta\phi})$.

Now write the pattern at the wall as an integral of the slit pattern $S(x)$, where x can be taken to equal a as defined in problem 1. To find the correct phase difference for each x, assume the condition from 1 ($\frac{y^2}{L} << \lambda$)holds (resulting in $\Delta\phi = -2\pi\frac{a\sin\theta}{\lambda}$, $\Delta\phi$ is the phase change relative to the contribution of the central spherical wave).

The integral The integral is as follows, where A' is the solution at the plane of observation:

$$A'(\theta) \propto \int_{-y}^{y} e^{-2\pi i\frac{y\sin\theta}{\lambda}} S(y)dy \tag{13}$$

Take the small angle approximation on θ ($\sin\theta = \frac{y'}{L}$), explain how this makes the integral resemble the Fourier transform.

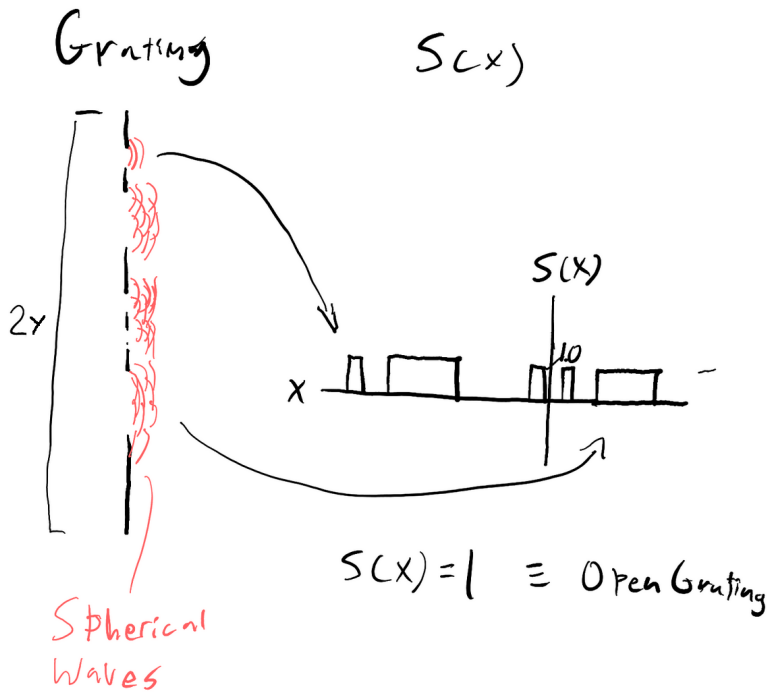

Figure 2. Diffraction over a length L from two slits.

Solution The approximation on θ ($\sin\theta = \frac{y'}{L}$)

$$A'(y') \propto \int_{-y}^{y} e^{-2\pi i\frac{y\cdot y'}{\lambda L}} S(y)dy \tag{14}$$

$$A'(y') \; looks \; like \; FT_y\{S(y)\} \; where \; the \; frequency \; is \; \; scaled\frac{1}{\lambda L} \tag{15}$$

Proportionality is usually all one cares for, since conservation of energy can find the magnitude of the proportionality, and global phase has no physical effect.

2. SAMPLING THE PHOTON AMPLITUDE

The python packages numpy and matplotlib will provide extremely useful functionality. First the boundary conditions for diffraction slits will be shown and sampled.

2.1 Sampling and Showing Photon Intensity

While figure 1 shows a diffraction slit that is continuous in space (x can be any real number between $-l$ and l, l is chosen rather than y to reduce notational confusion), the finite size of a computer demands a discretization be made. The pattern will be sampled at 32 points, which is shown in figure 2.1. The photon intensity proportional to $|A|^2$ is plotted first.

For code the following imported python modules, and definitions are always assumed:

```
import numpy as np
import matplotlib.pyplot as plt
#be explicit about units
meter = 1
millimeter = 10**-3 * meter
nanometer = 10**-9 * meter

```

The slit function $S(x)$ can be formulated using logical operators and inequalities on the argument x, since it is a sum of step functions

```
#l defines the physical extent of the diffraction slits
l = 10**-1 * millimeter

#function that defines the 1d slit pattern
def s(x):

  #np.logical_and returns 1.0 if both conditions are true, 0.0 otherwise, very useful
  #, add up each slit in the pattern this way
  res = np.logical_and(-l < x, x < -0.9 * l)

  res += np.logical_and(-0.7*l < x, x < -0.4 * l)

  res += np.logical_and(-0.05*l < x, x < 0.0 * l)

  res += np.logical_and(0.05*l < x, x < 0.1 * l)

  res += np.logical_and(0.45*l < x, x < 0.9 * l)

  return res

