Extra project

Yue Dong

January 22, 2024

1 The Spectrum of the star-forming region (NGC 3256 SF1)

The following is the spectrum plotted from the txt file.

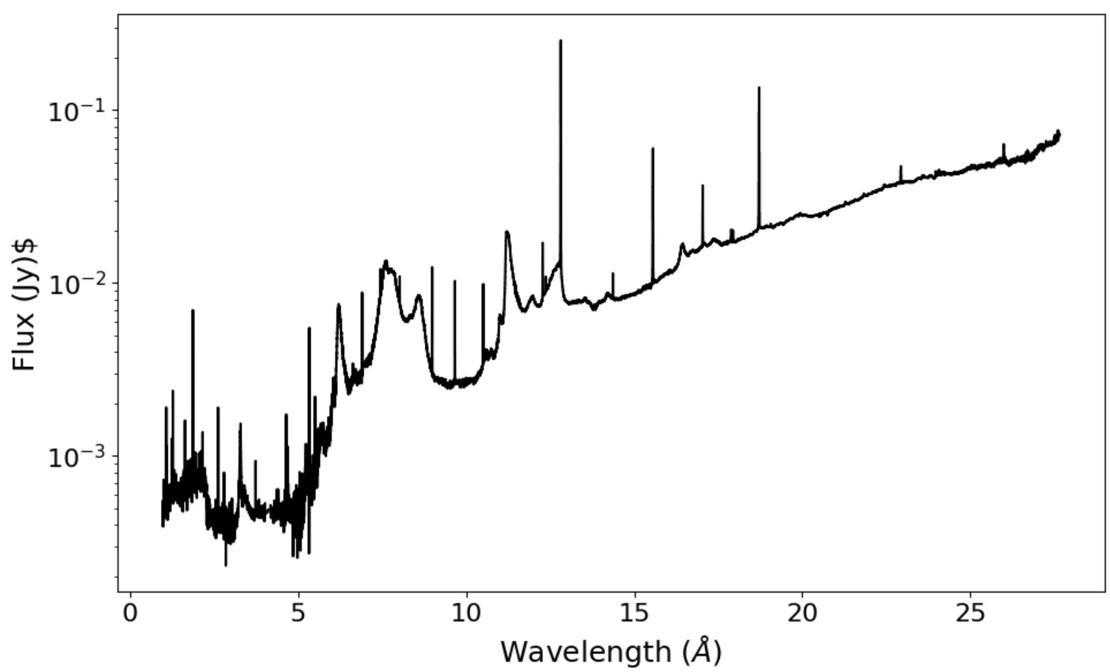

Figure 1: The Spectrum of NGC 3256 SF1

2 The Spectrum of the subBack_Level3_ch1-long_s3d.fits

Figure 2 is the spectrum plotted by Python from the fits file. A screenshot of the spectrum line from Qfitsview(Figure 3)is shown here as a comparison. The difference is the axis and scale, cuz I am not sure about the units of the subBackLevel3ch1-longs3d.fit raw data.

To analyze in detail the peak around $\lambda = 650$ (here I am referring to the unscaled axis in Figure 2, hence not using any units just to refer to the relative position on the graph). By zooming into the region of $\lambda = 639$ to $\lambda = 660$, and fitting the peak using a Gaussian, the fitted model is presented as Figure 4:

Then the flux is calculated by integrating the Gaussian where for this case, flux = 91141.89483885483(still in this "not that correct scale"). Python code for fitting and integrating is shown below.

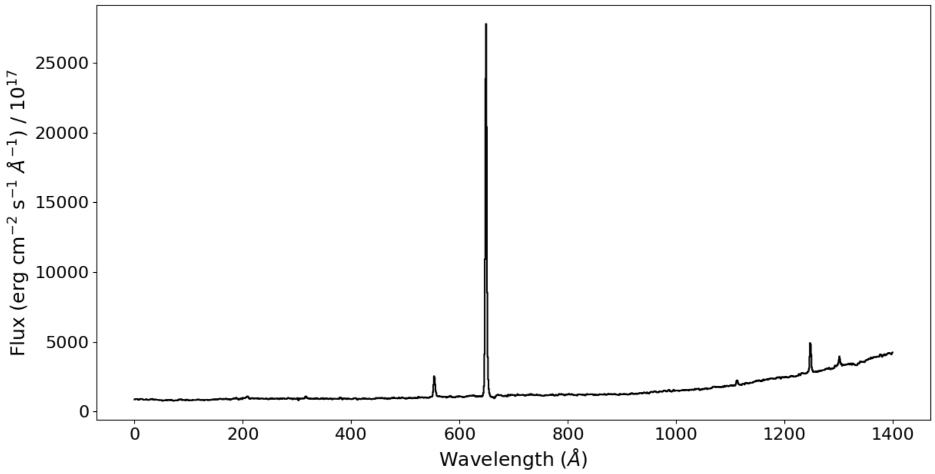

Figure 2: The Spectrum of Ch1_long

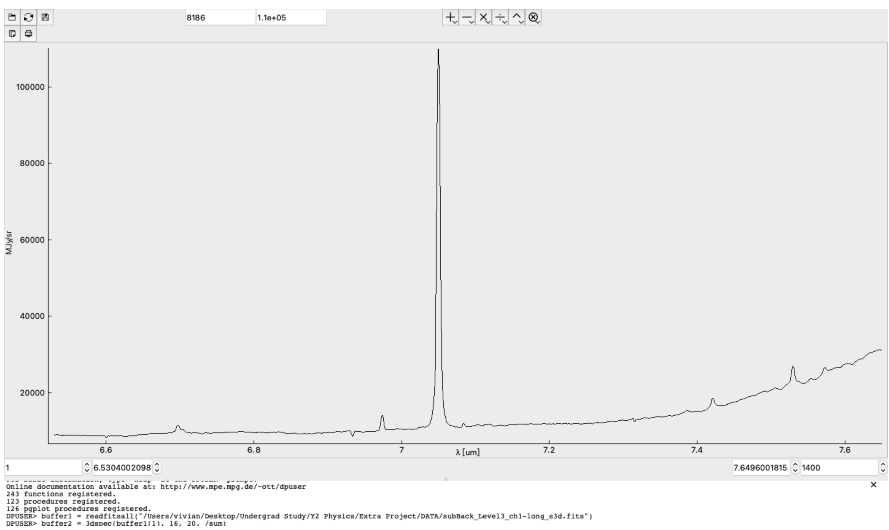

Figure 3: The Spectrum of Ch1_long from Qfitsview

Python script is shown here:

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

path = '/Users/vivian/Desktop/Undergrad Study/Y2 Physics/Extra Project/DATA/spectrum_subBack_Level3_ch1-long_s3d'
hdul = fits.open(path)
hdul.info()
hdr = fits.getheader(path) # gets the header of the file
print(hdr)
# initial value of the data
data = hdul[0].data
flux_max = max(data)
print(flux_max)

x = np.arange(0, 1400, 1)

# plot the spectrum
fig = plt.figure(figsize=(14,7))
```

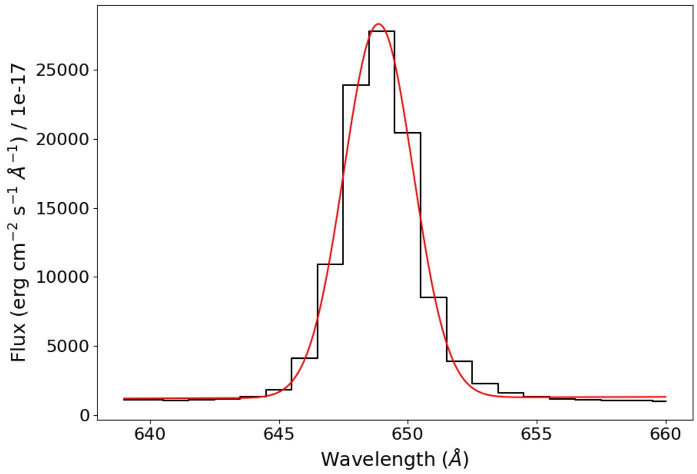

Figure 4: Fitted using Gaussian, parameters: peak = 27067.2667, mean = 648.867264, width = 1.34333310; errors: peak_err = 338. 146801, mean_err = 0.019069, width_err = 0.02035

```
plt.step(x, data, where='mid', color='k')
plt.xlabel(r"Wavelength ($\AA$)", fontsize=18)
plt.ylabel(r"Flux (erg cm$^{-2}$ s$^{-1}$ $\AA^{-1}$) / $10^{17}$", fontsize=18)
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.show()

hdr = fits.getheader(path) # gets the header of the file
# print(hdr)

# fitting emission lines and calculte line fluxes
# zoom into the specific region
hb_sliced_index = np.where(np.logical_and(x >= 639, x<= 660))
x_hb = x[hb_sliced_index]
flux_hb = data[hb_sliced_index]

fig = plt.figure(figsize=(10,7))
plt.step(x_hb, flux_hb, where='mid', color='k')
plt.xlabel(r"Wavelength ($\AA$)", fontsize=18)
plt.ylabel(r"Flux (erg cm$^{-2}$ s$^{-1}$ $\AA^{-1}$) / 1e-17", fontsize=18)
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.show()

# use AstroPy --initial estimations for the Gaussian parameters
cont = models.Polynomial1D(1)

g1 = models.Gaussian1D(amplitude=25000, mean=640, stddev=5)

# define the total model to fit the continuum and emission line
g_total = g1 + cont

# choose a fitter to fit our model
# AstroPy's LevMarLSQFitter
fit_g = fitting.LevMarLSQFitter()
# fit the model to the data
g = fit_g(g_total, x_hb, flux_hb, maxiter = 1000)

# define an array of wavelength values across the wavelength range
x_g = np.linspace(np.min(x_hb), np.max(x_hb), 10000)
plt.figure(figsize=(10, 7))
plt.step(x_hb, flux_hb, where='mid', color='k') # plot the data
plt.plot(x_g, g(x_g), color='red')
plt.xlabel(r"Wavelength ($\AA$)", fontsize=18)
```

```
plt.ylabel(r"Flux (erg cm$^{-2}$ s$^{-1}$ $\AA^{-1}$) / 1e-17", fontsize=18)
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.show()

# call the parameters of the final fit
print(g.parameters)
peak, mean, width, m, c = g.parameters
# [ 2.70672667e+04  6.48867264e+02  1.34333310e+00 -2.35511977e+03 5.55677147e+00]

# standard diviations
fit_errs = np.sqrt(np.diag(fit_g.fit_info['param_cov']))
peak_err, mean_err, width_err, m_err, c_err = fit_errs
print(fit_errs)
# [3.38146801e+02 1.90692865e-02 2.03539010e-02 9.16940355e+03 1.41100530e+01]

# create a function for calculating line flux and the associated uncertainty
def line_flux(peak, width, peak_err, width_err):
    flux_value = peak * width * np.sqrt(2 * np.pi)
    flux_err = flux_value * np.sqrt((peak_err / peak) ** 2 + (width_err/width) **2 )
    return flux_value, flux_err

flux_value, flux_err =line_flux(peak, width, peak_err, width_err)
print(flux_value)
# 91141.89483885483
```

3 Using Qfitsview to fit the spectrum

Figure 5 is fitted using the Gaussian/Lorentzian(Lorentzian not used here) to fit the spectrum in Qfitsview.

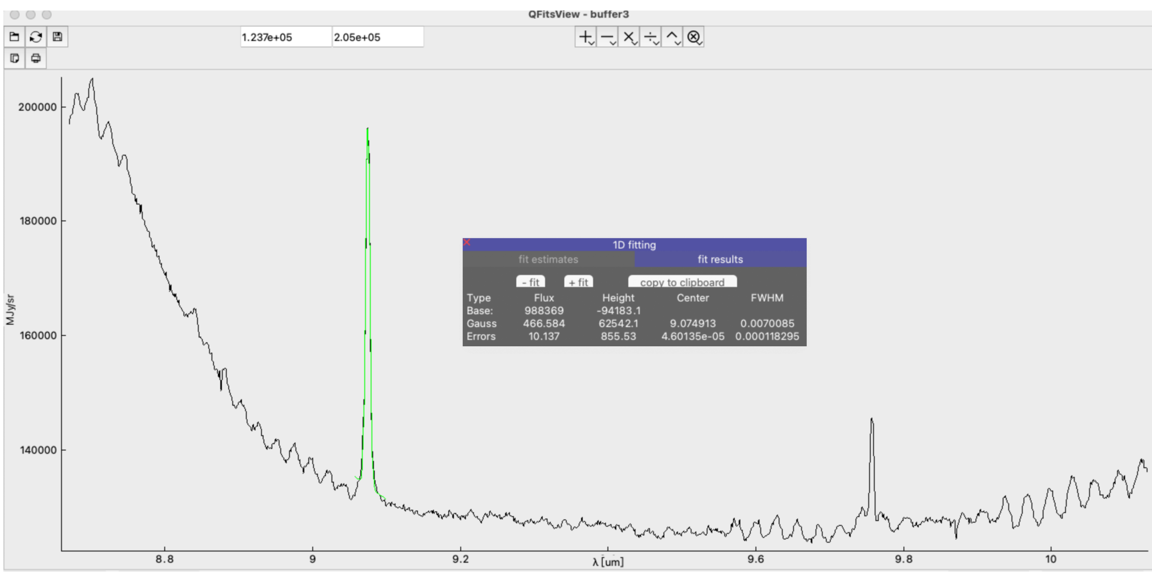

Figure 5: Qfitsview fitting a random spectrum