Lecture Notes 3 Introduction to Fourier Optics

Benjamin Schreyer

University of Maryland, College Park, Maryland

June 2nd 2024

1. THE INVERSE SQUARE LAW AND CONVENIENT COORDINATES

The problem of far field diffraction (diffraction given the Fraunhofer condition) for a 1 dimensional pattern has been shown, and solved numerically. The utility of this numerical approach is certain, and will not be surpassed for some applications, evidently as the approximation holds, but in addition the growing scale factor of the coordinates of discrete amplitude samples $\lambda L \frac{N}{(2l)^2}$ will prove useful.

The reason this is convenient that all sources of light in free space will eventually follow the inverse square law. This was shown rigorously for the case of a plane boundary condition (like a diffraction slit) in lecture one with the wave equation. Each part of the boundary contributed a half spherical wave to the radiative field. Once the distance to the source is large enough, detail is lost, and the factor of $\frac{1}{r}$ in the amplitude is $\frac{1}{r^2}$ in the intensity. The propagation considered is through vacuum, and so is lossless. By this principle, the intensity through far away surfaces from the sources must be spread over a larger and larger area, regardless of how the phases exactly interfere. This spread will be proportional to r^2, such that the product of the intensity and area are constant, to be physically accurate to the assumed constant power source of electromagnetic radiation. In the case of only considering one dimension of the diffraction pattern, a linear scaling of the pattern is observed, combined with the linear scaling of the other transverse dimension would yield the expected growth of the area proportional to the square of distance r.

Later fixed window size, or simulation grid optical propagation algorithms will be shown which prove to be extremely useful and general. One should appreciate that the provided coordinate growth of the Fourier transform far field propagation saves a lot of headache for problems where the light pattern needs to be known for large distances, which guarantees it will spread out dramatically as in figure 1. If the coordinate scales were fixed, the difficulty of far field diffraction would grow with the distance of desired propagation. Additionally this is the fastest algorithm that will be looked at, because of the approximations allowed, and the single fast Fourier transform which has been meticulously optimized on most computing systems.

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

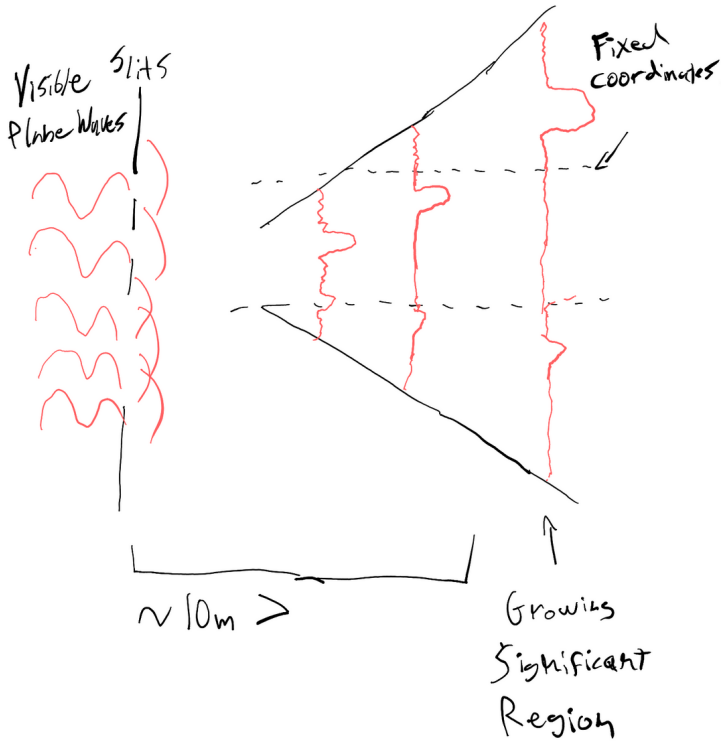

Figure 1. Linear growth of a 1d diffraction pattern. If coordinates were fixed,the results could lose information for large distances.

2. THE CONVOLUTION

The convolution is a mathematical concept that finds utility in huge portions of mathematics, from the work to be done here in optics (which is of partial differential equations), to probability theory, or signal analysis. The convolution finds a simple motivation in a realization of diffraction pattern measurement. The diffraction patterns that are predicted by the Fourier transform far field method have a resolution set by coordinates that change with distance. The reality of measurement is often that such simulation coordinates give a much finer picture of the optical pattern than can be measured.