```

Now apply sampling, *np.linspace* allows the programmer to easily generate evenly spaced sample coordinates, by specifying the domain to sample over ($(-l, l)$ in the current case) and the number of samples.

```
#Take a large amount of sample coordinates (1024)
extreme_sampling = np.linspace(-l,l, 1024)
#Plot these samples using plot, there are so many it is
#reasonable to treat them as continuous , take np.abs squared to show the photon
    intensity only
plt.plot(extreme_sampling / millimeter, np.abs(s(extreme_sampling))**2, label = "
    underlying function", c ="orange")
```

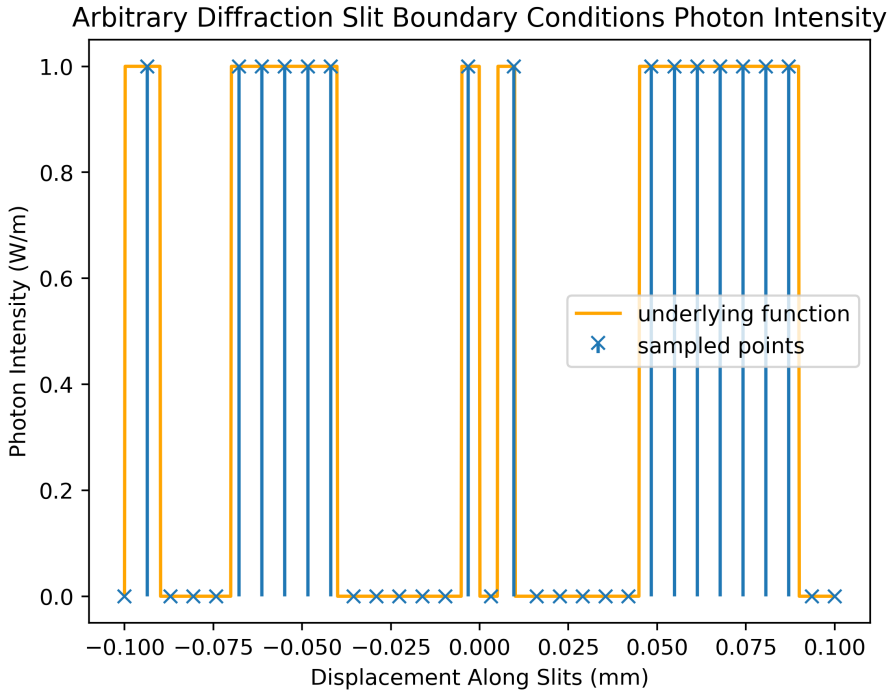

Figure 3. Diffraction slit pattern in 1 dimension with sampling shown.

```

#Take a small amount of sample coordinates for illustrative purposes
sample_points = np.linspace(-1, 1, 32)
#Use a stem plot, since the sample count is low, take np.abs squared to show the
    photon intensity only
plt.stem(sample_points / millimeter, np.abs(s(sample_points))**2, markerfmt = 'x',
    basefmt=" ", label = "sampled points")

#plotting format omitted...

```

2.2 Showing Phase

As discussed previously, complex numbers play an important role in calculating the propagation of light, but the plotting so far has not shown the complex nature of the photon amplitude for the diffraction slit. The photon intensity proportional to $|A|^2$ has been plotted, but the phase must additionally be plotted. The angle function in numpy will give the phase of a complex number in radians, exactly what is needed. The following code produces a phase plot for the 1d arbitrary patterned slit. All that is changed from the photon intensity plotting example is replacement of abs squared by the numpy angle function. The resulting plot is boring, because for the arbitrary pattern slit problem, the diffraction slits only block light, and never change the phase. Later examples will show that components like the thin lens (useful for focusing light), will change phase, and make such plotting much more informative.

```
#Take a large amount of sample coordinates (1024)
extreme_sampling = np.linspace(-1,1, 1024)
#Plot these samples using plot, there are so many it is
#reasonable to treat them as continuous
```

