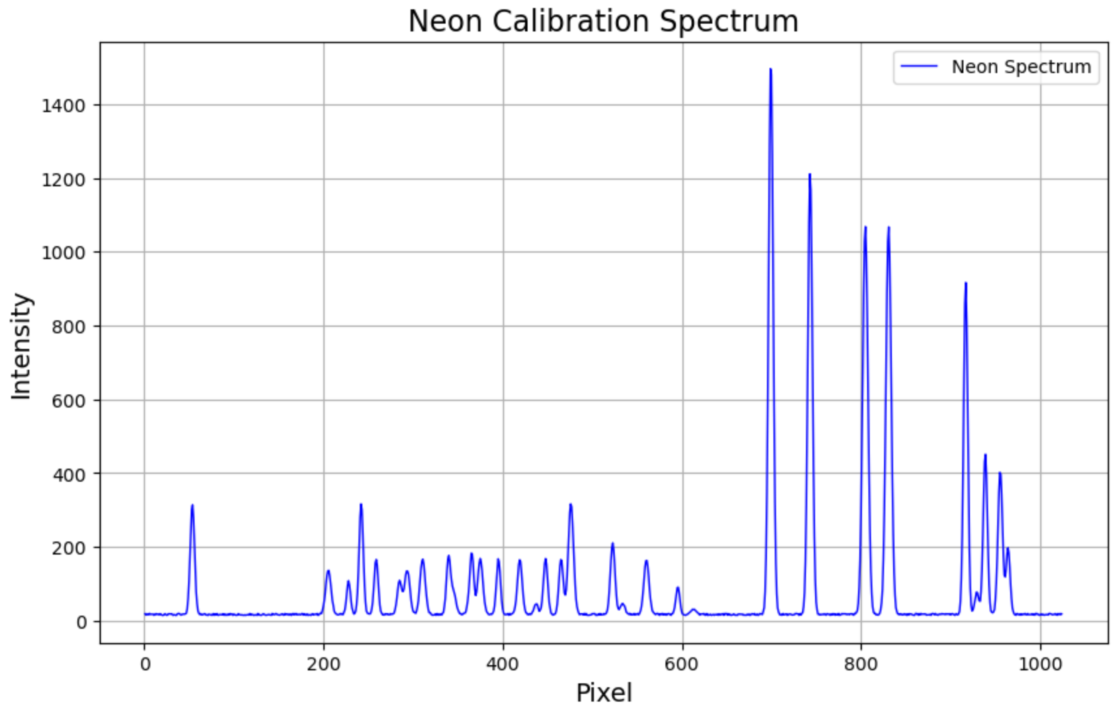
AST326 Lab 2 (1)

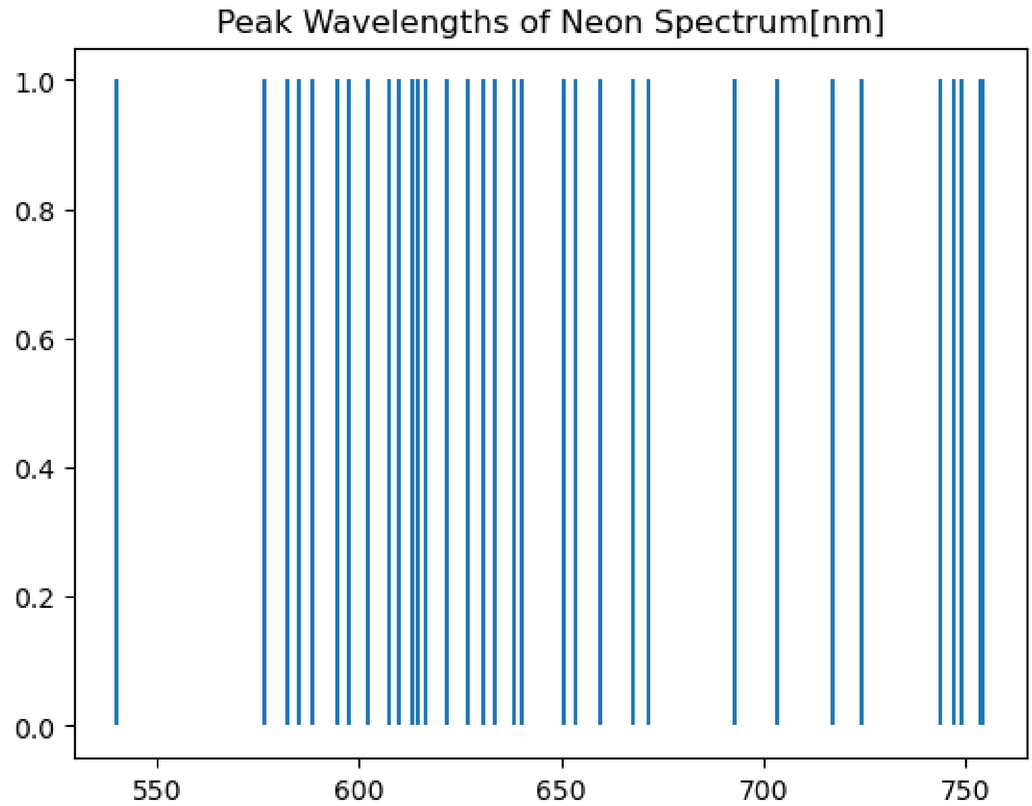
December 31, 2023

[3]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import math
import scipy as sp
from scipy import stats
from scipy.stats import chisquare
from scipy.stats import poisson
from scipy.stats import gamma
from scipy.stats import norm
import seaborn as sb
from scipy.optimize import curve_fit
from astropy.io import fits
with open('1Ne_calib.dat', 'r') as file:
    data = [float(line.strip()) for line in file]

pixels = list(range(1, len(data) + 1))

plt.figure(figsize=(10, 6))
plt.plot(pixels, data, color='b', linewidth=1, label='Neon Spectrum')
plt.xlabel('Pixel', fontsize=14)
plt.ylabel('Intensity', fontsize=14)
plt.title('Neon Calibration Spectrum', fontsize=16)
plt.legend()
plt.grid()
plt.show()
```

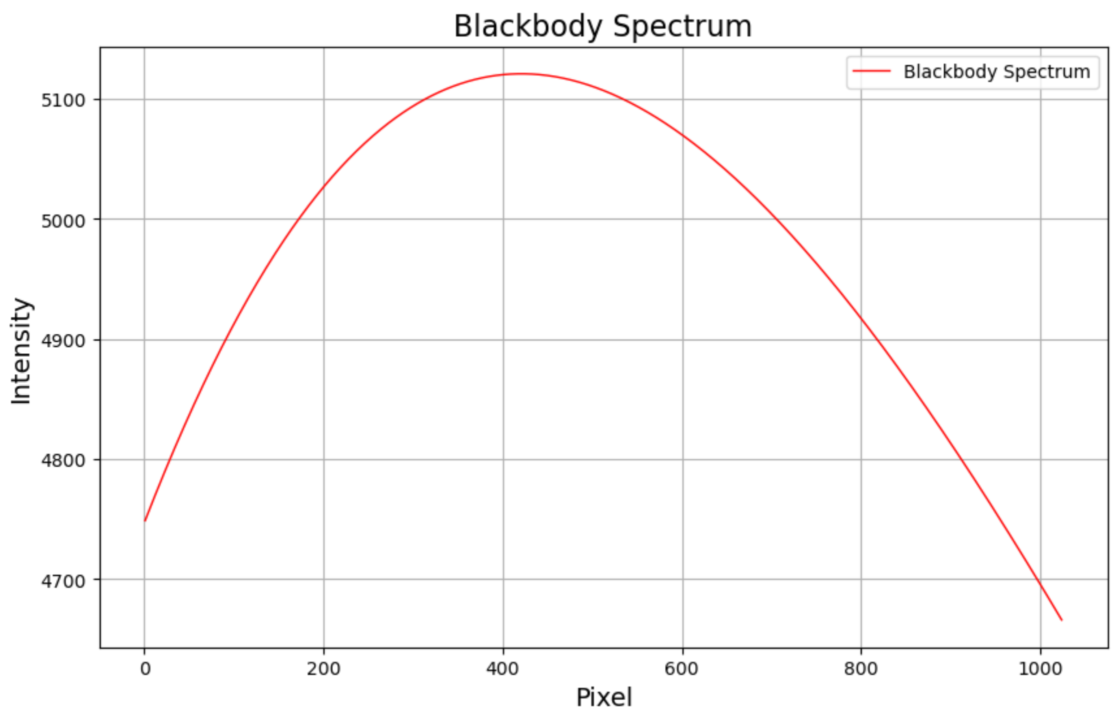


[4]:
```
wavelengths_neon = np.loadtxt('1wavelength_neon.txt')
fig = plt.figure()
plt.vlines(wavelengths_neon, 0, 1)
plt.title('Peak Wavelengths of Neon Spectrum[nm]')
```

[4]: Text(0.5, 1.0, 'Peak Wavelengths of Neon Spectrum[nm]')



[5]:
```python
with open('1Group_Z_BB.dat', 'r') as file:
    data = [float(line.strip()) for line in file]


pixels = list(range(1, len(data) + 1))

