Simulating Diffraction Patterns of Irregular Apertures using Fast Fourier Transform

Storm Lin, Glenn LeBlanc

5CL Capstone

December 3, 2018

Overview

The Fraunhofer Diffraction Equation

Consider an aperture of some shape given by region D in the xy plane. The Huygens-Fresnel principle tells us that given a uniform plane wave incident on the aperture, we can consider each point $\mathbf{r} \in D$ to contribute to the outgoing field as a spherical wavelet.

Hence to find the field $E_f(\mathbf{r}')$ we have

$$E_f(\mathbf{r}') \propto \int_D E_0(\mathbf{r}) e^{-i\mathbf{k}\cdot(\mathbf{r}-\mathbf{r}')} dA_r$$

In Cartesian Coordinates

With the shape of the aperture given by region D in the xy plane and with incident light heading in the $+z$ direction, we have (using $\lambda \gg \frac{d^2}{z'}$):

$$\begin{aligned} E_f(x', y', z') &\propto \int\int_D E_0(x, y, 0) e^{-ik(x\frac{x'}{z'} + y\frac{y'}{z'})} dydx \\ &\propto \int\int_D E_0(x, y, 0) e^{-\frac{2\pi i}{\lambda z'}(x'x + y'y)} dydx \\ &\propto \mathcal{F}[E_0]_{x'/\lambda z', y'/\lambda z'} \end{aligned}$$

Discrete Fourier Transform

The DFT maps a sequence of n complex numbers $\{x_k\}$ to another sequence of n complex numbers, $\{X_k\}$ by the following relation:

$$X_k = \sum_{m=0}^{n-1} x_m e^{\frac{-2\pi i}{n} km}$$

One can see the similarity between this and the continuous Fourier transform. This essentially allows us to approximate the value of a function's continuous Fourier transform evaluated at a series of equally spaced points.

If we take each sequence to be a vector, this is equivalent to matrix multiplication (next slide).

Fast Fourier Transform

From previous slide:

$$X_k = \sum_{m=0}^{n-1} x_m e^{\frac{-2\pi i}{n} km}$$

The FFT provides an efficient method for computing the discrete fourier transform. The FFT multiplies an n-dimensional vector by the matrix $M_n(\omega)$, where $\omega = e^{-2\pi i/n}$ is a complex nth root of unity and $M_n(\omega)_{mk} = \omega^{mk}$:

$$X_k = \sum_{m=0}^{n-1} \left[M_n(\omega) \right]_{mk} x_k \implies \mathbf{X} = M_n(\omega)\mathbf{x}$$

This matrix can be easily split into sub-matrices, reducing the problem into smaller problems (divide-and-conquer). This is the source of FFT's speedup.

In 2D

Previous discussion holds; sum over two indices instead of one.
Hence FFT becomes two successive matrix multiplications.

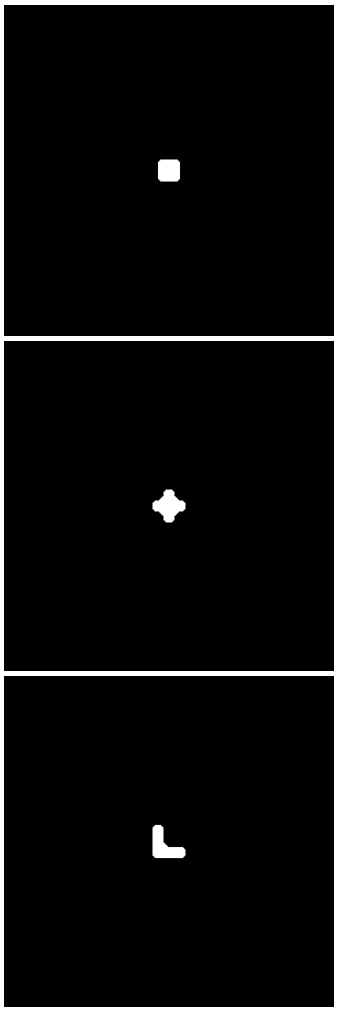
In Python

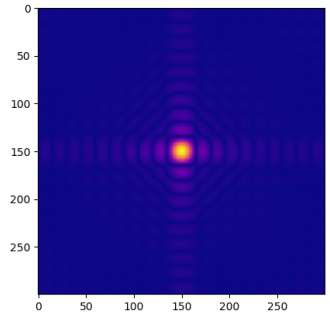
Rather than writing our own implementation of FFT, we used numpy for simplicity.

