Waveoptics

FYS2150 Lab Report

Nicholas Karlsen

May 14, 2018

Abstract

1 Introduction

2 Theory

2.1 Diffraction & Interference

When a monochromatic light wave of wavelength λ is passed through a slit of width a, the emerging diffraction pattern's measured intensity, E, in line, x, tangential to the path of the light satisfies the proportionality Eqn. 1, where $u \equiv x/\lambda R$, R denoting the distance between the slit and the line, x.

$$E_1(x) \propto a^2 \left(\frac{\sin(\pi a u)}{\pi a u} \right) \tag{1}$$

The intensity is at a minimum when $sin(\pi a u) = 0$, which occurs when au is an integer and not zero.

For two parallel slits with a distance A between, the intensity is instead given by Eqn. 2, which will have intensity minima when either $au = n$ for nonzero $n \in \mathbb{Z}$ or $Au = 2n$ for $n \in \mathbb{Z}$.

$$E_2(x) \propto 4a^2 \cos^2(\pi A u) \left(\frac{\sin(\pi a u)}{\pi a u} \right)^2 \tag{2}$$

This can be generalized for N parallel slits satisfying Eqn. 3

$$E_N(x) \propto a^2 \left(\frac{\sin(N\pi A u)}{\sin(\pi A u)} \cdot \frac{\sin(\pi a u)}{\pi a u}^2 \right) \tag{3}$$

3 Experimental Procedure

3.1 Diffraction Grating

To investigate the specifications of a selection of slits, and their effect on laser, we used an apparatus as sketched in Fig. 1, where a laser lined up with two lenses and a diffraction slit(s) secured on an optical track. Laser passing through is then reflected by a mirror onto a screen in order to effectively increase the distance R, in which the laser has passed. This reflection does presumably cause the measurements to deviate slightly from the theoretical model used, but for the purposes of this experiment this deviation is taken to be negligible. Additionally, neither the laser source nor the optical track were fastened to the table and both were easily moved. We took care not to move them, but due to a very limited workspace this may have happened.

As we swapped between different types slits, the patterns projected onto the screen was recorded by outlining their features on a piece of paper held up to the screen with a pen. Drawing the lines in the "correct" position was not easy to do in a precise manner, and is likely the source of a significant error in our final results across all of the different measurements. Afterwards, distance between the lines was measured using a ruler.

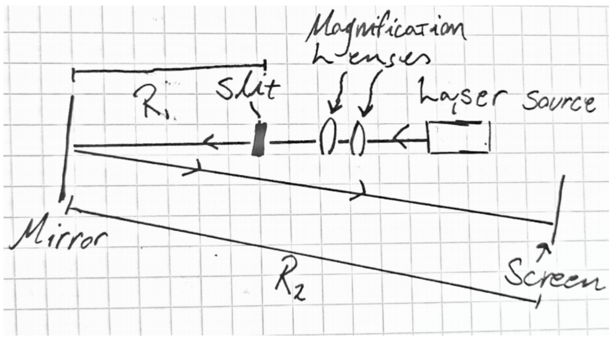


Figure 1: Sketch of aparature used to measure diffraction lines of a laser

3.2 Diffraction spectroscopy

In order to determine the wavelength of some of the spectral lines in Hydrogen and helium, a spectrometer similar to the one depicted in Fig. 2 was used. Both the Collimator and the grating were fixed, and whilst the telescope was only fixed radially (relative to the center of the grating). The telescope was connected to a vernier scale, which read its angle θ relative to the collimator.

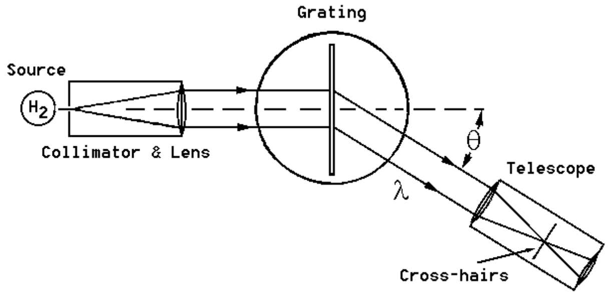


Figure 2: Sketch of spectrometer used to measure angle of diffraction(Source: http://felix.physics.sunysb.edu/~allen/252/PHY251_H_spectrum_fig1.gif)

Light coming from the source is passed through the collimator, hits the grating at a tangent and is diffracted. The visible wavelengths was then be observed through the telescope, and their angle of diffraction recorded by the vernier scale to an accuracy of 10^{-1} deg. The diffracted wavelengths were mirrored on both sides, and by taking the difference in their angle on the vernier scale we get θ satisfying Braggs' law INSERT REFERENCE TO BRAGGS LAW for $n = 1$, used to determine the wavelength of the observed spectral line. In addition to recording the angle, we also made note of the color we "think" we saw, which was later used as a way to check the validity of our calculated wavelengths.

