section_4_4

February 13, 2024

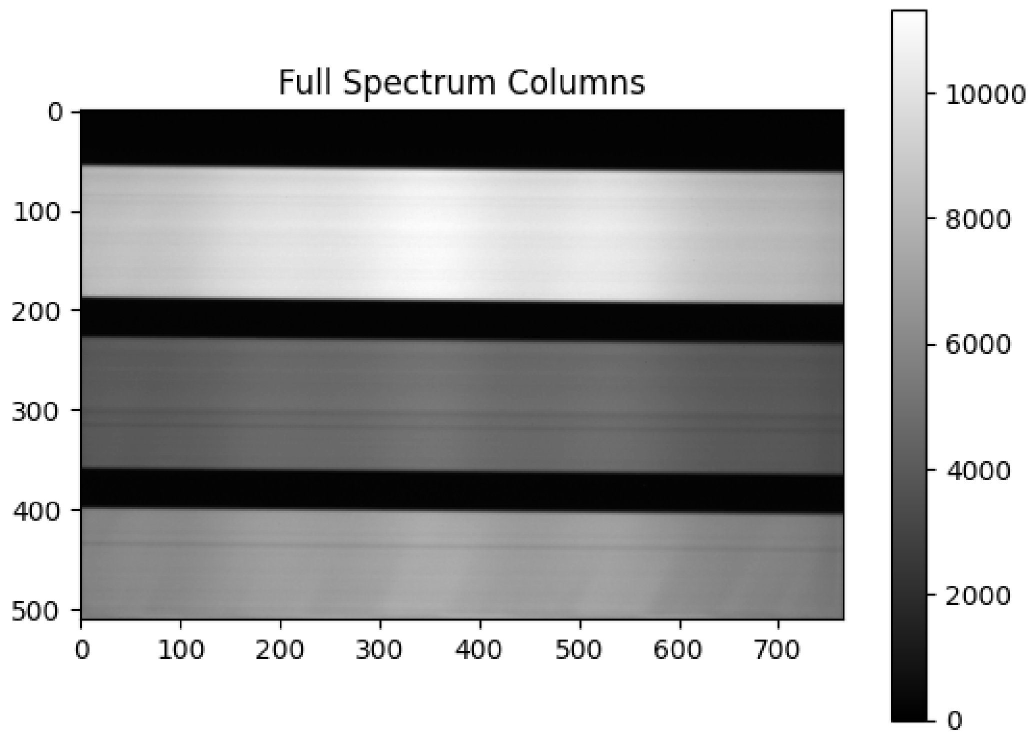
1 Section 4.4 - Spectroscopic Calibration

1.1 4.4.1

1. Below is the image of the 3 slits from the spectrograph with a Neon arc lamp:

[2]:
```python
from astropy.io import fits
import matplotlib.pyplot as plt
import numpy as np

spectra_image = fits.open('../Lab_1/images/lab1_3_4_8.00000009.FIT')

data = spectra_image[0].data
plt.imshow(data,cmap='gray')
plt.colorbar()
plt.title('Full Spectrum Columns')
```

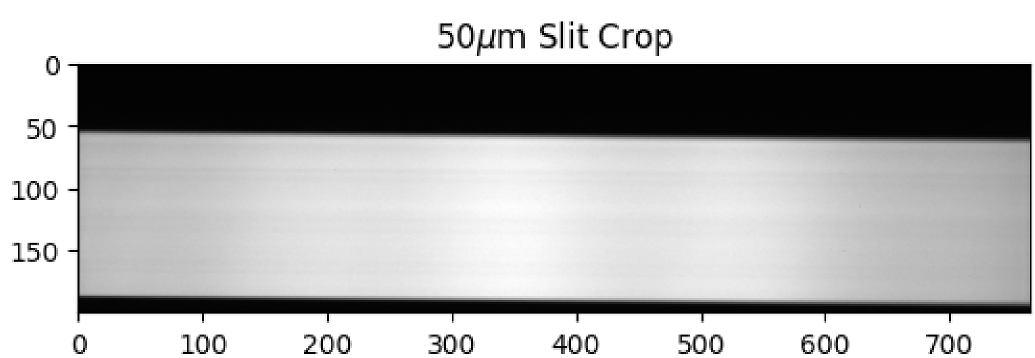
[2]:
```
Text(0.5, 1.0, 'Full Spectrum Columns')
```



