Measuring AU Using Solar Observation

Anita Bahmanyar Ayushi Singh Morten Nyborg Stostad
Department of Astronomy and Astrophysics, University of Toronto

Written by: Anita Bahmanyar

anita.bahmanyar@mail.utoronto.ca

Student Number: 998909098

April 4 2014

Abstract

In this lab our goal is to measure the distance between Sun and the Earth (AU) in meters using simple geometry. The solar data was taken on March 18, 2014 using the 8 inch refractor telescope located on top of the Burton Tower at University of Toronto. Using Toshiba CCD and spectrometer, we measure the rotational velocity of the sun by computing the Doppler shift between spectra of the limbs of the sun. We used two methods of covariance and Fourier Transform in order to find the wavelength shift between two edges f the sun. Using our calculated rotational velocity of the sun, we estimated AU to be $1.5506 \times 10^8 \pm 3.656 \times 10^6$ km which makes the true value to be 96.176 % of the obtained value.

1 Introduction

Astronomical distances and scales are large enough that is gets bothersome to use usual distance units such as meters and kilometers. Therefore, astronomers defined a new unit for length measurements called Astronomical Unit(AU), which is defined as the distance between the Sun and the Earth. In this lab we will use simple geometry, circular motion and properties of light to measure this unit. In order to do so, we found the wavelength solution of the Charged Coupled Device(CCD) that related the pixel numbers to wavelengths as explained in section [...], obtained the solar spectra along its diameter and by taking advantage of cross-correlation method as discussed in

section [...], we figured out the pixel shift and therefore the Doppler Shift in the solar spectra with respect to the starting point(in order to determine the radial velocity of the Sun. From this, we can compute the radius of the sun by knowing the period of the Sun. Then finally we can compute AU using geometry as will be explained more in section [...]. The goal of this lab is to learn to find the wavelength solution for CCDs, cross-correlate data sets with each other and find the rotational velocity of the sun in order to find its radius and finally to measure AU using simple geometry as described later.

2 Observation and Data Acquisition

2.1 Neon and Mercury Lamps

We used HR1 spectrometer which consists of Toshiba 1×3652 CCD. Size of each pixel is $8\ \mu \times 200\ \mu$. The wavelength range of this spectrometer is 523-573 nm. We used Neon and Mercury lamps to find the wavelength solution for this spectrometer. We set the integration times such that the pixels are not saturated. Therefore, we used 200 ms and 400 ms integration time for neon lamp and 10 ms and 50 ms integration times for the mercury lamp. We also took data of dark counts with the same exposure times for each lamp.

2.2 Solar Observing

We went up to 16th floor of Burton Tower to use the 8-inch refractor telescope for our purpose. We mounted another telescope cap in front of the telescope aperture in order to reduce the amount of light coming into telescope from the Sun so that we do not hurt our eyes. Then we mounted the projection screen as it can be seen in Figure 1 so that we get the Sun on the screen. Next step was to line up the shadow of the finder scope until it became circle as in Figure 2 and then focused the the telescope to get a sharp image(We did this by looking at the sharpness of the sun spots). The next thing we need to consider is the temperature of the spectrometer since the spectral resolution and dispersion of the the spectograph or in general, the wavelength solution changes with temperature and for this mean we placed the spectrometer inside a box to keep the temperature approximately as it was in the astronomy lab, where we collected Neon and Mercury data to solve for the wavelength solution. The box consisted of a resistor that heats up the box. Then we connected the fibre coupler to collimate [....]

Figure 1: This figure shows the projection screen mounted on the telescope to get and the image of the Sun on the screen(indirect observing)

Figure 2: This figure shows the shadow of the finder scope as a circle which means it is lined up correctly. We also focused the telescope.

3 Analysis

3.1 Wavelength Solution

We calibrated the wavelength of the spectrometer since we need to figure out how wavelength is related to the pixel numbers to the wavelength. We used neon and mercury lamps for this purpose. We took 200, 400 and 1000 ms integration times for neon and 10 and 50 ms integration times of data for mercury as well as dark data sets for all of these integration times. We then subtracted dark from the data sets. Figure [....] shows the the intensity vs. pixel number of 10ms mercury and 200ms neon lamp graphs. We then found the peaks of the graph using centroid method as discussed in Appendix section. Then we compared these peaks with a reference peak wavelength for neon and mercury in the spectrometer wavelength range and plotted the wavelengths vs. pixel numbers in order to find the wavelength solution. We used linear least squares to fit the graph. This was done manually in Python as explained in Appendix, so it means we did not use the built-in package of Python. Fig. [...] shows the wavelength solution, which is $y = 0.014x + 524.61$, where y is the wavelength and x is the pixel number.