To experimentally measure a diffraction pattern at some distance, a lens and photodiode (electronic device that responds to incoming photons with electric signal), or simply a photodiode, in either case of finite size will be swept across the diffraction pattern, and the measurements marked down are the displacement of the photodiode, and its measured signal (which is proportional to the intensity). Since the measurement apparatus has finite size all the light intensity that reaches it will be added up. This can be represented as an averaging filter. Figure 2 shows a pictorial representation of the diffraction measurement experiment for a specific case. The steps in space the lens takes as it measures may be smaller than the size of the lens, in this case, it is not sufficent to bucket our predicted diffraction pattern to check it against experimental data. Bucketing would be a good model in the case of a CCD camera, where each pixel has some area, and constant spacing to other pixels where it collects photons. Instead the following integral should be taken on the predicted diffraction pattern, which reflects the averaging. This integral is called the continuous convolution of I_{true} and B_w, or $[I_{true}(y) * B_w(y)](y')$.

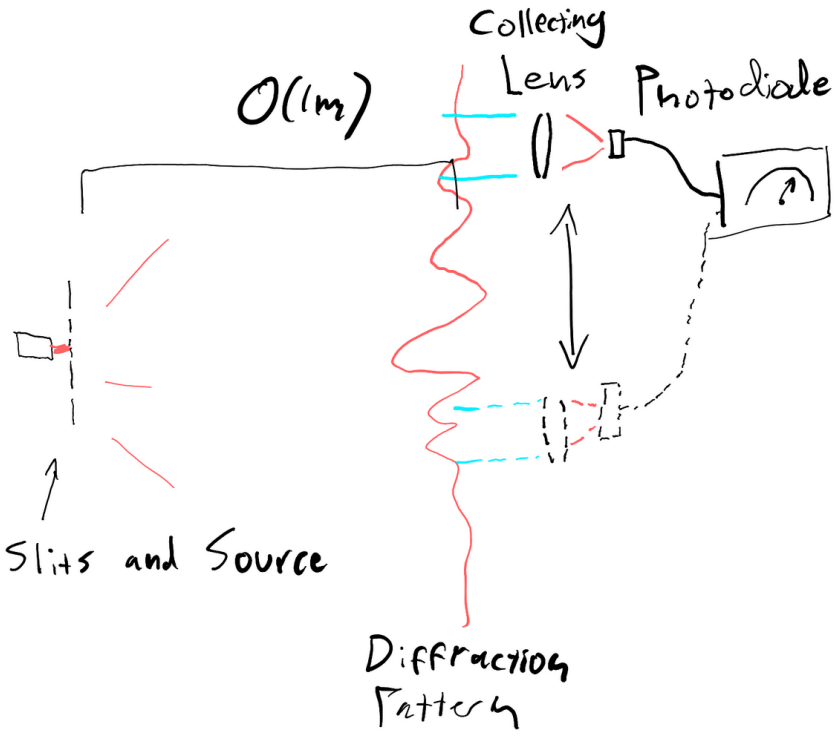

Figure 2. An experimental setup that uses a lens to collect light intensity over an area, making the finite resolution of data collection non negligible.

$$I_{observed}(y') = [I_{true}(y) * B_w(y)](y') = \int_{-\infty}^{\infty} I_{true}(y) B_w(y' - y) dy \tag{1}$$

Where $I_{observed}$ and I_{true} are experimental and theoretical distributions of intensity, and $B_w(x)$ is the boxcar function or tophat function of width w as show in figure 2, which serves as the averaging filter. In numerical computations, the integral is not taken to infinity, rather to the bounds of the simulation grid, which should be larger than the smaller width of $B_w(x)$ or $I_{true}(x)$. One may wonder why the negative sign in the argument $-y$ in $B_w(y' - y)$, in this case it is unimportant, the specific function being convolved $B_w(x)$ is even so either is equivalent. This negative sign makes other formulas on the convolution much more elegant, but necessitates some caution. An example convolution for the boxcar and a diffraction pattern is shown in figure 2.

