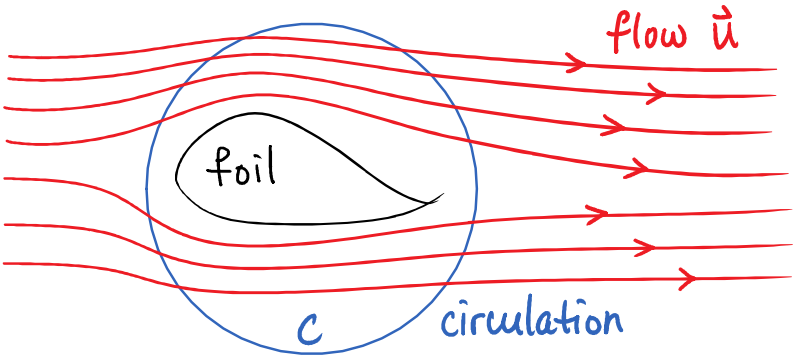
Lab 6 Report

Math 391 - Introduction to Modern Physics Lab
Professor Wayne Lau
University of Michigan



The Kutta–Joukowski Lift Theorem

Jinyan Miao

Fall 2022

Contents

Lab 6 - Spectroscopy of Atomic Hydrogen and Deuterium

Introduction

The observations in optical spectroscopy in the 19th century led to the development of quantum theory. Among those observations, the line spectrum for hydrogen atoms was one of the most important ones. In the visible region, the line spectrum, showed by Johann Balmer, is well described by the geometric series:

$$\lambda = G\frac{n^2}{n^2-4} \qquad \forall n \in \mathbb{N} - \{1,2\} \tag{6.1}$$

where G is an empirical constant. In 1889, Johannes Rydberg proposed a more general expression for the spectral line of hydrogen atom:

$$\frac{1}{\lambda} = R_H\left(\frac{1}{n_1^2} - \frac{1}{n_2^2}\right) \qquad n_2 > n_1 \tag{6.2}$$

and the Balmer series characterized by (6.1) is a special case characterized by (6.2) when $n_1 = 2$. The Rydberg constant is given by the following:

$$R_H = \left(\frac{1}{4\pi\epsilon_0}\right)^2 \frac{me^4}{4\pi\hbar^3 c} = 10.9737316\,\text{micron}^{-1} \tag{6.3}$$

Here the spectral lines predicted by (6.2) are given by the followings:

Line Designation	Transition	Wavelength (nm)
H_α	$3 \to 2$	656.27
H_β	$4 \to 2$	486.13
H_γ	$5 \to 2$	434.05
H_δ	$6 \to 2$	410.17
H_ϵ	$7 \to 2$	397.01
H_ζ	$8 \to 2$	388.91

Note that R_H is the value that would be appropriate if the nucleus is fixed. In practice, the mass of the electron should be replaced by the reduced mass $m_e M_N/(m_e + M_N)$ where M_N is the mass of the nucleus. Thus the spectrum has a dependence on the nucleus mass. Hence one can find the ratio between the electron mass and the proton mass by evaluating the frequency shift $\Delta\nu = \nu_H - \nu_D$ of the alpha lines in the hydrogen atom spectrum and the deuterium atom spectrum:

$$\frac{\Delta\nu}{\nu_H} = \left(1 - \frac{m_e}{M_D}\right) - \left(1 - \frac{m_e}{M_P}\right) \approx \frac{m_e}{2M_P} \tag{6.4}$$

where M_D is the nucleus mass of the deuterium atom, and we approximate M_D by $2M_P$, with M_P being the mass of a proton.