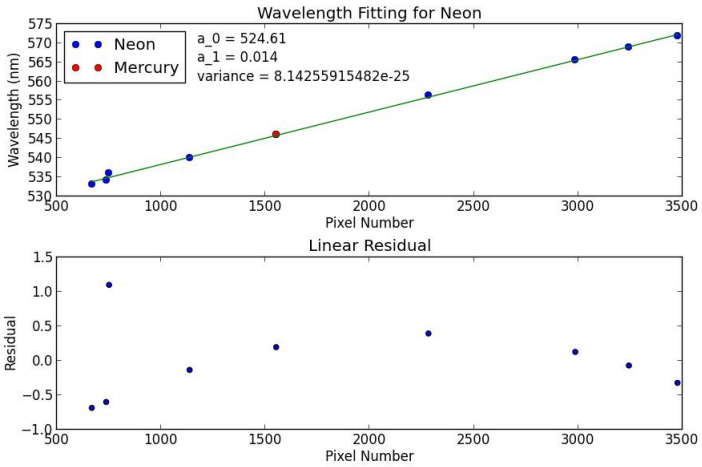

Figure 3: The upper image shows the wavelengths vs. pixel numbers of neon and mercury peaks as compared with their reference spectrum. The wavelength solution is written in the graph. The lower image shows the error in the data and as it is visible it is very small and the variance is also very small that is negligible.

Figures [...] and [...] show the intensity vs. wavelength for neon 200ms and mercury 10ms.

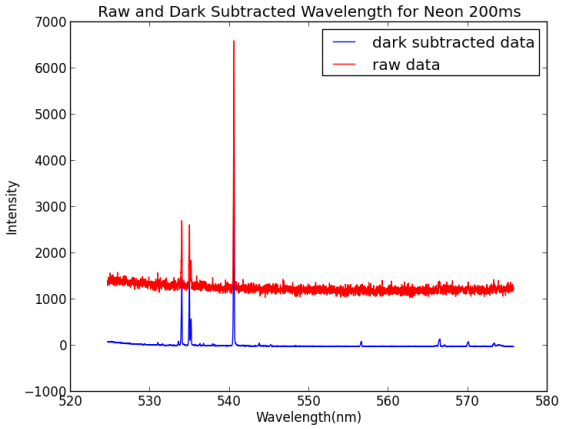

Figure 4: This figure shows the intensity vs. wavelength of neon 200ms. The raw data is very noisy unlike the mercury raw data.

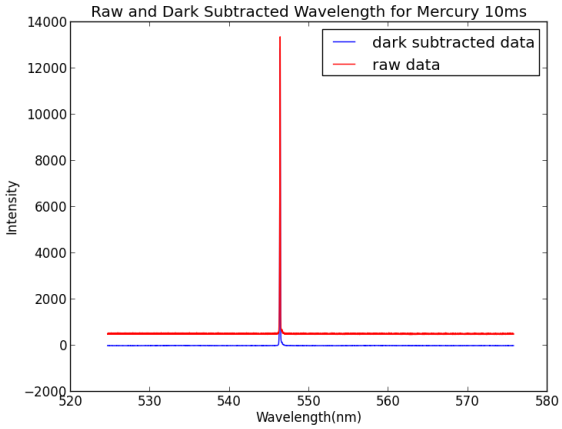

Figure 5: This figure shows the intensity vs. wavelength of mercury 10ms.

3.2 Solar Data

After finding the wavelength solution of the spectrometer, we should work with solar data. One of the solar data spectrums is plotted vs. wavelength in Fig. [...].

We collected 170ms and 200ms data sets on March 13, 2014 and three sets of data with 125ms integration time on March 18, 2014. We found the mean of each data file and plotted these mean values vs. time. The times where extracted from the file names as it will be discussed in details in Appendix. Then we considered the limb darkening which is given by equation [...]:

$$I = \frac{2}{5} I_0 \left(\frac{3}{5} + \sqrt{\frac{(t - t_0)^2}{\Delta t^2}}\right) \tag{1}$$

I_0 is the intensity of the center value, Δt is the difference in the time of the two start and end point data and t_0 is the time of the central data.

Table 1: Limb Darkening Parameters

	I_0	$t_{center}[s]$	Δt [s]
Calculated	5087.608	1718.079	64.693
Error	8.927e+02	1.355e-01	1.723e-01

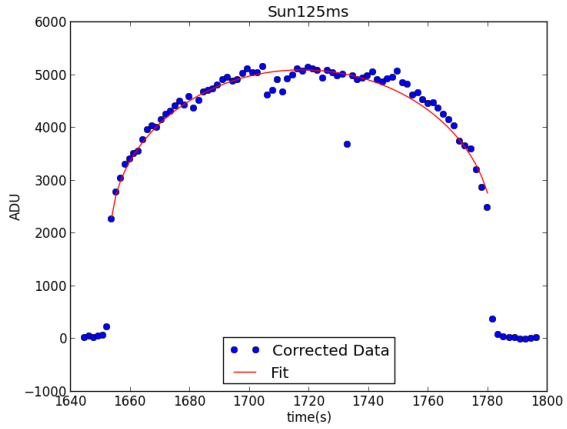

Figure 6: This figure shows the intensity vs. time of sun with integration time of 125ms. The blue dots show the averaged spectra across the sun and the red curve shows the fit to the data. There is one dot really off from the other points which has been removed from the data for correlations that come later in the sections.

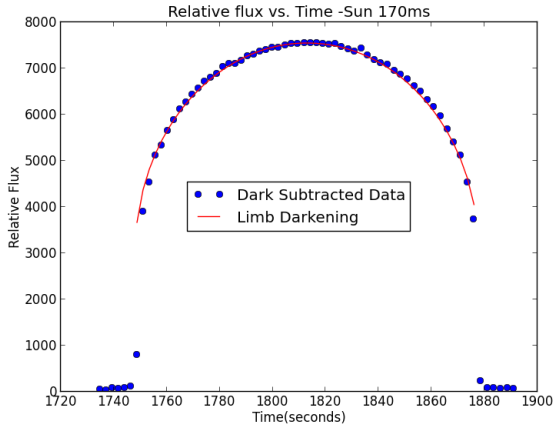

Figure 7: This figure shows the intensity vs. time of sun with integration time of 170ms. Similar to the left picture, the blue dots show the averaged spectra and the red curve is the fit to these data points. We did not remove any of the points from the data.

3.3 Flattening and Smoothing

We wanted to make the solar spectrum to look flat rather than having a blackbody shape since it is easier to see the features such as the shift and also the absorption lines. In order to do so, I tried different methods which some worked and some did not. Here I will go over the methods. First of all, I tried fitting a blackbody curve to it with considering the surface temperature of the sun (5777 K), but since the spectrum is covering only a small range of the wavelength, the blackbody values did not change much and therefore division of the spectrum by the black body basically did nothing. The second method I used was using smoothing Python code found online called savitzky_golay.py . This code generated a function that had a similar overall shape of our data. We should note that we should use large window sizes so that by smoothing we do not loose spectral features. Then by dividing our data by this smoother function we can get rid of the slope in the spectrum and still preserve the spectral features. However, this generated huge amount of noise at the end of the spectrum since we did not have much information on the end parts of the spectrum so the noise(uncertainty) on the end points got big. Therefore, I used a third method using gaussian functions, and it worked better than the others in that it flattened the spectrum and reduced the amount of noise at the end parts. The code explanation can be found in Appendix.

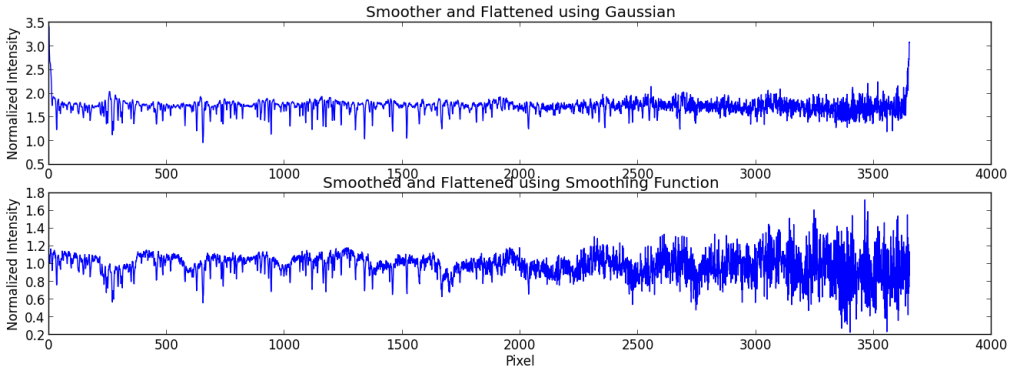

Figure 8: This figure shows flattened and smoothed spectra of the sun with two different methods. *Up* : Smoothed and flattened using the gaussian function. *Down* : Smoothed and flattened using savitzky golay.py code. As it is obvious using gaussian function have better results in that its end parts are less noisy and it is because in the second method we had less information about the end parts so it became more noisy at these parts.

3.4 Velocity Resolution of Spectrometer

Each spectrometer has a specific velocity resolution which corresponds to its wavelength and therefore, its pixel resolving power. We used Mercury peak in the wavelength range of 523nm-573nm (since t only has one peak in this wavelength range) in order to compute this velocity resolution. The method we used is that we fitted a gaussian function on the peak and calculated the Full Width half Maximum(FWHM) which is given by Eq. [...]:

$$FWHM = 2\sqrt{n \ln 2}\sigma \tag{2}$$

where σ is the standard deviation of the data. We found the σ for the gaussian fit to be 2.816 $\pm$7.0532e-05 pixels which corresponds to FWHM of 3.315 $\pm$ 7.0532e-05pixels and by Eq. [...] this gives the velocity resolution of about 29.885 km/s, which is close to the 33 km/s value given in the lecture slides. The percentage difference is 9.44 % which is good enough. Figure [...] shows the mercury 50ms peak and the gaussian fit on it.

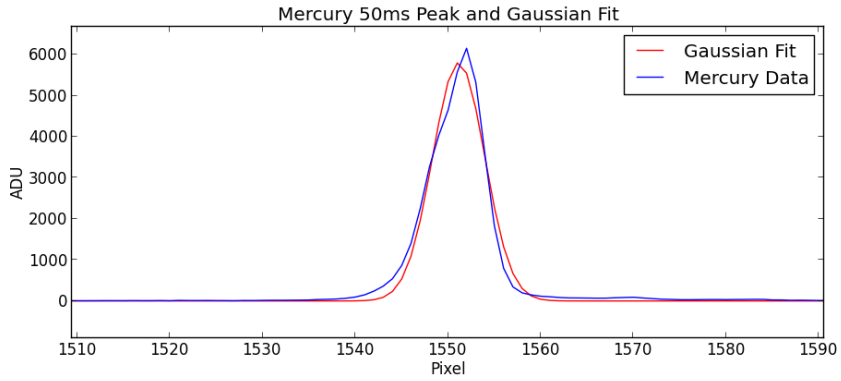

Figure 9: This figure shows the mercury 50ms peak in blue and the gaussian fit on to of it in red. We then computed the σ of the gaussian function and used it to calculate FWHM of it in order to find the velocity resolution of the spectrometer using Doppler Shift Equation.

4 Determining Doppler Shift

As we were collecting the solar data, the sun was rotating. We need to account this rotation since it affects the wavelength of the light that we observe. As the sun rotates, one edge is turning toward us, making the light to appear blue-shifted(light has shorter wavelength) and the other edge is turning away from us, making it to appear red-shifted(light has longer wavelength). This is known as Doppler Shift and it is given by

equation [...]:

$$\frac{\Delta\lambda}{\lambda} = \frac{\Delta v}{c} = \frac{2v_{rad}}{c} \tag{3}$$

$$v_{rad} = \frac{c\Delta\lambda}{2\lambda} \tag{4}$$

Where Δv is the change in velocity, $\Delta\lambda$ is the change in wavelength given some reference wavelength, λ and c is the speed of light in vacuum. The spectrometer we used has a velocity resolution of 33 km/s. Figure [...] shows the spectrums of the two edges of the sun. As it can be seen in the image, even the two furthest spectrums are very close to each other.

