Lecture Notes 5 Introduction to Fourier Optics

Benjamin Schreyer

University of Maryland, College Park, Maryland

June 11th 2024

1. 2D BOUNDARIES AND DIFFRACTION

Until now all calculations have existed in the realm of 1d diffraction. In reality, when one places a double slit in front of a laser and watches it diffract, or looks up at the stars with a telescope, the optical information available is 2d, and only due to the phenomena of the separability of two two transverse dimensions (called x and y) was the simplification of a 1d diffraction calculation allowable. First 2d light fields will be introduced, then it will be shown that their diffraction reduces to 2 1d calculations, and further the already shown Fraunhofer and Fresnel single Fourier transform methods for calculating diffraction will be extended for propagation of light between 2d planes.

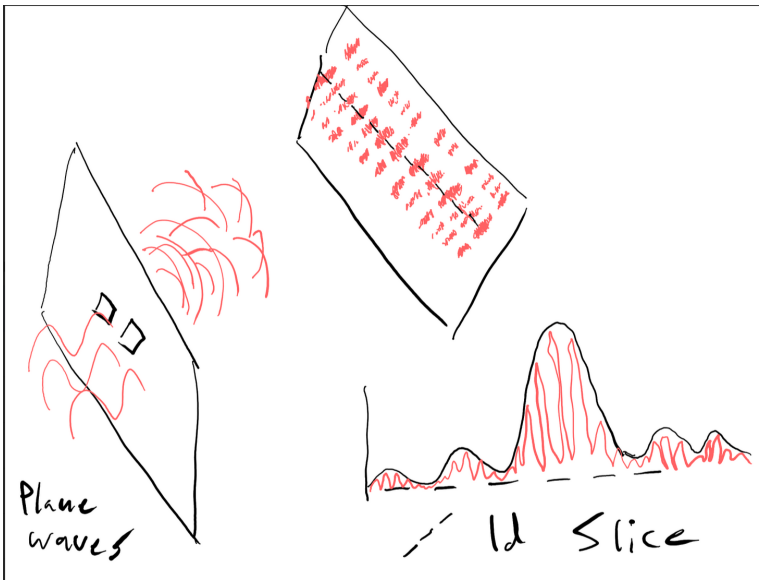

Figure 1. A real diffraction pattern is the multiplication of two 1d diffraction patterns, so the double slit experiment shows single slit modulation in the transverse axis that up until now has been ignored.

1.1 Two dimensional Wave Amplitude

First to ensure there is familiarity with the mathematical objects at hand, the 2d E-field or scalar wave amplitude is a complex valued function of two real variables, the transverse coordinates x and y. Just like in 1d, the square of this E-field should have units of $\frac{W}{m^2}$ or just $\frac{1}{m^2}$, to represent the intensity of radiation, or the intensity of photons, which are carrying energy in proportion to their number. As in 1 dimension, the wave amplitude can usually be scaled by a constant phase or magnitude, without destroying information, as long as one does not care about the total power of the radiative phenomena, or keeps it handy to correct normalization.

The wave amplitudes that were previously only analyzed in 1d are now sampled on a 2d grid of coordinates, the following function using the numpy library to generate such a 2d grid of coordinates. The resultant array

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

now has three indices, the grid index for the x axis i, the grid index for the y axis j, and the coordinate value x or y. In our code the last axis will also be the choice of x or y coordinate value. One can also eliminate the last index by keeping record of two doubly indexed arrays, one for x and one for y, rather than one array with three indices. Generating coordinates of the 2d samples can be efficient but requires a new numpy built in *mgrid*.

```
def generate_2d_coordinates(grid_size,grid_dimension):

  #Build an array of appropriate size, square of grid points, each point has 2 values,
     its x coordinate, and y coordinate
  coordinate_array =   grid_dimension / grid_size * np.mgrid[- grid_size // 2:
    grid_size // 2,- grid_size // 2: grid_size // 2]

  #Return the grid, place the non spatial index last (x,y coordinate pair), this is
    generally advised, helps with array manipulations
  # but is not necessary
  return np.swapaxes(coordinate_array,0,2)
```

To understand use of this code, consider the case grid_size $=$ 12, then the shape of the output array will be $(12, 12, 2)$, with two slots for the last index, the x coordinate, and y coordinate of the given grid cell.