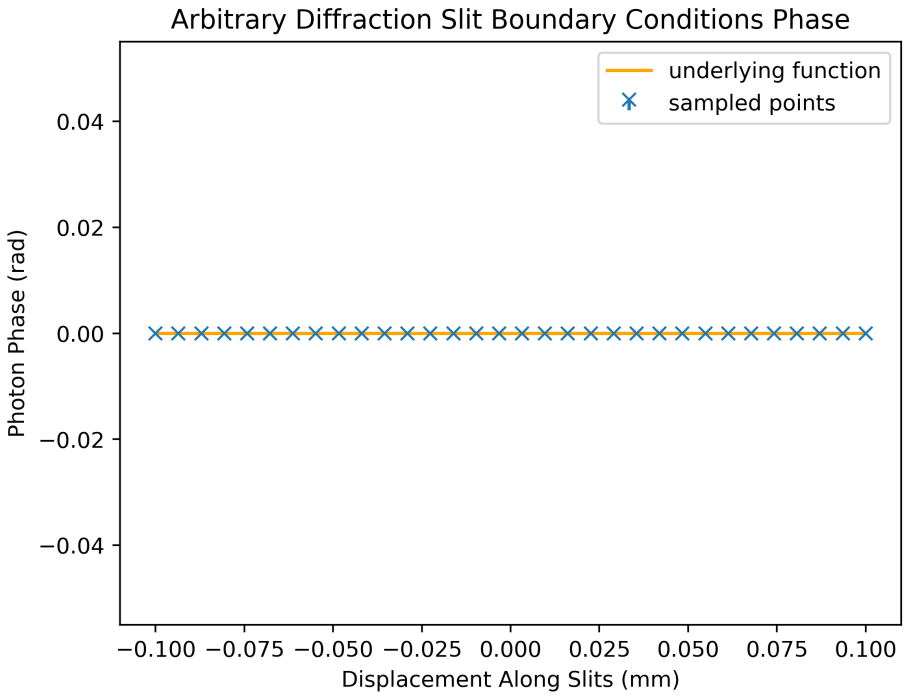

Figure 4. Diffraction slit pattern photon phase in 1 dimension with sampling shown.

```
plt.plot(extreme_sampling / millimeter, np.angle(s(extreme_sampling)), label = "
    underlying function", c ="orange")

#Take a small amount of samples for illustrative purposes
sample_points = np.linspace(-1, 1, 32)
#Use a stem plot, since the sample count is low
plt.stem(sample_points / millimeter, np.angle(s(sample_points)), markerfmt = 'x',
    basefmt=" ", label = "sampled points")

#plotting format omitted...
```

2.3 Utility Functions

Useful functions are introduced:

1. *normalize* Take a photon amplitude, and scale it so that the sum of its magnitudes square $|A_j|^2$ is 1.
2. *normalize_max* Take a photon amplitude, and scale it so that its maximum magnitude is 1, nice for plotting.
3. *plot_intensity_1d* Plot the intensity of a 1d boundary/plane amplitude.
4. *plot_phase_1d* Plot the phase of a 1d boundary/plane amplitude.

3. PROPAGATING TO A FAR PLANE

The code implemented thus far has not given new information about the diffraction problem, and specified it on a computer. (14) will be used (after verifying the approximations made are acceptable) to calculate the diffraction pattern far away from the slits.

3.1 Approximations

The approximations made were that the angle from the slits to any points on the observation plane is small, and the Fraunhofer condition (12). Since y is $10^{-4}m$, and we will assume light of wavelength $500nm$, propagating a distance of $10m$ will be sufficient to satisfy the approximation. The calculation is done explicitly.

$$\frac{(10^{-4}m)^2}{10m}? << 500nm \tag{16}$$

$$10^{-9}m << 5 \cdot 10^{-7}m \tag{17}$$

For the small angle, the calculated photon amplitude $L = 10m$ will be cropped, such that $\frac{y'}{L}$ is small, ensuring a small angle.

3.2 The Discrete Fourier Transform

To calculate the Fourier transform in code, the discrete Fourier transform (DFT) is used, a formulation of the Fourier transform assuming instead of a continuous domain, a discrete, evenly spaced domain of samples and sample coordinates. The fast Fourier transform (FFT) algorithm calculates the DFT much more quickly, and is implemented in the numpy library. Additionally the DFT also assumes the function to be periodic at least within the sampled domain (in the current case $(-y, y)$).

The Fourier transform that has been seen so far is

$$FT\{A(x)\}(\tilde{x}) = \int_{-\infty}^{\infty} e^{-2\pi i \tilde{x} x} A(x) dx \tag{18}$$

Usually the bounds of integration have been cut off, say $(-l, l)$ because the photon amplitude is of finite extent in space, so there is no need to integrate to infinity.