[3]:
```
#Crop data to only include the top band (50um)

crop = data[:200,:]
print(crop.shape)
plt.imshow(crop,cmap='gray')
plt.title(r'50$\mu$m Slit Crop')
```

```
(200, 765)
```

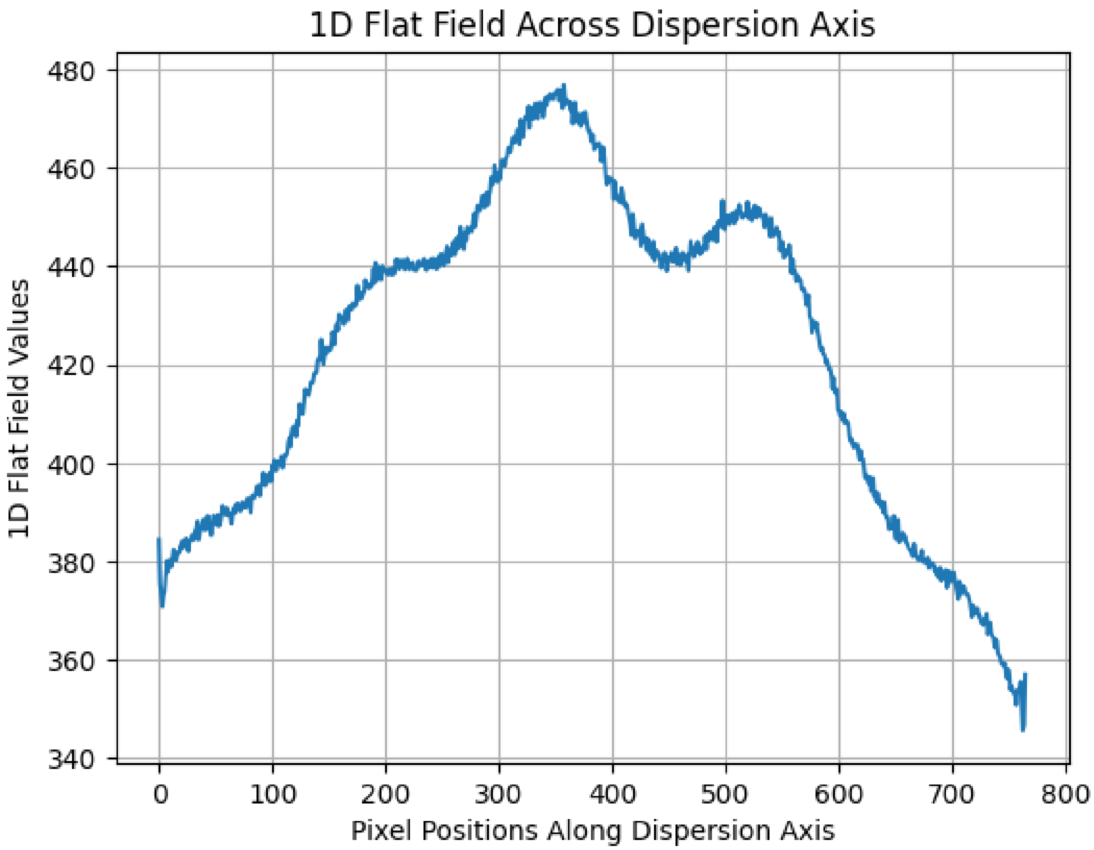
[3]:
```
Text(0.5, 1.0, '50$\\mu$m Slit Crop')
```



