Draft version November 7, 2018
Typeset using LaTeX **preprint** style in AASTeX62

Lab 2: Astronomical Spectroscopy

Jessica Birky, Julian Beaz-Gonzalez, Russell Van-Linge

ABSTRACT

In this lab...

1. INTRODUCTION

Astronomical spectroscopy, components of spectrographs, types of spectrographs

2. OBSERVATIONS

Describe ocean lab and KAST spectrographs specifications and diagrams

3. DATA REDUCTION & METHODS

Centroid routine, wavelength calibration, centroid error and calibration error
Bias subtraction and normalization

4. DATA ANALYSIS & MODELING

Show centroids, wavelength calibration plots, application of wavelength solution, plots of astronomical spectra

5. DISCUSSION

What kind of astronomical sources

6. CONCLUSION

7. AUTHOR CONTRIBUTIONS

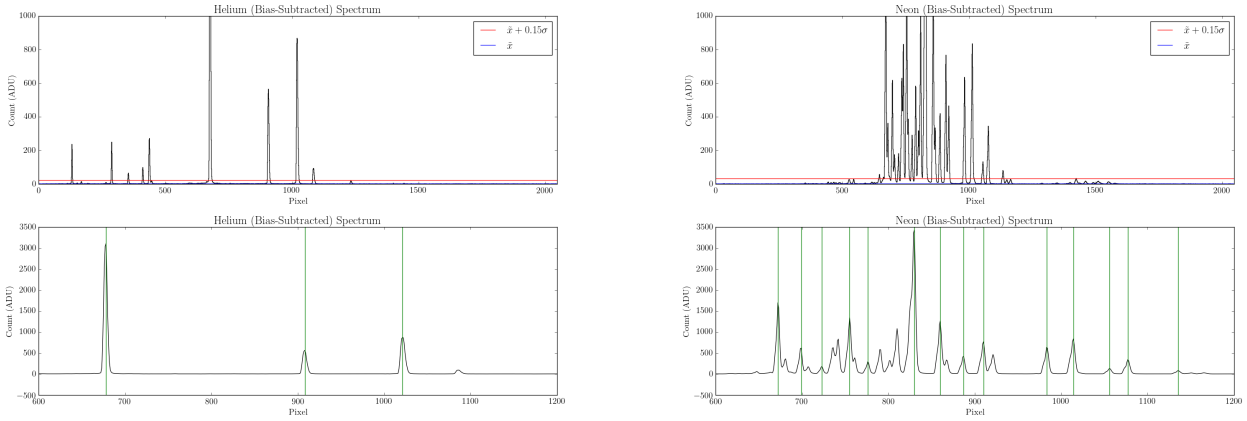

Figure 1. Centroid finding routine.

Corresponding author: Jessica Birky (A13002163)
jbirky@ucsd.edu

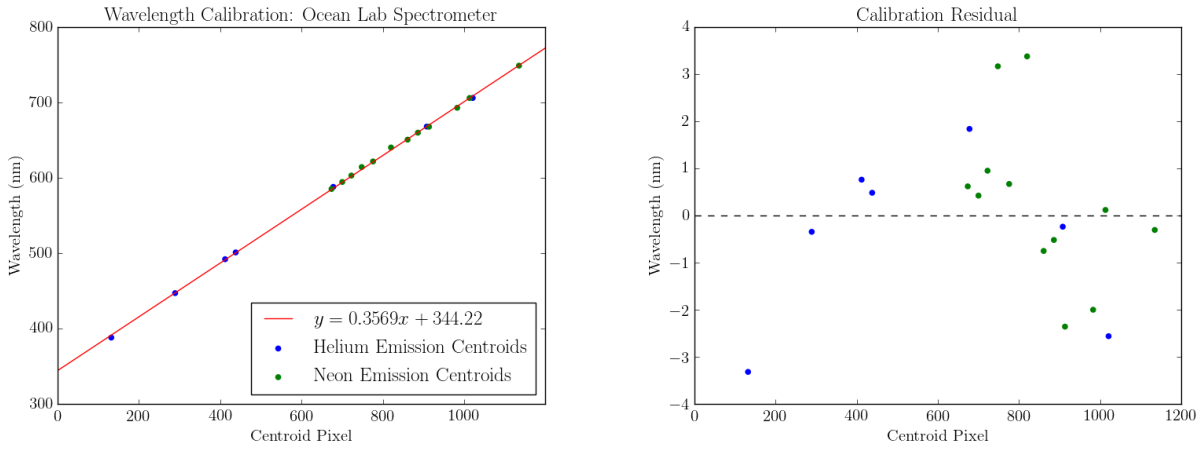

Figure 2. Wavelength calibration for the Ocean Lab spectrometer.