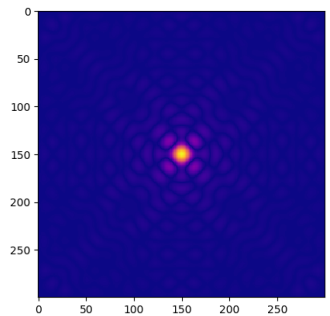
```
1   import numpy as np
2   import matplotlib.pyplot as plt
3   import matplotlib.cm as cm
4   import cv2
5   from mpl_toolkits.mplot3d import Axes3D
6   # Import square image with appropriate filepath for aperture. white-> opening; black-> blockage
7   img_path='./aperture9.png'
8   img=cv2.imread(img_path,0).astype(float)*.001
9   res=len(img)
10
11  # Do fft on image with numpy
12  img_fft=np.abs(np.fft.fftshift(np.fft.fft2(img)))
13  plt.imshow(img_fft, cmap=cm.plasma)
14  plt.show()
```

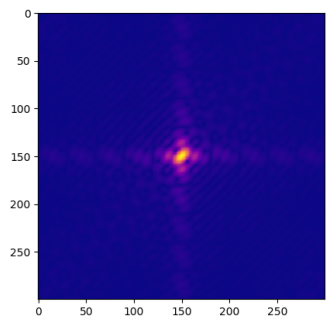
Although intensity is given by the square of the modulus, we only take the modulus for better visibility.