An example 2d field could be $A(x, y) = B(x)B(y)$. Where B is the tophat or boxcar function. This 2d function would show up as a single slit diffraction boundary, but, in 2d, hence a boxcar function in both coordinates x and y. Figure 1.1 shows such a field amplitude's intensity, plotted as an image which is a common approach to plotting for 2d fields. Matplotlib provides the function *imshow* that will interpret a 2d array of values as a black and white image. A helper function is given that plots 2d wave amplitude intensity.

```
#Plotting utility for 2d wave amplitudes
def plot_2d_intensity(coordinates, amplitude, name = ""):
  plt.title(name + " 2d Intensity")

  #The intensity is maximum normalized, and the coordinate grid is assumed to be
    square
  #Use the extent =[...] parameter to change the axes to having units of length,
    rather than index
  plt.imshow(np.square(np.abs(normalize_max(amplitude))), extent = [np.min(coordinates
    / millimeter), np.max(coordinates / millimeter)] * 2)
  plt.xlabel("x (mm)")
  plt.ylabel("y (mm)")
  plt.colorbar()
```

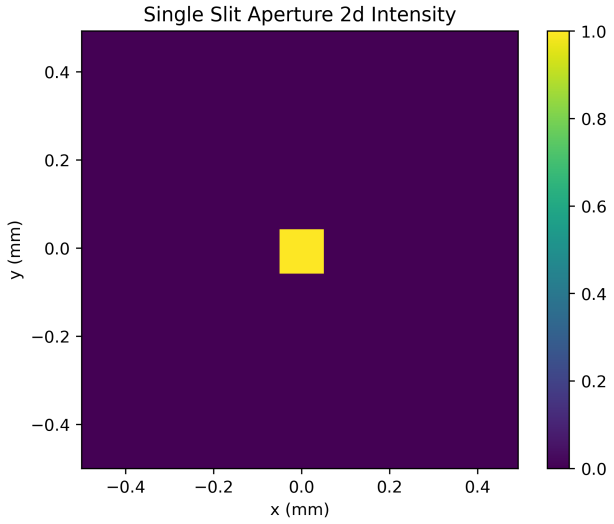

Figure 2. Two dimensional product of boxcar functions of the x and y coordinates, forms a rectangular region of full amplitude, and zero amplitude everywhere else. Color indicates intensity normalized from 0-1.

A double slit can easily be modeled by making one side of the boxcars shorter, and additively including a second slit shifted over some. Figure 1.1 shows the double slit pattern, as a superposition of two 2d boxcar amplitudes. The following code generates a double slit using the previously implemented *boxcar* function. Note the use of each coordinate, with accessor [:,:,0] denoting all x sample coordinates, and accessor [:,:,1] denoting all y sample coordinates. For each slit, one boxcar function is called on each coordinate, then they are multiplied.

```
#Double slit in 2d formed out of boxcar functions
def double_slit_2d(coordinates,separation, width):

 #width * 10 is so that the slits are "slit-like"
 #rather than squares, taller than they are wide

 #left slit
 amplitude = boxcar(width * 10, coordinates[:,:,1])  * boxcar(width , coordinates
   [:,:,0])
 #right slit
 amplitude += boxcar(width * 10, coordinates[:,:,1])  * boxcar(width, coordinates
   [:,:,0] - separation)
 return amplitude