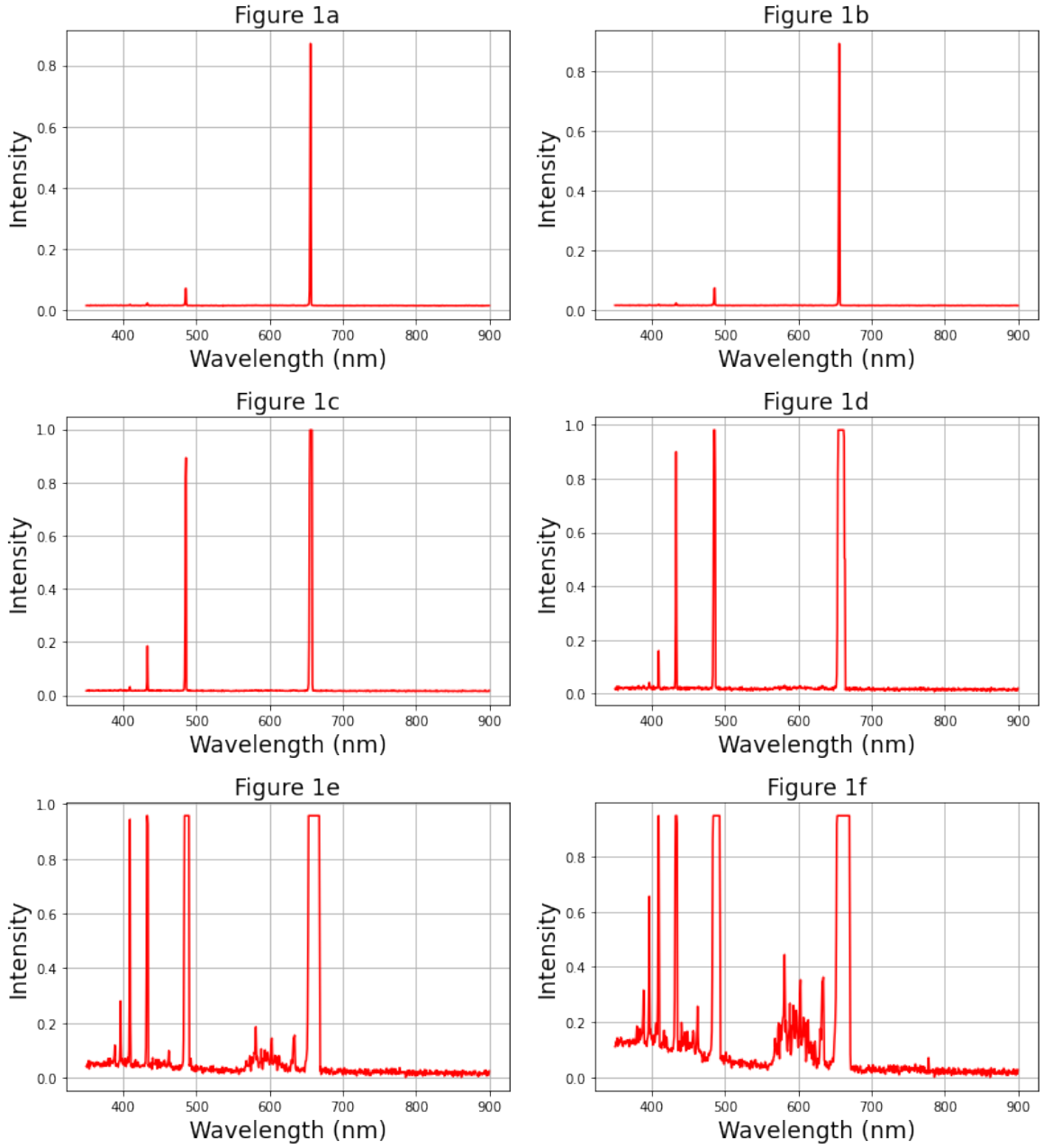
In Lab 6 of Physics 391, we measure six lines of the Balmer series ranging from $656\,nm$ to $388\,nm$ with a hydrogen discharge tube as the light source. We then compare our results to the predicted values given by (6.2) and used our data to estimate R_H. The estimated R_H is given by $11.0624 \pm 0.0742\,\text{micron}^{-1}$, capturing the theoretical value given by (6.3). We then use the same setup to make precise measurements on the alpha lines for both the hydrogen and the deuterium atom, estimate the frequency shift, and hence the electron-proton mass ratio via (6.4).

Experimental setup

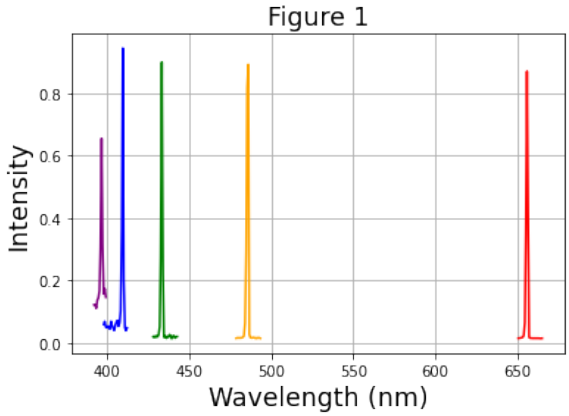
First, we measure the wavelengths of the first six lines of the Balmer series using a hydrogen discharge tube as a light source and a spectrometer manufactured by Vernier. The light sources are narrow glass gas discharge tubes that are mounted in a vertical support which supplies the voltage. We adjust the spectrometer exposure for each line to optimize the accuracy, and the readout is performed by Logger Pro which controls all aspects of the data acquisition and logging. We obtain one set of data that contains wavelengths and the corresponding detected intensities of the light from the hydrogen discharge tube. Then we use a similar setup and use a deuterium discharge tube to obtain data for estimating the frequency shift. We take a total of 20 different data runs, 5 for hydrogen, 5 for deuterium, then again 5 for hydrogen and 5 for deuterium.

Visualizing the data

For the measurement of the wavelengths of the Balmer series, we make 6 measurements, with different exposure of the spectrometer, and obtain the followings:



We truncate each dataset and limit the wavelengths to be around the wavelengths of the Balmer series, and concatenate into a single data set, obtaining the following figure:



Each line in Fig. 1 represents a spectral line in the Balmer series.

We then estimate the wavelengths of the peaks of the intensity in the truncated-concatenated dataset via mean position:

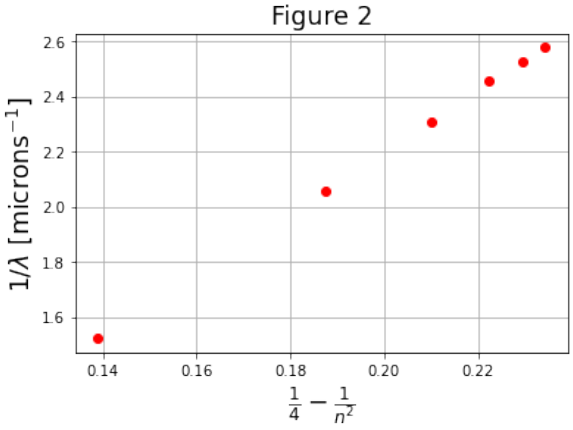
$$\bar{x} = \frac{\sum_i x_i y_i}{\sum_i y_i}$$

where $\{x_i\}$ is the set of wavelengths near the local peak and $\{y_i\}$ is the associated set of intensities. As a result, we obtain the estimated wavelengths of the Balmer series:

