Afrah Ali

Physics 139

Lab 7

```
# Importing

import numpy as np
import matplotlib.pyplot as plt
import math
import scipy as sp
import scipy.stats as stats
from astropy.io import fits
from astropy.modeling import models, fitting
```

PART ONE

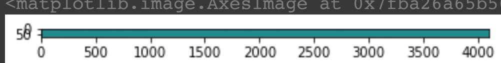
```
# Opening and analyzing file

slit = fits.open("slit.m1217A.085B.fits")
slit.info()
last = slit[1].data
flux = last[0]['FLUX']
wave = last[0]['WAVE']
```

```
Filename: slit.m1217A.085B.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU       4   ()
  1  slit          1 BinTableHDU    101   1R x 3C   [286720E, 286720E, 286720E]
```

```
# The lab instructions specify that the slitlet is 70 pixels long and 4096
# pixels wide
```

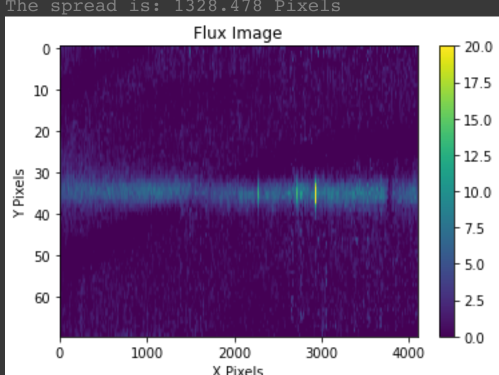
```
plt.imshow(flux)
```

```
<matplotlib.image.AxesImage at 0x7fba26a65b50>
```




```
plt.imshow(flux,aspect='auto',vmin=0,vmax=20)
plt.colorbar()
plt.title("Flux Image")
plt.xlabel("X Pixels")
plt.ylabel("Y Pixels")

spread = wave.max() - wave.min()
print("The spread is: " + str(spread) + " Pixels")
```

```
The spread is: 1328.478 Pixels
```

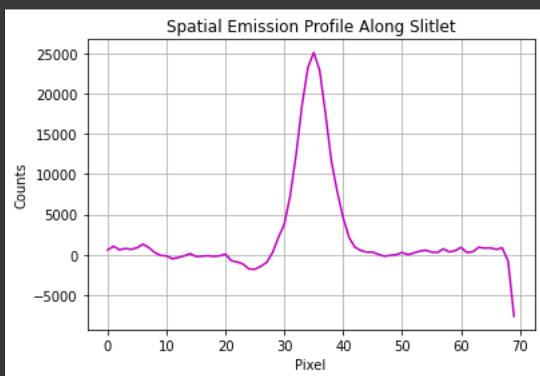


PART TWO

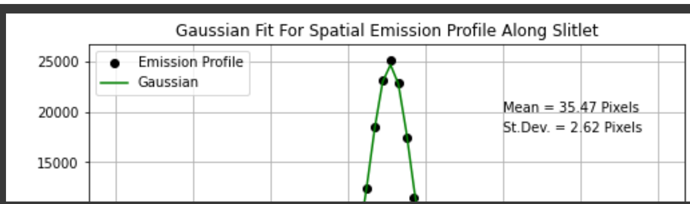
```
# This function sums the 2D flux array along the dispersion direction and
# creates a 1D array of total flux as a function of spatial position along the
# slitlet.

def func(arg):
  sums = []
  for i in range(len(arg)):
    sums.append(arg[i].sum())

  return sums
```

```
sum_obs = func(flux)
```

```
x = np.linspace(0,100)
plt.plot(sum_obs,color='m',linewidth=1.5)
plt.xlabel("Pixel")
plt.ylabel("Counts")
plt.title("Spatial Emission Profile Along Slitlet")
plt.grid()
```




```
# Using code from lab 6 to fit a Gaussian to the emission profile curve

# Initializing
xdata = np.linspace(0,70,70)
ydata = sum_obs
g_init = models.Gaussian1D(amplitude=np.max(ydata), mean=np.mean(xdata), stddev=np.std(xdata)) + models.Const1D(amplitude=1.)
fit_g = fitting.LevMarLSQFitter()
g = fit_g(g_init, xdata, ydata)

# Plotting
plt.figure(figsize=(8,5))
plt.plot(xdata, ydata, 'ko', label="Emission Profile")
plt.plot(xdata, g(xdata), color='green',label='Gaussian', linewidth=1.5)
plt.xlabel('Pixel')
plt.ylabel('Counts')
plt.grid()
plt.title("Gaussian Fit For Spatial Emission Profile Along Slitlet")
plt.legend(loc=2)

# Calculating Values
mean_obs = round(g[0].mean.value,2)
std_obs = round(g[0].stddev.value,2)

# Adding Values to Graph
plt.annotate('Mean = '+ str(mean_obs) + " Pixels",xy = (50,20000))
plt.annotate('St.Dev. = '+ str(std_obs) + " Pixels",xy = (50,18000))

plt.show()
```