1.2 4.4.2

[4]:
```
flat_list = []

# find and open the Flat Fields and store them all in one list
file_prefix = "lab1_3_4_8.0000000"
file_end = ".FIT"
pixel_data = [None]*9

for i in range(0, 9, 1):
    filename = file_prefix + str(i) + file_end
    list = fits.open('../Lab_1/images/'+filename)
    image_data = list[0].data
    pixel_data[i] = image_data



# open a 2d list to store median values
median_values = np.zeros((200, 765)) #the cropped image is a 200 x 765 pixel␣
 ↪array

# run through and calculate the median value for each pixel, 200 for y-axis and␣
 ↪765 for x-axis
for i in range(0, 200):
    for j in range(0, 765):
        values = [pixel_data[x][i][j] for x in range(0,9)]
        median_values[i][j] = np.median(values)
```

[20]:
```
# Calculate the 1D flat field values by averaging along the y-axis
flat_field_1d = np.mean(median_values, axis=0)

# Plotting the 1D flat field values against pixel positions along the␣
 ↪dispersion axis
dispersion_axis = np.arange(765)

plt.plot(dispersion_axis, flat_field_1d)
plt.xlabel('Pixel Positions Along Dispersion Axis')
plt.ylabel('1D Flat Field Values')
plt.title('1D Flat Field Across Dispersion Axis')
plt.grid(True)
plt.show()
```