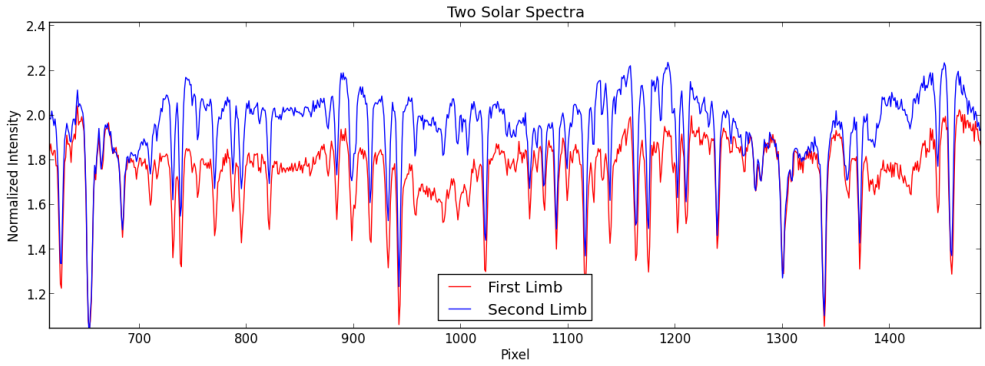

Figure 10: *Top* : Zoomed view of two spectra of the two edges of the sun over-plotted. As it can be seen, the two spectra are very close to each other. *Bottom* : The two spectra are zoomed so the features are better shown.

4.1 Cross-Correlation

Cross-correlation is a method used in this lab in order to show how similar the different spectra are with respect to a reference spectrum of the sun. Covariance is a statistical parameter that shows how correlated two variables are. Covariance can be computed by Eq. [...]:

$$s_j^2 = \frac{1}{N-1}\Sigma(x_i - \bar{x})(y_i - \bar{y}) \tag{5}$$

Where N is the size of the data set, $\bar{x}$ and $\bar{y}$ are the mean values of the two data sets. In theory, the maximum value of the covariance shows where the two data sets are most similar. In order to find the shift in the pixel values of the two data sets, we considered an array from -10 to 10 and iterated it by 1 pixel each time, and computed the covariance. Then we plotted covariance vs. the array we formed. Then we should

find the centroid of the cross-correlation in order to find the pixel shift. This can be computed by Eq. [...]:

$$< j >= \frac{\Sigma_j j X_j}{\Sigma_j X_j} \tag{6}$$

4.2 Fourier Transform

A Fourier Series is where any function expressed as the sum of sines and cosines. Fourier Transform also transforms a function that depends on time to a function that depends on the frequency. The way we correlated our function is given by Eq. [...].