PART THREE

```
# The function below extracts the object spectrum and creates a new 1D spectrum
# of relative object flux as a function of pixel position (and thus wavelength.


def extract(obs,wave,mean,width):
    # Initializing
    mean = int(mean)
    width = int(width)
    SSBU = obs[mean-(width):mean+(width)]
    SSBB = wave[mean-(width):mean+(width)]
    start = mean - width

    # Creating holders
    flx = []
    wvlh  = wave[mean]

    hold1 = []
    for k in range(len(SSBU)):
        corect = []
        fint = np.interp(last['WAVE'][0,start + k-1], last['WAVE'][0,start+k], last['FLUX'][0,start+k])

        for f in range(len(SSBU[0])):
            corect.append(SSBU[k][f]+fint[f])
        hold1.append(corect)

    for p in range(len(SSBU[0])):
        helper = []
        for k in range(len(SSBU)):
            helper.append(hold1[k][p])
        flx.append(np.sum(helper))

    return wvlh,flx
```

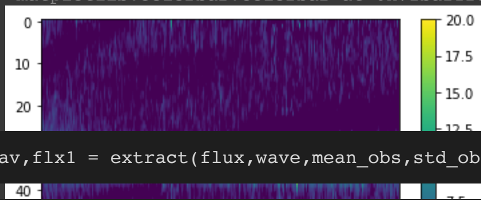
```
lst = []

for p in range(69):
  if p<70:
    f = np.interp(last['WAVE'][0,p], last['WAVE'][0,p+1], last['FLUX'][0,p+1])

    antherlst = []
    for i in range(4096):
      antherlst.append(f[i] + flux[p][i])

    lst.append(antherlst)
```

```
plt.imshow(lst)
```

```
<matplotlib.image.AxesImage at 0x7fba24c41220>
```


```
plt.imshow(lst,aspect='auto',vmin=0,vmax=20)
plt.colorbar()
```

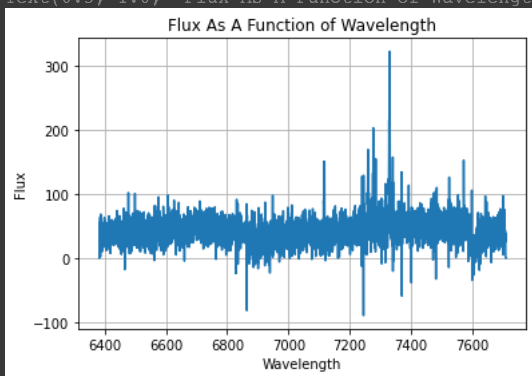
```
<matplotlib.colorbar.Colorbar at 0x7fba212c9220>
```



```
wav,flx1 = extract(flux,wave,mean_obs,std_obs)
```

```
plt.plot(wav,flx1)
plt.xlabel('Wavelength')
plt.ylabel('Flux')
plt.grid()
plt.title("Flux As A Function of Wavelength - 1")
```