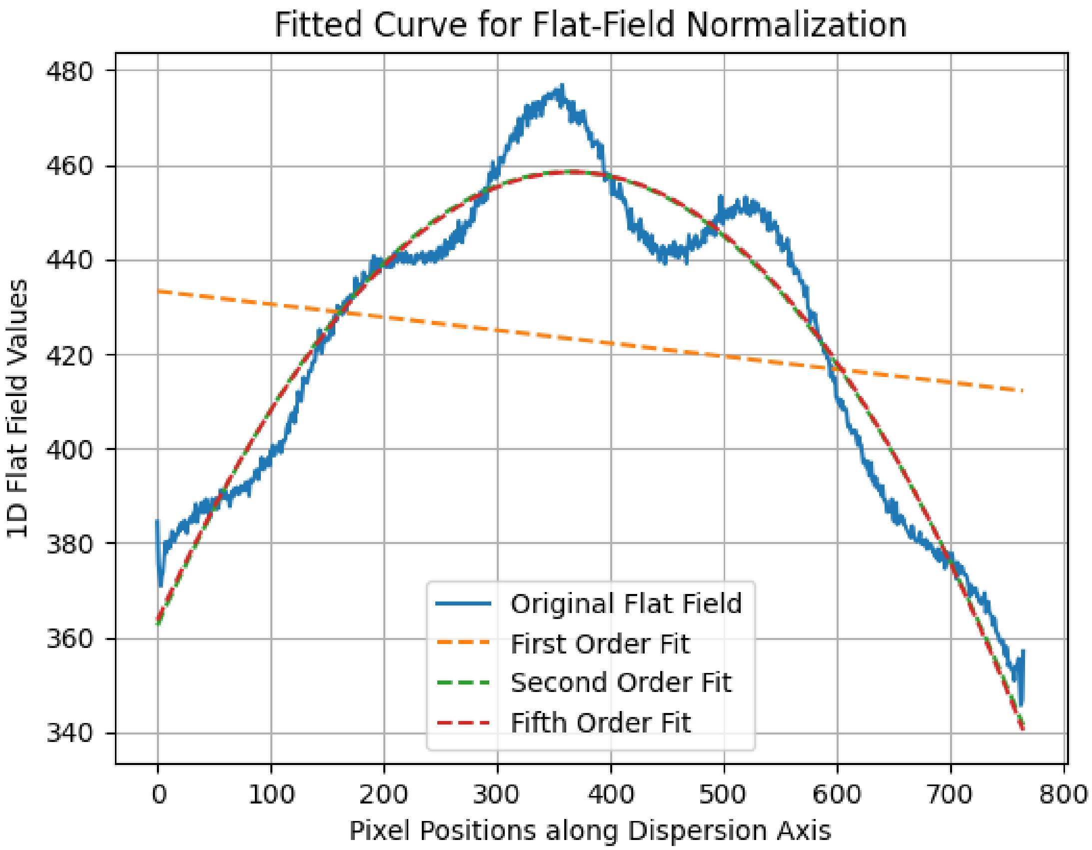


1.3 4.4.3

[6]:
```python
from astropy.modeling import models, fitting

# Pixel positions along the dispersion axis
dispersion_axis = np.arange(765)  #765 is the length of the dispersion axis

# Initialize various polynomial models for fitting
first_order = models.Polynomial1D(degree=1)
second_order = models.Polynomial1D(degree=2)
fifth_order = models.Polynomial1D(degree=5)


# Initialize a fitting algorithm
fitter = fitting.LinearLSQFitter()

# Fit the models to the  1D flat-field spectrum
first_fit = fitter(first_order, dispersion_axis, flat_field_1d)
second_fit = fitter(second_order, dispersion_axis, flat_field_1d)
fifth_fit = fitter(fifth_order, dispersion_axis, flat_field_1d)
```

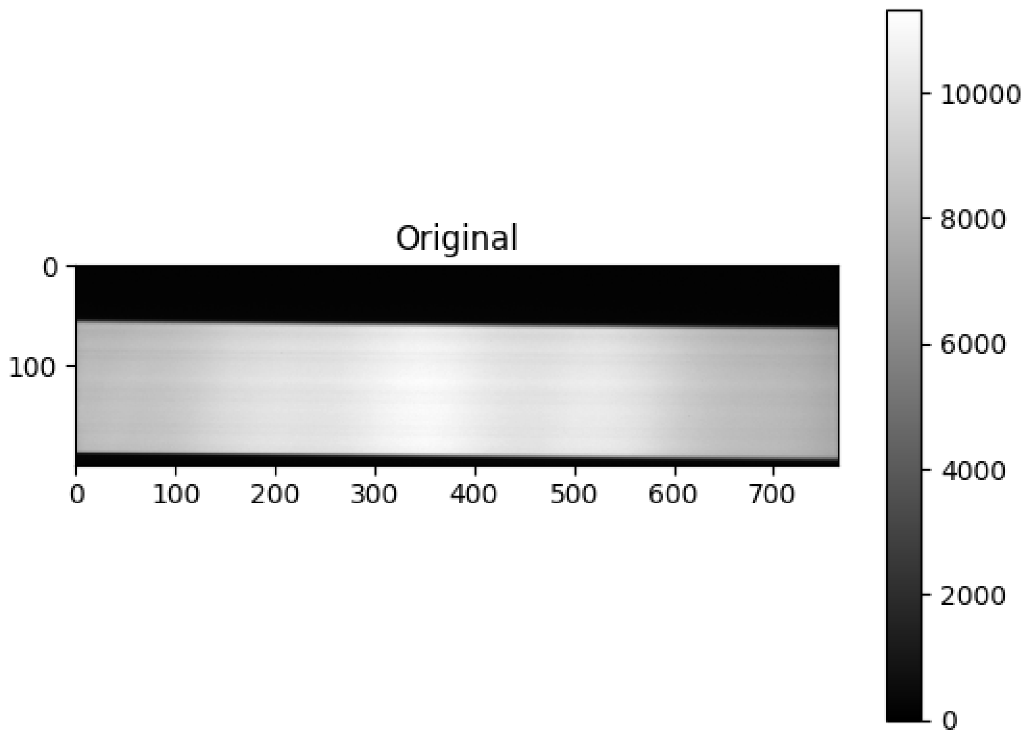
```python
# Generate the fits using the fitted model
first_fit_curve = first_fit(dispersion_axis)
second_fit_curve = second_fit(dispersion_axis)
fifth_fit_curve = fifth_fit(dispersion_axis)

# Plot the original flat-field and the fitted curves
plt.plot(dispersion_axis, flat_field_1d, label='Original Flat Field')
plt.plot(dispersion_axis, first_fit(dispersion_axis), label='First Order Fit',␣
 ↪linestyle='--')
plt.plot(dispersion_axis, second_fit(dispersion_axis), label='Second Order␣
 ↪Fit', linestyle='--')
plt.plot(dispersion_axis, fifth_fit(dispersion_axis), label='Fifth Order Fit',␣
 ↪linestyle='--')
plt.xlabel('Pixel Positions along Dispersion Axis')
plt.ylabel('1D Flat Field Values')
plt.title('Fitted Curve for Flat-Field Normalization')
plt.legend()
plt.grid(True)
plt.show()

# Divide the 2D flat-field by the fits to obtain the normalized flat-field
first_normalized_flat_field = median_values / first_fit_curve
second_normalized_flat_field = median_values / second_fit_curve
fifth_normalized_flat_field = median_values / fifth_fit_curve
```

```
WARNING: The fit may be poorly conditioned
 [astropy.modeling.fitting]
```