Examples




0
50
100
150
200
250
0
50
100
150
200
250

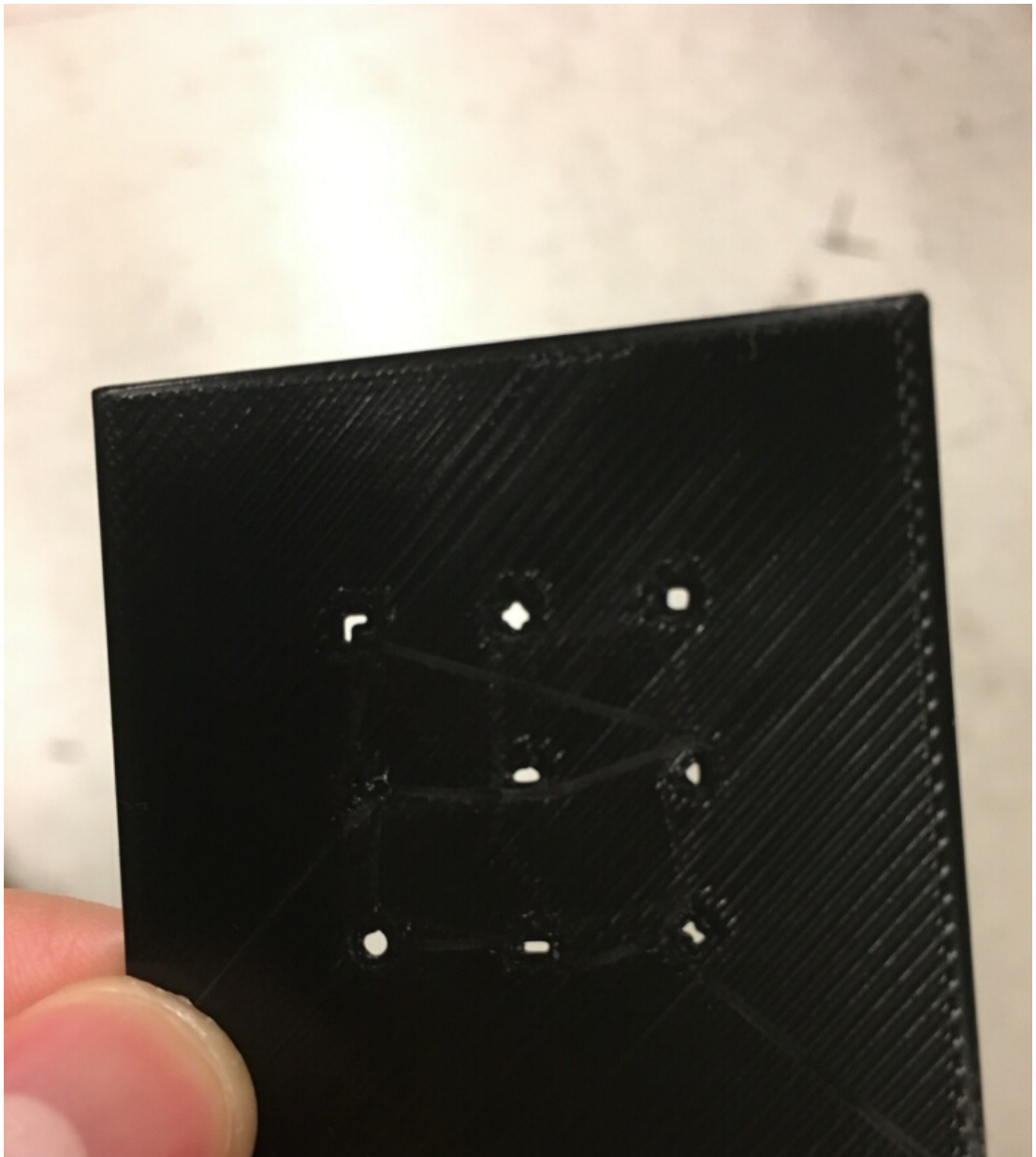

0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250

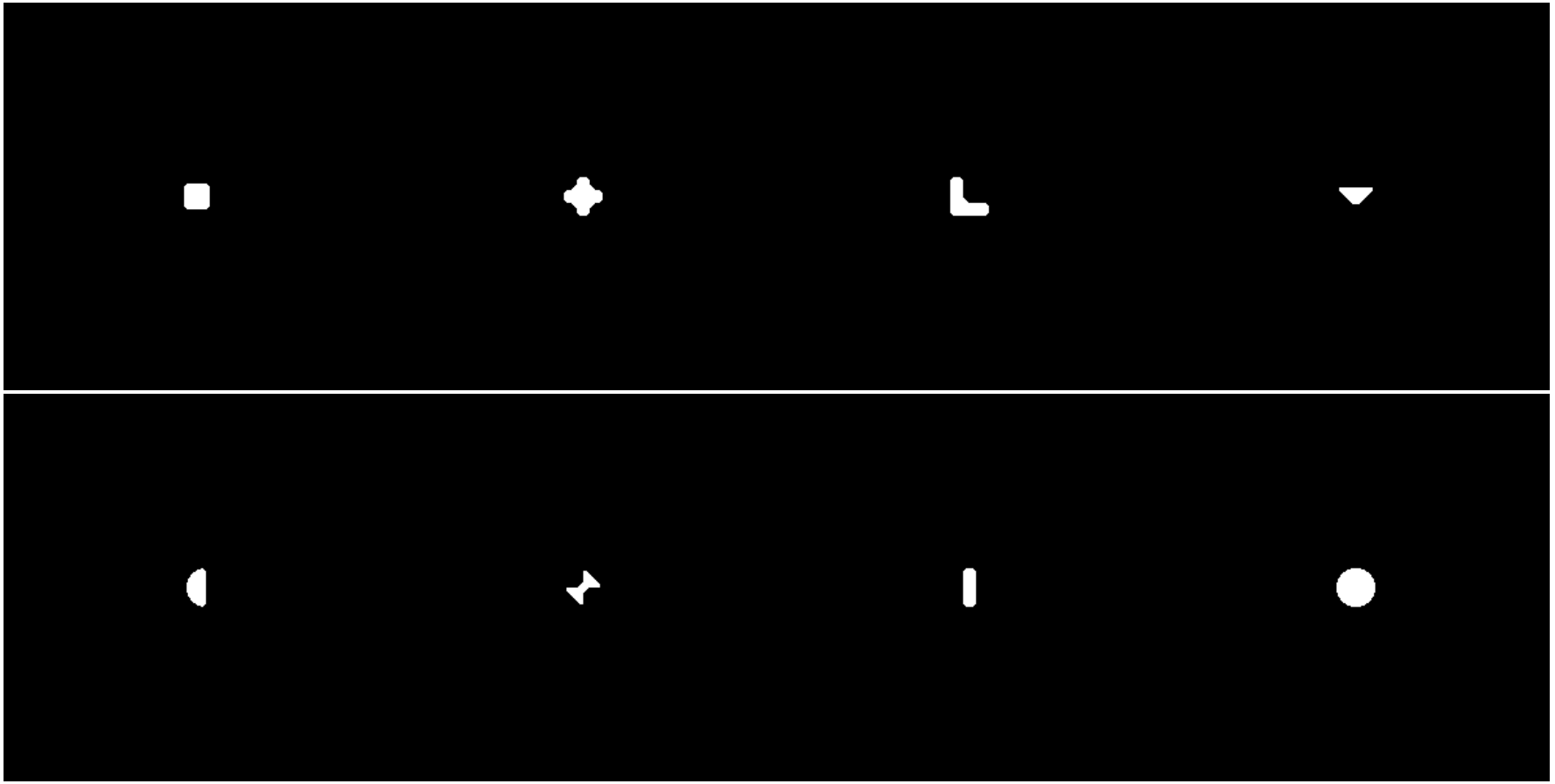
Materials

1. Bench Stand
2. HeNe laser
3. Converging lens with focus 15 cm