plt.figure(figsize=(10, 6))
plt.plot(pixels, data, color='r', linewidth=1, label='Blackbody Spectrum')
plt.xlabel('Pixel', fontsize=14)
plt.ylabel('Intensity', fontsize=14)
plt.title('Blackbody Spectrum', fontsize=16)
plt.legend()
plt.grid()
plt.show()
```

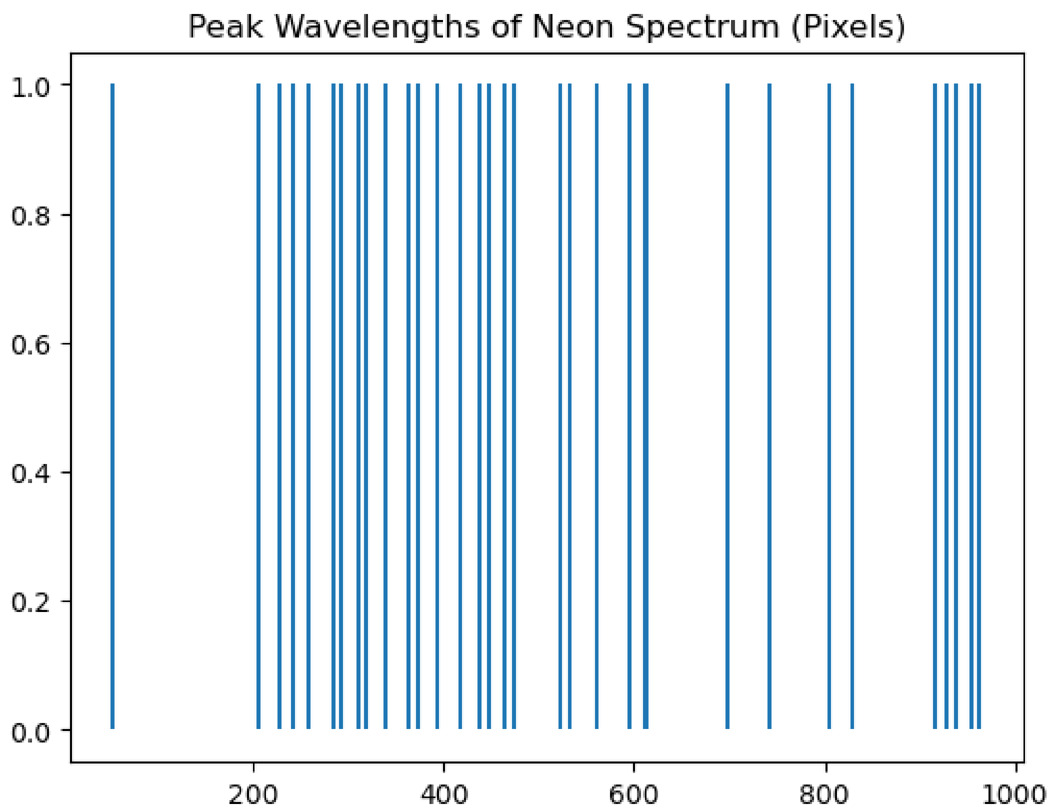


[6]:
```
data = np.loadtxt('1Ne_calib.dat')
```

[7]:
```
pixelpeaks_neon = sp.signal.find_peaks(data,height=22)
print(pixelpeaks_neon[0])
plt.vlines(pixelpeaks_neon[0], 0, 1)
plt.title('Peak Wavelengths of Neon Spectrum (Pixels)')
```

```
[ 53 205 227 241 258 284 292 310 318 339 364 374 394 418 437 447 464 475
 522 533 560 595 611 613 698 742 804 830 916 928 938 954 963]
```

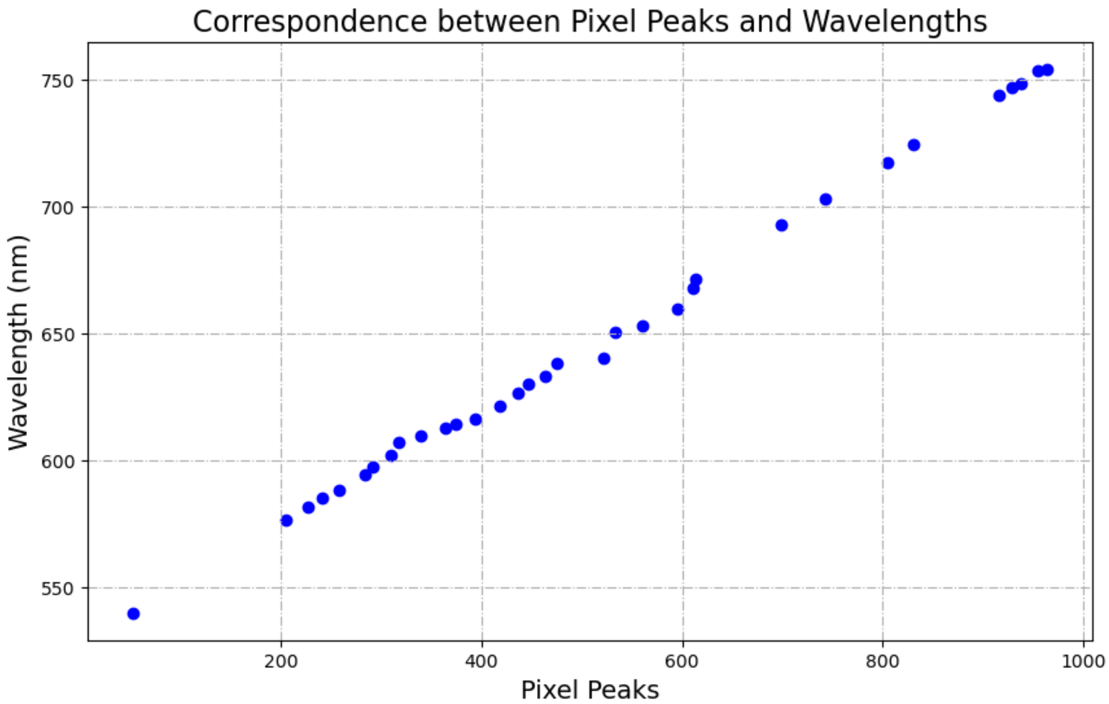
[7]:
```
Text(0.5, 1.0, 'Peak Wavelengths of Neon Spectrum (Pixels)')
```



[8]:
```
print(pixelpeaks_neon[0])
print((wavelengths_neon))
print(len(pixelpeaks_neon[0]))
print(len(wavelengths_neon))
```

```
[ 53 205 227 241 258 284 292 310 318 339 364 374 394 418 437 447 464 475
 522 533 560 595 611 613 698 742 804 830 916 928 938 954 963]
[540.056 576.441 582.015 585.249 588.189 594.483 597.553 602.    607.433
 609.616 612.884 614.306 616.359 621.728 626.649 630.479 633.442 638.299
 640.225 650.653 653.288 659.895 667.828 671.704 692.947 703.241 717.394
 724.512 743.89  747.244 748.887 753.577 754.404]
33
33
```

[9]:
```
plt.figure(figsize=(10, 6))
plt.scatter(pixelpeaks_neon[0], wavelengths_neon, color='b')
plt.xlabel('Pixel Peaks', fontsize=14)
plt.ylabel('Wavelength (nm)', fontsize=14)
plt.title('Correspondence between Pixel Peaks and Wavelengths', fontsize=16)
plt.grid(ls='-.')
plt.show()
```



[10]:
```
def chi_2_r(y, prediction, sigma, m):
    """Determines the X_r^2 metric that characterizes the model fit.
    """
    N=len(y)
    chi_squared=np.sum(((y-prediction)/sigma)**2)
    chi_squared_reduced=chi_squared/(N-m)
    return chi_squared_reduced
```

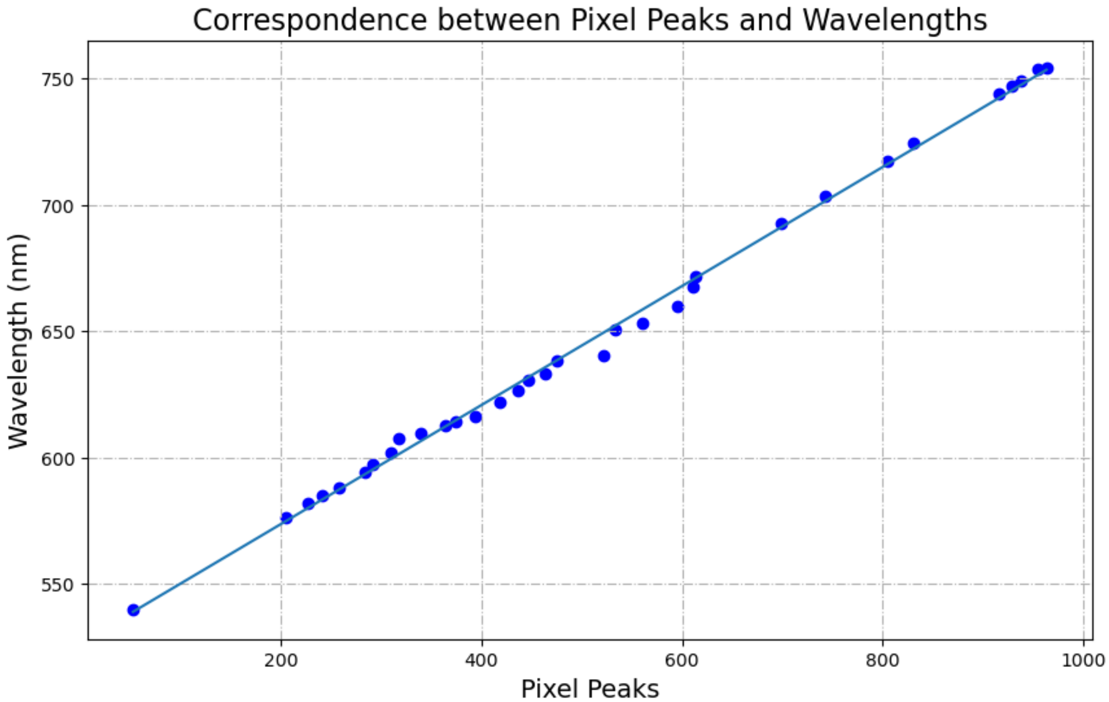
[11]:
```
def linear_model(x, a, b):
    return a*x + b
```

[12]:
```
popt1, pcov1 = curve_fit(linear_model, pixelpeaks_neon[0], wavelengths_neon,␣
 ↪absolute_sigma=True)
pstd1 = np.sqrt(np.diag(pcov1))

