Elizabeth Champagne

PHYS 332

Computational Project: Simulating Multi-Slit Diffraction

Introduction

For this project, I wanted to focus on optics, so I chose to simulate diffraction slits. My first step was to get my code to work for the single- and double-slit experiments, since there is ample information about these online. Then I extended this to multiple slit gratings, to simulate a transmission diffraction grating like the ones used in spectroscopy. Finally, I plotted all wavelengths in Mercury’s emission spectrum to simulate the use of a diffraction grating in a spectroscopic application.

How the Code Works

I wrote my program in C++. To avoid having to recompile my code every time I wanted to change something, I programmed it to prompt the user several parameters, including the wavelength, the number, width (d), and spacing (sigma) of the slits, the distance to the screen (y), and the “width” of the screen that is measured (which determines the “zoom” and resolution of the data). When I present results and plots, I will specify the parameters used so you can check my results. Note that my code does not plot the data itself; it merely saves the output to a text file. All of the plots were then made with GNU Plot, sometimes with another graph plotted on top for comparison.

Many explanations of diffraction describe wave fronts as the superposition of tiny spherical waves across the whole surface. To model this, I found the electric field contributions from n = 100 points at increments across each slit. To produce an intensity of I0 (normalized to 1), the electric field should have an amplitude of , but I divided this by n so that the contributions from each segment would add up to the total desired amplitude. (I ignored the prefactors, since they would just cancel out when I converted back to intensity at the end.)

To approximate the phase interaction, I found the sum of the cosines of the wave number times the distance to each segment, and likewise the sum of the sine of k times distance. I multiplied the cosine and the sine terms by the amplitude of the segmented electric field. Note- I tried doing this with just the cosine term, since when taking the real part of the electric field only the cosine term remains. However, this does not accurately represent the time averaged amplitude, because any point that is not at full amplitude at t = 0 will be diminished in the final result. Using the sine term preserves the information of the maximum amplitude, whenever it occurs. To convert to frequency, I squared the cosine and sine terms separately and then added them together.

Results and Plots

Single Slit experiment

On the *Hyperphysics* website, I found an equation for the intensity as a function of position across a screen[1], which I plotted along with my data. The equation is:

Where λ = wavelength, a = the width of the slit, y is the position on the screen, and D is the distance to the screen. Unfortunately, this notation does not match mine.

This equation has a lot more spread than my results, but I corrected for this by replacing y with (y\*10), and the graphs line up. I am not sure what caused this discrepancy, but my data is at least qualitatively correct.

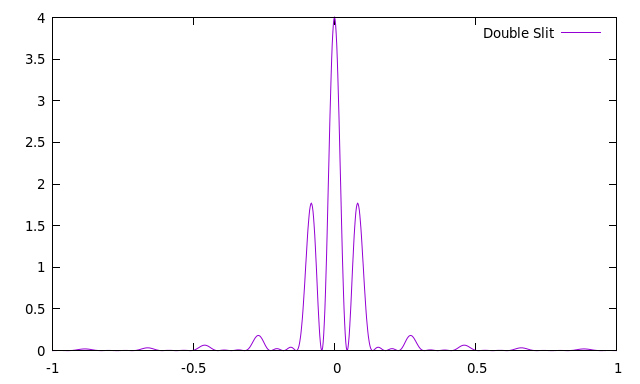
|  |  |
| --- | --- |
| Parameter | Value |
| Filename | User’s choice |
| How many slits? | 1 |
| y (m) | 2 |
| d (m) | 0.000005 |
| sigma (m) | 0.000005 |
| Screen width (m) | 2 |
| Wavelength (nm) | 450 |

A screenshot of a computer

Description automatically generated

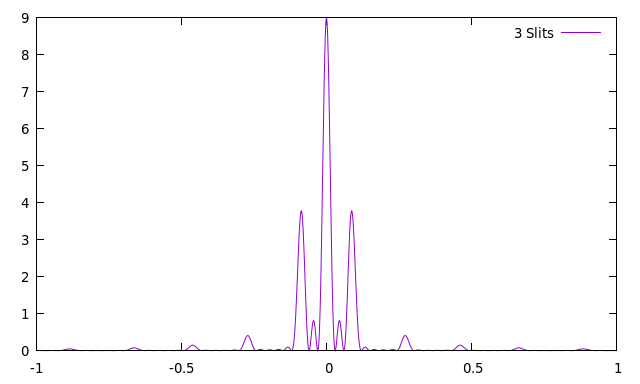
Double Slit Experiment

|  |  |
| --- | --- |
| Parameter | Value |
| Filename | User’s choice |
| How many slits? | 2 |
| y (m) | 2 |
|  |  |
| d (m) | 0.000005 |
| sigma (m) | 0.000005 |
| Screen width (m) | 2 |
| Wavelength (nm) | 450 |