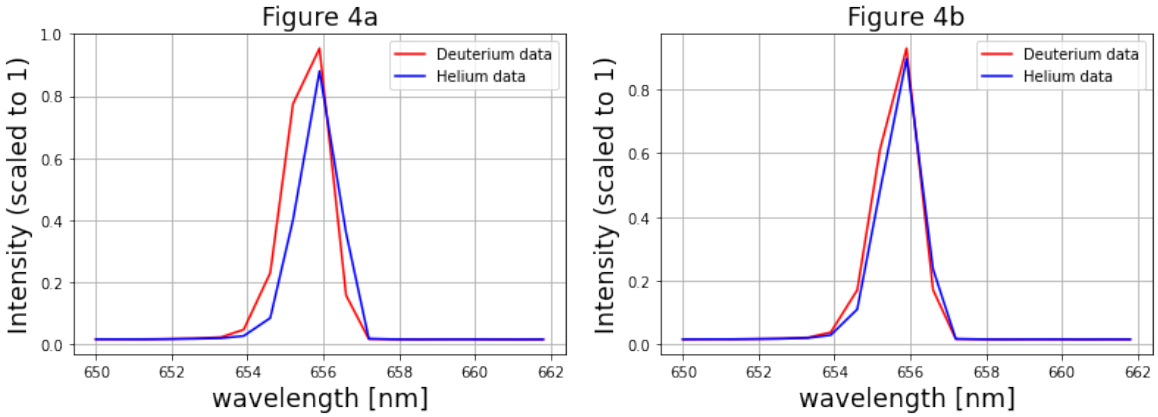
Line Designation	Estimated λ (nm)	Theoretical λ (nm)	Deviation
H_α	656.22	656.27	0.01%
H_β	485.66	486.13	0.10%
H_γ	433.86	434.05	0.04%
H_δ	407.33	410.17	0.69%
H_ϵ	395.97	397.01	0.26%
H_ζ	388.21	388.91	0.17%

Note that the deviation from the theoretical values is higher for those with lower wavelengths, this is because the lower wavelengths have smaller intensities, and hence it is more difficult for the equipment to make precise measurements.

We plot $1/\lambda$ over $1/4 - 1/n^2$ where n is the n_2 of the spectral line given in (6.2):



For the comparison between the hydrogen atom alpha line and deuterium alpha line, we group the first 5 hydrogen data runs and the first 5 deuterium data runs as experiment 1, and the rest as experiment 2. Then we calculate the average intensity recorded for each wavelength.

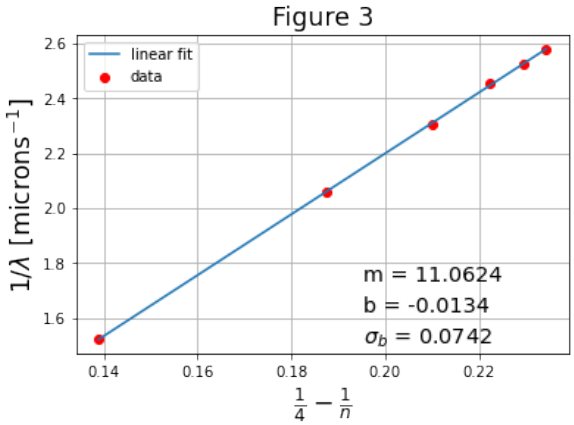


Here Fig. 4a displays data for experiment 1, and Fig. 4b displays data for experiment 2. Note that in both experiments, the peaks of the deuterium spectra have smaller wavelengths.

Analyzing the data

6.4.1 Estimation of the Rydberg constant

Here we perform a linear fit $y = mx + b$ to the dataset displayed in Fig. 2:



According to (6.2), R_H is given by the slope of the linear fit, in our case, we see that R_H is estimated to be $11.0624\,\text{micron}^{-1}$, with a standard deviation 0.0742. Note that the y-axis interception $b = -0.0134$ is negligible, hence confirming the validity of (6.2). Moreover, the theoretical value of R_H is around 10.9737, and we have:

$$\frac{|11.0624 - 10.9737|}{0.0742} = 1.195$$

hence the theoretical value of R_H is captured within 3σ from our estimation, suggesting that the theoretical value is well-predicted by our data, and our data gives a good approximation to the true R_H.

6.4.2 Comparison of deuterium and hydrogen alpha lines

First, we perform individual Gaussian fit to the data via the following model:

$$y_H = a_H + b_H e^{-\frac{(x-(656-\delta_H))^2}{2\sigma_H^2}} \qquad y_D = a_D + b_D e^{-\frac{(x-(656-\delta_D))^2}{2\sigma_D^2}} \tag{6.5}$$

where y_H, y_D are the intensities of the hydrogen spectrum and the deuterium spectrum, respectively, a_H, a_D, b_H, b_D are linear parameters, δ_H, δ_D, σ_H, σ_D are non-linear parameters, and x is the wavelengths. For individual fit, we obtain the followings:

