Introduction to Spectroscopy

AST 325: LAB 2

Anntara Khan

anntara.khan@mail.utoronto.ca (1002321891)

Abstract

Spectroscopy is described to be the study of emission and absorption of any radiation matter including light. It involves splitting a light into its base wavelengths similarly to using a prism to separate light into its rainbow colors.(ref3) Spectroscopy is an important tool for astronomy as it allows the observation of celestial bodies and at large the universe. Using spectroscopy we can determine the various chemical and physical characteristics of the objects that float around in the vastness of space. Few such characteristics that will be mentions in this experiment are temperature and velocity. In the first part of this experiment, 1-dimensional spectral data from Neon lamps were reduced using wavelength calibration to determine the temperature of a black body spectrum. Following that 2-dimensional spectral data (2-D dispersed image) of OH telluric sky lines would be reduced using wavelength Calibration to determine the velocity of ionized iron gas from a supernova explosion.

Introduction

The purpose of this lab is to understand how Spectroscopy works. Spectroscopy is an important part of Physical science, especially Astronomy and Astrophysics, where all findings and outcomes of hypotheses heavily rely on emission and absorption data (ref). Focus of this lab would be on wavelength calibration which is a crucial element of spectroscopy. Spectral data obtained from a spectroscope provides peak intensity. Which can be used to obtain peak wavelengths of the emission lines shown in the spectral data. Using the wavelength and Wien's displacement law the temperature of a black body object can be calculated, where the calculated temperature would be, $T = 4671.16$. And using the wavelength and the Doppler shift formula the velocity of the object emitting Fe II line scan be calculated to be $v = 3531/s$.

Observations and Data

The Material used in this experiment is a USB4000 spectrograph. The spectrometer takes in light, breaks it into it's spectral components, digitizes the signal as a function of wavelength

and displays the data on a connected computer(ref1).

1-D Spectral Data

For one dimensional spectral data, there were two set of data given, 'Group_G_BB.dat' which contains the intensity of a radiating blackbody. and 'Ne_calib.dat' which contains the intensity of a neon lamp. Wavelength range of Neon gas line transmission can range between 500 nm to 800 nm are shown in figure 4 of the lab manual. These wavelengths of Neon lights occurred at different intensity is shown on figure 5 of the lab manual.

2-D Spectral Data

The the given data "Near-Infrared.fits" was a real data file of a 2-dimensional dispersed image of ionized iron gas from a supernova explosion saved in the FITS (Flexible Image Transport System) file format. The data is stored in a 3-d array, and it can be plotted to get the image in figure 1. The wavelength of the five brightest OH lines was provided in

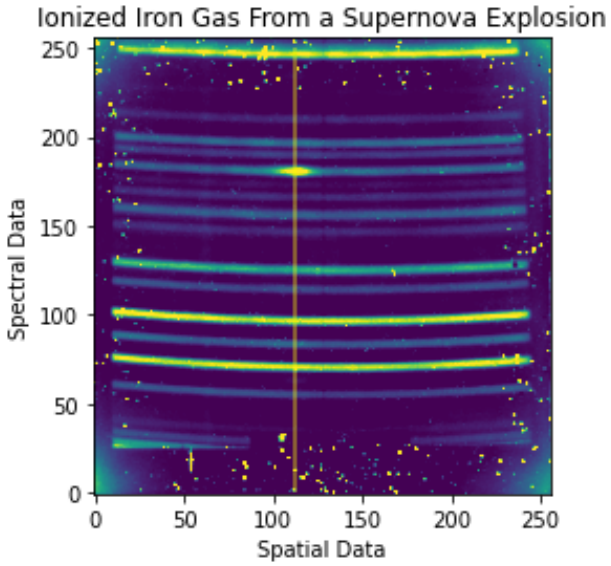


Fig. 1: Dispersed image of ionized iron gas from a supernova explosion. The vertical yellow line indicates the position of Fe II emission line. The horizontal blue and light green lines (curves) are the OH sky telluric lines.

figure 7 of the lab manual. Five bright OH sky lines are easily identifiable using the relative intensities of the known OH spectrum given in figure 8 of the lab manual.

Astronomical Observations to Obtain Spectra

Astronomical observation was conducted to obtain spectra of bright objects (e.g. Jupiter, Saturn, Markab, Enif/Epsilon Pegasi, Mirach, etc) using a spectrograph on the campus

telescope located on the 16th floor of the McLennan Physics tower. The spectrograph used was a Shelyak Alpy 600 spectrograph. The first set of data obtained were from Jupiter. The telescope was pointed towards Jupiter and the position of it was adjusted till the bright point of Jupiter alighted with the slit of the spectorgraph. Once aligned the exposure time was set to be 10 seconds and the image data was recorded and saves as "JUPITER_10secs.fit". Following this same process the spectral data was collected from; Saturn ("SATURN_10secs.fit"), Markab ("MARKAB_30secs.fit"), Enif/Epsilon Pegasi ("ENIF_30secs.fit") and Mirach ("MIRACH_30secs.fit"). In order to calibrate the data, three bias images, three dark images with one second exposure, three dark images with ten second exposure, three dark images with thirty second exposure, three flat images with one second exposure were taken.

Data Reduction and Methods

All collected and given data for the lab were loaded into Python for reduction (see appendix A). following sections takes the data introduced in the previous section and reduces them to obtain the wavelength solutions.

1-D Spectral Data

The goal of the first part of this lab is to determine the temperature of a blackbody. To do this the wavelength of the blackbody spectra is needed. To Obtain the wavelength the first step that needed to be completed would be to find the peak intensity location and height of the neon lamp data. The peak intensity of the neon lines can be seen in figure 2.
Using the Neon lamp data the peak height and the pixel location was determined using full width at half maximum (FWHM) method. The data was then plotted onto the peak location to get figure 2 (see appendix C for details). The peak heights can be compared