```

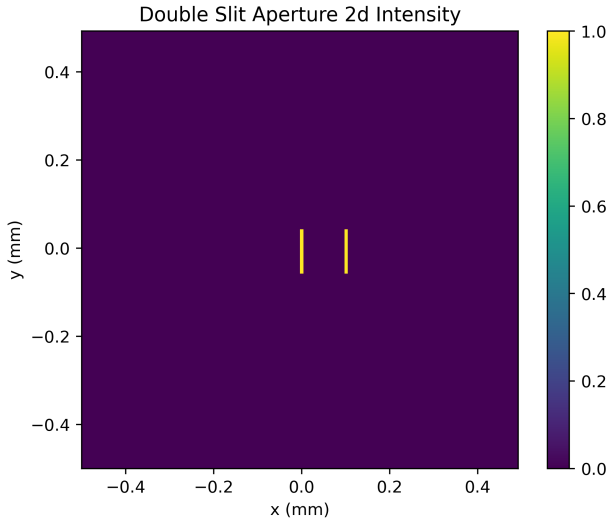

Figure 3. Two dimensional double slit intensity plot, the 2d double slit is a superposition of a product of boxcar functions of the x and y coordinates. Color indicates intensity normalized from 0-1.

As a final example, a two dimensional Gaussian intensity is shown. This is an extremely important example from laser optics. To a great approximation, many laser systems output a wave amplitude that has a Gaussian amplitude fall off in both transverse coordinates, with uniform phase. Figure 1.1 shows an example with $\sigma = 0.1mm$.

$$A_{laser}(x, y) \propto e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{1}$$

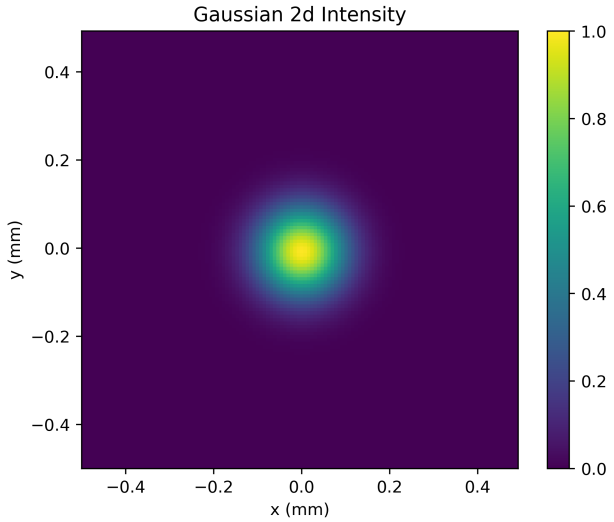

Figure 4. Two dimensional Gaussian laser beam intensity. Color indicates intensity normalized from 0-1.

It is often useful to look at a cut of a 2d amplitude or phase graph along one axis x or y. Figure 1.1 shows a cross section of a the above 2d Gaussian amplitude for a constant value of the y coordinate.

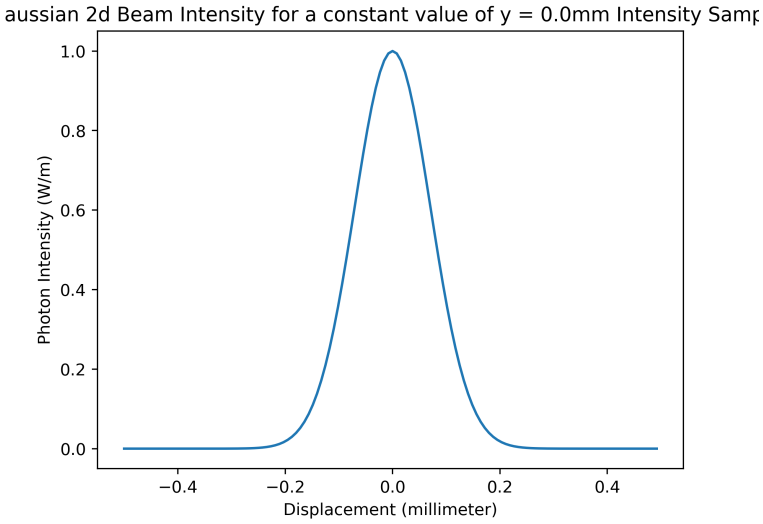

Figure 5. 1d cut of the 2d Gaussian intensity.