a_estimate1, b_estimate1 = popt1
```

[13]:
```
predicted_slope1 = linear_model(pixelpeaks_neon[0], *popt1)
```

[14]:
```
plt.figure(figsize=(10, 6))
plt.scatter(pixelpeaks_neon[0], wavelengths_neon, color='b')
plt.plot(pixelpeaks_neon[0], predicted_slope1)
plt.xlabel('Pixel Peaks', fontsize=14)
plt.ylabel('Wavelength (nm)', fontsize=14)
```

```
plt.title('Correspondence between Pixel Peaks and Wavelengths', fontsize=16)
plt.grid(ls='-.')
plt.show()
```



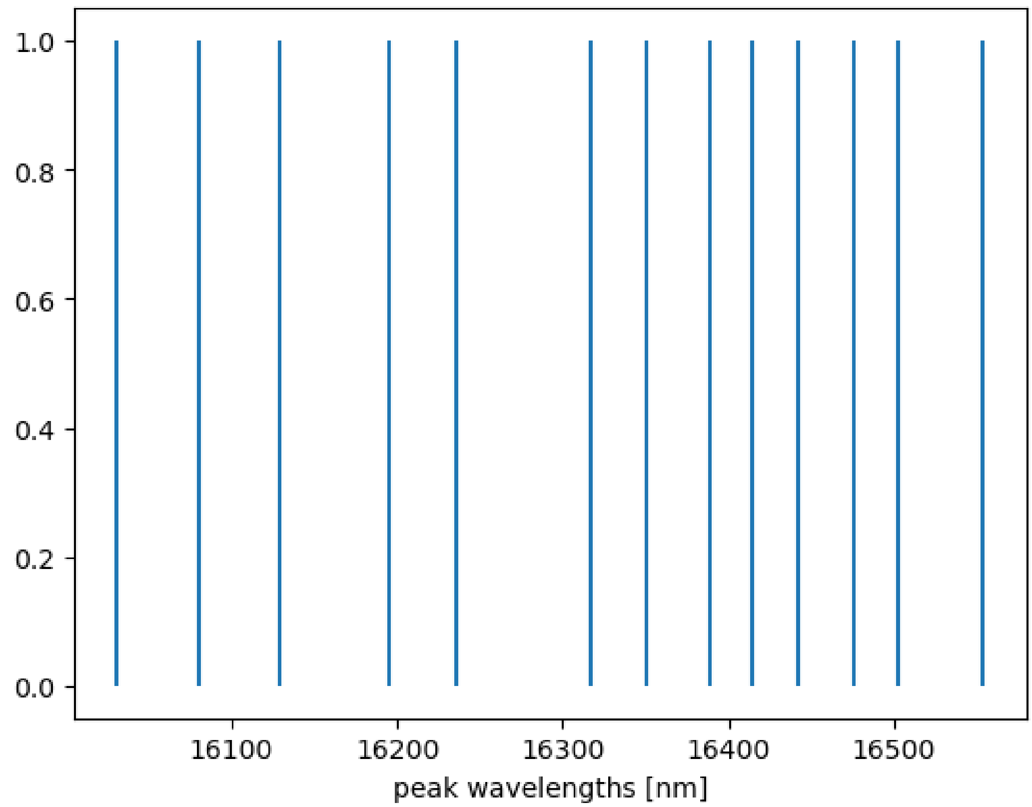
Part 2.2

[15]:
```
wavelength2D = np.loadtxt('2wavelength_OH.txt')
len(wavelength2D)
```

[15]: 13

[16]:
```
fig = plt.figure()
plt.vlines(wavelength2D, 0,1)
plt.xlabel('peak wavelengths [nm]')
```

[16]: Text(0.5, 0, 'peak wavelengths [nm]')




```
[17]: iron = np.loadtxt('2D_Signal')
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[17], line 1
----> 1 iron = np.loadtxt('2D_Signal')

File /opt/conda/lib/python3.10/site-packages/numpy/lib/npyio.py:1338, in␣
↪loadtxt(fname, dtype, comments, delimiter, converters, skiprows, usecols,␣
↪unpack, ndmin, encoding, max_rows, quotechar, like)
   1335 if isinstance(delimiter, bytes):
   1336     delimiter = delimiter.decode('latin1')
-> 1338 arr = _read(fname, dtype=dtype, comment=comment, delimiter=delimiter,
   1339             converters=converters, skiplines=skiprows, usecols=usecols,
   1340             unpack=unpack, ndmin=ndmin, encoding=encoding,
   1341             max_rows=max_rows, quote=quotechar)
   1343 return arr

File /opt/conda/lib/python3.10/site-packages/numpy/lib/npyio.py:975, in␣
↪_read(fname, delimiter, comment, quote, imaginary_unit, usecols, skiplines,␣
↪max_rows, converters, ndmin, unpack, dtype, encoding)
    973     fname = os.fspath(fname)
```

```
    974 if isinstance(fname, str):
--> 975     fh = np.lib._datasource.open(fname, 'rt', encoding=encoding)
    976     if encoding is None:
    977         encoding = getattr(fh, 'encoding', 'latin1')

File /opt/conda/lib/python3.10/site-packages/numpy/lib/_datasource.py:193, in␣
 ↪open(path, mode, destpath, encoding, newline)
    156 """
    157 Open `path` with `mode` and return the file object.
    158
   (…)
    189
    190 """
    192 ds = DataSource(destpath)
--> 193 return ds.open(path, mode, encoding=encoding, newline=newline)

File /opt/conda/lib/python3.10/site-packages/numpy/lib/_datasource.py:533, in␣
 ↪DataSource.open(self, path, mode, encoding, newline)
    530     return _file_openers[ext](found, mode=mode,
    531                               encoding=encoding, newline=newline)
    532 else:
--> 533     raise FileNotFoundError(f"{path} not found.")

FileNotFoundError: 2D_Signal not found.
```

[ ]:
```
fig=plt.figure(figsize=(7,5))
pixels2 = []
intensity2 = []
for i in range(len(iron)):
    pixels2.append(iron[i][0])
    intensity2.append(iron[i][1])
plt.plot(pixels2, intensity2)
plt.xlabel("Pixel")
plt.ylabel("Intensity")
```

[18]:
```
peak2 = sp.signal.find_peaks(intensity2, height=1700)
print(peak2[1]['peak_heights'])
print(len(peak2[1]['peak_heights']))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[18], line 1
----> 1 peak2 = sp.signal.find_peaks(intensity2, height=1700)
      2 print(peak2[1]['peak_heights'])
      3 print(len(peak2[1]['peak_heights']))
```

```
NameError: name 'intensity2' is not defined
```

[19]:
```
plt.figure()

plt.plot(pixels2, intensity2)
peaks_x_2 = peak2[0]+1
peaks_y_2 = peak2[1]['peak_heights']
plt.plot(peaks_x_2, peaks_y_2, color = 'coral', ls = '',
         marker = 'x',  ms = 5, label = 'Peaks')

plt.xlabel('Pixel')
plt.ylabel('Intensity')

plt.legend(loc = 'best', fontsize = 10)
plt.grid(ls = '-.')
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[19], line 3
      1 plt.figure()
----> 3 plt.plot(pixels2, intensity2)
      4 peaks_x_2 = peak2[0]+1
      5 peaks_y_2 = peak2[1]['peak_heights']

NameError: name 'pixels2' is not defined
```

```
<Figure size 640x480 with 0 Axes>
```

Part 3

[20]:
```
capella_10s = fits.getdata('Capella10s.fit')
aldebran_10s = fits.getdata('Aldebaran-10s.fit')
elnath_10s = fits.getdata('Elnath-betaTau-10s.fit')
darks0001_10s = fits.getdata('Dark-0001_10s.fit')
darks0002_10s = fits.getdata('Dark-0002_10s.fit')
darks0003_10s = fits.getdata('Dark-0003_10s.fit')
flat1s_01=fits.getdata('Flat-1s_01.fit')
flat1s_02=fits.getdata('Flat-1s_02.fit')
flat1s_03=fits.getdata('Flat-1s_03.fit')
darks0001_1s = fits.getdata('Dark-0001_1s.fit')
darks0002_1s = fits.getdata('Dark-0002_1s.fit')
darks0003_1s = fits.getdata('Dark-0003_1s.fit')
```

[21]:
```
median_dark_10s = np.median([darks0001_10s, darks0002_10s, darks0003_10s],␣
 ↪axis=0)
median_flat_1s = np.median([flat1s_01, flat1s_02, flat1s_03], axis=0)
```