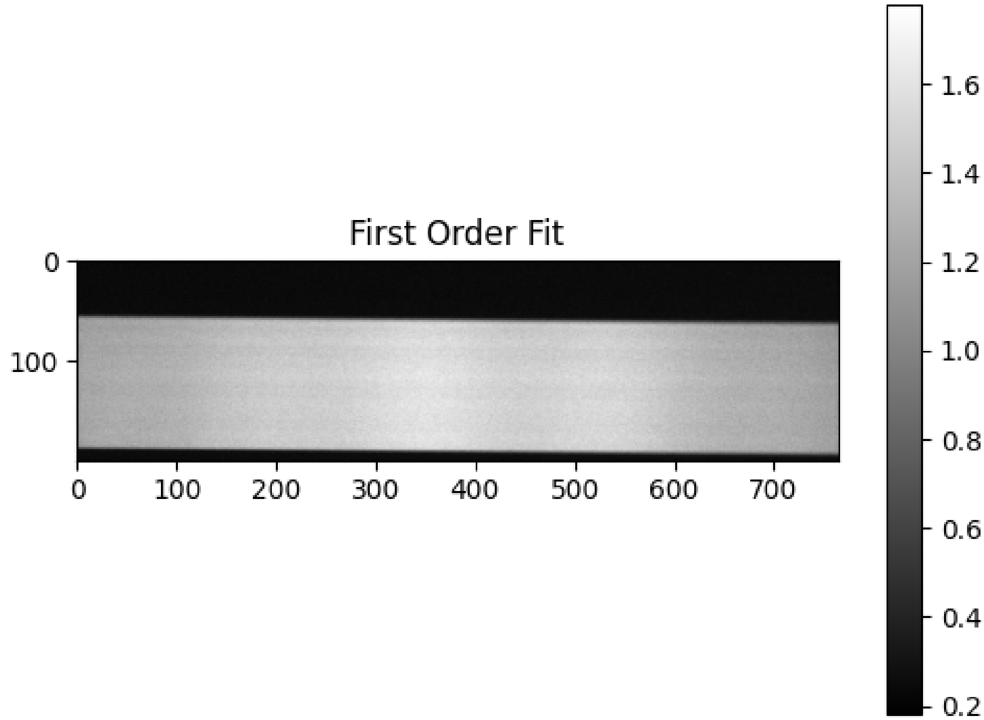
[7]:
```
plt.imshow(crop,cmap='gray')
plt.title('Original')
plt.colorbar()
```