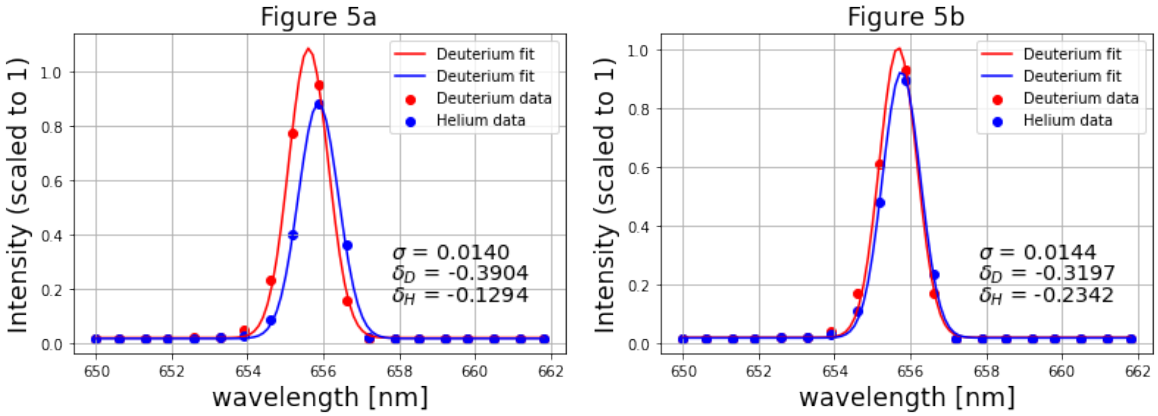


Fig. 5a displays the data and the fit for experiment 1, and Fig. 5b displays the data and the fit for experiment 2. For both experiments, we see that the peaks for the deuterium spectrum have lower wavelengths. More specifically, we can compute:

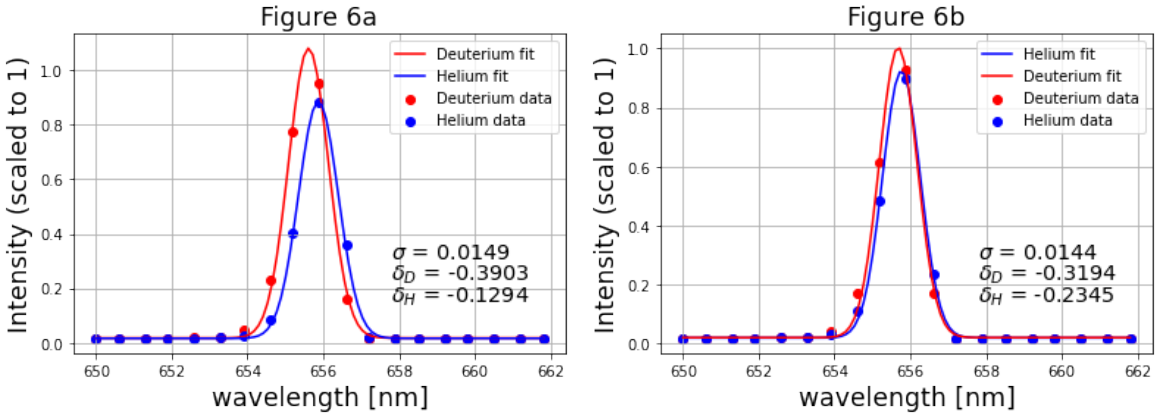
$$\delta_H - \delta_D = 0.2610\,nm \qquad \text{(experiment 1)}$$
$$\delta_H - \delta_D = 0.0855\,nm \qquad \text{(experiment 2)}$$

and the standard deviation for $\delta_H - \delta_D$

$$\sigma = \sqrt{\sigma_D^2 + \sigma_H^2} = 0.0140\,nm \qquad \text{(experiment 1)}$$
$$\sigma = \sqrt{\sigma_D^2 + \sigma_H^2} = 0.0144\,nm \qquad \text{(experiment 2)}$$

The theoretical difference $\Delta = \delta_H - \delta_D$ is $0.18\,nm$, and we see that such theoretical value is not captured within 3σ by either of the two experiments. In particular, experiment 1 gives an overestimation (5.78σ away) of the true Δ, while experiment 2 gives a underestimation (6.6σ away) of Δ. Moreover, we note that the estimated value of δ_D and δ_H in both experiments are less than zero, but the true value of δ_D and δ_H should be greater than zero as predicted by (6.2), this is a systematic error. One possible cause of such an error is the change in temperature of the hydrogen discharger tube, and the calibration of the equipment in the experiments.

We then perform a joint fitting of the Gaussian curves in the two experiments. For each experiment, we again fit the data via (6.5), but with $\sigma_D = \sigma_H$, and we obtain the following results:



Notice that the difference between this fit and the previous one is minimal. Experiment 1 gives an underestimation of the true Δ, and experiment 2 gives an overestimation of the true Δ.

	estimated $\Delta = \delta_H - \delta_D$	σ	difference from true Δ
Experiment 1	0.2608	0.0149	5.7σ
Experiment 2	0.0849	0.0144	6.6σ

6.4.3 Estimation of electron-proton mass ratio

Rearranging (6.4), we obtain:

$$\frac{m_e}{M_P} = 2 \cdot \frac{\Delta\nu}{\nu_H}$$

and we obtain the following results:

	m_e/M_P (via joint fitting)	st. dev. (via joint fitting)
Experiment 1	0.000795	0.056959
Experiment 2	0.000259	0.169572

	m_e/M_P (via individual fitting)	st. dev. (via individual fitting)
Experiment 1	0.000796	0.053458
Experiment 2	0.000261	0.168148

The true ratio between electron and proton mass is given by 0.000544. The standard deviation in our estimation is large, as a result, the true value is captured within 1σ from our estimated values in alll cases. This shows that the ratio between electron mass and proton mass is statistically well predicted from our experimental results.