3 Slits

|  |  |
| --- | --- |
| Parameter | Value |
| Filename | User’s choice |
| How many slits? | 3 |
| y (m) | 2 |
| d (m) | 0.000005 |
| sigma (m) | 0.000005 |
| Screen width (m) | 2 |
| Wavelength (nm) | 450 |



10 Slits

|  |  |
| --- | --- |
| Parameter | Value |
| Filename | User’s choice |
| How many slits? | 10 |
| y (m) | 2 |
| d (m) | 0.000005 |
| sigma (m) | 0.000005 |
| Screen width (m) | 2 |
| Wavelength (nm) | 450 |

A screenshot of a cell phone

Description automatically generated

100 Slits

|  |  |
| --- | --- |
| Parameter | Value |
| Filename | User’s choice |
| How many slits? | 100 |
| y (m) | 2 |
| d (m) | 0.000005 |
| sigma (m) | 0.000005 |
| Screen width (m) | 2 |
| Wavelength (nm) | 450 |

A screenshot of a social media post

Description automatically generated

1000 Slits

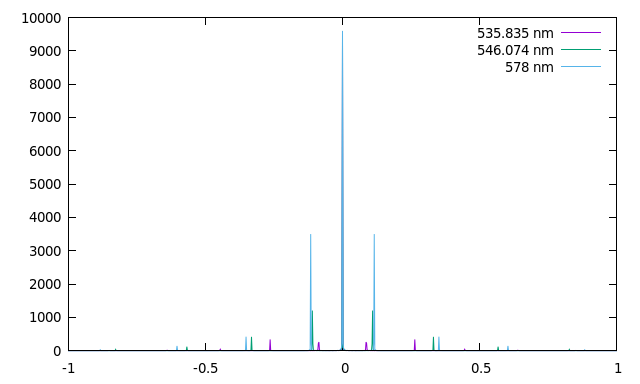
|  |  |
| --- | --- |
| Parameter | Value |
| Filename | User’s choice |
| How many slits? | 1000 |
| y (m) | 2 |
| d (m) | 0.000005 |
| sigma (m) | 0.000005 |
| Screen width (m) | 2 |
| Wavelength (nm) | 450 |

A screenshot of a cell phone

Description automatically generated

Mercury Spectrum

|  |  |
| --- | --- |
| Parameter | Value |
| Filename | User’s choice |
| How many slits? | 100 |
| y (m) | 2 |
| d (m) | 0.000005 |
| sigma (m) | 0.000005 |
| Screen width (m) | 2 |
| Wavelength (nm) | 435.835, 546.074, 578. (average of the double orange line) |



In case it is difficult to tell in black and white, the longer wavelengths are further from the central peak in each given order.

Unresolved Problems

As a check on my work, I wanted to confirm energy confirmation by finding the power output at the source and the power input at the screen. Since , I approximated the this in 1D as along the screen and along the slits, and found the relative error. I have summarized my findings in the following table for various numbers of slits (λ = 450 nm, d = σ = 0.000005 m, y = 2 m). The results are very wrong. The possible reasons for this are:

1. My simulation is wrong (although given that the single slit matches so well with the diffraction equation I doubt this is the cause).
2. My integration method for the power was sloppy and faulty.
3. I failed to correctly include some behaviors such as intensity fall off from the inverse square law.
4. The initial assumption that I could calculate power in this way is wrong (most likely, none of the others would explain errors so large).