4. Converging lens with focus 5.5 cm
5. 3D-printed apertures
6. Meter stick

3D Printed Apertures



Approximate shapes, Accounting for Curvature



Setup

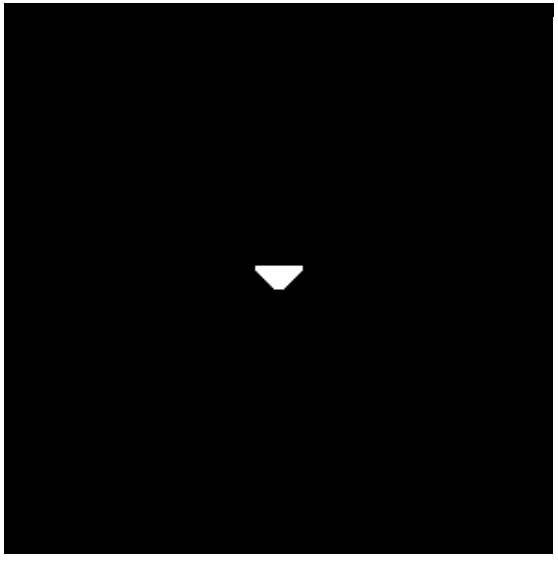
HeNe Laser source $\rightarrow$ beam expander $\rightarrow$ aperture $\rightarrow$ wall