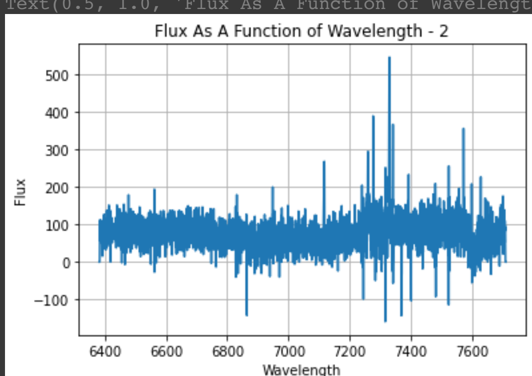
```
Text(0.5, 1.0, 'Flux As A Function of Wavelength')
```




```
# Initializing variables to cancel noise from and plot

wav,flx2 = extract(flux, wave, mean_obs, 2.5*std_obs)
wav,flx4 = extract(flux, wave, mean_obs, 4*std_obs)
```

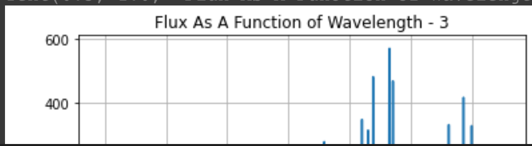
```
plt.plot(wav,flx2)
plt.xlabel('Wavelength')
plt.ylabel('Flux')
plt.grid()
plt.title("Flux As A Function of Wavelength - 2")
```

```
Text(0.5, 1.0, 'Flux As A Function of Wavelength - 2')
```




```
plt.plot(wav,flx4)
plt.xlabel('Wavelength')
plt.ylabel('Flux')
plt.grid()
plt.title("Flux As A Function of Wavelength - 3")
```

```
Text(0.5, 1.0, 'Flux As A Function of Wavelength - 3')
```

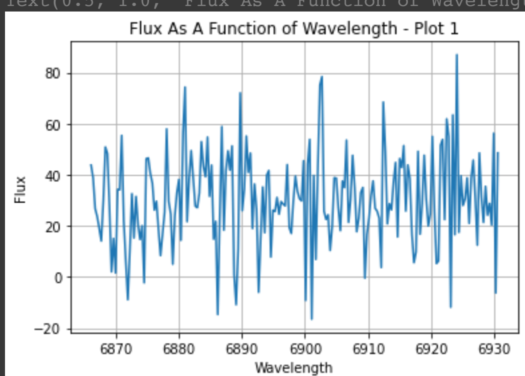



```
def func2(flux, x1, x2):
  q = flux[x1:x2]
  m = np.mean(q)
  std = np.std(q)
  m_over_s = m/std

  print("Mean: "+str(m)+", Standard Deviation: "+str(std))
  print("Signal due to Noise Ratio: "+str(m_over_s))

  return plt.plot(wav[x1:x2],q)
```

```
# The following plots all pertain to regions that are at 200 pixels in length
# and that are relatively free of sky line residuals. Signal-to-noise ratio is
# measured for each of these plots.
```

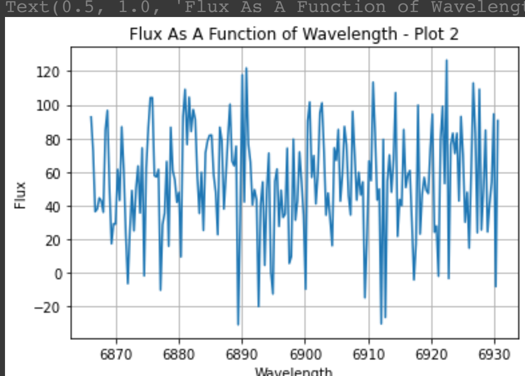
```
func2(flx1,1500,1700)
plt.xlabel("Wavelength")
plt.ylabel("Flux")
plt.grid()
plt.title("Flux As A Function of Wavelength - Plot 1")
```

```
Mean: 30.335264513113188, Standard Deviation: 17.94281333719827
Signal due to Noise Ratio: 1.6906637740149377
Text(0.5, 1.0, 'Flux As A Function of Wavelength - Plot 1')
```




```
func2(flx2,1500,1700)
plt.xlabel("Wavelength")
plt.ylabel("Flux")
plt.grid()
plt.title("Flux As A Function of Wavelength - Plot 2")
```

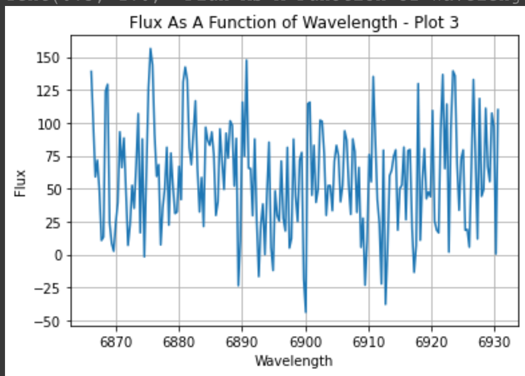
```
Mean: 54.29973742103926, Standard Deviation: 31.757332080744053
Signal due to Noise Ratio: 1.7098330956448233
Text(0.5, 1.0, 'Flux As A Function of Wavelength - Plot 2')
```




```
func2(flx4,1500,1700)
plt.xlabel("Wavelength")
plt.ylabel("Flux")
```

```
plt.grid()
plt.title("Flux As A Function of Wavelength - Plot 3")
```

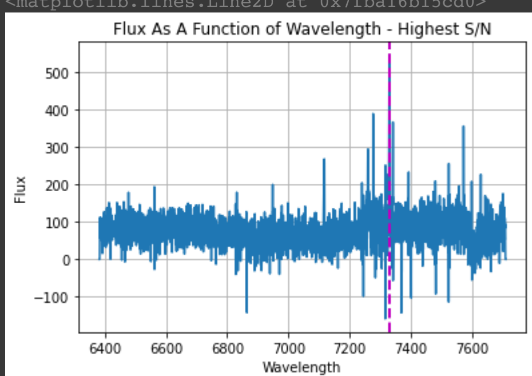
```
Mean: 58.85709843884358, Standard Deviation: 40.99374751998668
Signal due to Noise Ratio: 1.4357579386990065
Text(0.5, 1.0, 'Flux As A Function of Wavelength - Plot 3')
```




```
# Using the extraction window that yields the highest S/N, display the 1D
# spectrum as a function of wavelength. Are there any notable features in this
# spectrum?
```

```
plt.plot(wav,flx2)
plt.xlabel("Wavelength")
plt.ylabel("Flux")
plt.grid()
plt.title("Flux As A Function of Wavelength - Highest S/N")
plt.axvline(7330,color="m",linewidth=2, linestyle='dashed')
```

```
<matplotlib.lines.Line2D at 0x7fba16b15cd0>
```



The highest peak of this spectrum is denoted by the dashed magenta line. The spectrum is homogenous in shape and quite horizontal, but it has several peaks between the wavelengths 7200 and 7600.

PART FOUR

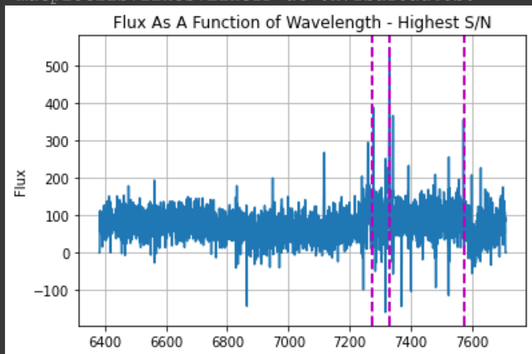
```
spread = (wav[-1] - wav[0])/4096
print(spread)
```

```
0.32387733459472656
```

```
# Finding three strongest emission lines
```

```
plt.plot(wav,flx2)
plt.xlabel("Wavelength")
plt.ylabel("Flux")
plt.grid()
plt.title("Flux As A Function of Wavelength - Highest S/N")
plt.axvline(7330,color="m",linewidth=2, linestyle='dashed')
plt.axvline(7274,color="m",linewidth=2, linestyle='dashed')
plt.axvline(7572,color="m",linewidth=2, linestyle='dashed')
```

```
<matplotlib.lines.Line2D at 0x7fba16ad48b0>
```

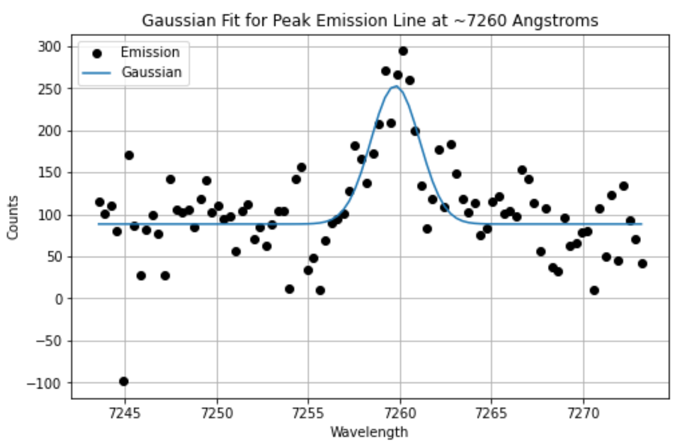



```
# To correspond pixel number to the specified wavelength, subtract a value of
# 6328 from each wavelength, then multiply the resulting value by the spread
# calculated previously.

peak1 = flx2[2662:2754] #7245 - 7275
peak2 = flx2[2862:2955] #7310 - 7340
peak3 = flx2[3634:3727] #7560 - 7590
```

Please refer to the second Google Colab file for the rest of part 4 of this lab.

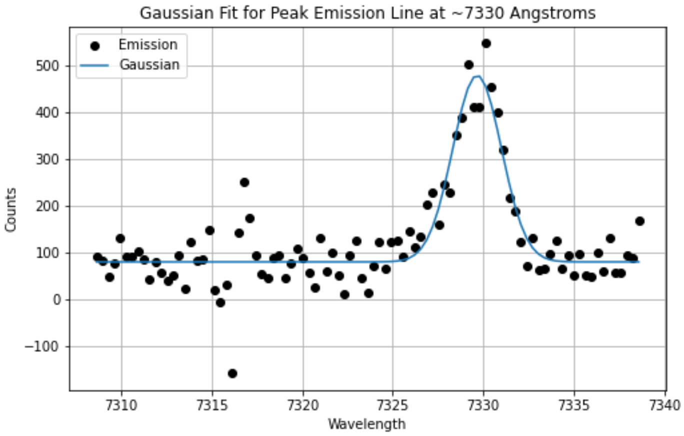
1s completed at 1:30 PM

```
xdata1 = wav[2662:2754]
ydata1 = peak1
g_init1 = models.Gaussian1D(amplitude=np.max(ydata1), mean=7260., stddev=1.) + models.Const1D(amplitude=1.)
fit_g1 = fitting.LevMarLSQFitter()
g1 = fit_g1(g_init1, xdata1, ydata1)
plt.figure(figsize=(8,5))
plt.plot(xdata1, ydata1,'ko', label="Emission")
plt.plot(xdata1, g1(xdata1), label='Gaussian')
plt.xlabel('Wavelength')
plt.grid()
plt.ylabel('Counts')
plt.legend(loc=2)
plt.title("Gaussian Fit for Peak Emission Line at ~7260 Angstroms")

plt.show()
print(g1)
```



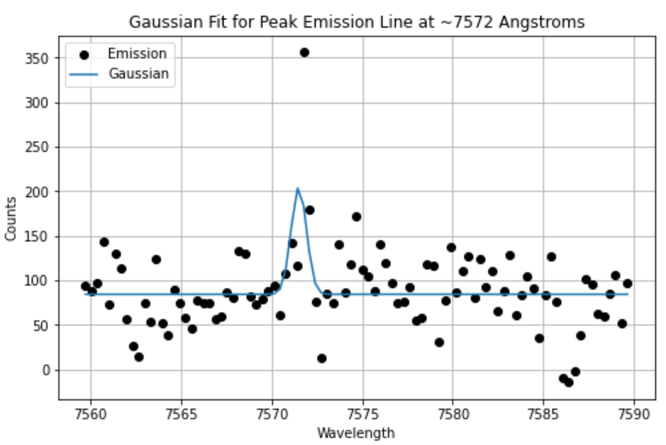

```
Model: CompoundModel
Inputs: ('x',)
Outputs: ('y',)
Model set size: 1
Expression: [0] + [1]
Components:
    [0]: <Gaussian1D(amplitude=164.53055648, mean=7259.75721613, stddev=1.327076

    [1]: <Const1D(amplitude=88.58866103)>
Parameters:
      amplitude_0          mean_0            stddev_0          amplitude_1
    ---------------- ----------------- ------------------ -----------------
    164.530556482689 7259.757216134146 1.3270762516403123 88.58866102799493
```

```
xdata2 = wav[2862:2955]
ydata2 = peak2
g_init2 = models.Gaussian1D(amplitude=np.max(ydata2), mean=7333., stddev=7.) + models.Const1D(amplitude=50.)
fit_g2 = fitting.LevMarLSQFitter()
g2 = fit_g2(g_init2, xdata2, ydata2)
plt.figure(figsize=(8,5))
plt.plot(xdata2, ydata2,'ko', label="Emission")
plt.plot(xdata2, g2(xdata2), label='Gaussian')
plt.xlabel('Wavelength')
plt.grid()
plt.ylabel('Counts')
plt.legend(loc=2)
plt.title("Gaussian Fit for Peak Emission Line at ~7330 Angstroms")

plt.show()
print(g2)
```




```
Model: CompoundModel
Inputs: ('x',)
Outputs: ('y',)
```

```
xdata3 = wav[3634:3727]
ydata3 = peak3
g_init3 = models.Gaussian1D(amplitude=np.max(ydata3), mean=7572., stddev=1.) + models.Const1D(amplitude=1.)
fit_g3 = fitting.LevMarLSQFitter()
g3 = fit_g3(g_init3, xdata3, ydata3)
plt.figure(figsize=(8,5))
plt.plot(xdata3, ydata3,'ko', label="Emission")
plt.plot(xdata3, g3(xdata3), label='Gaussian')
plt.xlabel('Wavelength')
plt.ylabel('Counts')
plt.legend(loc=2)
plt.grid()
plt.title("Gaussian Fit for Peak Emission Line at ~7572 Angstroms")

plt.show()
print(g3)
```




```
Model: CompoundModel
Inputs: ('x',)
Outputs: ('y',)
Model set size: 1
Expression: [0] + [1]
Components:
    [0]: <Gaussian1D(amplitude=120.59928914, mean=7571.51122944, stddev=0.42374557)>

    [1]: <Const1D(amplitude=84.37882329)>
Parameters:
       amplitude_0          mean_0            stddev_0          amplitude_1
    ----------------- ----------------- ------------------ -----------------
    120.5992891387782 7571.511229437897 0.4237455685304033 84.37882328688077
```

```
fit = [g1,g2,g3]
values = []
for i in range(3):
    values.append([fit[i][1].amplitude.value,fit[i][0].amplitude.value,fit[i][0].mean.value,fit[i][0].stddev.value])
```

```
print (values)
```

```
[[88.58866102799493, 164.530556482689, 7259.757216134146, 1.3270762516403123], [79.34207851400707, 398.3814527245603, 7329.6
```

H_Beta = 4861 A Corresponds to 7269 Peak

O3 = 4959 A Corresponds to 7330 Peak

O3 = 5007 A Corresponds to 7572 Peak

```
h = 4861.35
o1 = 4958.9
o2 = 5006.9
tru_wave = [h,o1,o2]
rdst = []
for i in range(3):
    rdst.append((fit[i][0].mean.value - tru_wave[i])/tru_wave[i])
print (rdst)
```

```
[0.49336238208196187, 0.4780837351848728, 0.5122153886512407]
```

The three measured redshifts are all within 0.04 of each other. The resultant spread in velocity is 9E6 Meters/Second.

```
print (fit[1][0].amplitude.value/fit[2][0].amplitude.value)
```

```
3.3033482665567586
```

The calculated amplitudes for the O3 lines align with the expected 1:3 ratio.

✓ 0s completed at 1:43 PM