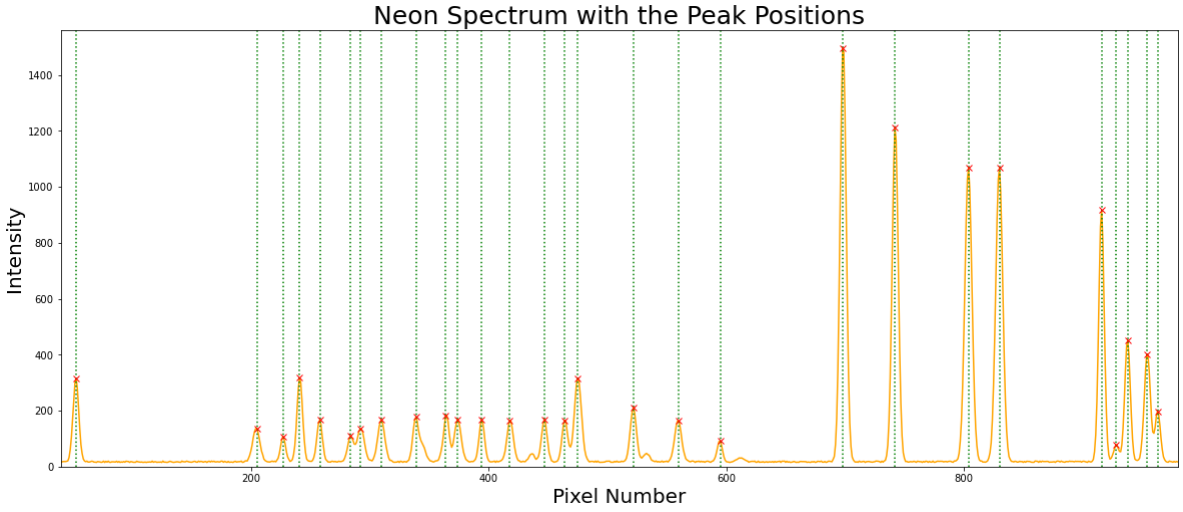


Fig. 2: Neon peak location illustrated by the the green lines and the peak height illustrated by the red crosses.

to the wavelength intensity shown in figure 5 of the lab manual to obtain a list of peak wavelength (Table 1) which was stored as "neonwl.txt".

Tab. 1: list of peak wavelength

Identified wavelengths of the Neon lines (in nm)		
511.367	616.359	703.241
511.650	621.728	717.394
540.056	626.649	724.512
576.441	633.442	743.890
585.249	638.299	748.887
588.189	640.225	753.577
602.000	650.653	754.404
607.433	659.895	837.761
614.306	692.947	849.536
878.375	—	—

Using `scipy.curvefit`, The wavelengths and pixels positions were fitted to approximate the wavelength solution in the form of $y = mx + b$. Linear, quadratic and cubic fit of the data was plotted along with the scatter plot of the wavelength and the pixel position in figure 3 (see appendix D for details). Going back to the blackbody spectrum, the peak intensity

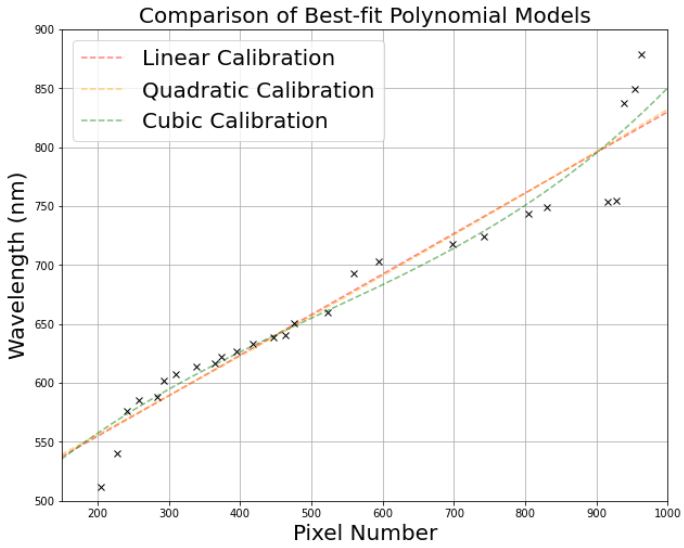


Fig. 3: Scatter plot approximating the possible wavelength solution, along with the linear fit, quadratic fit, and cubic fit of the wavelength solution for the Neon lamp spectra.

of the spectrum was determined using `scipy.signal.findpeaks()` which gave the pixel position of the peak in the blackbody spectra (see appendix E for details), which would be the value of x for the wavelength solution equation. Now that the vales of m, x, and b was known $y = mx + b$ could be solved to find the peak wavelength of the blackbody spectra. Using Wein's law and the wavelength, the Temperature of the blackbody can be calculated (see appendix F for details).

2-D Spectral Data

The goal for the second part of this lab was to calculate the velocity of an object emitting Fe II lines. To read the file into python `fits` package from `astropy.io` library was used. Once the given data was processed into python. `numpy` and `matplotlib` libraries were used to manipulate the data into a workable form and plot it. The given array was a 3-d array, which was indexed into a workable 2-d array. The matrix formed by the data was transposed using `numpy.transpose()`, as image files can sometimes be flipped. Once the 2d array was obtained it was plotted to generate the image (figure 1) using `imshow()` of the `matplotlib` library.

The x-axis of the plot contained the spatial information and y-axis of the plot contained the spectral information. The median of the spectral data between the pixel position 135 to 150 was taken using `numpy.median()`, which results in a 1-D array which was stored as `wave_int` (see appendix B for details). The reason behind using this specific location is because it is the closest position where the Fe II and one of the OH lines overlap. The peak intensity of the OH lines can be seen in figure 4. Using the OH line data (`wave_int`) the peak height