Using the same sampling scheme but on a Gaussian function looks like this is in code.

```
#Helper function to create a Gaussian falloff amplitude field
def gaussian_amplitude(coordinates, sigma, x0=0, y0=0):
  return np.exp(-1 * ((coordinates[:,:,0]-x0)**2 + (coordinates[:,:,1]-y0)**2)/2 /
    sigma**2)
```

1.2 Phase Diagrams

Up until now phase of the boundary has largely been ignored, because the example of diffraction slit patterns in its simplest form does not require phase of the boundary to vary. One of the simplest physical cases where phasing that varies across the boundary amplitude is with a mirror. If a mirror is slightly tilted, say half a degree for example, and light is reflected off of it, this can be modeled as applying a linearly varying phase to the amplitude field as shown in figure 1.2. This linear phase variation is the result of light across the mirror taking a longer path length before it is reflected back. Note only small tilts can generally be thought of in this way, since with a large tilt and large mirror, the beam will continue to diffract and change shape, before hitting different parts of the mirror, making the mirror's effect more than just a modification of the phase. As will be seen later, this is a similar approximation to the one made when considering the thin lens. Using some simple geometry, under such a conditions, the phase change for a mirror tilted by rotation in the y axis by angle θ is:

$$\Delta\phi(x) = 2\pi\frac{2\tan(\theta)x}{\lambda} \tag{2}$$

The simple path length model gives that the phase change is just the mirror surface height difference, which must be put into radians using the wavelength. Usually it is taken $\tan(\theta) = \theta$ because θ should be a small angle given the previous approximations. The factor of two in the numerator comes from the light making the trip to the mirrors surface, and then back.

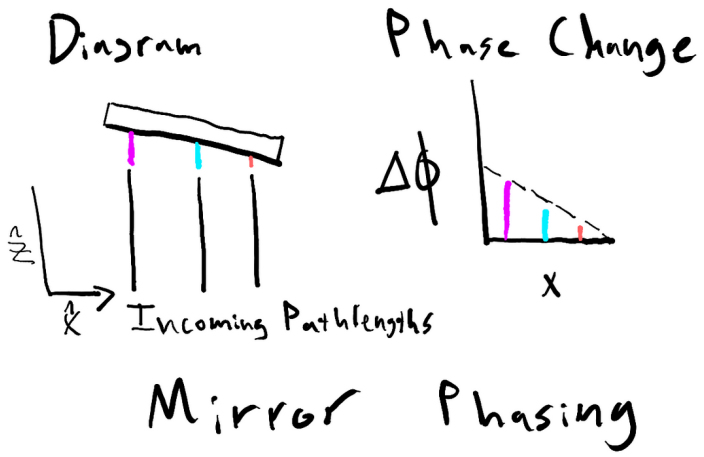

Figure 6. Simple path length model of a mirror, ignoring the change in intensity shape over a small distance. This model results in a linear phase change.

A small angle mirror's phasing can be implemented simply in code. To apply the phase change as given by this code, the returned phasor field is multiplied on an amplitude field, since when the phase is in the complex exponential $e^{i\alpha}$ phasor form, the addition of the phase argument is a multiplication for exponentials.

```
#return the phase change as a phasor array for a tilted mirror
def small_angle_mirror_phase(coordinates, photon_lambda, angle, axis = "y"):
  if angle > 0.2:
    print("Not a small angle mirror")
  if axis == "y":
    return np.exp(1.0j * 2 * np.pi * 2 * angle * coordinates[:,:,0] / photon_lambda)
  else:
    return np.exp(1.0j * 2 * np.pi * 2 * angle * coordinates[:,:,1] / photon_lambda)
```

A plot of the phasing by a perfectly flat mirror is compared to a mirror that is tilted about the y axis. The intuition about a mirror on a beam of light is that it reflects light in a rotated direction. This intuition is correct for the small angle mirror, and can be seen as the transverse phase induced causing transverse waves to travel across the light, causing it to propagate in the x or y axis and not just straight ahead in the z direction. Such transverse waves can be seen in the right plot of figure 1.2