Summary

In Lab 6 of Physics 391, we measure the wavelengths of the Balmer series for hydrogen atoms, and we have verified the validity of equation (6.2). We also estimated the Rydberg constant to be around $11.0624 \pm 0.0742\,\text{micron}^{-1}$, successfully capturing the true value of R_H. Moreover, we measured the frequency shift of the alpha line between the deuterium atom and hydrogen atom due to the effect of reduced mass. There is a systematic error present in our data, possibly due to the temperature change in the hydrogen discharging tube and some other equipment calibration issues. Lastly, we estimated the ratio of the electron mass over the proton mass, and the estimation captures the true ratio of the two masses.

Code

The code for computing statistics of the data sets is attached.

```
import numpy as np
from matplotlib import pyplot as plt
import pandas as pd
import scipy.optimize as opt
import os
import sys




## Define relevant experimental constants here
c = 2.9979e8
e = 1.6022e-19
h = 6.62607015e-34
me = 9.1093837e-31
mp = 1.67262192e-27


# Read in data from your data directory here.  Use pd.read_csv.
Hyd_df = pd.read_csv('data/Hydrogen.csv')
Da1_df = pd.read_csv('data/Da_1.csv')
Da2_df = pd.read_csv('data/Da_2.csv')
Ha1_df = pd.read_csv('data/Ha_1.csv')
Ha2_df = pd.read_csv('data/Ha_2.csv')
Hydt_df = pd.read_csv('data/Hydrogen_t.csv')


#  Make plots here
lambs_all = Hyd_df['Run 1: Wavelength (nm)']

name = ['a','b','c','d','e','f']

for i in range(6):
  plt.plot(lambs_all, Hyd_df['Run '+str(i+1)+': Intensity (rel)'], c='red')
  plt.grid()
  plt.title('Figure 1'+name[i], fontsize='xx-large')
  plt.ylabel(r'Intensity', fontsize='xx-large')
  plt.xlabel('Wavelength (nm)', fontsize='xx-large')
  plt.show()


plt.plot(Hydt_df['Run 1: Wavelength (nm)'], Hydt_df['Run 1: Intensity (rel)'],
    c='red')
plt.plot(Hydt_df['Run 3: Wavelength (nm)'], Hydt_df['Run 3: Intensity (rel)'],
    c='red')
plt.plot(Hydt_df['Run 4: Wavelength (nm)'], Hydt_df['Run 4: Intensity (rel)'],
    c='red')
plt.plot(Hydt_df['Run 5: Wavelength (nm)'], Hydt_df['Run 5: Intensity (rel)'],
    c='red')
plt.plot(Hydt_df['Run 6: Wavelength (nm)'], Hydt_df['Run 6: Intensity (rel)'],
    c='red')
plt.grid()
plt.title('Figure 1', fontsize='xx-large')
plt.ylabel(r'Intensity', fontsize='xx-large')
plt.xlabel('Wavelength (nm)', fontsize='xx-large')
plt.show()


def calculate_mean_peak(wavelength, intensity) :
```

```
  '''Calculates the mean peak intensity via Eqn 7
  '''
  return np.sum([x*y for (x,y) in zip(wavelength, intensity)])/np.sum(intensity
    )
  #### FILL THIS OUT TO RETURN THE MEAN WAVELENGTH ####

alpha = calculate_mean_peak(Hydt_df['Run 1: Wavelength (nm)'], Hydt_df['Run 1:
    Intensity (rel)'])
beta = calculate_mean_peak(Hydt_df['Run 3: Wavelength (nm)'], Hydt_df['Run 3:
    Intensity (rel)'])
delta = calculate_mean_peak(Hydt_df['Run 4: Wavelength (nm)'], Hydt_df['Run 4:
    Intensity (rel)'])
gamma =  calculate_mean_peak(Hydt_df['Run 5: Wavelength (nm)'], Hydt_df['Run 5:
     Intensity (rel)'])
epsilon = calculate_mean_peak(Hydt_df['Run 6: Wavelength (nm)'][:11], Hydt_df['
    Run 6: Intensity (rel)'][:11])
zeta = calculate_mean_peak(Hydt_df['Run 7: Wavelength (nm)'][:11], Hydt_df['Run
     7: Intensity (rel)'][:11])


def gaussian_model(x, a, b, xp, sigma) :
  '''Gaussian model
  '''
  return a + b * np.exp(-(x-xp)**2/(2*sigma**2))



### OPTIONAL SPACE TO FIT EACH CURVE HERE AND IN CELLS BELOW TO EXTRACT PEAK
    LOCATIONS ###
params, params_covariance = opt.curve_fit(gaussian_model, Hydt_df['Run 1:
    Wavelength (nm)'], Hydt_df['Run 1: Intensity (rel)'], p0 =
    [0.01,0.01,650,1])
perr = np.sqrt(np.diag(params_covariance))

# Calculate each peak here, and comparison to known values
err_alpha = np.abs(alpha-656.27)/656.27
err_beta = np.abs(beta-486.13)/486.13
err_delta = np.abs(delta-434.05)/434.05
err_gamma = np.abs(gamma-410.17)/410.17
err_epsilon = np.abs(epsilon-397.01)/397.01
err_zeta = np.abs(zeta-388.91)/388.91