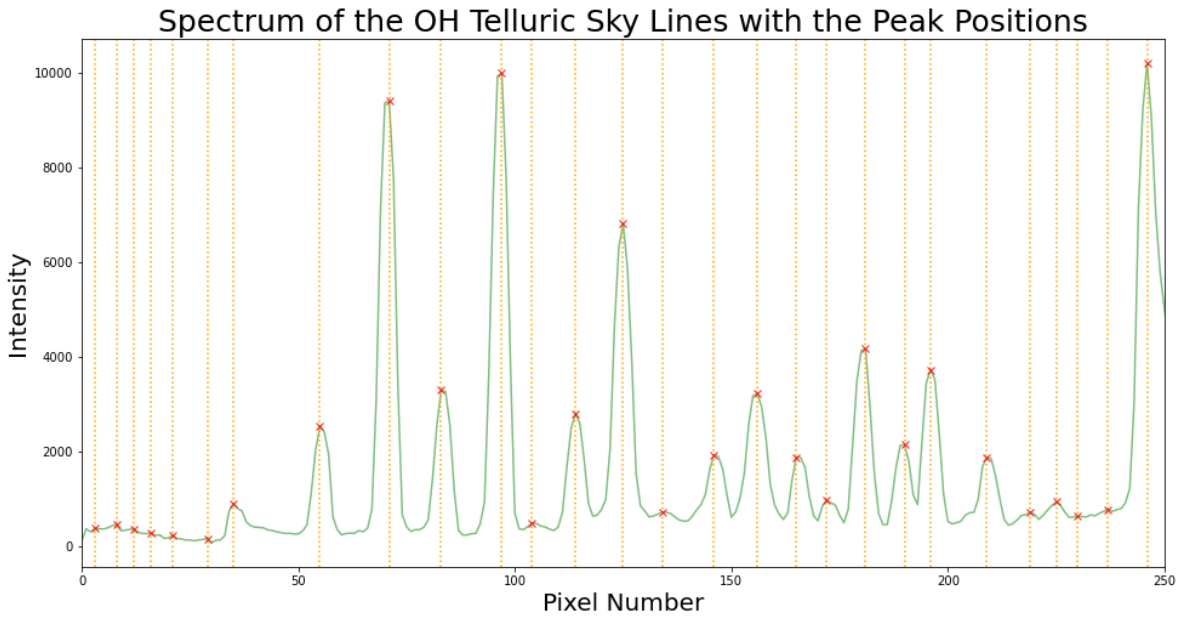


Fig. 4: OH peak location illustrated by the the orange lines and the peak height illustrated by the red crosses.

and the pixel location was determined using full width at half maximum (FWHM) method. The data was then plotted onto the peak location to get figure 4. (see appendix c for details) The peak heights can be compared to the wavelength intensity plot in figure 7 and 8 of the lab manual to obtain a list of peak wavelength (Table 2) which was stored as "ohwl.txt". Using `scipy.curvefit`, The wavelengths and pixels positions were fitted to approximate the wavelength solution in the form of $y = mx + b$. Linear, quadratic and cubic fit of the data was plotted along with the scatter plot of the wavelength and the pixel position in figure 5 (see appendix D for details).

Going back to Figure 1, the location of the Fe II emission line was obtained from the

Tab. 2: list of peak wavelength

Identified wavelengths of the OH lines (in Å)		
16128.608	16442.155	16702.639
16194.618	16447.616	16708.852
16235.376	16479.061	16724.739
16302.278	16502.365	16732.479
16317.161	16553.814	16755.155
16341.755	16586.322	16763.572
16351.954	16612.053	16802.368
16360.385	16689.201	16840.481
16388.492	16692.380	16903.679
16414.737	—	—

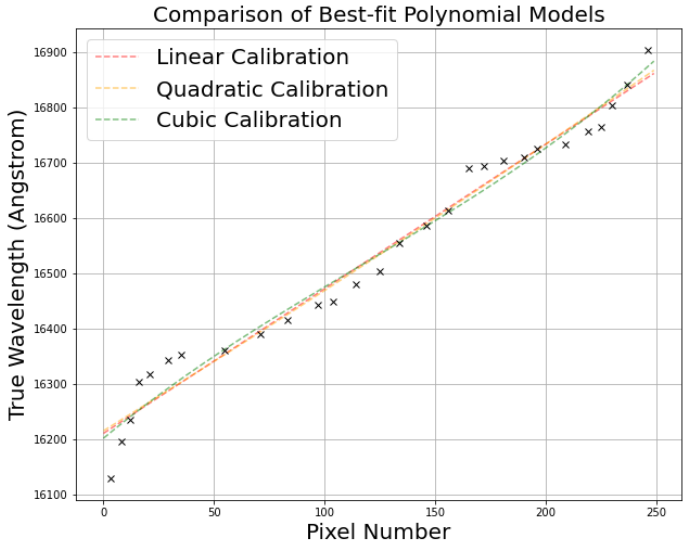


Fig. 5: Scatter plot approximating the possible wavelength solution, along with the linear fit, quadratic fit, and cubic fit of the wavelength solution for the OH line spectra.

location of the yellow line, the peak intensity of the spectrum was determined using `scipy.signal.findpeaks()`, which gave the pixel positions of the peaks in the OH line spectra (see appendix D for details), since it resulted in multiple spectra, the highest peak pixel location and peak value was taken from figure 5 (see appendix E for details). This pixel location would be the value of x for the wavelength solution equation. Now that the vales of m, x, and b was known $y = mx + b$ could be solved to find the peak wavelength of the Fe II emission lines. Using Doppler Shift equation and the wavelength, the velocity of the gas emitting the Fe II lines can be calculated (see appendix F for details).

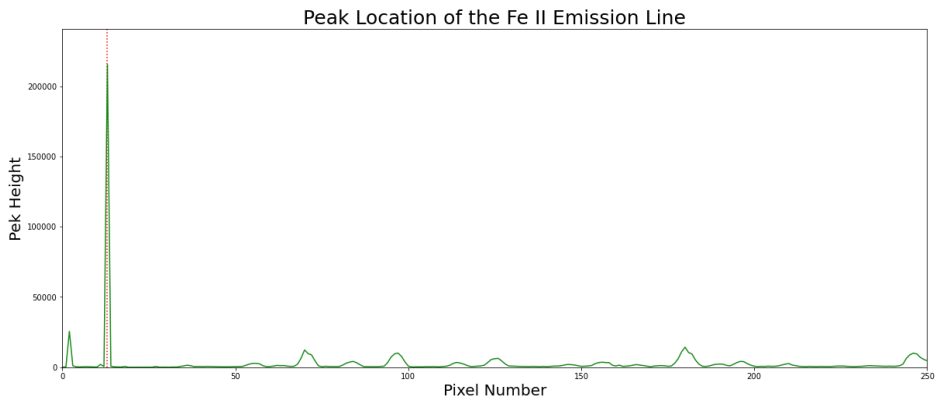


Fig. 6: Scatter plot approximating the possible wavelength solution for the OH line Spectra

Astronomical Observations to Obtain Spectra

The goal of this part of the lab is to reduce the imaged collected from the telescope and spectograph and extracting the spectra. To read the file into python `fits` package from `astropy.io` library was used. Once the given data was processed into python. `numpy` and `matplotlib` libraries were used to manipulate the data into a workable form and plot it. To calibrate the image the median of the biases was taken by stacking the three images together and taking the median to obtain a 2-D array. Similarly the median of darks (1s, 10s, and 30s) and the flats were taken. to obtain the 2-D arrays. the fist part of the reduction process would include (a) subtracting the image of the bright object from the dark of same exposure time, (b) and subtracting the flats from the darks of 1s exposure time (see appendix G for details). Then (a) was divided by (b) to obtain the calibrated image. The images were then plotted using `matplotlib.imshow()` which can be seen in figure 7 (see appendix H for details). Once the location of the spectra was determined the index of the array was extracted that contained the spectra. which was then plotted against the spectral position.