```
median_dark_1s = np.median([darks0001_1s, darks0002_1s, darks0003_1s], axis=0)
```

[22]:
```
capella_10s_subtracted=capella_10s-median_dark_10s
aldebran_10s_subtracted=aldebran_10s-median_dark_10s
elnath_10s_subtracted=elnath_10s-median_dark_10s
```

[23]:
```
flat_minus_dark=median_flat_1s-median_dark_1s
```

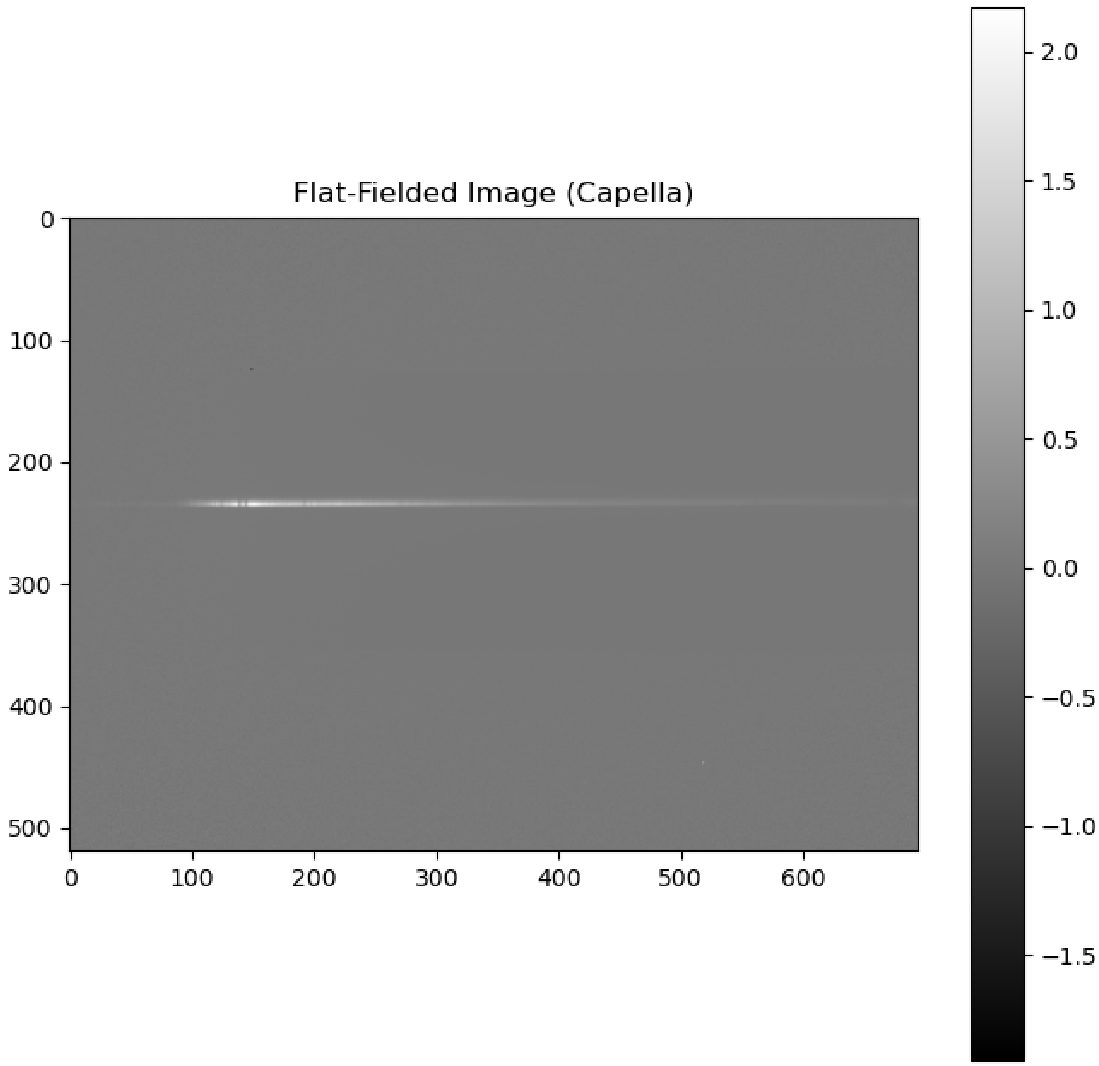
[24]:
```
flat_fielded_image=capella_10s_subtracted/flat_minus_dark
flat_fielded_image2=aldebran_10s_subtracted/flat_minus_dark
flat_fielded_image3=elnath_10s_subtracted/flat_minus_dark
```

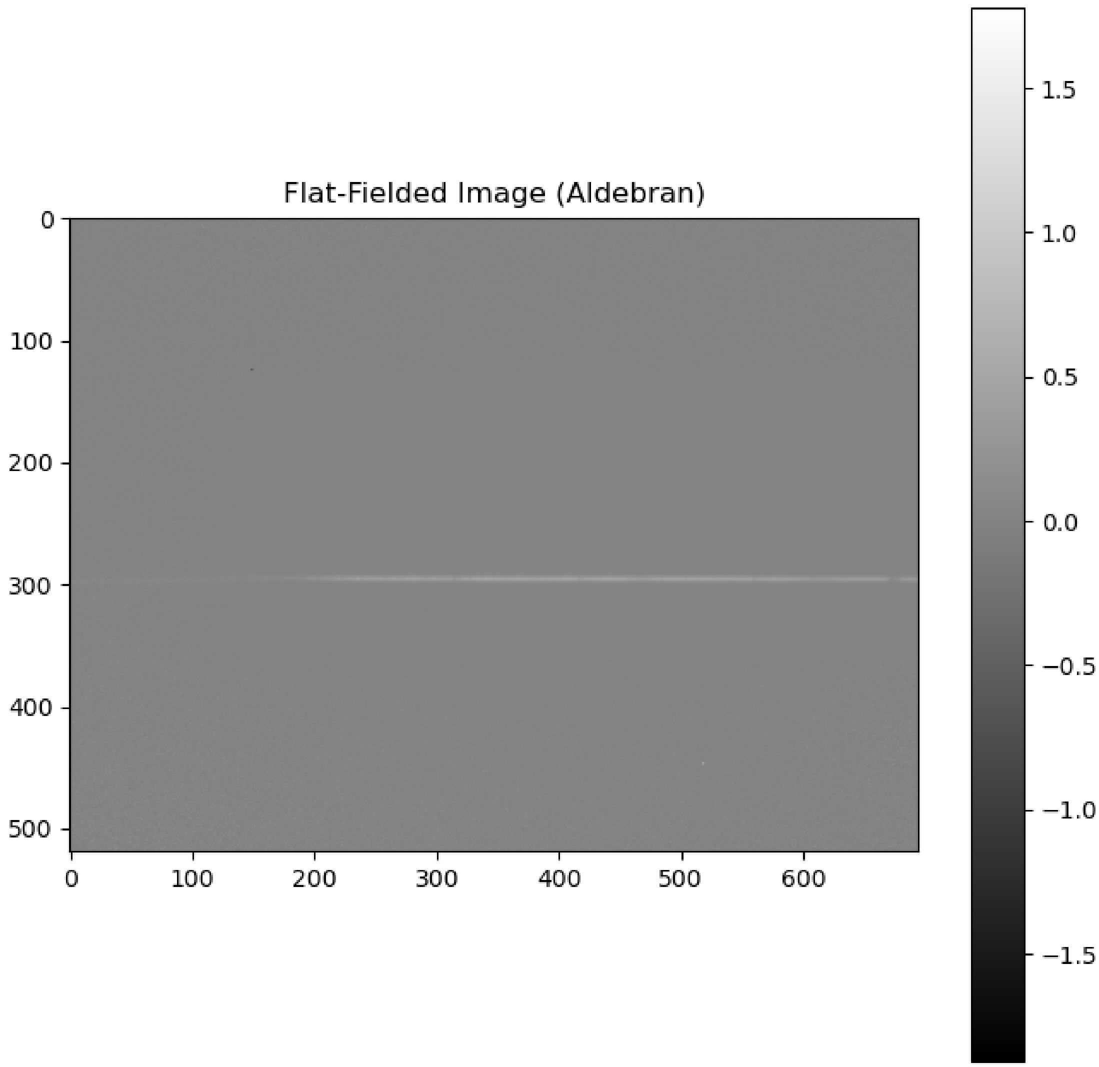
[25]:
```
print(flat_fielded_image)
```

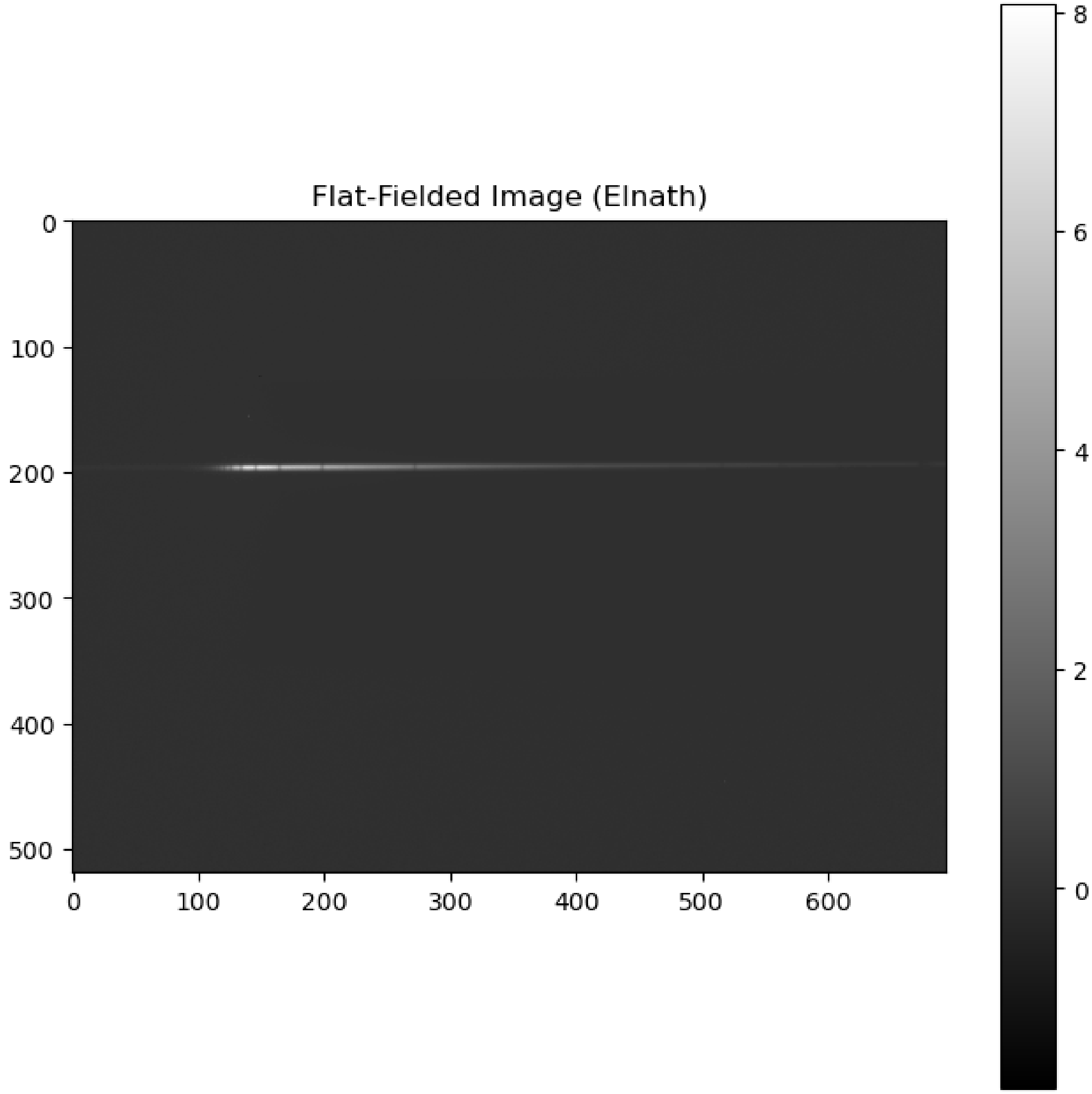
```
[[ 0.03053435 -0.02131783  0.01888668 …  0.06227106 -0.03202847
   0.00361011]
 [ 0.01242236 -0.005923   -0.04170804 …  0.00934579 -0.01518438
  -0.01976285]
 [ 0.04933586 -0.00902708 -0.02597403 …  0.02747253  0.00626305
  -0.02621723]
 …
 [-0.14049587 -0.04008439  0.0332681  … -0.00681818 -0.04140787
  -0.03921569]
 [ 0.14893617 -0.00238663  0.01455301 …  0.02067183 -0.00497512
  -0.04497354]
 [ 0.0373444   0.02444444  0.06311637 …  0.01647059 -0.00222717
   0.00471698]]
```

[26]:
```
import matplotlib.pyplot as plt