It is worth noting, however, that all my errors are on the same order of magnitude, which indicates consistency for different numbers of slits. As the number of slits increases, the power received by the screen increases proportionally to the amount of light introduced by each new slit.

|  |  |
| --- | --- |
| Number of Slits | Relative Power Error |
| 1 | 35520.9 |
| 2 | 35521.3 |
| 3 | 35521.4 |
| 4 | 35521.4 |
| 5 | 35521.5 |
| 10 | 35521.5 |
| 50 | 41940.1 |
| 100 | 73203.2 |
| 500 | 43324.5 |
| 1000 | 37624.3 |

Further Research

There are a number of parameters that I included in my program, but did not experiment with much in my analysis. For instance, I consistently used slit widths and spacings of 0.000005 m instead of looking for the limits where the diffraction equation breaks down. Also, except for the mercury spectrum, I exclusively used 450 nm for the wavelength. Varying more parameters would help me better understand the limits of my program.

Additionally, as mentioned already, the intensity to power conversion needs to be checked. Energy conservation is a powerful way to check the sanity of results. While my power errors indicate the consistency of my data, without knowing whether my simulation or my power calculation is at fault, I am unsure of the accuracy of all of my data.

References

[1] “Single Slit Diffraction Intensity.” *Hyperphysics*, http://hyperphysics.phy-astr.gsu.edu/hbase/phyopt/sinint.html#c3

//grating.cpp

#include <stdlib.h>

#include <iostream>

#include <fstream>

#include <string>

#include <cmath>

using namespace std;

double distance(double x1, double x2, double y) {

return sqrt((x2 - x1)\*(x2 - x1) + y\*y);

}

int main() {

double c = 3.00E8;

double PI = 4 \* atan(1);

string filename;

cout << "Please enter the name of the file: ";

cin >> filename;

ofstream file;

file.open(filename);

double y; // in meters

double d;

double sigma;

int slits;

cout << "How many slits? ";

cin >> slits;

slits = abs(slits);

cout << "Please enter y, d, and sigma in meters: "; //Check for spacings = 0;

cin >> y >> d >> sigma;

sigma = abs(sigma);

double screenWidth;

cout << "Please enter the desired screen width: ";

cin >> screenWidth;

double a = -screenWidth/2.; //Endpoints on screen

double b = screenWidth/2.;

double startAt = -(d \* slits + sigma \* (slits - 1))/2;

//double ax1 = -(d + sigma/2);

//double ax = -d / 2.; //ax and bx are x coors of endpoints of slit

//double bx = d / 2.;

double lambda;

cout << "Please insert the wavelength in nm: ";

cin >> lambda;

lambda \*= 1E-9;

double f = c / lambda;

double omega = 2 \* PI \* f;

double k = 2 \* PI / lambda;

int n = 100; //Number of intervals for slit- loops go n times

int m = 1000; //Intervals along screen, although it's really m + 1 so I get the endpoints

double dx = (b - a) / m;

double screenPower = 0;

double I0 = 1.; //Normallize intensity to 1

double E0 = sqrt(I0) / n;

double sourcePower = I0 \* (slits \* d);

for(int i = 0; i < m + 1; i++) {

double x = a + (b - a) / m \* i;

double Exsin = 0;

double Excos = 0;

for (int j = 0; j < slits; j++) { //Which slit

for(int l = 0; l < n; l++) { //How far along slit \*\*\*K IS THE WAVE NUMBER\*\*\*

double slitX = startAt + (d + sigma) \* j + d / n \* (l + 0.5);

double dist = distance(x, slitX, y);

Excos += cos(k \* dist) \* E0;

Exsin += sin(k \* dist) \* E0;

}

}

double Ix = Exsin \* Exsin + Excos \* Excos;

screenPower += Ix \* dx;

//cout << x << "\t" << Ix << "\n";

file << x << "\t" << Ix << "\n";

}

file.close();

//My "energy conservation" attempts were unsuccessful. I don't think this means my program is wrong, since my graphs match others I've found; I think something is wrong my my power integration.

cout << "The source emits a power of " << sourcePower << " m \* arbitrary intensity units.\n";

cout << "The screen recieves a power of " << screenPower << " m \* arbitrary intensity units.\n";

double error = abs(screenPower - sourcePower) / abs (sourcePower);

cout << "This gives a relative error of " << error << ".\n";

//I initially experimented with having the program give the gnuplot command itself, but this lagged considerably and didn't always work

//I had it output the command so I could copy and paste and save myself a little trouble when graphing

//This also serves to confirm that the program executed correctly

string command = "gnuplot -p -e \"p \'" + filename + "\' u 1:2 w l\"\n";

cout << command;

//system(command.c\_str());

return 0;

}