Data Analysis and Modeling

1-D Spectral Data

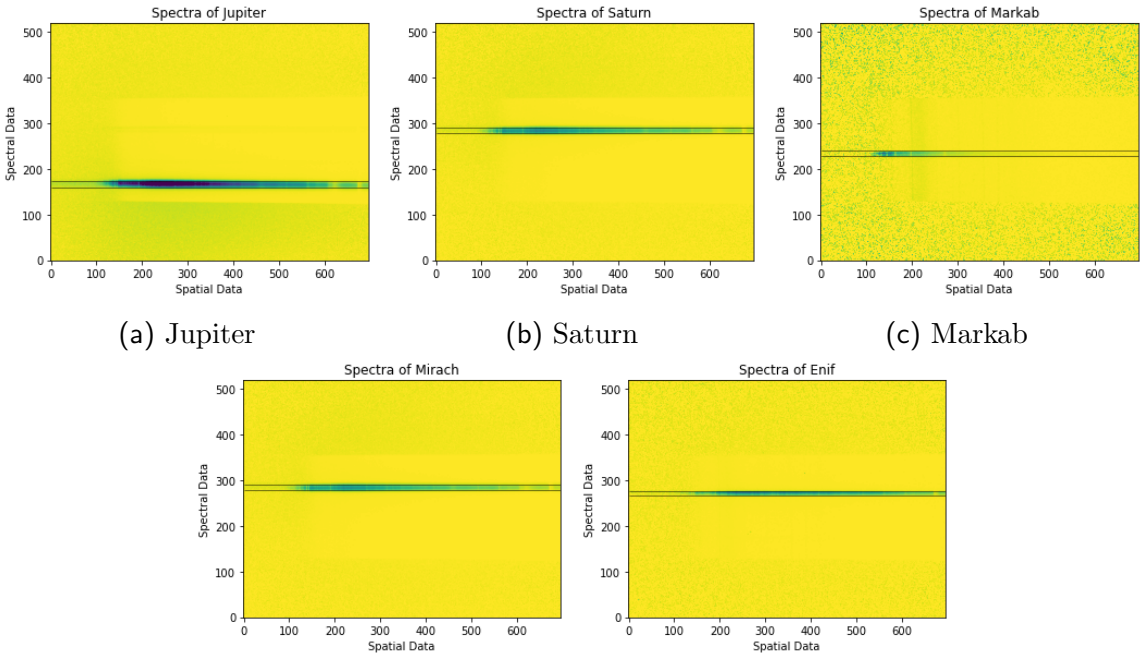
Once the fitted data was plotted against the pixel position, looking at figure 3 it can be observed that the cubic model provides the best fit solution. Using `scipy.stats.linregress()` (linear regression) was used to find the wavelength solution (see appendix F for details);

$$y = 0.9376x + 154.4098 \tag{1}$$

Since x which is the pixel position of the peak in the blackbody spectrum was determined to be 497, the wavelength solution in equation (1) could be solved to find the value of y which would be;

$$y = 0.9376 \times 497 + 154.4098$$

$$y = 620.40\ nm$$



(a) Jupiter (b) Saturn (c) Markab

(d) Mirach (e) Enif

Fig. 7: Spectral image of bright objects with the location of the spectral peaks highlighted with black lines.

Wien's displacement law states that the black-body radiation wavelengths are inversely proportional to the temperature. And this relation can be be used to find the temperature using the constant of proportionality b:

$$\lambda = \frac{b}{T} \tag{2}$$

$$T = \frac{b}{\lambda} \tag{3}$$

λ is calculated to be $\lambda = 620.40\ nm$ and the constant of proportionality is given to be $b = 2898\ \mu m.K$. Therefore using equation (3) the temperature can be calculated to be;

$$T = 4671.16\ K$$

(See appendix F for details)

2-D Spectral Data

Once the fitted data was plotted against the pixel position it can be observed from figure 5 that the cubic model provides the best fit solution. Using `scipy.stats.linregress()` (linear regression) was used to find the wavelength solution (see appendix F for details);

$$y = 2.5966x + 16212.721 \tag{4}$$

Since x which is the pixel position of the peak in the Fe II emission spectrum was determined to be 13, the wavelength solution in equation (4) could be solved to find the value of y which

would be;

$$y = 2.5966 \times 13 + 16212.721$$

$$y = 16246.47\ \mathring{A}$$

Doppler effect suggest that the change in wavelength due to Doppler shift is proportional to the velocity of the moving object and the intrinsic wavelength. And this relation can be be used to find the velocity of the object using the speed of light.

$$\Delta\lambda = \lambda_{int}\frac{v}{c} \tag{5}$$

Doppler shift in this case would be the difference between the given wavelength in the lab manual which is $\lambda_{int} = 16439.981\ \mathring{A}$ and the $\lambda_{obs} = 16246.477\ \mathring{A}$ which is the calculated wavelength from equation (4). Therefore;

$$\Delta\lambda = \lambda_{int} - \lambda_{obs}$$

And equation (5) can be rewritten as;

$$v = (\lambda_{int} - \lambda_{obs})\frac{c}{\lambda_{int}} \tag{6}$$

Using the speed of light as $c = 300000\ km/s$ the velocity can be calculated in equation (6) to be;