# Assuming 'flat_fielded_image' is the array you provided

# Plotting the flat-fielded image
plt.figure(figsize=(8, 8))
plt.imshow(flat_fielded_image, cmap='gray')  # Change the cmap as per your␣
 ↪preference
plt.colorbar()
plt.title('Flat-Fielded Image (Capella)')
plt.show()

# Plotting the flat-fielded image
plt.figure(figsize=(8, 8))
plt.imshow(flat_fielded_image2, cmap='gray')  # Change the cmap as per your␣
 ↪preference
plt.colorbar()
plt.title('Flat-Fielded Image (Aldebran)')
plt.show()
```

```python
# Plotting the flat-fielded image
plt.figure(figsize=(8, 8))
plt.imshow(flat_fielded_image3, cmap='gray')  # Change the cmap as per your␣
↪preference
plt.colorbar()
plt.title('Flat-Fielded Image (Elnath)')
plt.show()
```




Flat-Fielded Image (Aldebran)
0
100
200
300
400
500
600
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5

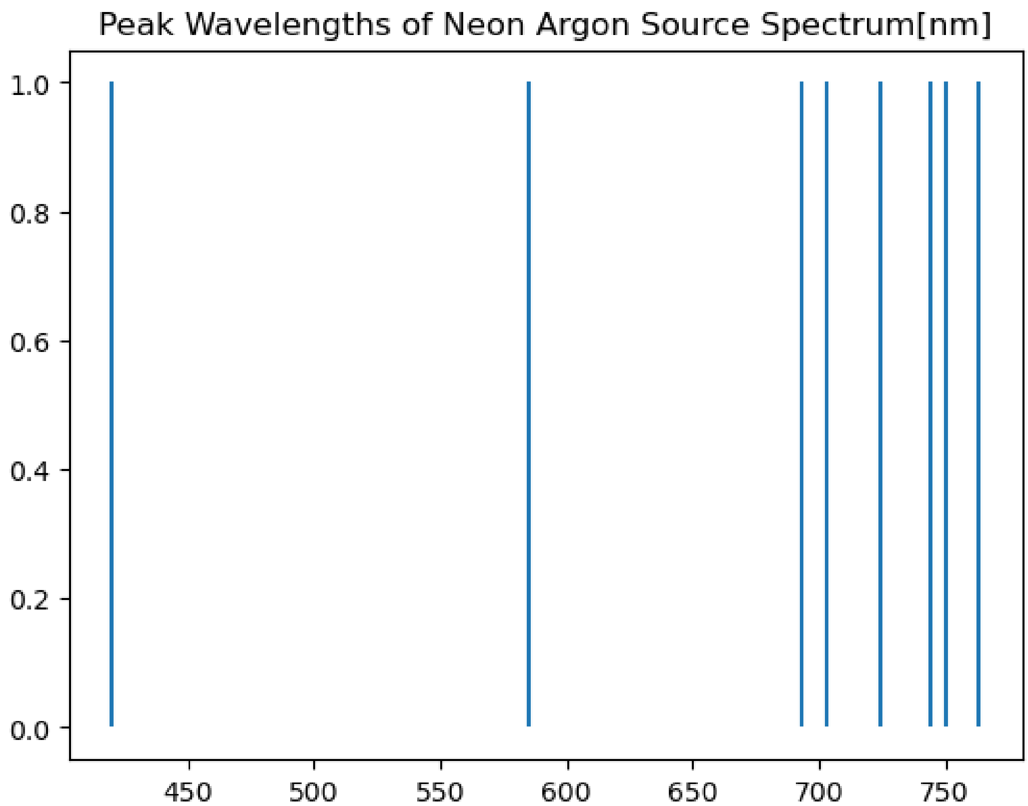


[27]:
```
from astropy.io import fits

# Assuming 'flat_fielded_image' is the array you want to save

# Define the output file name
output_filename = 'flat_fielded_image_aldebran.fits'

# Save the array as a FITS file
hdu = fits.PrimaryHDU(flat_fielded_image2)
hdu.writeto(output_filename, overwrite=True)


# Assuming 'flat_fielded_image' is the array you want to save
```

```python
# Define the output file name
output_filename = 'flat_fielded_image_elnath.fits'