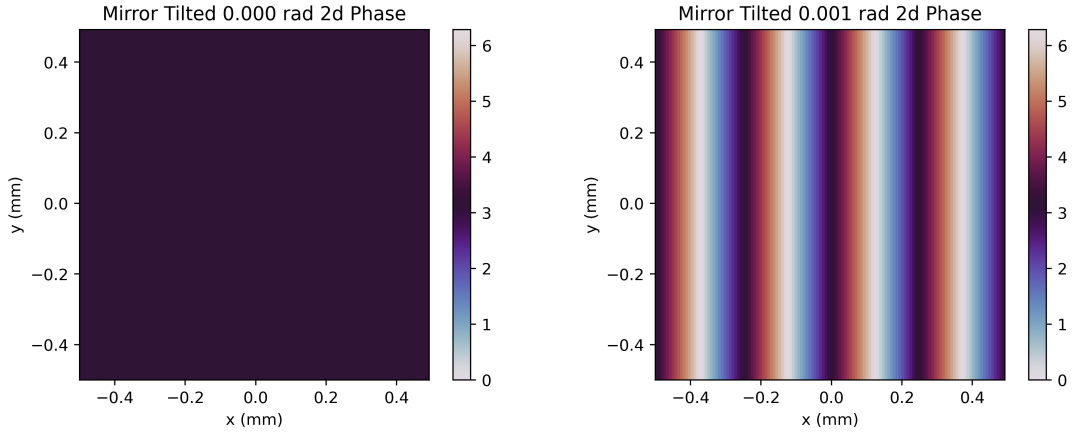

Figure 7. Phasing by an untilted mirror (left) compared with phasing by a tilted mirror. Coloring indicates phase 0 to 2π

To generate these plots, a helper function was used again using matplotlib's *imshow*. The important parts of the function are its use of numpy's *angle* to return the phase of a complex number as scalar $-\pi$ to π, and the cyclic red and blue colormap, which allows the phases 0 and 2π to be plotted as the same color, since the phase they determine is the same.

2. SEPARATION OF THE TRANSVERSE AXES

DONT SEPARATE, JUST SHOW 2d CIRCULAR WAVE FRONTS GIVE FRESNEL FT FORM

There should be a clear picture of what boundary conditions look like in 2d. Now the actual problem of propagating these boundary conditions is introduced. Adding dimensions to a problem is physics is more generally, not easy. It will be shown here however that in the case of linear optics, the problem will not become any more complicated, save for keeping track of more dimensions in arrays, (or function arguments in the unsampled case).

Consider the partial differential form of the wave equation

$$\frac{1}{c^2}\frac{\partial^2}{\partial t^2}A = (\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2})A \tag{3}$$

To solve diffraction problems of $2d$ planes more simply, one must show that the solutions of the wave equation separate. This is done exactly by the method of separation of variables. Separating out one axis x or y is of interest here, so assume $A = X(x)B(y,z,t)$. Plugging this assumption into (2) is the first step.

$$\frac{1}{c^2}\frac{\partial^2}{\partial t^2}X(x)B(y,z,t) = (\frac{\partial^2 X}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2})X(x)B(y,z,t) \tag{4}$$

Now note that derivatives with any other variables vanish for X, and the derivative with x for B is also zero. This may be seen as treating X and B as constants under the right derivatives, or as application of the product rule.

$$\frac{1}{c^2}X(x)\frac{\partial^2 B(y,z,t)}{\partial t^2} = \frac{\partial^2 X}{\partial x^2}B(y,z,t) + X(x)(\frac{\partial^2 B(y,z,t)}{\partial y^2} + \frac{\partial^2 B(y,z,t)}{\partial z^2})) \tag{5}$$

Now continuing to procedure to separate x, move all dependence on x to one side of the equation, and all other variables to the other, showing that both sides of the equation must be equal to some constant λ.

$$[\frac{1}{c^2}\frac{\partial^2 B(y,z,t)}{\partial t^2} - \frac{\partial^2 B(y,z,t)}{\partial y^2} - \frac{\partial^2 B(y,z,t)}{\partial z^2}]\frac{1}{B(y,z,t)} = \frac{\partial^2 X}{\partial x^2}\frac{1}{X(x)} = \lambda \tag{6}$$

It is useful to look at the equation for X:

$$\lambda X = \frac{\partial^2 X}{\partial x^2} \tag{7}$$

This admits solutions $e^{\pm i\sqrt{|\lambda|}x}$.