Under the assumption of discrete sampling, and periodicity within the sampling domain, the discrete Fourier transform approximates the Fourier transform. A_j are the samples taken of the continuous amplitude $A(x)$, the index j represents discrete samples of the domain x occupies. The number of samples is N. In python the indices are used to access numpy arrays, which represent the mathematical objects A_j, $DFT\{A_i\}_m$ on the computer.

$$DFT\{A_i\}_m = \sum_{j=0}^{N-1} A_j e^{-i\frac{(2\pi)jm}{N}} (\frac{2l}{N}) \tag{19}$$

A_j is analogues to $A(x)$, and $(\frac{2y}{N})$ to dx in (18). Index m is the index of the resulting discrete Fourier transform. The discrete Fourier transform is an example of a change of basis in linear algebra.

3.3 Propagation

Let us now take a double slit source, and apply (14) to find how it diffracts, in code. This simple example is chosen because the expected form is known and easily recognizable.

The boundary amplitude emitted from the slits is as expected for a double slit, shown in figure (3.3). The phase is not plotted as it is uniform. From (14) the propagated amplitude is more specifically written

$$A'(y') = FT\{A(y))\}(\frac{y'}{\lambda L}) \tag{20}$$

This formula must be discretized. The immediate approach is to go to a DFT rather than a normal Fourier transform.

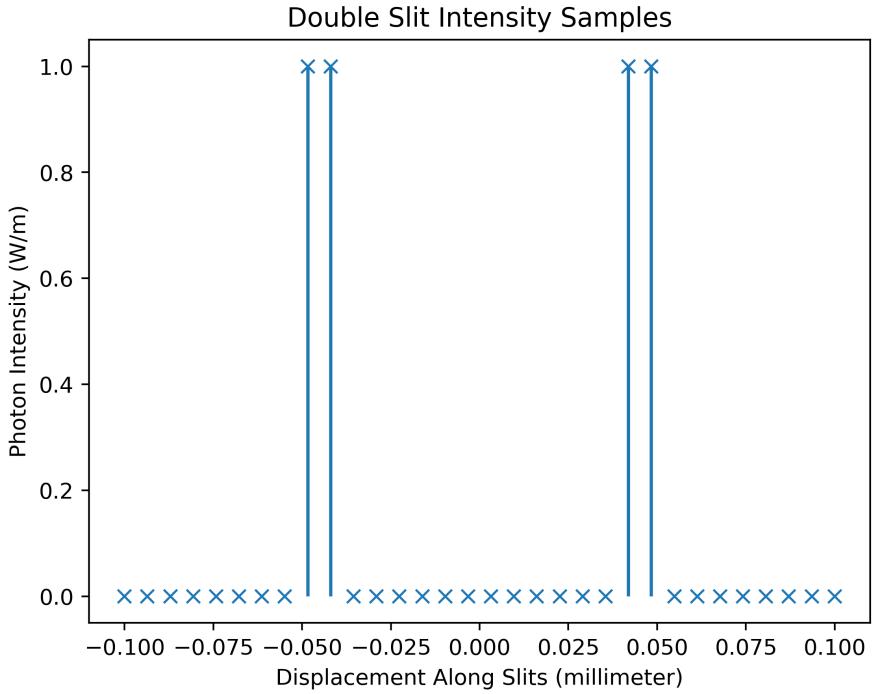

Figure 5. Intensity over the double slit at the plane of emission.

$$A'_m = DFT\{A_j\} \tag{21}$$

The issue is, how does one account for the $\frac{1}{\lambda L}$ scaling of the coordinate? Later we will see this scaling is physically related to the inverse square law. The solution is not to try and scale the integer index m by $\frac{1}{\lambda L}$, but instead, to scale the coordinate associated with each index appropriately.

So a fully rigorous expression in the discretized case requires introducing the spatial coordinate of each amplitude sample Y_j, then defining new linearly scaled coordinates Y'_j at the plane of observation.

$$A'_m = DFT\{A_j\},\ Y'_j = Y_j \cdot \lambda L \frac{N}{(2l)^2} \tag{22}$$

By accounting the scaling of the continuous case into the coordinates, the problem is solved. The secondary equation is just a scaling of the boundary amplitude coordinates. The appearance of λL is obvious, but what accounts for $\frac{N}{(2l)^2}$? To get spatial coordinates out at the end the factor scaling Y_j the array of coordinates must be unit less, but this is not the full story. When the DFT is taken, the frequency components have units of $\frac{1}{2l}$, since it was assumed the longest period of the photon amplitude was $2l$. Since the units given are spatial $(-l, l)$, or discretized difference of $\frac{2l}{N}$, the process of taking the Fourier transform must have changed the units by scaling $\frac{2l}{N}$, into $\frac{1}{2l}$, which is a multiplication by $\frac{N}{(2l)^2}$. Then add on the scaling λL apparent in the continuous form (20) to get $Y'_j = Y_j \cdot \lambda L \frac{N}{(2l)^2}$.