This procedure was performed for both Helium and Hydrogen, for which all clearly visible spectral lines were recorded in succession from the central top (parallel with the collimator) in both the "left" and "right" direction. The angles were recorded in succession from the center in order to ensure that each successive left angle would be in accordance with the corresponding right angle. In addition, we made sure their recorded color matched and that we got the same number of measurements on both sides.

Lastly, in order for the lines to be visible, the room in which the experiment was performed was kept dark by covering the windows. For the Hydrogen source in particular, additional measures had to be taken by covering the apparatus in a plastic bag whilst finding the spectral lines, in an attempt to filter out make them more visible. This was only partially successful, as the lines were still quite difficult to see clearly.

3.3 Zeeman effect

4 Results

4.1 Diffraction Patterns

Table 1: Single slit

Diameter of primary minima [cm]	Calculated Width of Slit [mm]	Stated Width of Slit [mm]
2.35	0.56	0.48
4.70	0.28	0.24
10.60	0.12	0.12

Table 2: Two parallel slits

Observed No. Peaks	Expected No. Peaks	Width of slits [mm]	Separation of slits [mm]
9	9	0.12	0.6
5	5	0.24	0.6
9	11	0.24	1.2

4.2 Spectral Lines

Table 3: Hydrogen Lines

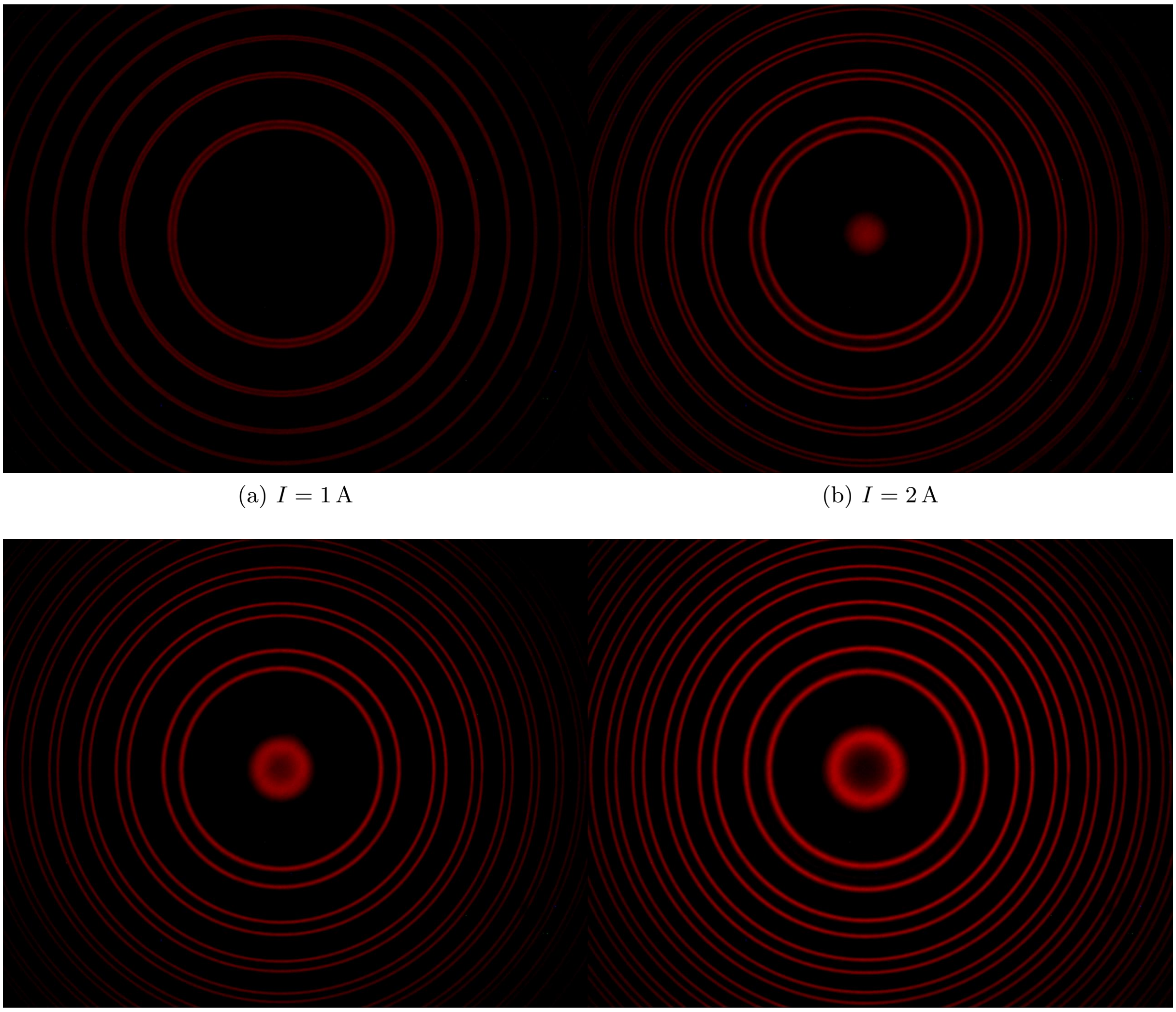
α_v	α_h	θ	λ [nm]
$167.40 \pm 0.01°$	$228.80 \pm 0.01°$	$30.70 \pm 0.01°$	432.28 ± 5.17
$163.10 \pm 0.01°$	$223.30 \pm 0.01°$	$30.10 \pm 0.01°$	424.63 ± 5.20
$146.10 \pm 0.01°$	$248.80 \pm 0.01°$	$51.35 \pm 0.01°$	661.25 ± 3.82