# Make plot here and/or in a new set of cells below.
n=np.array([3,4,5,6,7,8])
n_x = np.array([(1/4)-(1/(i*i)) for i in n])
lambs = np.array([alpha, beta, delta, gamma, epsilon, zeta])/1000
lambs_y = 1/lambs

plt.scatter(n_x, 1/lambs, c='red')
plt.grid()
plt.title('Figure 2', fontsize='xx-large')
plt.ylabel(r'1/$\lambda$ [microns$^{-1}$]', fontsize='xx-large')
plt.xlabel(r'$\frac{1}{4} - \frac{1}{n}$', fontsize='xx-large')
plt.show()


def linear_model(x, m, b) :
  '''
  x - list, x-dataset
  m - float, slope
  b - float, y-intercept
  '''
```

```
  return m*x+b


# Perform your fit
popt, pcov = opt.curve_fit(linear_model, n_x, lambs_y, np.array([1,1]))
sig_m = np.sqrt(pcov[0])



# Overplot data and best fit model using plt.plot and plt.scatter,
plt.scatter(n_x, 1/lambs, c='red', label = 'data')
plt.plot(n_x, linear_model(n_x, popt[0], popt[1]), label = 'linear fit')
plt.grid()
plt.title('Figure 3', fontsize='xx-large')
plt.legend()
plt.annotate('m = %.4f '%popt[0], (0.63,0.35),
               xycoords='figure fraction',
               fontsize='x-large')
plt.annotate('b = %.4f'%popt[1], (0.63,0.28),
               xycoords='figure fraction',
               fontsize='x-large')
plt.annotate(r'$\sigma_b$ = %.4f'%sig_m[0], (0.63,0.21),
               xycoords='figure fraction',
               fontsize='x-large')
plt.ylabel(r'1/$\lambda$ [microns$^{-1}$]', fontsize='xx-large')
plt.xlabel(r'$\frac{1}{4} - \frac{1}{n}$', fontsize='xx-large')
plt.show()


### READ IN CSV FILES HERE AS DATAFRAMES ###
da = pd.read_csv('data/Da_1.csv')
da = pd.concat([da, pd.read_csv('data/Da_2.csv')], axis=1)
da.columns = ['L1', 'I1',
              'L2', 'I2',
              'L3', 'I3',
              'L4', 'I4',
              'L5', 'I5',
              'L6', 'I6',
              'L7', 'I7',
              'L8', 'I8',
              'L9', 'I9',
              'L10', 'I10',]

ha = pd.read_csv('data/Ha_1.csv')
ha = pd.concat([ha, pd.read_csv('data/Ha_2.csv')], axis=1)
ha.columns = ['L1', 'I1',
              'L2', 'I2',
              'L3', 'I3',
              'L4', 'I4',
              'L5', 'I5',
              'L6', 'I6',
              'L7', 'I7',
              'L8', 'I8',
              'L9', 'I9',
              'L10', 'I10',]

da1 = da[['L1', 'I1', 'L2', 'I2', 'L3', 'I3', 'L4', 'I4', 'L5', 'I5']]
da2 = da[['L6', 'I6', 'L7', 'I7', 'L8', 'I8', 'L9', 'I9', 'L10', 'I10']]
ha1 = ha[['L1', 'I1', 'L2', 'I2', 'L3', 'I3', 'L4', 'I4', 'L5', 'I5']]
ha2 = ha[['L6', 'I6', 'L7', 'I7', 'L8', 'I8', 'L9', 'I9', 'L10', 'I10']]


# Create your summed/averaged spectra here
```

```
da1_avgI = da1['I1'].values
ha1_avgI = ha1['I1'].values
for i in range(2,6):
  da1_avgI = da1_avgI + da1['I'+str(i)].values
  ha1_avgI = ha1_avgI + ha1['I'+str(i)].values
da1_avgI = da1_avgI/5
ha1_avgI = ha1_avgI/5

da2_avgI = da2['I6'].values
ha2_avgI = ha2['I6'].values
for i in range(7,11):
  da2_avgI = da2_avgI + da2['I'+str(i)].values
  ha2_avgI = ha2_avgI + ha2['I'+str(i)].values
da2_avgI = da2_avgI/5
ha2_avgI = ha2_avgI/5


# Plots here
plt.plot(da1['L1'], da1_avgI, c='red', label = 'Deuterium data')
plt.plot(da1['L1'], ha1_avgI, c='blue', label = 'Helium data')
plt.grid()
plt.title('Figure 4a', fontsize='xx-large')
plt.legend()
plt.ylabel(r'Intensity (scaled to 1)', fontsize='xx-large')
plt.xlabel(r'wavelength [nm]', fontsize='xx-large')
plt.show()


plt.plot(da1['L1'], da2_avgI, c='red', label = 'Deuterium data')
plt.plot(da1['L1'], ha2_avgI, c='blue', label = 'Helium data')
plt.grid()
plt.title('Figure 4b', fontsize='xx-large')
plt.legend()
plt.ylabel(r'Intensity (scaled to 1)', fontsize='xx-large')
plt.xlabel(r'wavelength [nm]', fontsize='xx-large')
plt.show()


#  Fit for delta_h from Experiments hydrogen and deuterium data
def gaussian_model(x, a, b, xp, sigma) :
  '''Gaussian model
  '''
  return a + b * np.exp(-(x-xp)**2/(2*sigma**2))


lambs = da1['L1']
parD1, corD1 = opt.curve_fit(gaussian_model, lambs, da1_avgI, p0 =
    [0.1,0.1,660,10])
sigD1 = np.sqrt(np.diag(corD1))
del_D1 = parD1[2]-656