# Save the array as a FITS file
hdu = fits.PrimaryHDU(flat_fielded_image3)
hdu.writeto(output_filename, overwrite=True)
```

[28]:
```python
wavelengths_NeonArgonSource = np.loadtxt('NeonArgonSource')
fig = plt.figure()
plt.vlines(wavelengths_NeonArgonSource, 0, 1)
plt.title('Peak Wavelengths of Neon Argon Source Spectrum[nm]')
```

[28]: Text(0.5, 1.0, 'Peak Wavelengths of Neon Argon Source Spectrum[nm]')



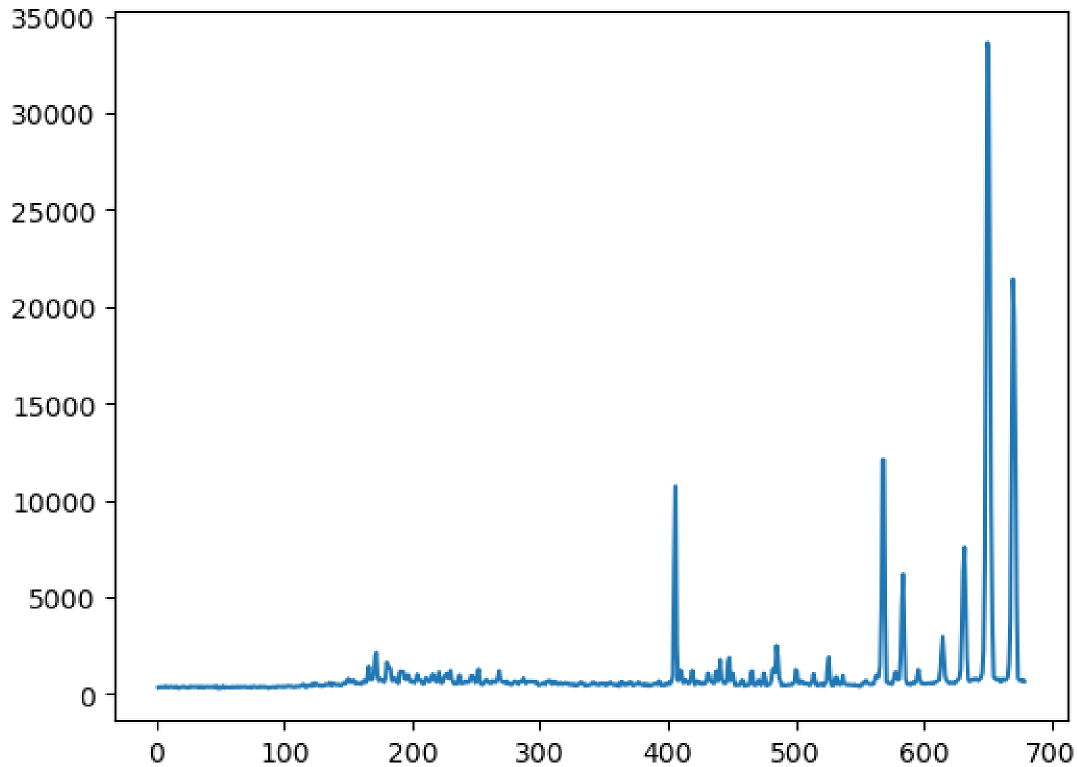
[29]:
```python
print(wavelengths_NeonArgonSource)
```

```
[420.067 585.249 692.947 703.241 724.517 743.89  750.387 763.511]
```

[30]:
```python
intensity3_neon = np.loadtxt('Neon_calib_part3')
pixelsneon3 = []
```

```
intensityneon3 = []
for i in range(len(intensity3_neon)):
    pixelsneon3.append(intensity3_neon[i][0])
    intensityneon3.append(intensity3_neon[i][1])
plt.plot(pixelsneon3, intensityneon3)
```

[30]: [<matplotlib.lines.Line2D at 0x7fb9b0bcc670>]

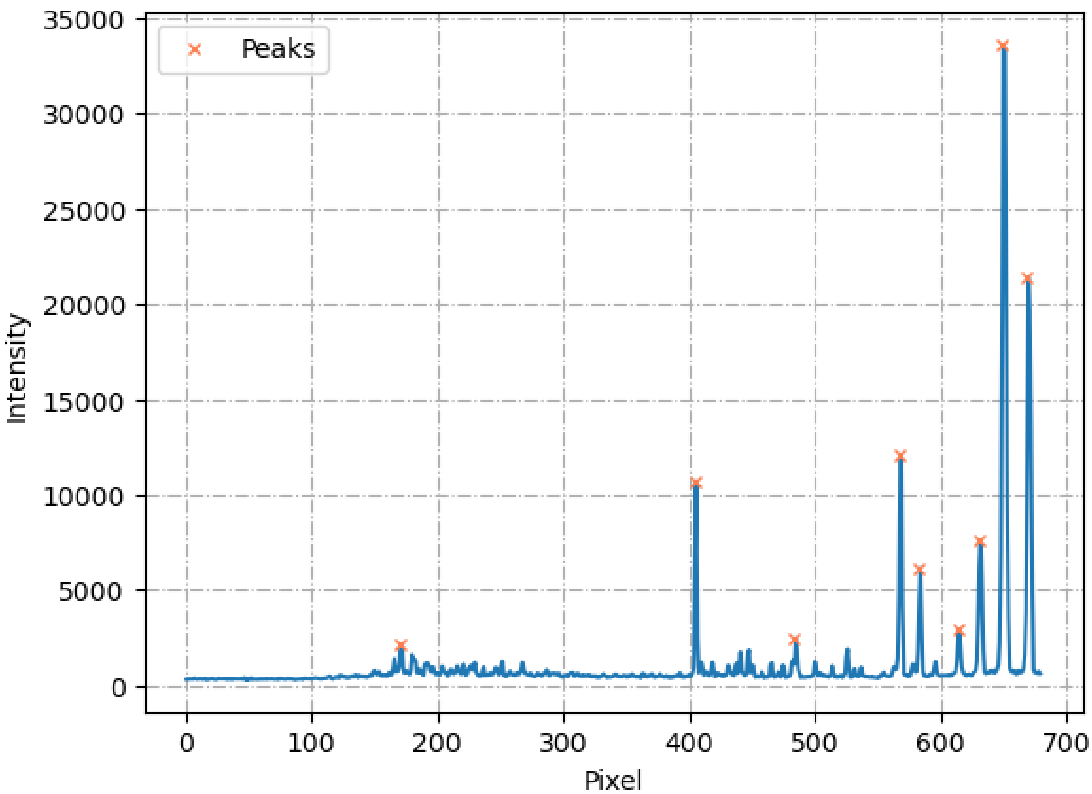


[31]:
```
pixelpeaks3_neon = sp.signal.find_peaks(intensityneon3,height=2000)
print(len(pixelpeaks3_neon[1]['peak_heights']))
print(len(wavelengths_NeonArgonSource))
```

```
9
8
```

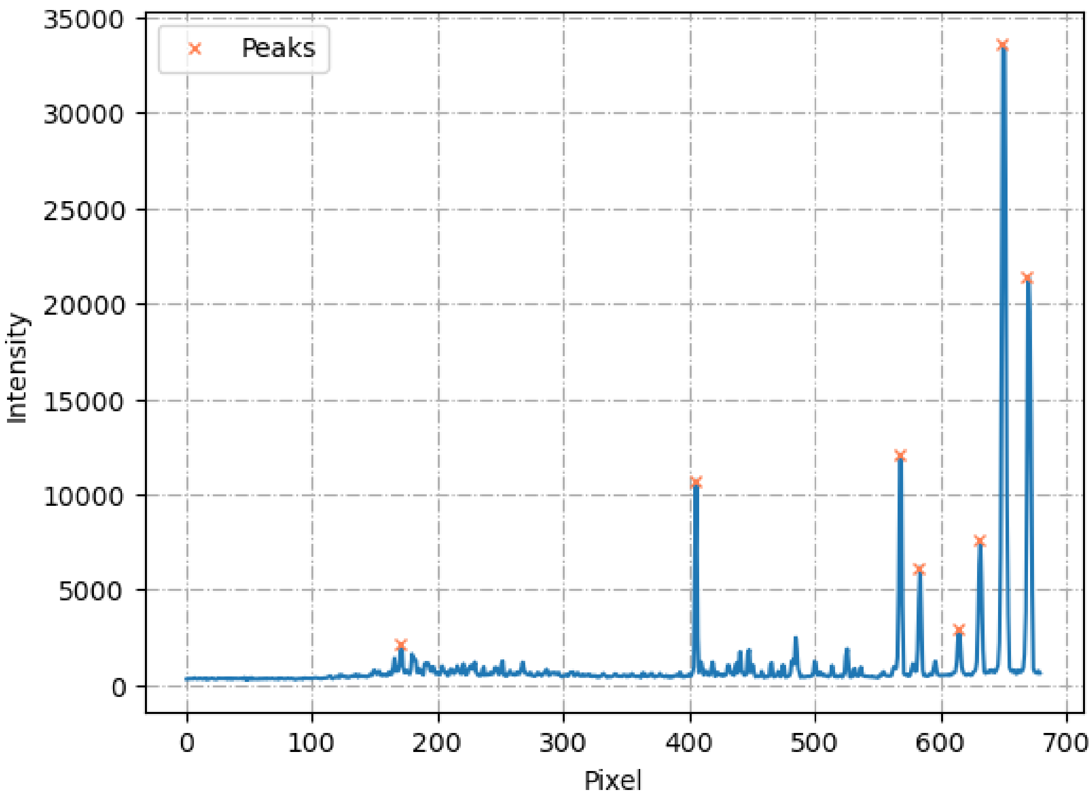
[32]:
```
plt.figure()

plt.plot(pixelsneon3, intensityneon3)
peaks_x_neon3 = pixelpeaks3_neon[0]
peaks_y_neon3 = pixelpeaks3_neon[1]['peak_heights']
plt.plot(peaks_x_neon3, peaks_y_neon3, color = 'coral', ls = '',
         marker = 'x',  ms = 5, label = 'Peaks')
```

```
plt.xlabel('Pixel')
plt.ylabel('Intensity')

plt.legend(loc = 'best', fontsize = 10)
plt.grid(ls = '-.')
plt.show()
```



[33]:
```
# Remove the third peak from the arrays
peaks_x_neon3 = np.delete(peaks_x_neon3, 2)
peaks_y_neon3 = np.delete(peaks_y_neon3, 2)

# Plot the updated graph
plt.figure()
plt.plot(pixelsneon3, intensityneon3)
plt.plot(peaks_x_neon3, peaks_y_neon3, color='coral', ls='', marker='x', ms=5,␣
  ↪label='Peaks')
plt.xlabel('Pixel')
plt.ylabel('Intensity')
plt.legend(loc='best', fontsize=10)
plt.grid(ls='-.')
```

```python
plt.show()
```



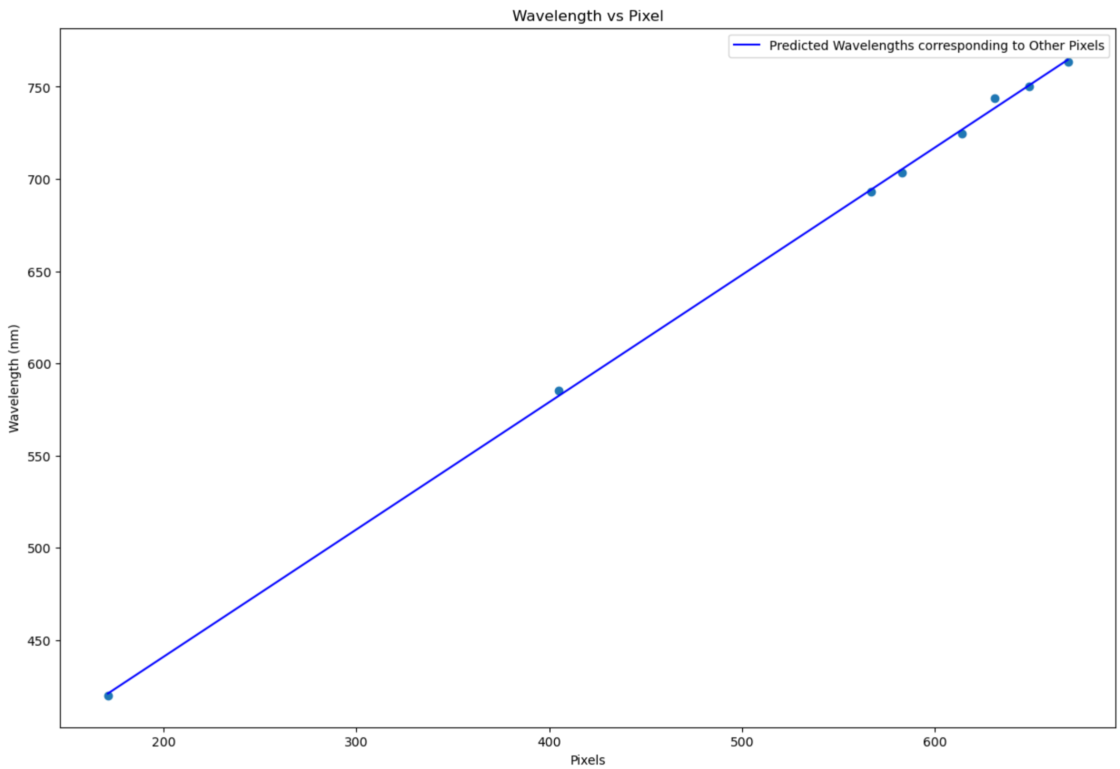
[34]:
```python
popt4, pcov4 = curve_fit(linear_model, peaks_x_neon3,␣
 ↪wavelengths_NeonArgonSource,absolute_sigma = True)

pstd4 = np.sqrt(np.diag(pcov4))

print("Gradient: ", popt4[0])
print("Uncertainty Gradient: ", pstd4[0])

print("Intercept: ", popt4[1])
print("Uncertainty Intercept: ", pstd4[1])
```

```
Gradient:  0.6899820219943742
Uncertainty Gradient:  0.0022409874554043903
Intercept:  303.05951346888287
Uncertainty Intercept:  1.2523899847069118
```

[35]:
```python
predicted_slope4 = linear_model(peaks_x_neon3, *popt4)
```

[36]:
```
fig=plt.figure(figsize=(15,10))
plt.scatter(peaks_x_neon3, wavelengths_NeonArgonSource)
plt.plot(peaks_x_neon3, predicted_slope4, color= 'blue', label= 'Predicted␣
  ↪Wavelengths corresponding to Other Pixels' )
plt.xlabel('Pixels')
plt.ylabel('Wavelength (nm)')
plt.title("Wavelength vs Pixel")
plt.legend()
```

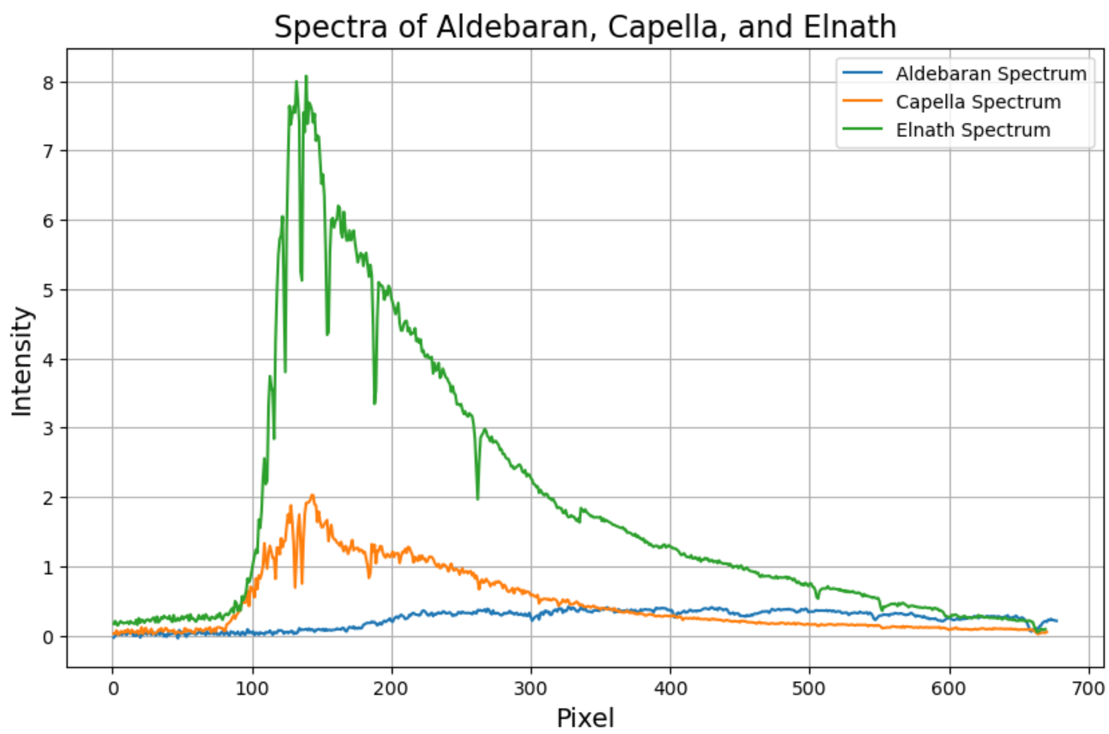
[36]: <matplotlib.legend.Legend at 0x7fb9b0a22bf0>



[37]:
```
slope_val = popt4[0]
slope_uncertainty = pstd4[0]
intercept_val = popt4[1]
intercept_uncertainty = pstd4[1]

print("The relationship between pixels and wavelength is Wavelength(nm) = ({} ±␣
  ↪{}) * Pixel + ({} ± {})".format(slope_val, slope_uncertainty, intercept_val,␣
  ↪intercept_uncertainty))
```