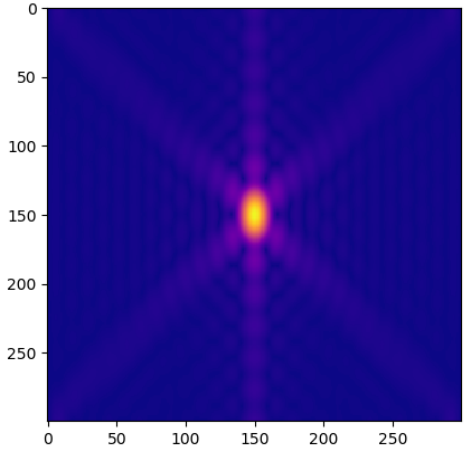
What we recorded

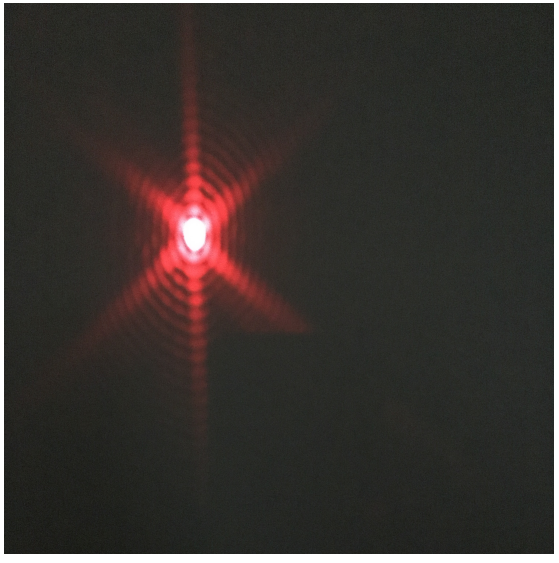
We measured the distance between intensity maxima for each diffraction pattern. Due to large qualitative differences in each pattern, what we chose to measure varied for each aperture.

Every aperture produces a diffraction pattern that qualitatively agrees with simulation (examples to follow).

Sample data set (Right triangle; Nice)



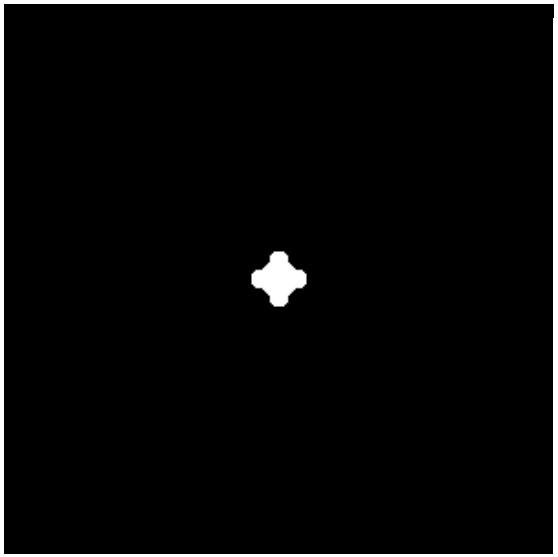


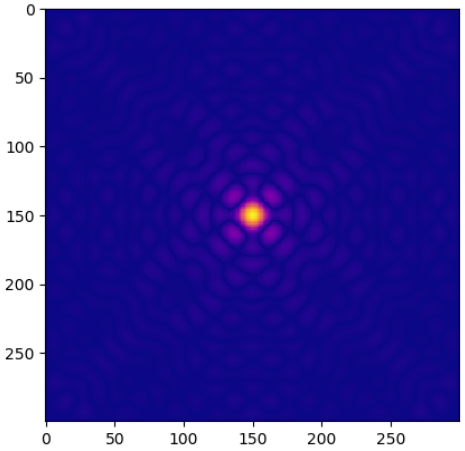


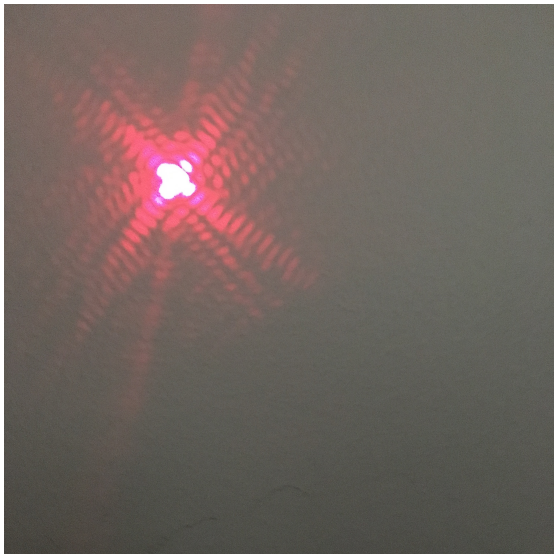
Distance from central maximum to nth intensity minimum at $L = 520 \pm 5$ *cm*

*n*th minimum	Along vertical (cm)	Along diagonal (cm)
1	$.8300 \pm .0005$	$.5500 \pm .0005$
2	$1.3350 \pm .0005$	$.9700 \pm .0005$
3	$1.930 \pm .0005$	$1.3950 \pm .0005$
4	$2.4550 \pm .0005$	$1.8700 \pm .0005$
5	$3.0350 \pm .0005$	$2.2550 \pm .0005$

Sample data set (Cross; less nice)