$$v = 3531\ km/s$$

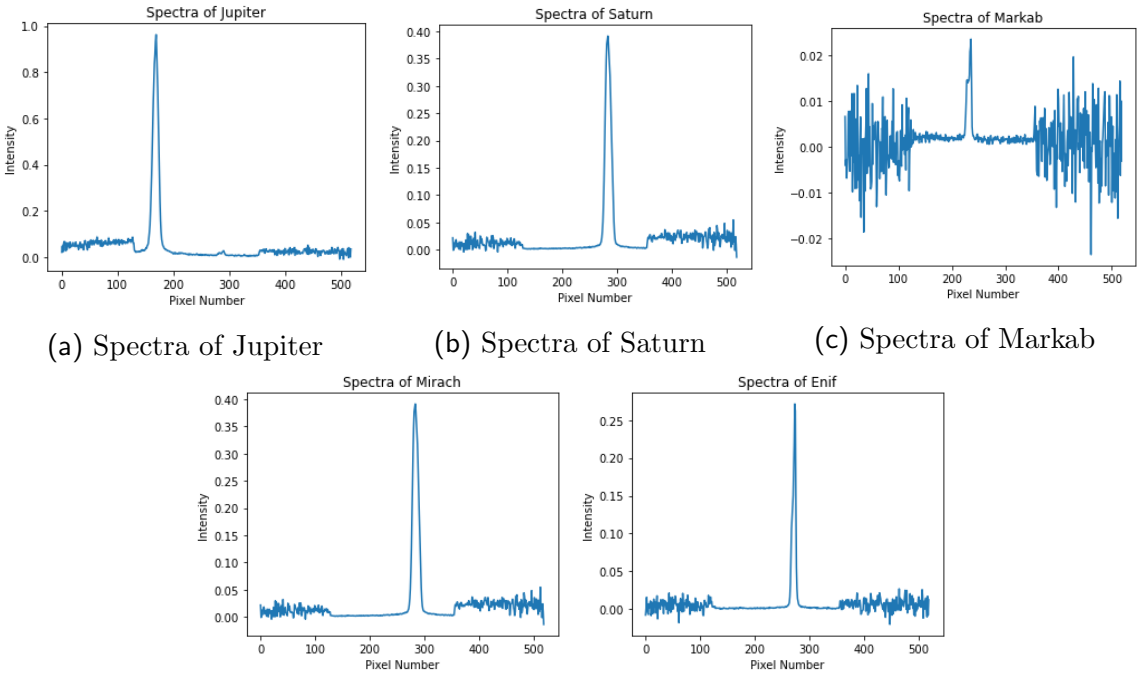
(See appendix F for details)

Astronomical Observations to Obtain Spectra

In figure 7 the position of the spectra was determined by highlighting the spectra with black lines. Using that information the array with the spectral data was extracted and the median of that data was taken using `numpy.median()` to obtain the peaks. once the peak data was obtained and stored in an array it was plotted against the pixel position to obtain the spectral plot in figure 8 (See appendix I for details).

Discussion

The calculated temperature and the velocity value in this report varied significantly from the reports of other members of Group G. There could be a few different reasons behind. for this part those discrepancies would be discussed. Firstly, the wavelength solution was obtained using curve-fitting was an approximation only. When figure 3 and figure 5 are observed it can be seen that the scatter plots and the regression line don't align. There are some residuals for both the OH line spectra and Neon line spectra. since the cubic fit is the closest to the most number of data points it was selected to form the wavelength solution. Using the Neon lamp data to find the temperature of the black body using cubic fit, the standard deviation error for the slope was 2.998×10^{-07} and the standard deviation error for y-intercept was 4.746×10^{-04} (See appendix I for details). Using the OH skylines data to



(a) Spectra of Jupiter (b) Spectra of Saturn (c) Spectra of Markab

(d) Spectra of Mirach (e) Spectra of Enif

Fig. 8: Spectra of bright object with the location of the spectral peaks highlighted with black lines.

find the velocity of the object emitting the Fe II gas using cubic fit, the standard deviation error for the slope was 2.353×10^{-05} and the standard deviation error for y-intercept was 8.851×10^{-03} (See appendix I for details).

The next part of the error is the most likely one is that the method used for finding the peaks (Full Width Half Max) might not have been accurate and as a result the peak location was estimated incorrectly. Furthermore, the curve fitting itself might have been wrong, as the python code for that was was take from the tutorials but not much changes to it were made (to accurately adapt to the requirements of this lab). And due to that reason it could be that the data obtained from the curve fitting is wrong.

The errors mentioned above are due to user error, but there could also be error in collecting the data, the equipment could have been calibrated incorrectly to account for additional noise or discrepancies.

Mechanical or equipment error also could be the reason being the incorrect data collection for part three of this lab, astronomical Observations to Obtain Spectra. Even though the biased, flats, and darks were taken to calibrate the image, it is likely that all the noise in the images could have been reduced, looking closely at figure 7, it can be seen that there are in fact some noise remaining even in the calibrated image. due to this the extracted spectra might not be full accurate representative of known spectra of that object. Furthermore the python code used to extract and plot the data might not have been correct as the spectra shows up as a peak and not a wave spectrum (Figure 8).

Conclusion

The purpose of this lab was to learn to read and analyze spectral data and images. Spectroscopy is an important tool when it comes to physical sciences. And in terms of astrophysics and astronomy it is one of the most important tool a scientist have because other that spectral and wave data astronomers do not have much to go on. Therefore the goal of this lab being; reading, analyzing, and finding physical attributes using spectral data. It can be concluded that the goal of this lab was accomplished.

Appendix A

Loading the given data into python arrays, opening and loading FITS files into workable arrays:

```
#load data and organize
#blackbody spectra
bbdat = np.loadtxt('Group_G_BB.dat', unpack=True)
pixel = np.arange(len(bbdat))
peak_int = np.max(bbdat)
#neon lamp data
ne_calib = np.loadtxt('Ne_calib.dat', unpack=True)
pixel1 = np.arange(len(ne_calib))

#loading the fits file for the OH and FeII sprecta
nifr_fits = fits.open('Near-Infrared.fits')
#converting the data
data = nifr_fits[0].data
#indexing the data to get rid of the 3rd dimension
data1 = data[0, :, :]
#transposing the data
data2 = np.transpose(data1)

#Loading the data from the astronomy observation
##JUPITER
jfit = "C:\\Users\\annta\\OneDrive\\Desktop\\School\\AST325\\GroupG\\JUPITER_10secs.fit"
jfit1 = fits.open(jfit)
jupiter = jfit1[0].data

##SATURN
sfit = "C:\\Users\\annta\\OneDrive\\Desktop\\School\\AST325\\GroupG\\SATURN_10secs.fit"
sfit1 = fits.open(sfit)
saturn = sfit1[0].data

##MARKAB
mafit = "C:\\Users\\annta\\OneDrive\\Desktop\\School\\AST325\\GroupG\\MARKAB_30secs.fit"
mafit1 = fits.open(mafit)
markab = mafit1[0].data

##MIRACH
mifit = "C:\\Users\\annta\\OneDrive\\Desktop\\School\\AST325\\GroupG\\MIRACH_30secs.fit"
mifit1 = fits.open(mifit)
mirach = mifit1[0].data

#ENIF
```

```
efit = "C:\\Users\\annta\\OneDrive\\Desktop\\School\\AST325\\GroupG\\ENIF_30secs.fit"
efit1 = fits.open(efit)
enif = efit1[0].data

#DARK1S
d1afit = "C:\\Users\\annta\\OneDrive\\Desktop\\School\\AST325\\GroupG\\DARK_1sec_01.fit"
d1afit1 = fits.open(d1afit)
d1a = d1afit1[0].data

d1bfit = "C:\\Users\\annta\\OneDrive\\Desktop\\School\\AST325\\GroupG\\DARK_1sec_02.fit"
d1bfit1 = fits.open(d1bfit)
d1b = d1bfit1[0].data

d1cfit = "C:\\Users\\annta\\OneDrive\\Desktop\\School\\AST325\\GroupG\\DARK_1sec_03.fit"
d1cfit1 = fits.open(d1cfit)
d1c = d1cfit1[0].data

#DARK10S
d10afit = "C:\\Users\\annta\\OneDrive\\Desktop\\School\\AST325\\GroupG\\DARK_10secs_01.fit"
d10afit1 = fits.open(d10afit)
d10a = d10afit1[0].data

d10bfit = "C:\\Users\\annta\\OneDrive\\Desktop\\School\\AST325\\GroupG\\DARK_10secs_02.fit"
d10bfit1 = fits.open(d10bfit)
d10b = d10bfit1[0].data

d10cfit = "C:\\Users\\annta\\OneDrive\\Desktop\\School\\AST325\\GroupG\\DARK_10secs_03.fit"
d10cfit1 = fits.open(d10cfit)
d10c = d10cfit1[0].data

#DARK30S
d30afit = "C:\\Users\\annta\\OneDrive\\Desktop\\School\\AST325\\GroupG\\DARK_30secs_01.fit"
d30afit1 = fits.open(d30afit)
d30a = d30afit1[0].data

d30bfit = "C:\\Users\\annta\\OneDrive\\Desktop\\School\\AST325\\GroupG\\DARK_30secs_02.fit"
d30bfit1 = fits.open(d30bfit)
d30b = d30bfit1[0].data

d30cfit = "C:\\Users\\annta\\OneDrive\\Desktop\\School\\AST325\\GroupG\\DARK_30secs_03.fit"
d30cfit1 = fits.open(d30cfit)
d30c = d30cfit1[0].data

#FLAT1S
f1afit = "C:\\Users\\annta\\OneDrive\\Desktop\\School\\AST325\\GroupG\\FLAT_1sec_01.fit"
```

```
f1afit1 = fits.open(f1afit)
f1a = f1afit1[0].data

f1bfit = "C:\\Users\\annta\\OneDrive\\Desktop\\School\\AST325\\GroupG\\FLAT_1sec_02.fit"
f1bfit1 = fits.open(f1bfit)
f1b = f1bfit1[0].data

f1cfit = "C:\\Users\\annta\\OneDrive\\Desktop\\School\\AST325\\GroupG\\FLAT_1sec_03.fit"
f1cfit1 = fits.open(f1cfit)
f1c = f1cfit1[0].data
```

Appendix B

Taking the indices of the OH spectrum, for the median OH lines and the Fe II emission lines:

```
#indexing the data again to take a slice of the spectrum
data_mat = data2[135:150, :]
# taking the median value of each row in the rectangle
wave_int = np.median(data_mat, axis = 0)
```

Appendix C

Using FMHW to find the peak heights and peak pixel location. code source: Tutorial 5 'Tutorial Spectral Centroiding and Error Analysis (Oct 17th)'

```

##FWHM for Neon spectra
peaks = [] #pixel value at peak
peaks_height = [] #signal value at peak
threshold = 74
#iterate over signal array, starting at index 2, end at len(signal)-2
for j in range(len(ne_calib)- 4):
    i = j+2
    #conditions that need to be met
    threshold_met = ne_calib[i]>threshold
    higher_than_neighbour = (ne_calib[i]>ne_calib[i-1])&(ne_calib[i]>ne_calib[i+1])
    higher_than_neighbour2 = (ne_calib[i]>ne_calib[i-2])&(ne_calib[i]>ne_calib[i+2])
    if threshold_met & higher_than_neighbour & higher_than_neighbour2:
        peaks.append(pixel1[i])
        peaks_height.append(ne_calib[i])
# plot the peaks we find
plt.figure(figsize = (20,8))
```

```
plt.title('Neon Spectrum with the Peak Positions', size = 25)
plt.plot(pixel1, ne_calib, color = 'orange',linestyle='-', label="Lamp Spectrum")
plt.ylabel("Intensity", size = 20)
plt.xlabel("Pixel Number", size = 20)
plt.xlim(40, 980)
plt.ylim(0, 1560)
for i in range(len(peaks)):
    plt.axvline(peaks[i], color='green', ls=':', label='Approx. Peak Location')
    plt.plot(peaks[i], peaks_height[i], 'rx')
plt.show()

##FWHM for OH and FeII spectra
peaks = [] #pixel value at peak
peaks_height = [] #signal value at peak
threshold = 74
#iterate over signal array, starting at index 2, end at len(signal)-2
for j in range(len(wave_int)- 4):
    i = j+2
    #conditions that need to be met
    threshold_met = wave_int[i]>threshold
    higher_than_neighbour = (wave_int[i]>wave_int[i-1])&(wave_int[i]>wave_int[i+1])
    higher_than_neighbour2 = (wave_int[i]>wave_int[i-2])&(wave_int[i]>wave_int[i+2])
    if threshold_met & higher_than_neighbour & higher_than_neighbour2:
        peaks.append(pixel2[i])
        peaks_height.append(wave_int[i])
#print(len(peaks))
# plot the peaks we find
plt.figure(figsize = (16,8))
plt.xlim(0,250)
plt.title('Spectrum of the OH Telluric Sky Lines with the Peak Positions', size = 25)
plt.plot(pixel2, wave_int, 'g-', alpha=0.5, label="Lamp Spectrum")
plt.ylabel("Intensity", size = 20)
plt.xlabel("Pixel Number", size = 20)
for i in range(len(peaks)):
    plt.axvline(peaks[i], color='orange', ls=':', label='Approx. Peak Location')
    plt.plot(peaks[i], peaks_height[i], 'rx', alpha=0.9)
plt.show()
```