```
The relationship between pixels and wavelength is Wavelength(nm) =
(0.6899820219943742 ± 0.0022409874554043903) * Pixel + (303.05951346888287 ±
1.2523899847069118)
```

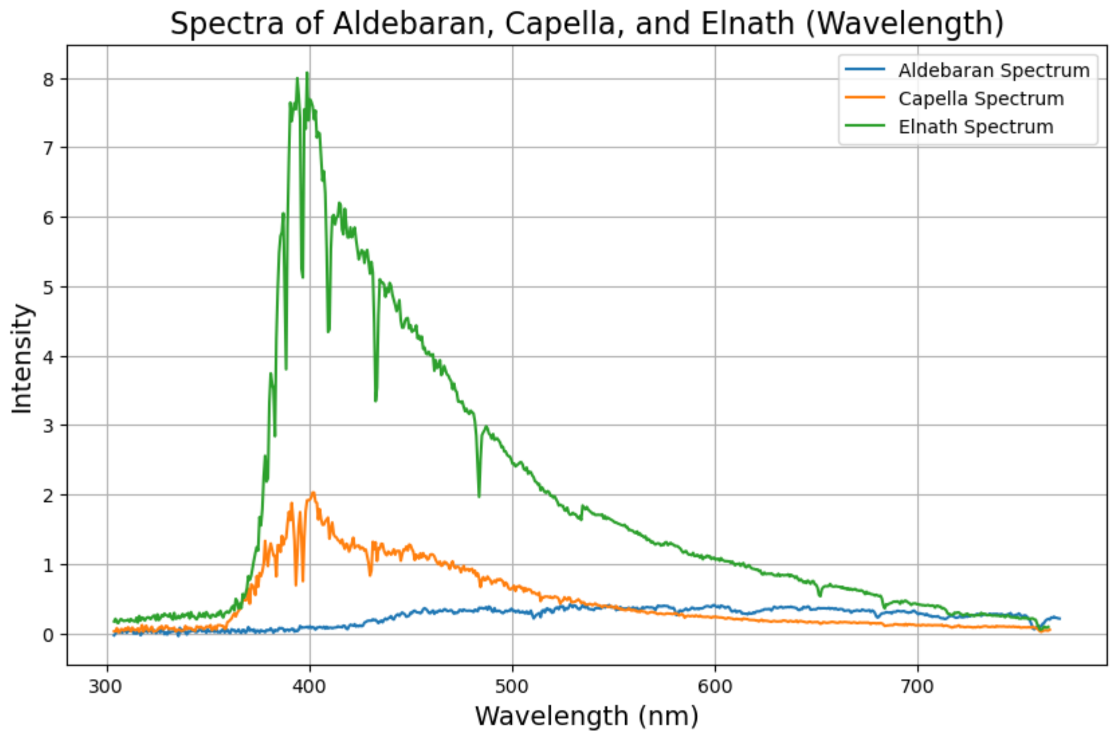
[38]:
```
data_aldebaran = np.loadtxt('Aldebran_Spectrum')
data_capella = np.loadtxt('Capella_Spectrum')
data_elnath = np.loadtxt('Elnath_Spectrum')

# Unpack data
pixels_aldebaran, intensity_aldebaran = data_aldebaran[:, 0], data_aldebaran[:,␣
↪1]
pixels_capella, intensity_capella = data_capella[:, 0], data_capella[:, 1]
pixels_elnath, intensity_elnath = data_elnath[:, 0], data_elnath[:, 1]

# Plotting the data
plt.figure(figsize=(10, 6))

# Plot for Aldebaran
plt.plot(pixels_aldebaran, intensity_aldebaran, label='Aldebaran Spectrum')

# Plot for Capella
plt.plot(pixels_capella, intensity_capella, label='Capella Spectrum')

# Plot for Elnath
plt.plot(pixels_elnath, intensity_elnath, label='Elnath Spectrum')

plt.xlabel('Pixel', fontsize=14)
plt.ylabel('Intensity', fontsize=14)
plt.title('Spectra of Aldebaran, Capella, and Elnath', fontsize=16)
plt.legend()
plt.grid()
plt.show()
```



[39]:

```
# Assuming you have the slope and intercept from the previous calculations
slope = popt4[0]
intercept = popt4[1]

# Convert the pixels to wavelength using the conversion formula
wavelength_aldebaran = slope * pixels_aldebaran + intercept
wavelength_capella = slope * pixels_capella + intercept
wavelength_elnath = slope * pixels_elnath + intercept

# Plotting the data with wavelength on the x-axis
plt.figure(figsize=(10, 6))

# Plot for Aldebaran
plt.plot(wavelength_aldebaran, intensity_aldebaran, label='Aldebaran Spectrum')

# Plot for Capella
plt.plot(wavelength_capella, intensity_capella, label='Capella Spectrum')

# Plot for Elnath
plt.plot(wavelength_elnath, intensity_elnath, label='Elnath Spectrum')

plt.xlabel('Wavelength (nm)', fontsize=14)
plt.ylabel('Intensity', fontsize=14)
```

```python
plt.title('Spectra of Aldebaran, Capella, and Elnath (Wavelength)', fontsize=16)
plt.legend()
plt.grid()
plt.show()
```



[40]:
```python
import numpy as np

# Constants for calculations
b = 2.8977729e-3  # Wien's displacement constant in meters kelvin
speed_of_light = 3.0e8  # speed of light in meters per second

# Temperature values for different star types
temp_aldebaran = 4050  # K5 star
temp_capella = 5010  # G5 star
temp_elnath = 12700  # B7 star

# Calculate the original wavelengths using Wien's displacement law
wavelength_aldebaran = b / temp_aldebaran
wavelength_capella = b / temp_capella
wavelength_elnath = b / temp_elnath

# Assume you have the necessary data and functions to calculate the temperature
↪and velocity for each star
```

```python
# For example, you might have the peak wavelength, observed change, and functions for spectral analysis

# Constants for calculations
b = 2.8977729e-3  # Wien's displacement constant in meters kelvin
speed_of_light = 3.0e8  # speed of light in meters per second


peak_wavelength_aldebaran_m = np.argmax(intensity_aldebaran) / 1e9  # Convert from nanometers to meters
observed_change_aldebaran = peak_wavelength_aldebaran_m - wavelength_aldebaran # Example observed change for Aldebaran

peak_wavelength_capella_m = np.argmax(intensity_capella) / 1e9  # Convert from nanometers to meters
observed_change_capella = peak_wavelength_capella_m - wavelength_capella  # Example observed change for Capella

peak_wavelength_elnath_m = np.argmax(intensity_elnath) / 1e9  # Convert from nanometers to meters
observed_change_elnath = peak_wavelength_elnath_m - wavelength_elnath  # Example observed change for Elnath


velocity_aldebaran=observed_change_aldebaran*speed_of_light/wavelength_aldebaran
velocity_capella=observed_change_capella*speed_of_light/wavelength_capella
velocity_elnath=observed_change_elnath*speed_of_light/wavelength_elnath

temperature_aldebaran=b/peak_wavelength_aldebaran_m
temperature_capella=b/peak_wavelength_capella_m
temperature_elnath=b/peak_wavelength_elnath_m

# Print the results
print("Estimated Temperature and Velocity of Aldebaran:")
print("Temperature: ", temperature_aldebaran, "Kelvin")
print("Velocity: ", velocity_aldebaran, "m/s")

print("Estimated Temperature and Velocity of Capella:")
print("Temperature: ", temperature_capella, "Kelvin")
print("Velocity: ", velocity_capella, "m/s")

print("Estimated Temperature and Velocity of Elnath:")
print("Temperature: ", temperature_elnath, "Kelvin")
print("Velocity: ", velocity_elnath, "m/s")

print(np.argmax(intensity_capella))
```

```
Estimated Temperature and Velocity of Aldebaran:
Temperature:  8888.873926380369 Kelvin
Velocity:  -163312269.9159758 m/s
Estimated Temperature and Velocity of Capella:
Temperature:  20406.851408450704 Kelvin
Velocity:  -226348265.593898 m/s
Estimated Temperature and Velocity of Elnath:
Temperature:  20998.35434782609 Kelvin
Velocity:  -118557209.9180029 m/s
142
```

[ ]:

[86]:
```
wavelength_aldebaran = slope * pixels_aldebaran + intercept
def uncert_mx(m,x,x_u,m_u):
    uncert_mx = m*x*((x_u/x)**2+(m_u/m)**2)**0.5
    return uncert_mx

def uncert_wavelength(uncert_mx, uncert_intercept):
    uncert_w = ((uncert_mx)**2 + (uncert_intercept)**2)**0.5
    return

uncert_mx_aldebaran = uncert_mx(slope,peak_wavelength_aldebaran_m,0.5,pstd4[0])
uncert_w_aldebaran = ((uncert_mx_aldebaran)**2 + (pstd4[1])**2)**0.5
print(uncert_mx_aldebaran)
print(uncert_w_aldebaran )

uncert_mx_capella = uncert_mx(slope,peak_wavelength_capella_m,0.5,pstd4[0])
uncert_w_capella = ((uncert_mx_capella)**2 + (pstd4[1])**2)**0.5
print(uncert_mx_capella)
print(uncert_w_capella)

uncert_mx_elnath = uncert_mx(slope,peak_wavelength_elnath_m,0.5,pstd4[0])
uncert_w_elnath  = ((uncert_mx_elnath)**2 + (pstd4[1])**2)**0.5
print(uncert_mx_elnath)
print(uncert_w_elnath)
```

```
0.34499101099718715
1.2990379022426712
0.3449910109971871
1.2990379022426712
0.3449910109971871
1.2990379022426712
```

[87]:
```
uncert_t_aldebaran = temperature_aldebaran*peak_wavelength_aldebaran_m/
 ↪uncert_w_aldebaran
```

```
uncert_t_capella = temperature_capella*peak_wavelength_capella_m/
 ↪uncert_w_capella
uncert_t_elnath = temperature_elnath*peak_wavelength_elnath_m/uncert_w_elnath
print(uncert_t_aldebaran)
print(uncert_t_capella)
print(uncert_t_elnath)
```

```
0.002230706967823847
0.002230706967823847
0.002230706967823847
```

[ ]:
```
uncert_velocity_a = uncert_w_aldebaran*c/
```