parD2, corD2 = opt.curve_fit(gaussian_model, lambs, da2_avgI, p0 =
    [0.1,0.1,660,10])
sigD2 = np.sqrt(np.diag(corD2))
del_D2 = parD2[2]-656

parH1, corH1 = opt.curve_fit(gaussian_model, lambs, ha1_avgI, p0 =
    [0.1,0.1,656,10])
sigH1 = np.sqrt(np.diag(corH1))
del_H1 = parH1[2]-656

parH2, corH2 = opt.curve_fit(gaussian_model, lambs, ha2_avgI, p0 =
    [0.1,0.1,656,10])
```

```
sigH2 = np.sqrt(np.diag(corH2))
del_H2 = parH2[2]-656

expt1_sig = np.sqrt(sigD1[2]**2 + sigH1[2]**2)
expt2_sig = np.sqrt(sigD2[2]**2 + sigH2[2]**2)

print(del_D1,del_D2)
print(del_H1,del_H2)

print(parH1)

#  Overplot best-fit model and data for experiments
plt.scatter(da1['L1'], da1_avgI, c='red', label = 'Deuterium data')
plt.scatter(da1['L1'], ha1_avgI, c='blue', label = 'Helium data')
plt.plot(np.linspace(da1['L1'][0], da1['L1'].values.tolist()[-1], 100),
         gaussian_model(np.linspace(da1['L1'][0], da1['L1'].values.tolist()[-1], 100),
                        parD1[0], parD1[1], parD1[2], parD1[3]),
         c='red', label = 'Deuterium fit')
plt.plot(np.linspace(da1['L1'][0], da1['L1'].values.tolist()[-1], 100),
         gaussian_model(np.linspace(da1['L1'][0], da1['L1'].values.tolist()[-1], 100),
                        parH1[0], parH1[1], parH1[2], parH1[3]),
         c='blue', label = 'Deuterium fit')
plt.annotate(r'$\delta_H$ = %.4f'%del_H1, (0.68,0.28),
              xycoords='figure fraction',
              fontsize='x-large')
plt.annotate(r'$\delta_D$ = %.4f'%del_D1, (0.68,0.33),
              xycoords='figure fraction',
              fontsize='x-large')
plt.annotate(r'$\sigma$ = %.4f'%expt1_sig, (0.68,0.38),
              xycoords='figure fraction',
              fontsize='x-large')
plt.grid()
plt.title('Figure 5a', fontsize='xx-large')
plt.legend()
plt.ylabel(r'Intensity (scaled to 1)', fontsize='xx-large')
plt.xlabel(r'wavelength [nm]', fontsize='xx-large')
plt.show()


plt.scatter(da1['L1'], da2_avgI, c='red', label = 'Deuterium data')
plt.scatter(da1['L1'], ha2_avgI, c='blue', label = 'Helium data')
plt.plot(np.linspace(da1['L1'][0], da1['L1'].values.tolist()[-1], 100),
         gaussian_model(np.linspace(da1['L1'][0], da1['L1'].values.tolist()[-1], 100),
                        parD2[0], parD2[1], parD2[2], parD2[3]),
         c='red', label = 'Deuterium fit')
plt.plot(np.linspace(da1['L1'][0], da1['L1'].values.tolist()[-1], 100),
         gaussian_model(np.linspace(da1['L1'][0], da1['L1'].values.tolist()[-1], 100),
                        parH2[0], parH2[1], parH2[2], parH2[3]),
         c='blue', label = 'Deuterium fit')
plt.annotate(r'$\delta_H$ = %.4f'%del_H2, (0.68,0.28),
              xycoords='figure fraction',
              fontsize='x-large')
plt.annotate(r'$\delta_D$ = %.4f'%del_D2, (0.68,0.33),
              xycoords='figure fraction',
              fontsize='x-large')
plt.annotate(r'$\sigma$ = %.4f'%expt2_sig, (0.68,0.38),
              xycoords='figure fraction',
              fontsize='x-large')
plt.grid()
```

```
289 plt.title('Figure 5b', fontsize='xx-large')
290 plt.legend()
291 plt.ylabel(r'Intensity (scaled to 1)', fontsize='xx-large')
292 plt.xlabel(r'wavelength [nm]', fontsize='xx-large')
293 plt.show()
294
295 ## NOTE: You may need to change the name of the dataframe if this is not the
        variable name you used
296 ##   for df_hydrogen_expt1.
297 h1d1_avgI = np.concatenate((ha1_avgI, da1_avgI))
298 h2d2_avgI = np.concatenate((ha2_avgI, da2_avgI))
299
300 Len_ha = ha.shape[0]
301
302
303 def two_gaussian_model(wavelengths, aH, bH, lam_H, aD, bD, lam_D, sigma,
304                        length_hydrogen=Len_ha) :
305   ''' Model to fit both
306   '''
307
308   hyd_gaussian = gaussian_model(wavelengths[:length_hydrogen], aH, bH, lam_H,
        sigma)
309   deut_gaussian = gaussian_model(wavelengths[length_hydrogen:], aD, bD, lam_D,
        sigma)
310
311   return np.append(hyd_gaussian, deut_gaussian)
312
313
314
315
316
317 #  Perform your fit here
318 parj1, corj1 = opt.curve_fit(two_gaussian_model, lambs.append(lambs).values,