In code the boxcar is easily defined:

```
#define a boxcar function
def boxcar(w,x):
  return np.logical_and(x <  w/2 , x > -w/2)
```

For each observation point y', the observed signal by the photodiode is a sum of all the intensity in the surrounding area of extent w. An example diffraction pattern could be from four slits, in figure 2 such a diffraction pattern is shown, and a $4mm$ boxcar convolution is shown, to simulate how the pattern would come out in a real experiment, if the resolution was $4mm$. The smaller scale details have been reduced by the finite size of the measurement device, and in the limit of large photodiode/observation aperture, the pattern would be unresolvable, fully averaged out by a $B_w(x)$ wider than the pattern.

Numpy's discretized version of this integral was employed, called *numpy.convolve*. The user provides two arrays which hold the sampled values of the functions being convolved $B_w(x)$ and $I_{true}(x)$. Instead of an integral and continuous functions, numpy works with a sum and discrete arrays. Importantly, the intensity is what is convolved here, since the photodiode adds up photons collected in terms of number (proportional to intensity). The convolution function is used on the intensity in code as follows:

```
#generate the 4mm boxcar convolved diffraction pattern, to show predicted pattern if
    measured with a 4mm photodiode
experimental_intensity = normalize_max(np.convolve(boxcar(4 * millimeter, coordinates)
    , np.abs(np.square(amplitude)), mode = 'same'))

```

Where the amplitude is the diffraction pattern, simulated to have propagated from some slits, to the plane of measurement.

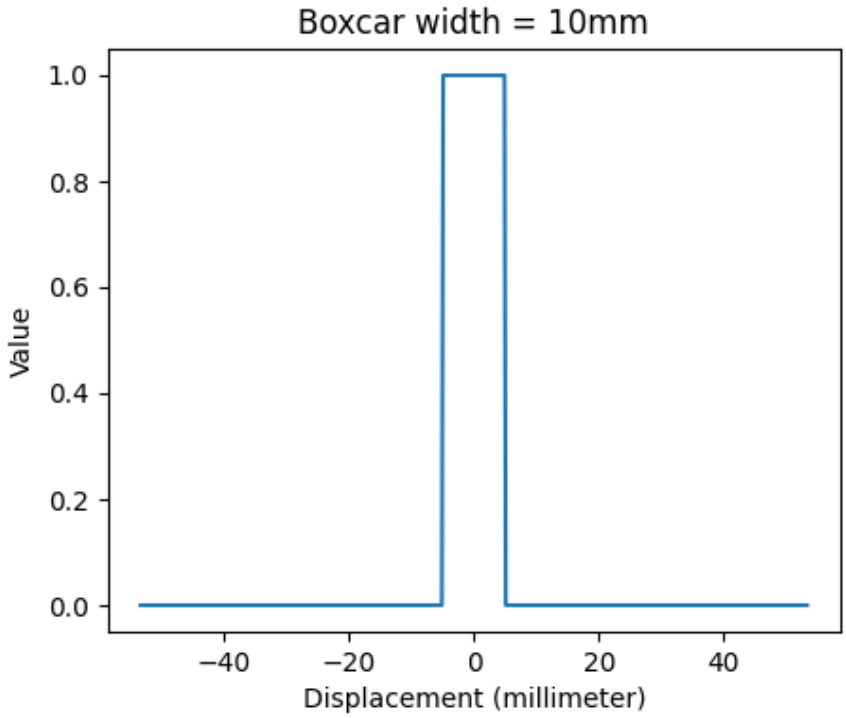

Figure 3. A Boxcar function, the name boxcar fits better when the function is plotted in a smaller window.