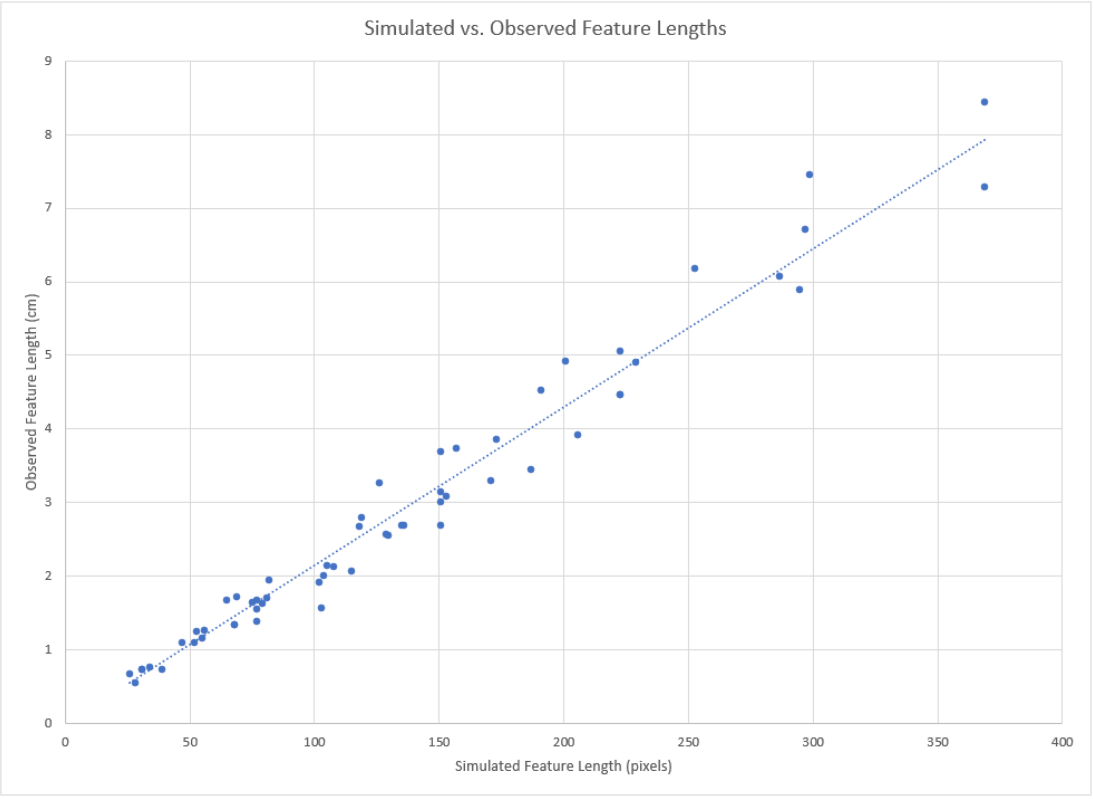
Distance from central maximum to nth intensity minimum at L=520 $\pm$ 5 cm

nth minimum	Along Diagonal (cm)
1	.3750 $\pm$ .0005
2	.6650 $\pm$ .0005
3	.9500 $\pm$ .0005
4	1.2750 $\pm$ .0005

Comparing Relative Distances Between Script and Experiment

Our script does not give us a way to directly determine real-world length scales from the output. Hence, our analysis involves comparing *relative* lengths between simulation and experimental results. That is, we measure the length of certain features of each pattern (in particular, the distances between intensity minima of the same degree) and try to show that there is a direct proportionality between the sizes of features in the simulated patterns and their sizes in real life.

Results of Linear Regression



We found that there is a very strong ($r^2 = 0.991$ and $\chi^2 = 1.82$) direct proportionality relationship between the length of diffraction pattern features predicted by the program and the measured lengths of those pattern features.

Conclusions

We conclude that our script produces qualitatively correct results for each of the eight apertures we tested. Performing a linear regression with a direct proportionality hypothesis provided quantitative support for this conclusion. We found a that there is a strong linear correlation between the length in pixels of a simulated pattern feature and the length in cm of that feature measured in real life. This indicates that our script's output accurately reflects the relative sizes of the real diffraction pattern's features.

Questions?