Table 4: Helium Lines

α_v	α_h	θ	λ [nm]
$144.40 \pm 0.01°$	$248.70 \pm 0.01°$	$52.15 \pm 0.01°$	668.57 ± 3.76
$152.80 \pm 0.01°$	$240.90 \pm 0.01°$	$44.05 \pm 0.01°$	588.70 ± 4.36
$160.40 \pm 0.01°$	$233.70 \pm 0.01°$	$36.65 \pm 0.01°$	505.42 ± 4.84
$160.60 \pm 0.01°$	$233.40 \pm 0.01°$	$36.40 \pm 0.01°$	502.45 ± 4.86
$161.50 \pm 0.01°$	$232.70 \pm 0.01°$	$35.60 \pm 0.01°$	492.88 ± 4.90
$163.20 \pm 0.01°$	$231.00 \pm 0.01°$	$33.90 \pm 0.01°$	472.24 ± 5.00
$165.20 \pm 0.01°$	$229.10 \pm 0.01°$	$31.95 \pm 0.01°$	448.06 ± 5.11

4.3 Zeeman Effect

The pictures taken of the split diffraction lines are shown in Fig.3 and the associated results computed using zeemanread.py (see appendix A) are shown in Table 5. The image shown in Fig. 3a could not be successfully processed by my script, and is therefore not included in Table 5.



(a) $I = 1\,\mathrm{A}$ (b) $I = 2\,\mathrm{A}$

(c) $I = 3\,\mathrm{A}$ (d) $I = 4\,\mathrm{A}$

Figure 3: Split difraction lines due to σ-transitions for different magnitudes of magnetic field

Table 5: Zeeman results

I [A]	$\bar{B}(I)$ [mT]	$\bar{d}_1(I)$ [px]	$\bar{d}_2(I)$ [px]	$\bar{d}_3(I)$ [px]	$\tilde{\mu_B}$ [J T^{-1}]
4	685.5	423.5	524.5	661.0	$8.977 \cdot 10^{-27}$
3	526.5	437.0	515.0	668.5	$9.123 \cdot 10^{-27}$
2	354.5	450.0	503.0	677.5	$9.195 \cdot 10^{-27}$

5 Discussion

6 Conclusion

References

*

A Scripts

scripts/zeemanread.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
'''
Finds diameter of diffraction rings from Zeeman experiment
in the FYS2150 Waveoptics lab
'''

import numpy as np
import matplotlib.pyplot as plt
from matplotlib.image import imread
from scipy import ndimage
# from PIL import Image
# import skimage.morphology as morph
# from skimage import filters


def rgb2gray(rgb):
    '''
    Converts shape=(N,M,rgb) array to (N, M) grayscale array
    '''
    return np.dot(rgb[..., :3], [0.299, 0.587, 0.114]).astype(int)


def gray2binary(gray, limBW=128):
    """Converts grayscale image to binary grayscale of 0 OR 255
    image must be array of shape=(N, M)
    gray: (N, M) array
    limBW: threshhold limit between B/W
    """
    bw = np.asarray(gray).copy()
```

```python
    bw[bw < limBW] = 0        # Black
    bw[bw >= limBW] = 255     # White
    return bw


def readZeeman(filename, lowerThresh, higherThresh, g2bThresh=20):
    #import skimage.color
    if isinstance(filename, basestring) is False:
        raise TypeError("Filename arguement not string")
    img = imread(filename)
    bwImg = rgb2gray(img)
    binImg = gray2binary(bwImg, 17)
    binCrop = binImg[475:525, 0:-1]

    plt.imshow(bwImg, cmap=plt.get_cmap('gray'))
    plt.close()
    plt.axhline(0, linestyle="-", color="r")
    plt.imshow(binCrop, cmap=plt.get_cmap('gray'))
    plt.close()

    edge_horizont = ndimage.sobel(binCrop, 0)
    edge_vertical = ndimage.sobel(binCrop, 1)
    magnitude = np.hypot(edge_horizont, edge_vertical)
    outlines = gray2binary(magnitude, g2bThresh)
    plt.imshow(outlines, cmap=plt.get_cmap("gray"))

    outline_indeces = []

    for i in range(len(outlines)):
        outline_indeces_row = [0]  # Setting first element to zero to make
    loop work
        for j in range(len(outlines[i])):
            if outlines[i, j] == 0:
                pass
            else:
                if abs(j - outline_indeces_row[-1]) < 4:
                    pass
                else:
                    outline_indeces_row.append(j)
        outline_indeces_row.pop(0)  # remove the zero
        outline_indeces.append(outline_indeces_row)
    d_outlines = outline_indeces[30]

    plt.plot(d_outlines, np.zeros_like(d_outlines) + 30, "ro")
    plt.title("Chose lower and higher threshhold")
    plt.close()

    d_outlines = filter(lambda f: f < higherThresh and f > lowerThresh,
    d_outlines)

    if len(d_outlines)%2 != 0:
        raise ValueError("outlines not even number, check threshhold")

```

```
    d_center = []
    counter = 0
    while counter < len(d_outlines):
            d_center.append((d_outlines[counter] + d_outlines[counter + 1])
    / 2.0)
            counter += 2
    plt.imshow(binCrop, cmap=plt.get_cmap("gray"))
    plt.plot(d_center, np.zeros_like(d_center) + 30, "ro")
    plt.yticks([])
    plt.xticks(d_center, rotation=-25)
    plt.close()

    d_3 = d_center[-1] - d_center[0]
    d_2 = d_center[-2] - d_center[1]
    d_1 = d_center[-3] - d_center[2]

    return d_1, d_2, d_3

d14, d24, d34 = readZeeman("figs/ZEEMAN4A.jpg", 255, 955)
d13, d23, d33 = readZeeman("figs/ZEEMAN3A.jpg", 255, 955)
d12, d22, d32 = readZeeman("figs/ZEEMAN2A.jpg", 255, 955)
#readZeeman("figs/ZEEMAN1A.jpg", 255, 955, 100)



def readZeemanAlt(filename):
    #import skimage.color
    if isinstance(filename, basestring) is False:
        raise TypeError("Filename arguement not string")
    img = imread(filename)
    bwImg = rgb2gray(img)
    binImg = gray2binary(bwImg, 17)
    binCrop = binImg[475:525, 0:-1]
    bwCrop = bwImg[475:525, 0:-1]
    bwRow = bwCrop[30]
    plt.imshow(bwImg, cmap=plt.get_cmap('gray'))
    plt.close()
    plt.subplot(212)
    plt.axhline(0, linestyle="-", color="r")
    plt.imshow(bwCrop, cmap=plt.get_cmap('gray'))
    plt.xlabel("Pixel")
    plt.yticks([])
    plt.subplot(211)
    plt.plot(bwRow)
    plt.ylabel("Intensity"); plt.xlabel("Pixel")
    plt.tight_layout()
    plt.close()

readZeemanAlt("figs/ZEEMAN1A.jpg")

def mu_B(B, d1, d2, d3):
    hc = 1.98644568E-25      # [CODATA]
    sigma = float(d2**2 - d1**2) / (d3**2 - d1**2)
```

```
    tx4 = 3.0 * 4.0

    return (hc / tx4) * (sigma / B)

mu_B_4 = mu_B(685.5e-3, d14, d24, d34)
mu_B_3 = mu_B(526.5e-3, d13, d23, d33)
mu_B_2 = mu_B(354.5e-3, d12, d22, d32)

def print_diameters(list):
    n = 1
    for item in list:
        print "d_%i = %.1f" % (n, item)
        n += 1
    return

print "\n4A Diameters"
print_diameters([d14, d24, d34])

print "\n3A Diameters"
print_diameters([d13, d23, d33])

print "\n2A Diameters"
print_diameters([d12, d22, d32])

print "\nMu_B:\n"
print "I = 4A -> %.4e" % mu_B_4
print "I = 3A -> %.4e" % mu_B_3
print "I = 2A -> %.4e" % mu_B_2


print "Mean mu_B", np.mean([mu_B_4, mu_B_3, mu_B_2])
```

scripts/balmerlines.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# by nicholas karlsen

import numpy as np
import matplotlib.pyplot as plt


# Helium lines in order
# Red, yellow, green1, green2, green3, blue, purple
He_av = np.array([144.4, 152.8, 160.4, 160.6, 161.5, 163.2, 165.2])
He_ah = np.array([248.7, 240.9, 233.7, 233.4, 232.7, 231.0, 229.1])

# Hydrogen lines in order
# Purple, green, red
H_av = np.array([167.4, 163.1, 146.1])
H_ah = np.array([228.8, 223.3, 248.8])

# Angles
```

```
He_theta = (He_ah - He_av) / 2.0
H_theta = (H_ah - H_av) / 2.0

d = 846.7e-9     # Gitterkonstant

def wlen(theta):
    "returns wavelength associated with angle theta"
    return d * np.sin(np.deg2rad(theta))

# Wavelengths
H_wlen = wlen(H_theta)
He_wlen = wlen(He_theta)


def dTheta(dah=0.01, dav=0.01):
    "error in measured angle"
    return 0.5 * np.sqrt(dah**2 + dav**2)


def dWlen(theta, Wlen, d_err=1E-9):
    "Error in measured wavelength"
    a = d_err / d
    b = dTheta() / np.tan(theta)
    return Wlen * np.sqrt(a**2 + b**2)


def balmerlines(last_n):
    "Returns balmer lines (theoretical spectral lines for Hydrogen)"
    R = 1.097E7     # Rydberg constant
    n = np.linspace(3, last_n, last_n - 3)
    return 1.0 / (R * (0.5**2 - (1.0 / n)**2))


# Compare Observed values with theoretical ones graphically
plt.scatter(H_wlen, np.zeros_like(H_wlen),
            label="Observed Hydrogen lines", color="r")
plt.scatter(balmerlines(20), np.zeros_like(balmerlines(20)) + 0.5,
            label="Predicted Balmer lines", color="black")
plt.xlim(400E-9, 700E-9)
plt.yticks([])
plt.legend()
plt.close()


def hydrogen_table():
    outfile = open("dat/hydrogenlines.dat", "w")  # Change to whatever outfile you want
    outfile.write("$\\alpha_v$ & $\\alpha_h$ & $\\theta$ & $\\lambda$ [nm]")
    outfile.write("\n")
    outfile.write("\\\\ \\hline ")
    for i in range(len(H_ah)):
        outfile.write("$%.2f \\pm 0.01 ^\\circ$" % H_av[i])
```

```
        outfile.write(" & ")
        outfile.write("$%.2f \\pm 0.01^\\circ$" % H_ah[i])
        outfile.write(" & ")
        outfile.write("$%.2f \\pm %.2f^\\circ$" % (H_theta[i], dTheta()))
        outfile.write(" & ")
        outfile.write("$%.2f \\pm %.2f$" % (H_wlen[i] * 1e9, dWlen(np.deg2rad(H_theta[i]), H_wlen[i]) * 1e9))
        outfile.write(" \\\\")
    outfile.close()


def helium_table():
    outfile = open("dat/heliumlines.dat", "w")  # Change to whatever outfile you want
    outfile.write("$\\alpha_v$ & $\\alpha_h$ & $\\theta$ & $\\lambda$ [nm]")
    outfile.write("\n")
    outfile.write("\\\\ \\hline ")
    for i in range(len(He_ah)):
        outfile.write("$%.2f \\pm 0.01 ^\\circ$" % He_av[i])
        outfile.write(" & ")
        outfile.write("$%.2f \\pm 0.01^\\circ$" % He_ah[i])
        outfile.write(" & ")
        outfile.write("$%.2f \\pm %.2f^\\circ$" % (He_theta[i], dTheta()))
        outfile.write(" & ")
        outfile.write("$%.2f \\pm %.2f$" % (He_wlen[i] * 1e9, dWlen(np.deg2rad(He_theta[i]), He_wlen[i]) * 1e9))
        outfile.write(" \\\\")
    outfile.close()


hydrogen_table()
helium_table()

print "\nDONE"
```