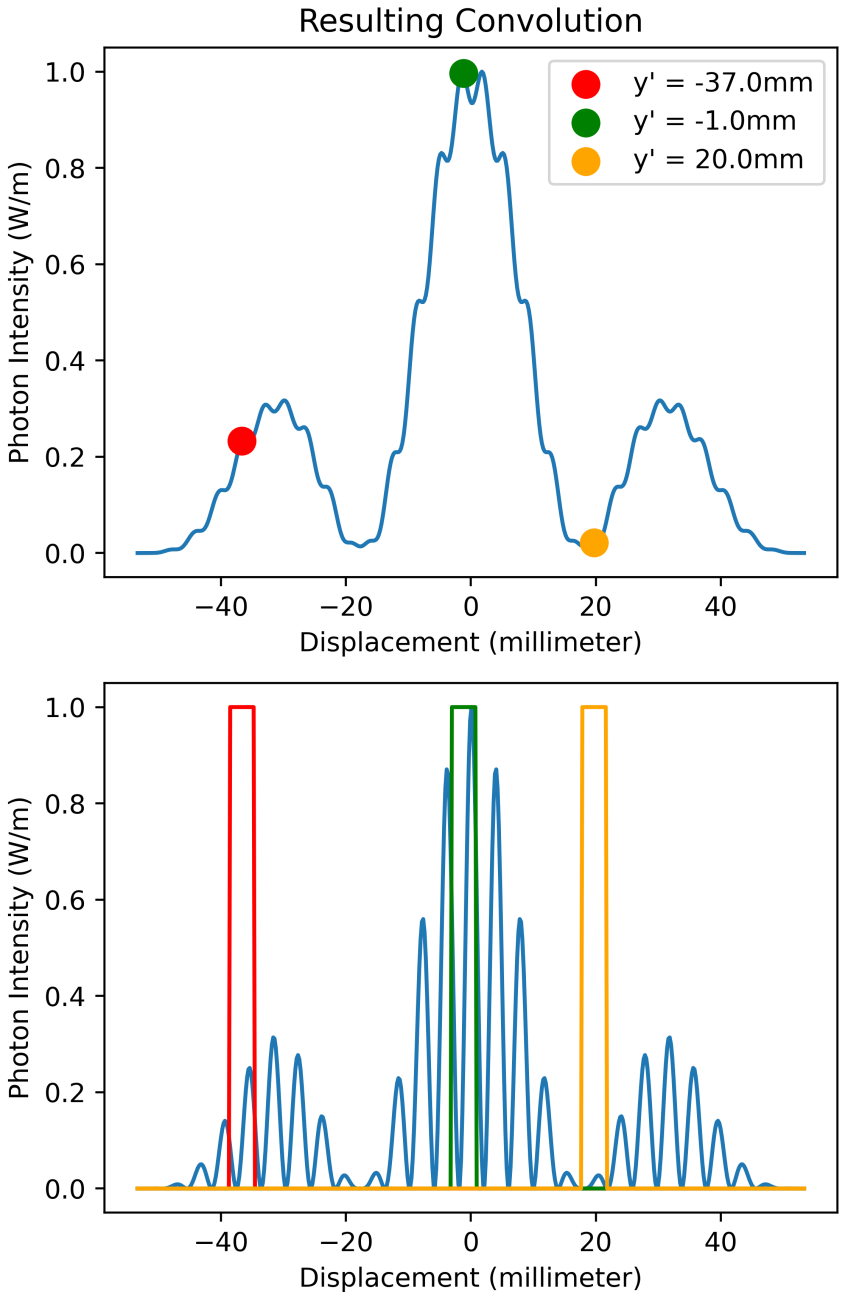

Figure 4.
Bottom: Highly sampled theoretical prediction from diffraction from four slits (two pairs of double slits) at $1m$ away. Top: The same prediction for four slits at $1m$, but with convolution of a $4mm$ boxcar function applied, to represent the loss of detail caused by a finite sized observation tool like a photodiode. One can see that the central triplet of peaks is nearly reduced to appearing as one peak.

Side by side showing how different overlap shifts of the two convolved functions produce values. Each colored point is calculated by integrating the product of the blue curve and corresponding color coded curve on the bottom plot. The example shown is exactly the local averaging convolution, which models a finite sized observation aperture in a real experiment. The orange curve shows how a peak or trough will become muted by the local averaging convolution, if the feature is not wide enough

2.1 Formalizing the Discrete Convolution

Convolution has been given some physical motivation, but like moving from the continuous Fourier transform to the DFT, defining the convolution in a way useful to optics computations requires the additional assumption of periodicity.