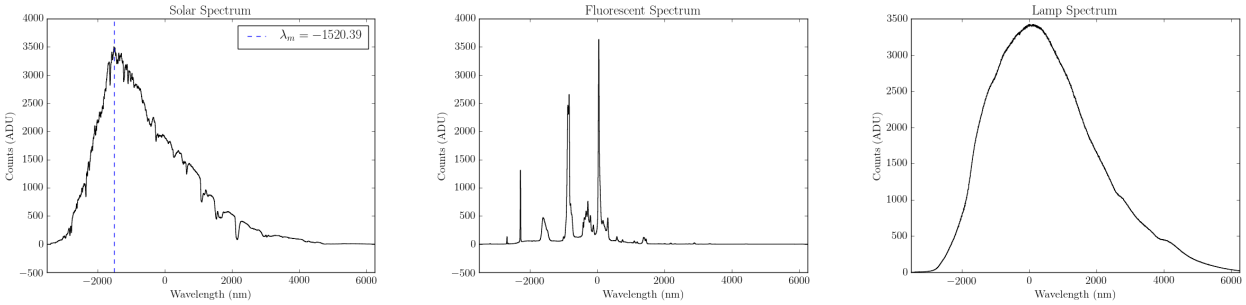

Figure 3. Three spectra taken on the Ocean Lab Spectrometer: the Sun, a fluorescent light bulb, and an incandescent lamp. Wavelength scale determined by calibration with emission peaks in Helium and Neon lamps.

8. APPENDIX

8.1. *Centroid Identification Routine*

```
def emission(data, **kwargs):
    thres = kwargs.get('thres', 1)

    x, y = np.array(data[0]), np.array(data[1])
    Npix = len(x)

    med = np.median(y)
    std = np.std(y)
    cut = med + thres*std

    count_cut = []
    for i in range(Npix):
        if y[i] >= cut:
            count_cut.append(i)

    emission = []
    arr = []
```

```
    for i in range(len(count_cut)-1):
        if (count_cut[i+1] - count_cut[i]) == 1:
            arr.append(count_cut[i])
        else:
            arr.append(count_cut[i])
            if len(arr) > 5:
                emission.append(np.array(arr))
            arr = []

return np.array(emission)


def centroid(data, feat_idx, **kwargs):

    x, y = np.array(data[0]), np.array(data[1])
    Npix = len(x)

    max_idx = []
    for feat in feat_idx:
        for idx in feat:
            if y[idx] == max(y[feat]):
                max_idx.append(idx)

    return np.array(max_idx)

centroid(data, emiss)
```

8.2. *Centroid Error*

```
#Produces float values for the centroids, calculating their error and width
centroids, errors, widths, intensities = [], [], [], []
for i in np.arange(len(cent)):
    inte = []
    for a in emiss[i]:
        inte.append(data[1][a])
    centr_f = sum(emiss[i]*inte)/sum(inte)
    err_f = sum(inte*((emiss[i]-centr_f)**2))/(sum(inte))**2
    width_f = sum(inte*((emiss[i]-centr_f)**2))/sum(inte)

    centroids.append(centr_f)
    errors.append(err_f)
    widths.append(width_f)
    intensities.append(max(data[1][emiss[i]]))
```

8.3. *Wavelength Calibration*

```
def linear_regression(x, y):

    A = np.array([[np.sum(x**2), np.sum(x)], \
                  [np.sum(x), len(x)]])
    a = np.array([np.sum(x*y), np.sum(y)])

    return np.dot(np.linalg.inv(A), a)
```