This separation can be repeated to show that each variable is intact separable and has complex exponential solutions, with the frequencies in time and space (which are related to the separation constants, the first λ) restricted by $c^2 = \frac{\omega^2}{k_x^2+k_y^2+k_z^2}$, which is just the limitation on the speed of light. Since the solutions to the equation can be written as products of functions of single coordinate x, y, z, t one can always fix a coordinate to be constant, and the behavior in the other axes will be the same.

Let this be applied to the single Fourier Transform expression of the Fresnel propagation, note L dependence only comes in once, so the two transverse coordinate propagations are not just being multiplied.

$$X'(x)Y'(y) \propto e^{\frac{i2\pi}{\lambda}L} e^{\frac{i\pi}{\lambda L}y'^2} FT_y\{Y(y)e^{\frac{i\pi}{\lambda L}y^2}\}(\frac{y'}{\lambda L}) \cdot e^{\frac{i\pi}{\lambda L}x'^2} FT_x\{X(x)e^{\frac{i\pi}{\lambda L}x^2}\}(\frac{x'}{\lambda L}) \tag{8}$$

The multiplication of the Fourier Transforms, importantly since they are on different variables, may be taken as a two dimensional Fourier Transform. This is starkly different from the case of the product of two Fourier transforms of the same variables, where the convolution is then involved.

$$X'(x)Y'(y) \propto e^{\frac{i2\pi}{\lambda}L} e^{\frac{i\pi}{\lambda L}(x'^2+y'^2)} FT_{x,y}\{Y(y)X(x)e^{\frac{i\pi}{\lambda L}(x^2+y^2)}\}(\frac{x'}{\lambda L}, \frac{y'}{\lambda L}) \tag{9}$$

There is a key detail, missing, the initial condition $Y(y)X(x)$ on its own is not fully descriptive. Rather a sum over $Y_i(y)X_j(x)$ must be taken to describe some amplitude $A(x, y)$ which has arbitrary dependence on both variables. Since the Propagation operation on $Y(y)X(x)$ is linear. Its action on $\sum \alpha_{i,j} Y_i(y)X_j(x)$ follows immediately:

$$A'(x', y') \propto e^{\frac{i2\pi}{\lambda}L} e^{\frac{i\pi}{\lambda L}(x'^2+y'^2)} FT_{x,y}\{\sum \alpha_{i,j} Y_i(y)X_j(x)e^{\frac{i\pi}{\lambda L}(x^2+y^2)}\}(\frac{x'}{\lambda L}, \frac{y'}{\lambda L}) \tag{10}$$

Taking $Y_i(y), X_j(x)$ to be orthogonal basis functions (an assumption that will not be justified) on a finite region, it is clear this sum can be any function $A(x, y)$.

$$A'(x', y') \propto e^{\frac{i2\pi}{\lambda}L} e^{\frac{i\pi}{\lambda L}(x'^2+y'^2)} FT_{x,y}\{A(x, y)e^{\frac{i\pi}{\lambda L}(x^2+y^2)}\}(\frac{x'}{\lambda L}, \frac{y'}{\lambda L}) \tag{11}$$

This gives a solution to Fresnel diffraction in 2d. The 2d Fourier transform is almost as simple to approximate on a computer as the 1d case. This is easily extended to the more approximated case of Fraunhofer diffraction, and yields the expression:

$$A'(x', y') \propto e^{\frac{i2\pi}{\lambda}L} FT_{x,y}\{A(x, y)\}(\frac{x'}{\lambda L}, \frac{y'}{\lambda L}) \tag{12}$$

The above derivation uses some technical tricks, like the completeness of the basis of solutions, and separability. The propagation algorithm described above could easily have been dervied in the same way the 1d solution was derived, by using the Fresnel approxilation on $\sqrt{L^2 + r^2}$ where $r^2 = x^2 + y^2$, in cooperation with the Green's function solution $e^{ikr-i\omega t}$ to reach the 2d result.

The propagation of an arbitrary wave amplitude plane to a very close plane is yet to be feasible with these methods. For extremely close propagation, finite difference could be feasible, or kernel/convolution methods, which will be discussed further.

3. CITATIONS/RESOURCES

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