        h1d1_avgI, p0 = [0.01,0.01, 655,0.01, 0.01, 655,20])
319 sigj1 = np.sqrt(np.diag(corj1))
320
321
322
323 parj2, corj2 = opt.curve_fit(two_gaussian_model, lambs.append(lambs).values,
        h2d2_avgI, p0 = [0.01,0.01,655,0.01,0.01, 655, 20])
324 sigj2 = np.sqrt(np.diag(corj2))
325
326 lam_space = np.linspace(da1['L1'][0], da1['L1'].values.tolist()[-1], 100)
327 lam_spaces = np.concatenate((lam_space, lam_space))
328 y_1 = two_gaussian_model(lam_spaces, parj1[0], parj1[1], parj1[2], parj1[3],
        parj1[4], parj1[5], parj1[6], 100)
329 y_2 = two_gaussian_model(lam_spaces, parj2[0], parj2[1], parj2[2], parj2[3],
        parj2[4], parj2[5], parj2[6], 100)
330
331 del_H1_j = parj1[2]-656
332 del_D1_j = parj1[5]-656
333 del_H2_j = parj2[2]-656
334 del_D2_j = parj2[5]-656
335
336
337 expt1_sig_j = np.sqrt(sigj1[2]**2 + sigj1[5]**2)
338 expt2_sig_j = np.sqrt(sigj2[2]**2 + sigj2[5]**2)
339
340 plt.scatter(da1['L1'], da1_avgI, c='red', label = 'Deuterium data')
341 plt.scatter(da1['L1'], ha1_avgI, c='blue', label = 'Helium data')
342 plt.plot(lam_spaces[100:], y_1[100:], c='red', label = 'Deuterium fit')
343 plt.plot(lam_spaces[:100], y_1[:100], c='blue', label = 'Helium fit')
344 plt.annotate(r'$\delta_H$ = %.4f'%del_H1_j, (0.68,0.28),
```

```
                xycoords='figure fraction',
                fontsize='x-large')
plt.annotate(r'$\delta_D$ = %.4f'%del_D1_j, (0.68,0.33),
                xycoords='figure fraction',
                fontsize='x-large')
plt.annotate(r'$\sigma$ = %.4f'%expt1_sig_j, (0.68,0.38),
                xycoords='figure fraction',
                fontsize='x-large')
plt.grid()
plt.title('Figure 5a', fontsize='xx-large')
plt.legend()
plt.ylabel(r'Intensity (scaled to 1)', fontsize='xx-large')
plt.xlabel(r'wavelength [nm]', fontsize='xx-large')
plt.show()


plt.scatter(da1['L1'], da2_avgI, c='red', label = 'Deuterium data')
plt.scatter(da1['L1'], ha2_avgI, c='blue', label = 'Helium data')
plt.plot(lam_spaces[:100], y_2[:100], c='blue', label = 'Helium fit')
plt.plot(lam_spaces[100:], y_2[100:], c='red', label = 'Deuterium fit')
plt.annotate(r'$\delta_H$ = %.4f'%del_H2_j, (0.68,0.28),
                xycoords='figure fraction',
                fontsize='x-large')
plt.annotate(r'$\delta_D$ = %.4f'%del_D2_j, (0.68,0.33),
                xycoords='figure fraction',
                fontsize='x-large')
plt.annotate(r'$\sigma$ = %.4f'%expt2_sig_j, (0.68,0.38),
                xycoords='figure fraction',
                fontsize='x-large')
plt.grid()
plt.title('Figure 5b', fontsize='xx-large')
plt.legend()
plt.ylabel(r'Intensity (scaled to 1)', fontsize='xx-large')
plt.xlabel(r'wavelength [nm]', fontsize='xx-large')
plt.show()


# Calculate here
c = 299792458


f_H1_j = c/parj1[2]
f_D1_j = c/parj1[5]
del_f_1j = f_H1_j - f_D1_j
ratio_1j = (f_H1_j - f_D1_j)/f_H1_j*2
sigfD1j = -c*sigj1[2]/(parj1[2]**2)
sigfH1j = -c*sigj1[5]/(parj1[5]**2)
sig_rj1 = np.sqrt((np.sqrt(sigfD1j**2 + sigfD1j**2)/del_f_1j)**2 + (sigfH1j/
    f_H1_j)**2)



f_H2_j = c/parj2[2]
f_D2_j = c/parj2[5]
del_f_2j = f_H2_j - f_D2_j
ratio_2j = (f_H2_j - f_D2_j)/f_H2_j*2
sigfD2j = -c*sigj2[2]/(parj2[2]**2)
sigfH2j = -c*sigj2[5]/(parj2[5]**2)
sig_rj2 = np.sqrt((np.sqrt(sigfD2j**2 + sigfD2j**2)/del_f_2j)**2 + (sigfH2j/
    f_H2_j)**2)


f_D1 = c/parD1[2]
```

```
406 f_H1 = c/parH1[2]
407 del_f_1 = f_H1 - f_D1
408 ratio_1 = (f_H1 - f_D1)/f_H1*2
409 sigfD1 = -c*sigD1[2]/(parD1[2]**2)
410 sigfH1 = -c*sigH1[2]/(parH1[2]**2)
411 sig_r1 = np.sqrt((np.sqrt(sigfD1**2 + sigfD1**2)/del_f_1)**2 + (sigfH1/f_H1)
        **2)
412
413
414 f_D2 = c/parD2[2]
415 f_H2 = c/parH2[2]
416 del_f_2 = f_H2 - f_D2
417 ratio_2 = (f_H2 - f_D2)/f_H2*2
418 sigfD2 = -c*sigD2[2]/(parD2[2]**2)
419 sigfH2 = -c*sigH2[2]/(parH2[2]**2)
420 sig_r2 = np.sqrt((np.sqrt(sigfD2**2 + sigfD2**2)/del_f_2)**2 + (sigfH2/f_H2)
        **2)
421
422
423 print(parj1[2]-parj1[5], parj2[2]-parj2[5], parH1[2]-parD1[2], parH2[2]-parD2
        [2])
424 print(ratio_1j, ratio_2j, ratio_1, ratio_2)
425 print(sig_rj1, sig_rj2, sig_r1, sig_r2)
```