[7]: <matplotlib.colorbar.Colorbar at 0x1476a7220>



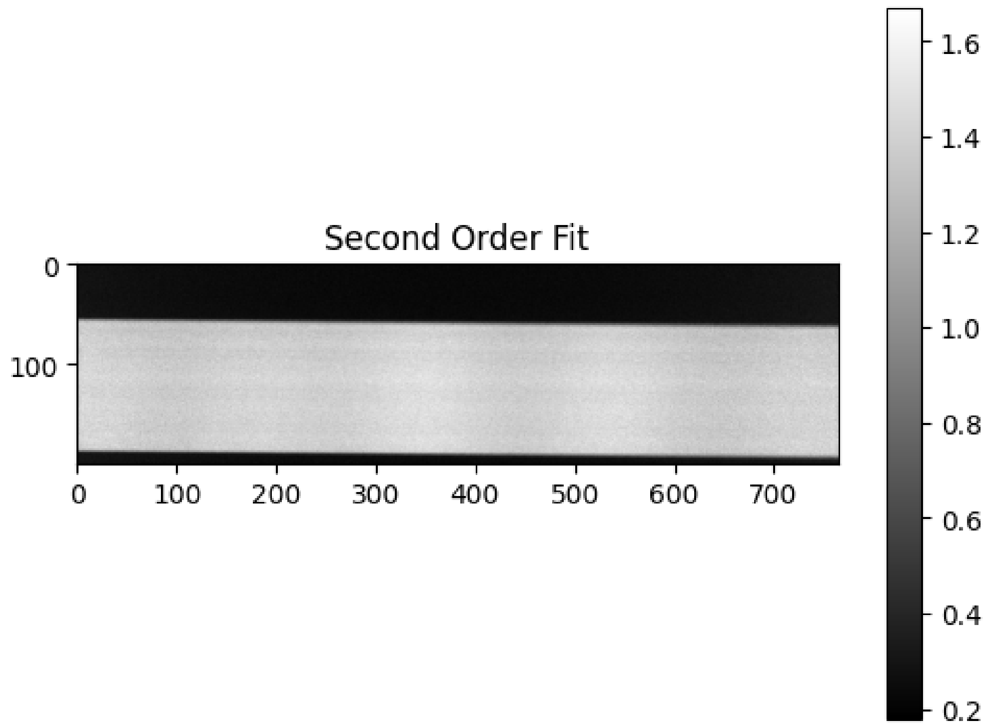

```
[8]: plt.imshow(first_normalized_flat_field,cmap='gray')
     plt.title('First Order Fit')
     plt.colorbar()
```

```
[8]: <matplotlib.colorbar.Colorbar at 0x147760d00>
```



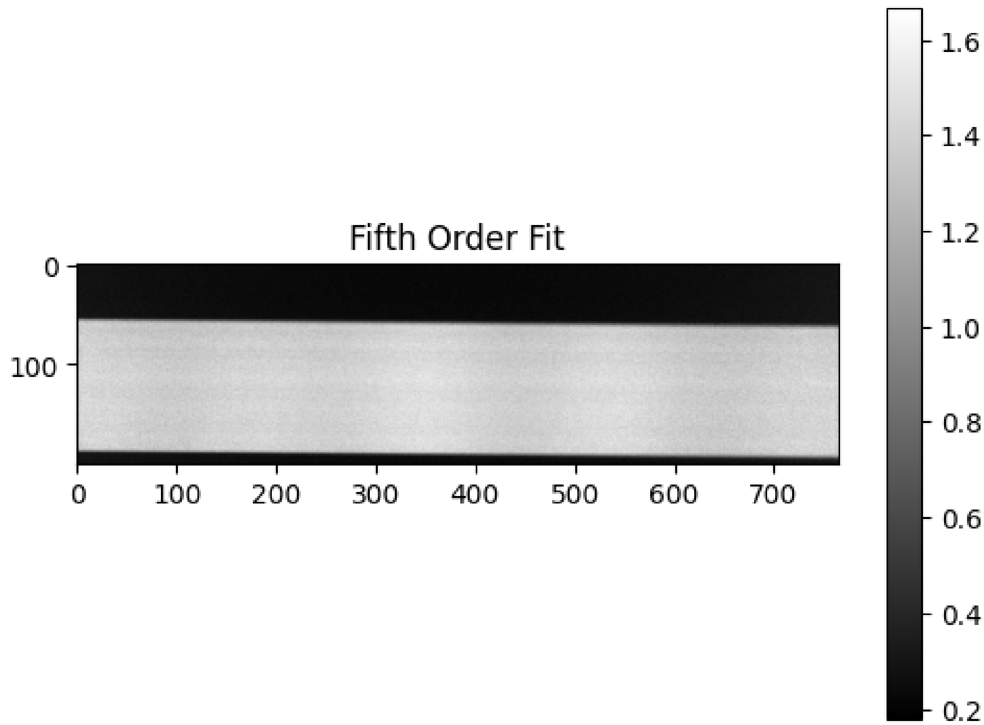
[9]:
```
plt.imshow(second_normalized_flat_field,cmap='gray')
plt.title('Second Order Fit')
plt.colorbar()
```