For the case of circularly convolving two functions sampled discretely (symbol $\circledast$), like the box car function and a diffraction pattern (B_j and A_j they will be denoted), the mathematical definition is as follows:

$$(B_j \circledast A_j)_n = \sum_{m=1}^{N} B_m \cdot A_{n-m(mod\ N)} \tag{2}$$

This seems very contrived, why add in this modulo, if as will be seen, it causes a headache on the boundaries of our sampled functions when they are convolved circularly? The answer is that the circular convolution obeys the following property:

$$(B_j \circledast A_j)_n = DFT_n^{-1}\{DFT\{B_j\} \cdot DFT\{A_j\}\} \tag{3}$$

This propertly allows the circular convolution to be computed extremely quickly, instead of a sliding sum between two arrays needed for brute force convolution(which is quadratically difficult in terms of addition operations) the problem is reduced to discrete Fourier transforms and place-wise multiplication. Since the discrete Fourier transform can be computed in sub quadratic time, $O(n\ log\ n)$, this algorithm becomes much faster for larger datasets. All that is required for this speed up is to take care to leave enough padded zero values at the boundary of our sampled functions, so that edge effects do not occur. An example of an edge effect is shown further.

The assumed periodicity means that instead of exactly the discrete approximation of the convolution integral, the convolution used in making propagation calculations will be different by the modulus N which causes the left and right boundaries of the diffraction pattern to bleed together when convolved with another function if one is not careful. In the case of circular convolution applied to the previous experimental application, if $B_w(x)$ has w too large the pattern could be caused to repeat itself for large displacements, which is non physical. Figure 2.1 shows the correct convolution of a cropped four slit diffraction pattern. Figure 2.1 shows the same pattern, but the convolution was calculated via circular convolution (3), which fails to be equivalent to the normal convolution, because points at the beginning or end of the pattern are mixed due to the periodic continuation. In the case of the uncropped figure 2, the circular convolution would produce equally correct results, because the pattern is calculated out to where its value is nearly zero (past $40mm$) and so the circular convolution suffers no edge effects.

The circular convolution can be quickly calculated in python code as follows:

```
#fft both boxcar and the pattern
fft_boxcar = np.fft.fft(boxcar(4 * millimeter, coordinates))
fft_pattern = np.fft.fft(np.abs(np.square(amplitude)))

#generate the 4mm boxcar convolved diffraction pattern, to show predicted pattern if
    measured with a 4mm photodiode
#take the CIRCULAR convolution by multiplying the FFT's, then come back to real space,
     rather than frequency space
#to see the pattern.
experimental_intensity = np.fft.fftshift(np.fft.ifft(fft_boxcar * fft_pattern))

```

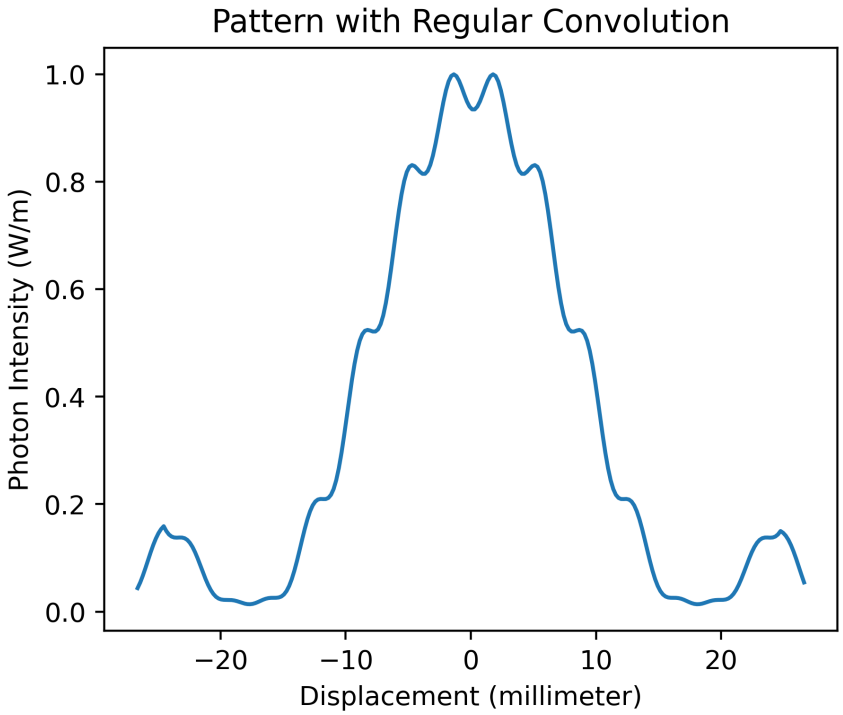

Figure 5. Convolution to model a finite observation scale, used in its most basic definition on a cropped down version of the pattern in figure 2.

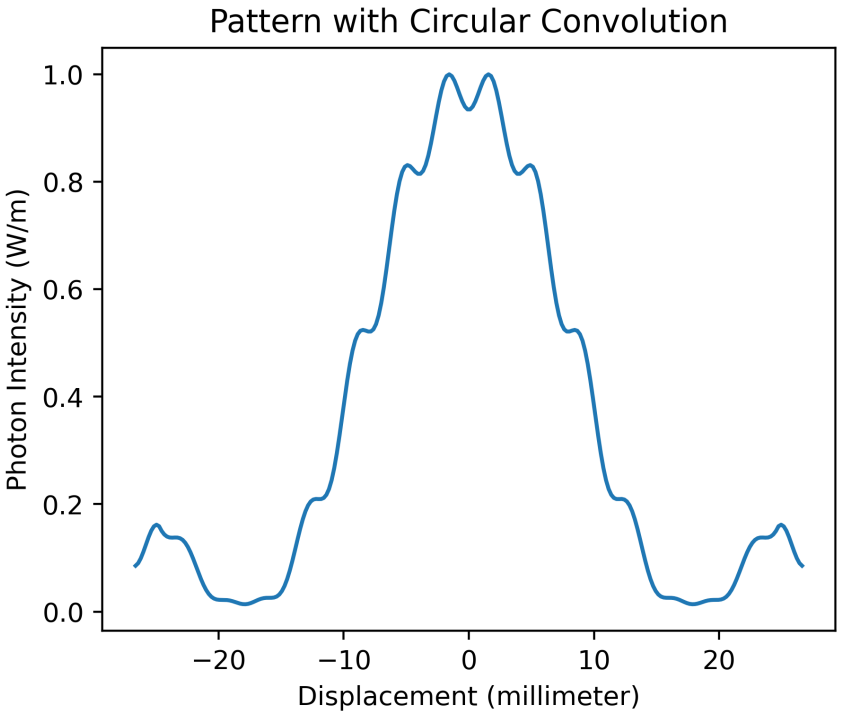

Figure 6. Circular convolution to model a finite observation scale, used in error, because the boundary of the diffracted intensity does not go to zero for at least the width of the boxcar function. The result is clearly not as expected from figure 2, and is in fact nonphysical.

Currently the only motivation for the circular convolution and convolutions in general is to model similar measurements to figure 2. Further, due to the shift invariance of a diffraction system, meaning the propagation of a pattern does not change when the system is shifted in the transverse coordinates, allows the circular convolution to be used to calculate linear optical propagation with amazing speed. For now, these properties should be interpreted and understood in preparation to understand further propagation methods.

3. SHIFTS AND DFT^{-1}

Two more of some of the most useful properties of the DFT will be shown here, one allows shifting in real or frequency space of a sampled function to be represented in the conjugate space. The other is simply a statement that the inverse DFT is equally computable as the DFT.

3.1 DFT Shift

The DFT of a discrete sequence of N samples obeys nice properties under circular shifting in a similar vane to the convenient properties under circular convolution. These properties are as follows:

$$DFT_m\{A_{j-k\ (mod\ N)}\} = DFT_m\{A_j\} \cdot e^{-\frac{i2\pi}{N}mk} \tag{4}$$

$$DFT_m\{A_j \cdot e^{\frac{i2\pi}{N}mk}\} = DFT_{m-k\ (mod\ N)}\{A_j\} \tag{5}$$

3.2 DFT^{-1}

Simply stated, the DFT has an inverse that may be calculated equally quickly to the DFT, through a nearly identical fast Fourier transform algorithm. Intuitively, this comes from the following definitions in the continous space:

$$FT\{f(x)\}(\omega) \propto \int_{-\infty}^{\infty} f(x)e^{-i\omega x}dx \tag{6}$$

$$FT^{-1}\{g(\omega)\}(x) \propto \int_{-\infty}^{\infty} g(\omega)e^{i\omega x}d\omega \tag{7}$$

Besides choice of variables, these expressions are identical besides proportionality constants, and the sign of the exponent, which is a reflection about 0. The DFT and inverse DFT that follow from these definitions are similarly related. Numpy provides both as functions on arrays of discrete samples *numpy.fft.fft* and *numpy.fft.ifft*.

An immediate utility of the inverse DFT is that given our algorithm for calculating the diffraction of some diffraction slits, it is simple to find the reverse algorithm, or given some diffraction pattern a known distances from some unknown slits, the pattern of the slits may be determined. This is why the diffraction pattern in this lecture's code was imported, rather than calculated inline, to show the diffracting slits that created it can be found by reversing the far field propagation algorithm. From lecture two, the algorithm for numerically determining the far field diffraction is stated as:

$$A'_m = DFT\{A_j\},\ Y'_j = Y_j \cdot \lambda L \frac{N}{(2l)^2} \tag{8}$$

Employing the inverse DFT, and some algebra, this algorithm can easily be turned around to propagate the given diffraction pattern backwards, obtaining the original amplitudes and coordinates at the slit exits.

$$DFT^{-1}\{A'_m\} = A_j,\ \frac{(2l)^2}{\lambda N L}Y'_j = Y_j \tag{9}$$

The functional implementation in python mirrors the regular far field propagation with the DFT.

```
def propagate_reverse(Aprime, coordinates_prime, photon_lambda, L):
  #scale the input coordinates correctly, len(coordinates) is the number of samples N
  reverse_propagated_coordinates = coordinates_prime / photon_lambda / L * (np.max(
    coordinates_prime)-np.min(coordinates_prime))**2 / len(coordinates_prime)
  #apply the IFFT (fast IDFT) to the field to find the diffraction pattern
  reverse_propagated_amplitude = np.fft.ifftshift(np.fft.ifft(np.fft.fftshift(Aprime))
    )

  #return the calculted propagation
  return reverse_propagated_amplitude, reverse_propagated_coordinates
```

This function is used to recover the slit pattern that generated the example diffraction pattern. The result is shown in figure 3.2.

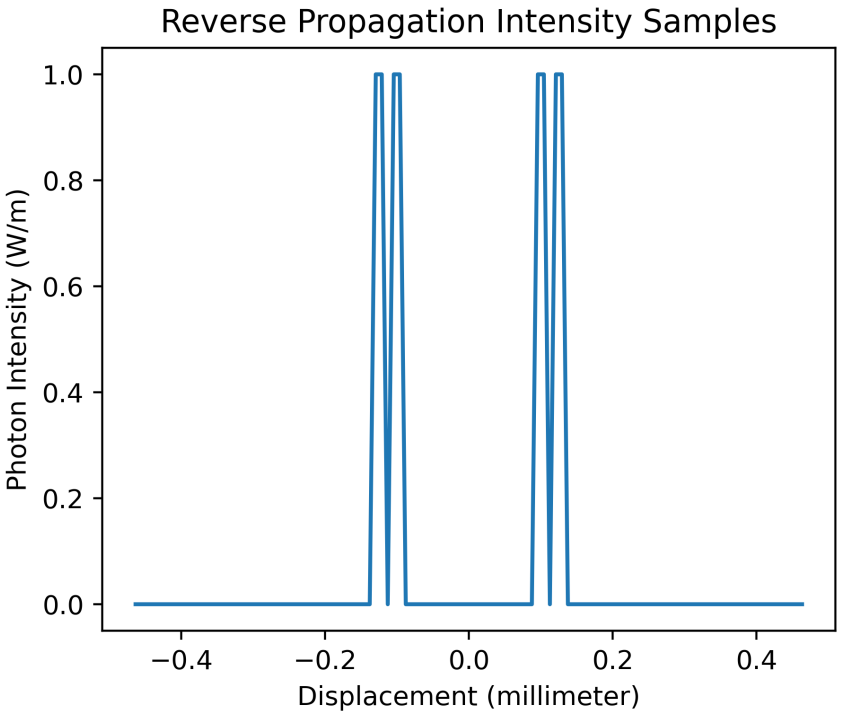

Figure 7. Diffraction pattern from figure 2 propagated back to the slits.

4. CITATIONS/RESOURCES

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

1. Richard Baraniuk et al., properties of the DTFS https://eng.libretexts.org/Bookshelves/Electrical_Engineering/Signal_Processing_and_Modeling/Signals_and_Systems_(Baraniuk_et_al.)/07%3A_Discrete_Time_Fourier_Series_(DTFS)/7.04%3A_Properties_of_the_DTFS *See section Signal Multiplication for the circular convolution*