Appendix D

Loading the given data into python arrays, opening and loading FITS files into workable arrays:

```
#Curve fitting of Ne
#Define the underlying data and calibration
pixel_number = np.arange(2048).astype('longlong')
calibration = 50*(pixel_number + 500) ** 0.3

data_x = x
data_y = y

#Define polynomial functions
def lin(x, a1, a2):
    y = a1*x + a2
    return y

def quad(x, a1, a2, a3):
    y = a1*x**2 + a2*x + a3
    return y

def cube(x, a1, a2, a3, a4):
    y = a1*x**3 + a2*x**2 + a3*x + a4
    return y

#Obtain best fit models with curvefit
curvefit_lin, cf_lin_pcov = curve_fit(lin, data_x, data_y)
curvefit_quad, cf_quad_pcov = curve_fit(quad, data_x, data_y)
curvefit_cube, cf_cube_pcov = curve_fit(cube, data_x, data_y)

#Now compute our best-fit models for all pixel values
lin_cal_curvefit =  curvefit_lin[0]*pixel_number + curvefit_lin[1]
quad_cal_curvefit =  curvefit_quad[0]*(pixel_number**2) + curvefit_quad[1]*pixel_number + curvefit_quad
cube_cal_curvefit =  (curvefit_cube[0]*np.power(pixel_number,3) +
                      curvefit_cube[1]*np.power(pixel_number,2) +
                      curvefit_cube[2]*pixel_number + curvefit_cube[3])

#Plot all of the best-fit models
plt.figure(figsize = (10,8))
plt.grid()
plt.plot(pixel_number, lin_cal_curvefit, '--', color = 'red',
         alpha = 0.5, label = 'Linear Calibration')
plt.plot(pixel_number, quad_cal_curvefit, '--', color = 'orange',
         alpha = 0.5, label = 'Quadratic Calibration')
plt.plot(pixel_number, cube_cal_curvefit, '--', color = 'green',
         alpha = 0.5, label = 'Cubic Calibration')

plt.plot(data_x, data_y, 'x', color = 'k')
```

```
plt.xlabel("Pixel Number", size = 20)
plt.ylabel("Wavelength (nm)", size = 20)
plt.title("Comparison of Best-fit Polynomial Models", size = 20)
plt.legend(prop={'size': 20})
plt.xlim(150,1000)
plt.ylim(500, 900)
plt.show()

#Curve fitting for OH
#Define the underlying data and calibration
pixel_number = np.arange(250).astype('longlong')
calibration = 50*(pixel_number + 500) ** 0.3

data_x = x
data_y = y

#Define polynomial functions, taken from tutorial session
def lin(x, a1, a2):
    y = a1*x + a2
    return y

def quad(x, a1, a2, a3):
    y = a1*x**2 + a2*x + a3
    return y

def cube(x, a1, a2, a3, a4):
    y = a1*x**3 + a2*x**2 + a3*x + a4
    return y

#Obtain best fit models with curvefit
curvefit_lin, cf_lin_pcov = curve_fit(lin, data_x, data_y)
curvefit_quad, cf_quad_pcov = curve_fit(quad, data_x, data_y)
curvefit_cube, cf_cube_pcov = curve_fit(cube, data_x, data_y)

#Now compute our best-fit models for all pixel values
lin_cal_curvefit =  curvefit_lin[0]*pixel_number + curvefit_lin[1]
quad_cal_curvefit =  curvefit_quad[0]*(pixel_number**2) + curvefit_quad[1]*pixel_number + curvefit_quad
cube_cal_curvefit =  (curvefit_cube[0]*np.power(pixel_number,3) +
                      curvefit_cube[1]*np.power(pixel_number,2) +
                      curvefit_cube[2]*pixel_number + curvefit_cube[3])

#Plot all of the best-fit models
plt.figure(figsize = (10,8))
plt.plot(pixel_number, lin_cal_curvefit, '--', color = 'red',
         alpha = 0.5, label = 'Linear Calibration')
```

```
plt.plot(pixel_number, quad_cal_curvefit, '--', color = 'orange',
         alpha = 0.5, label = 'Quadratic Calibration')
plt.plot(pixel_number, cube_cal_curvefit, '--', color = 'green',
         alpha = 0.5, label = 'Cubic Calibration')
plt.plot(data_x, data_y, 'x', color = 'k')
plt.grid()
plt.xlabel("Pixel Number", size = 20)
plt.ylabel("True Wavelength (Angstrom)", size = 20)
plt.title("Comparison of Best-fit Polynomial Models", size = 20)
plt.legend(prop={'size': 20})
plt.show()
```

Appendix E

Finding the peak of the Blackbody spectra and Fe II emission lines.

```
#peak of bb spectrum
peak1 = signal.find_peaks(bbdat)

#extract the array where the Fe2 line lies
feline =(data2[112,:])
fey = np.arange(len(feline))

plt.figure(figsize = (20,8))
plt.xlabel("Pixel Number", size = 20)
plt.ylabel("Pek Height", size = 20)
plt.title("Peak Location of the Fe II Emission Line", size = 25)
plt.vlines(13, 0, 250000, color='red', ls=':')
plt.plot(fey, feline, 'g-', alpha = 0.9)
plt.xlim(0,250)
plt.ylim(0, 240750)
plt.show()
#peak location observed from the plot to be 13
```

Appendix F

Using the wavelength solution to find the wavelength

```
#Blackbody
#linear regression
lin = stats.linregress(pixel_number, lin_cal_curvefit)
quad = stats.linregress(pixel_number, quad_cal_curvefit)
```

```
cube = stats.linregress(pixel_number, cube_cal_curvefit)

#assigning values to the wavelength eq
m = cube[0]
b = cube[1]
x = 497

#solving the eq
lam = m*x+b

#known values
lamda = lam* 0.001
b = 2898 #micrometer.K
#temp
T = b/lamda

#Fe II
#linear regression
lin = stats.linregress(pixel_number, lin_cal_curvefit)
quad = stats.linregress(pixel_number, quad_cal_curvefit)
cube = stats.linregress(pixel_number, cube_cal_curvefit)
#assigning values to the wavelength eq
m = cube[0]
b = cube[1]
x = 13

#solving the eq
lam = m*x+b

#known values
lamda = lam #angstom
lam_org = 1.6439981 * 10000 #angstom
c = 300000.0 #km/s

#Calculation
del_lam = lam_org-lamda
ratio = del_lam/lam_org
#Velocity
v = ratio*c
```

Appendix G

Calibrating Image using the darks, flats and biases.

```
#DARK1S MEDIAN
ds1 = np.stack([d1a, d1b, d1c])
dm1 = np.median(ds1, axis=0)

#DARK10S MEDIAN
ds10 = np.stack([d10a, d10b, d10c])
dm10 = np.median(ds10, axis=0)

#DARK30S MEDIAN
ds30 = np.stack([d30a, d30b, d30c])
dm30 = np.median(ds30, axis=0)

#FLATS MEDIAN
fs1 = np.stack([f1a, f1b, f1c])
fm1 = np.median(fs1, axis=0)

#DARK-FLAT
B = fm1-dm1

```

Appendix H

Reducing the images using calibrated images and plotting them with the spectra highlighted

```
#Reduction Jupiter
JA = jupiter - dm10
Jupiter_final = JA/B

#Reduction Saturn
SA = saturn - dm30
Saturn_final = SA/B

#Reduction Markab
MaA = markab - dm30
Markab_final = MaA/B

#Reduction Mirach
MiA = saturn - dm30
Mirach_final = MiA/B

#Reduction Enif
EA = enif - dm30
Enif_final = EA/B
```

```
20
21 #plotting the images
22 plt.xlabel('Spatial Data')
23 plt.ylabel('Spectral Data')
24 plt.title('Spectra of Jupiter')
25 plt.hlines(160, 0,700, colors = 'black', linewidth=.5)
26 plt.hlines(173, 0,700, colors = 'black', linewidth=.5)
27 plt.imshow(Jupiter_final,cmap='viridis_r', origin = 'lower', vmin=0, vmax=1)
28 plt.show()
29
30 plt.xlabel('Spatial Data')
31 plt.ylabel('Spectral Data')
32 plt.title('Spectra of Saturn')
33 plt.hlines(279, 0,700, colors = 'black', linewidth=.5)
34 plt.hlines(290, 0,700, colors = 'black', linewidth=.5)
35 plt.imshow(Saturn_final,cmap='viridis_r', origin = 'lower', vmin=0, vmax=.8)
36 plt.show()
37
38 plt.xlabel('Spatial Data')
39 plt.ylabel('Spectral Data')
40 plt.title('Spectra of Markab')
41 plt.hlines(240, 0,700, colors = 'black', linewidth=.5)
42 plt.hlines(228, 0,700, colors = 'black', linewidth=.5)
43 plt.imshow(Markab_final,cmap='viridis_r', origin = 'lower', vmin=0, vmax=0.09)
44 plt.show()
45
46 plt.xlabel('Spatial Data')
47 plt.ylabel('Spectral Data')
48 plt.title('Spectra of Mirach')
49 plt.hlines(279, 0,700, colors = 'black', linewidth=.5)
50 plt.hlines(290, 0,700, colors = 'black', linewidth=.5)
51 plt.imshow(Mirach_final,cmap='viridis_r', origin = 'lower', vmin=0, vmax=.9)
52 plt.show()
53
54 plt.xlabel('Spatial Data')
55 plt.ylabel('Spectral Data')
56 plt.title('Spectra of Enif')
57 plt.hlines(276, 0,700, colors = 'black', linewidth=.5)
58 plt.hlines(267, 0,700, colors = 'black', linewidth=.5)
59 plt.imshow(Enif_final,cmap='viridis_r', origin = 'lower', vmin=0, vmax=.4)
60 plt.show()
```

Appendix I

Extracting the spectral data and plotting it against pixle positions.

```
#Extracting the speactra
spec_jup1 = Jupiter_final[:, 160:173]
spec_sat1 = Saturn_final[:, 279:290]
spec_ma1 = Markab_final[:, 228:240]
spec_mi1 = Mirach_final[:, 279:290]
spec_en1 = Enif_final[:, 267:276]

#taking the median of the spectra
spec_jup = np.median(spec_jup1, axis = 1)
spec_sat = np.median(spec_sat1, axis = 1)
spec_ma = np.median(spec_ma1, axis = 1)
spec_mi = np.median(spec_mi1, axis = 1)
spec_en = np.median(spec_en1, axis = 1)

#plotthing the spectra
pixel = np.arange(len(spec_jup))

plt.figure(figsize=(5,4))
plt.xlabel('Pixel Number')
plt.ylabel('Intensity')
plt.title('Spectra of Jupiter')
plt.plot(pixel, spec_jup)
plt.show()

plt.figure(figsize=(5,4))
plt.xlabel('Pixel Number')
plt.ylabel('Intensity')
plt.title('Spectra of Saturn')
plt.plot(pixel, spec_sat)
plt.show()

plt.figure(figsize=(5,4))
plt.xlabel('Pixel Number')
plt.ylabel('Intensity')
plt.title('Spectra of Markab')
plt.plot(pixel, spec_ma)
plt.show()

plt.figure(figsize=(5,4))
plt.xlabel('Pixel Number')
plt.ylabel('Intensity')
```

```
plt.title('Spectra of Mirach')
plt.plot(pixel, spec_mi)
plt.show()

plt.figure(figsize=(5,4))
plt.xlabel('Pixel Number')
plt.ylabel('Intensity')
plt.title('Spectra of Enif')
plt.plot(pixel, spec_en)
plt.show()


```

Appendix J

Loading the given data into python arrays, opening and loading FITS files into workable arrays:

```

```

Appendix K

Loading the given data into python arrays, opening and loading FITS files into workable arrays:

```

```