[9]: <matplotlib.colorbar.Colorbar at 0x168070670>



[10]:
```
plt.imshow(fifth_normalized_flat_field,cmap='gray')
plt.title('Fifth Order Fit')
plt.colorbar()
```

[10]: <matplotlib.colorbar.Colorbar at 0x168137f70>



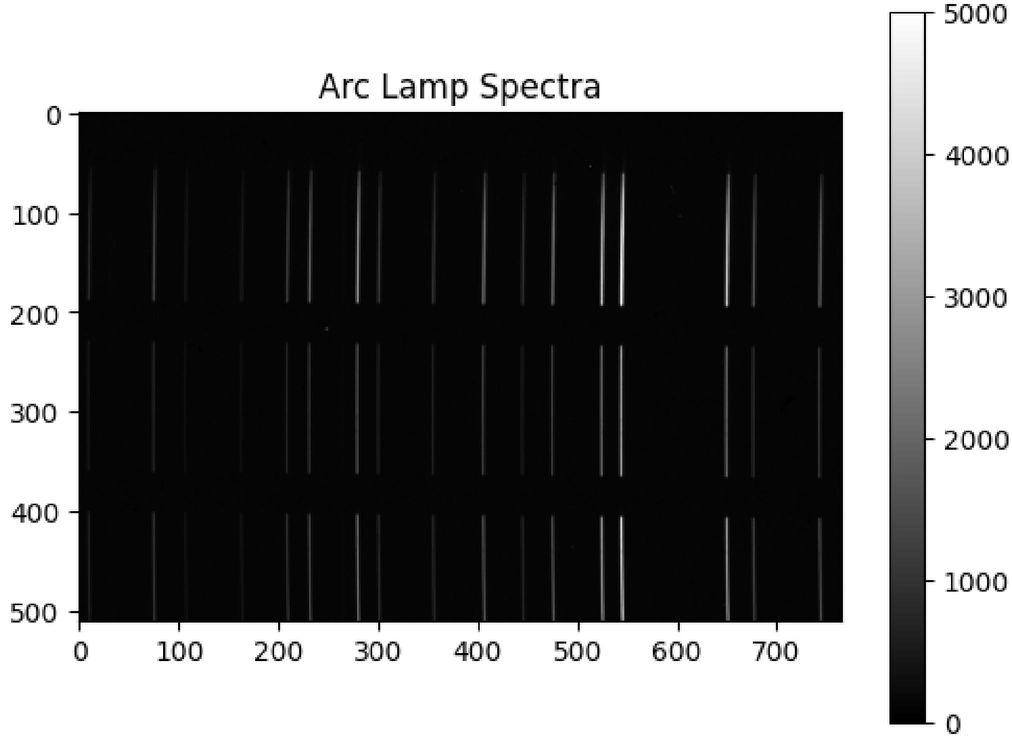
It seems as though the higher the order of the fit, the more evenly distributed the pixel counts.