All that is left now is to implement the DFT and coordinate scaling in code. First *double_slit* is implemented, in the same way 2.1. This is used to generate the boundary amplitude:

```
boundary_coordinates =  np.linspace(-l, l, 32)
boundary_amplitude_samples = double_slit(boundary_coordinates)
```

Then the propagation by distance L is implemented, care must always be taken that L is appropriately large, and that the angle subtended by the resultant pattern is sufficiently small.

```
def propagate(A, coordinates, photon_lambda, L):
  #scale the input coordinates correctly
  propagated_coordinates = boundary_samples  * photon_lambda * L / (np.max(coordinates
    )-np.min(coordinates))**2 * 32
  #apply the FFT (fast DFT) to the field to find the diffraction pattern
  #np.fft.fftshift swaps around the indices of the returned fft from np.fft.fft, since
     the
  #returned indices before fftshift are not in the normal convenient order physicists
    like to deal with
  propagated_amplitude = normalize(np.fft.fftshift(np.fft.fft(boundary_amplitude)))

  #return the calculted propagation
  return propagated_amplitude, propagated_coordinates
```

This propagation is tried on the double slit amplitude boundary. Note the use of *np.fft.fftshift* function, which orders the discrete Fourier transform indices in ascending order, needed as this is the assumption of the analytical expression. The result is shown for the intensity in figure 3.3

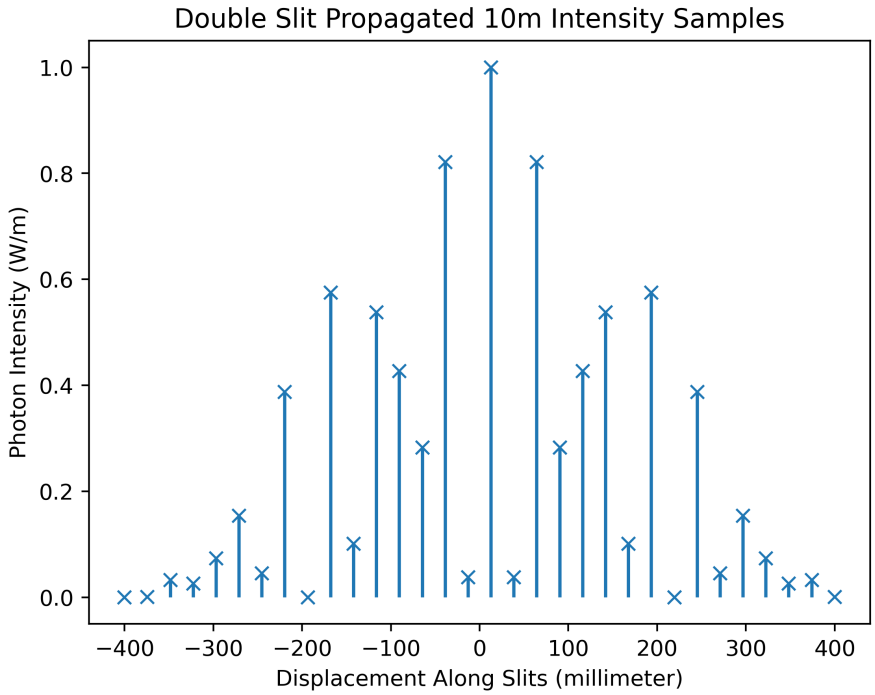

Figure 6. Intensity from a double slit $10m$ away.

The shape is correct. To ensure the scaling is correct, recall the spacing of modes for the double slit $\Delta y' = \frac{\lambda L}{d}$ where d is the slit separation. For this case $\Delta y' = 50mm$, which matches the numerical result. As a reminder $\Delta y'$ can be interpreted as the spacing between maxima, or minima. The angle subtended by this pattern is about 12 degrees, which is pushing the limit of a small angle.

This numerical method for calculating diffraction/propagation of light is limited, mainly by the Fraunhofer approximation, so that close in propagation ia not feasible. The small angle approximation limit can be side stepped by working in the original coordinate θ, r rather than L, y'.

Understanding this method will prepare one for a more capable version of the same numerical method, allowing not just far field diffraction calculations, and build familiarity with the DFT, crucial in other propagation formulations.

4. CITATIONS/RESOURCES

All the code to generate plots is made available with this PDF.

1. Hecht, Eugene (2017). "Problem 9.21". Optics (5th ed.). Pearson. p. 453. ISBN 978-1-292-09693-3