$$shjhscv \tag{7}$$

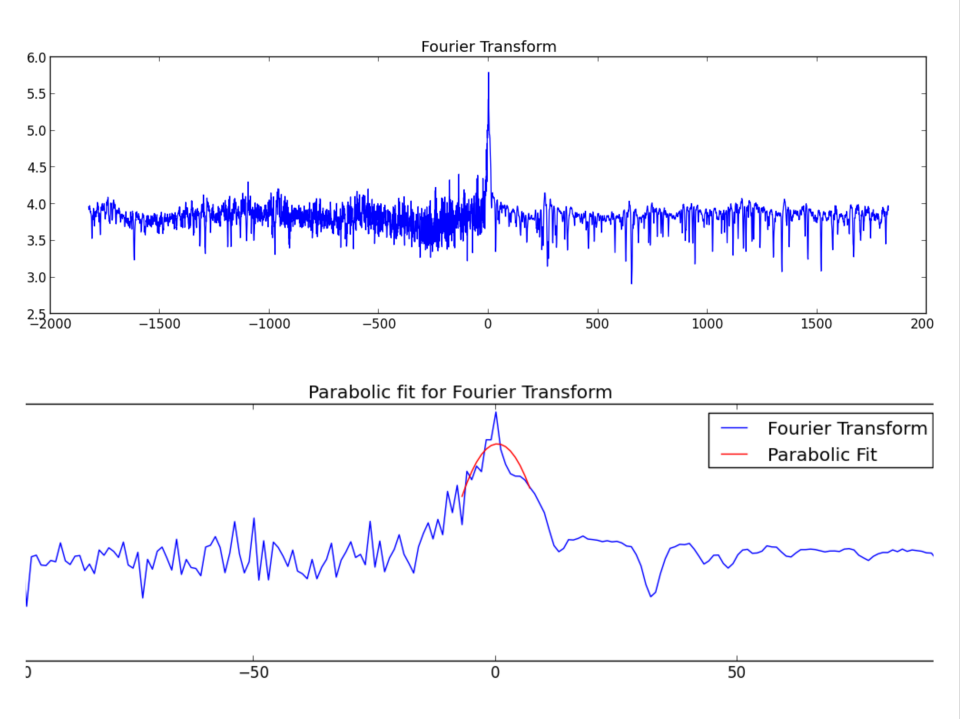

Figure 11: Fourier Transform of data. Up :

We had to roll the Fourier transformed data so that we get the peak at the center. We rolled it by half the size of the data. We then fitted a parabolic function ax^2+bx+c and the values we got from are as follows: a=0.0131, b=0.0078 and c=5.3726. For finding the pixel shift, we need to find the peak of the parabola which also shows the maximum point of the Fourier transformed data. We can find this point by setting derivative of parabolic equation to zero; therefore, the point would be at $x = \frac{-b}{2a}$. Table [...] shows these values along with their errors.

Table 2: Parabolic Fitting to Fourier Transformed Data Parameters

	a	b	c
Calculated	0.0131	0.0078	5.3726
Error	$6.156\ 10^{-5}$	$1.097\ 10^{-3}$	$4.938\ 10^{-2}$

4.3 Window Function

In applying cross-correlation we shift the spectra relative to one another. In doing so, we beginning and the tail of the spectra would not match and line up with other spectra since our data set is finite. Since our data is discrete (we cannot integrate from [-inf] to [inf], causes the energy from the true frequency to "leak" into adjacent frequencies. Leakage is one of the most common digital signal processing errors and it cannot be eliminated completely, it can only be minimized. Therefore, we could use the window functioning method that weights the beginning and end values of the data set to zero and it also minimizes edge effects and leakage of frequencies and amplitude of the signal. In this lab we used the Hanning window function which is given by Eq. [...] below, where M is the number of points we want the window function to return:

$$w(n) = 0.5(1 - \cos(\frac{2\pi n}{M-1})) \tag{8}$$

It is used for smoothing and it is also known as apodization which means that it "removes the foot", i.e.smoothing discontinuities at the beginning and end of the sampled signal) or tapering function.

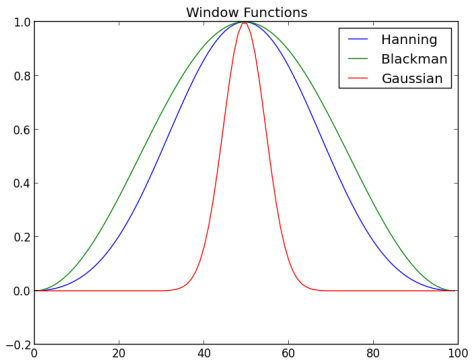

Figure 12: Different window functions shown. The blue curve is the Hanning window function.

5 Discussion

Using the methods discussed in section 4, we computed the pixel shift from the two end spectra using the one that starts earlier as the reference spectrum. Then we used the wavelength solution we found for the spectrometer to convert this pixel shift into wavelength shift so that we could use Doppler Shift equation given by Eq. [..] to compute the radial velocity of the Sun. The pixel shift found is 0.408 which corresponds to approximately 0.0064 nm shift in the wavelength. The radial velocity then would be 1.839 km/s .

Figures [..] and [...] show the spectrum of the beginning of the sun and the plot of the window function applied to the same data, respectively.

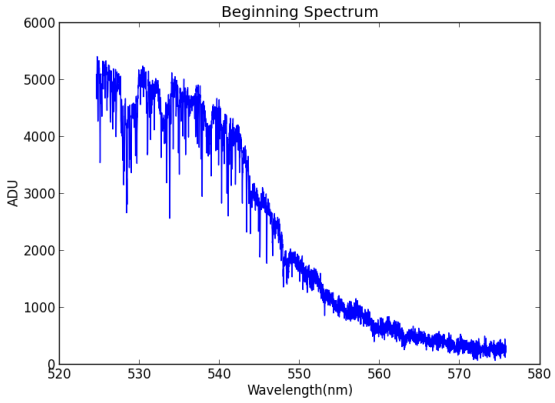

Figure 13: This figure shows the intensity vs. wavelength of beginning of the solar spectrum. This is dark subtracted and divided by flat.

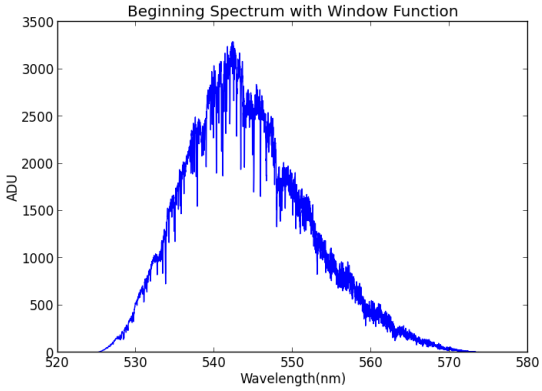

Figure 14: This figure shows the beginning spectrum of the sun which Hanning window function is applied on.

After applying the window function to the spectra, I recalculated the pixel shift and it was 0.508, so there is a small change in the pixel shift when window function was applied.

In order to find AU, we also need to find the angular size and radius of the Sun that are given by Eq. [..] and Eq. [...], respectively.

$$\frac{\Delta t}{24h} = \frac{\theta}{360^o} \tag{9}$$

$$v_{real} = \frac{v_{rad}}{\cos(\eta)\cos(\xi)} \tag{10}$$

$$R_{sun} = \frac{v_{real} T}{2\pi} \tag{11}$$

Using simple geometry, we can finally find the distance D, which is the distance between the Sun and the Earth. Fig. [...] shows the sketch of the geometry and the math is shown in Eq. [..].

$$D = \frac{R_{sun}}{\tan(\theta/2)} \tag{12}$$

Here, Δt is the time between the two limbs of the sun which is twice the Δt we used to plot the limb darkening, therefore, Δt is 126.138 s, which is equal to 2.1023 minutes. Using Eq. [...] θ has a value of 0.5334 $\pm$ 0.0007 degrees. The true angular size of the sun is approximately 0.5331 degrees as given on JPL HORIZONS [... reference] for the day of our observation, so the percentage difference with my value and the value on JPL is 0.06 %, which is very accurate.

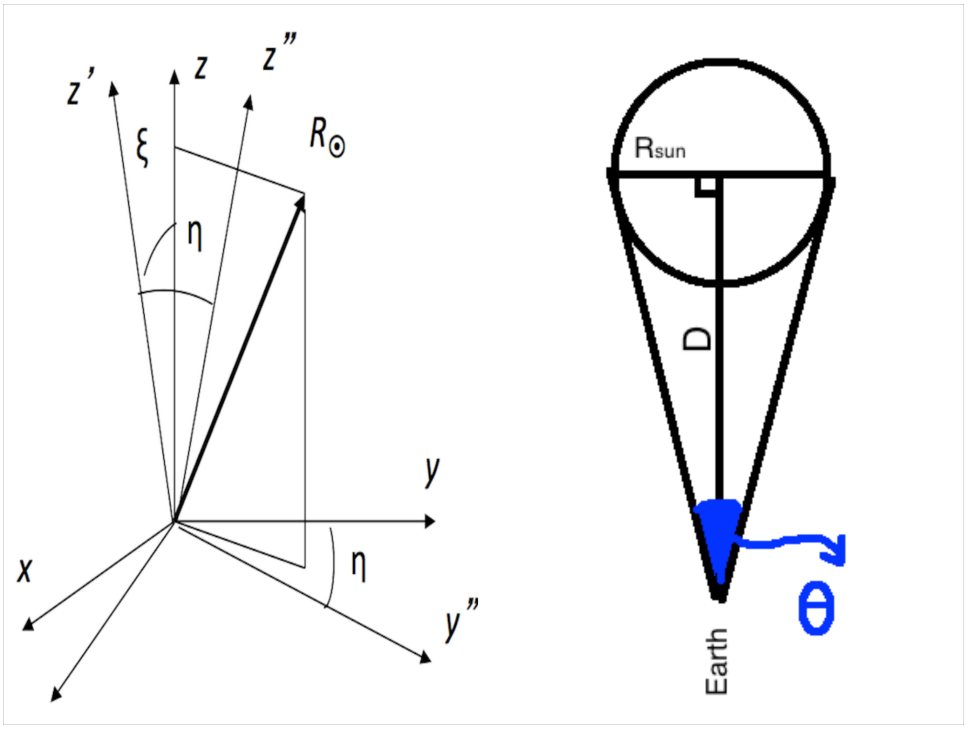

Figure 15: *Left* :Shows coordinate transformation for solar axis since it is not perpendicular to the ecliptic, meaning that the center of the sun as viewed from Earth is not b=0. Therefor, we should rotate it about y-axis by the angle ξ and about x-axis by the angle η .*Right* :This is a sketch of the simple geometry for calculating the AU using the computed values of angular size of the sun and its radius.

Table 3: Values computed by two methods of covariance and Fourier Transforms and their errors

	pixel shift	$\Delta\lambda$ (nm)	λ (nm)	v_{rad} (km/s)	v_{real} km/s
Calculated(Covariance Method)	0.408	0.0064	522.7807	1.839	2.044
Error			-	0.0432	0.0480
Calculated(Fourier Method)	0.408	0.0064	522.7807	1.839	2.044
Error			-	0.0432	0.0480

Table 4: Values computed by two methods of covariance and Fourier Transforms and their errors

	θ (degrees)	T (days)	η(deg.)	ξ(deg.)	R_{sun} (km)	AU (km)
Calculated(Covariance)	0.5256	25.38	335.0652	-7.09	713424.68	155547492.76
Error	0.0007	-	-	-	16767.97	3655909.1
Calculated(Fourier)	0.5256	25.38	335.0652	-7.09	713424.68	155547492.76
Error	0.0007	-	-	-	16767.97	3655909.1

5.1 Error Analysis

In order for the results to be valuable we need to show the error in the values measured as well. Here in this section I will explain how I got the errors and what are some sources of the errors that could be improved to make better results.

6 Conclusion

We can use simple geometry and some data analysis to measure the distance between Sun and the Earth.

7 References

1-http://www.aseq-instruments.com/HR1.html
2-http://en.wikipedia.org/wiki/Angular_diameter
3-The lecture slides

8 Appendix

We first discuss how we found the peaks of the spectrum of neon and mercury using centroid method. Below is the code I used to find the centroids. It goes through all of

the pixels and checks the intensity value of each pixel with its neighbour pixel intensity value. If it is higher than both its left and right pixels it points as a peak.

```
for i in range(0,3652,1):
    if values[i+1]>values[i]:
        i+=1
        if values[i+1]<values[i]:
            if values[i]>70:
                Pixellist.append(i)
                centroidintensity = 0.5*(values[i]+values[i+1])
                centroidintensitylist.append(centroidintensity)
```

Here is how fitting wavlength vs. pixel values work using linear sqaures.

```
ma =np.array([[np.sum(Pixels**2),np.sum(Pixels)],[np.sum(Pixels),len(Pixels)]])
mc =np.array([[np.sum(Pixels*Wavelengths)],[np.sum(Wavelengths)]])  #matrices
mai = np.linalg.inv(ma)
md = np.dot(mai,mc)
```

Next, we will discuss how the times of the solar observations were extracted from the file names.

```
times=[]
for k in range(0,66,1):
    names=np.loadtxt('sun170ms-filenames.txt',dtype='str')
    minutes=names[k][17:19]      # taking minute parts of the file name
    minutes=float(minutes)*60    # converting str to int and converting to seconds
    seconds=names[k][19:21]     # taking second parts of the file name
    seconds=float(seconds)        # converting str to int
    miliseconds=names[k][21:24] # taking mili second parts of the file name
    miliseconds=float(miliseconds)/1000 #converting str to int and to seconds
    time=minutes+seconds+miliseconds
    times.append(time)
t_center=times[5]+(times[59]-times[5])
Delta_t=(times[59]-times[5])/2
I_o=7550  #central intensity
I=I_o*((2./5)+((3./5)*(np.sqrt(1-(((times-t_center)**2)/(Delta_t**2))))))
```