1.4 4.4.4

Below is the full arc lamp spectra:

[11]:
```
from astropy.io import fits
import matplotlib.pyplot as plt
import numpy as np

spectra = fits.open('../Lab_1/images/lab1_3_4_7_actual.00000002.FIT')

spectra_data = spectra[0].data
plt.imshow(spectra_data,cmap='gray',vmax=5000)
plt.colorbar()
plt.title('Arc Lamp Spectra')
```

[11]:
```
Text(0.5, 1.0, 'Arc Lamp Spectra')
```

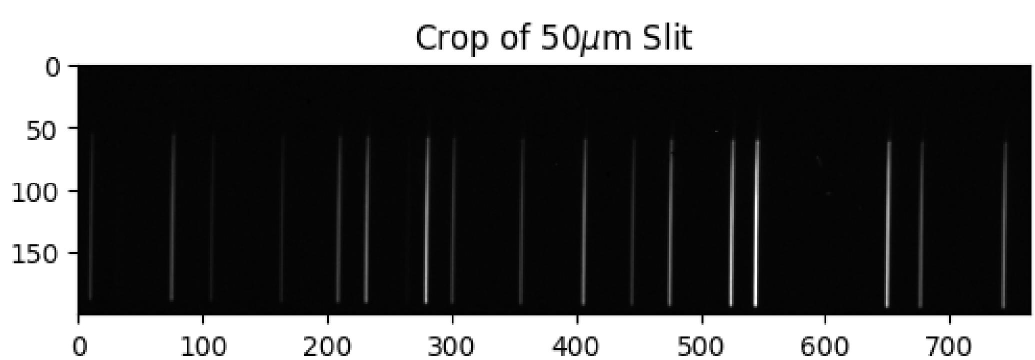


Taking the top row, which corresponds to the 50um slit, and applying the normalized flat field:

[17]:
```
spectra_crop = spectra_data[:200,:]
print(spectra_crop.shape)
plt.imshow(spectra_crop,cmap='gray',vmax=5000)
plt.title(r'Crop of 50$\mu$m Slit')
```

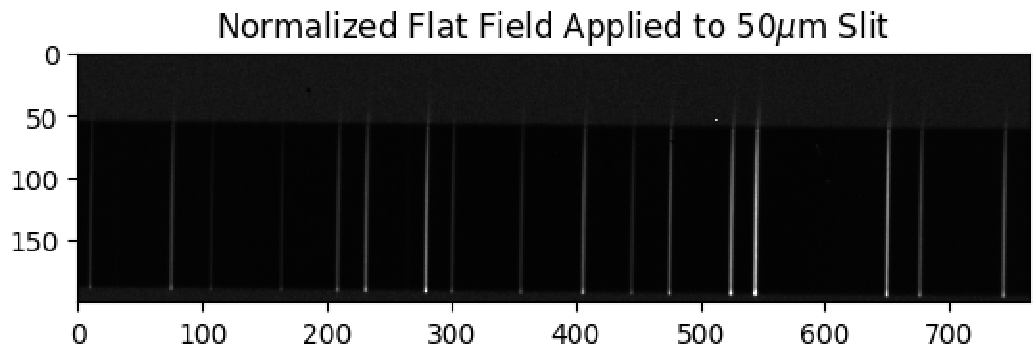
```
(200, 765)
```

[17]:
```
Text(0.5, 1.0, 'Crop of 50$\\mu$m Slit')
```



[18]:
```python
#Apply normalized flat field to arc lamp spectrum using the first order fit
normal = spectra_crop / first_normalized_flat_field

plt.imshow(normal,cmap='gray',vmax=5000)
plt.title(r'Normalized Flat Field Applied to 50$\mu$m Slit')
```

[18]: Text(0.5, 1.0, 'Normalized Flat Field Applied to 50$\\mu$m Slit')



[ ]: