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

Kirchoff's laws of spectral formation state that there are three different types of

Section 2 covers the schematics and specifications of two different spectrographs: the Ocean Lab spectrometer, and the KAST spectrograph mounted on the Shane 3m Telescope on Lick Observatory.

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

$$\begin{bmatrix} m \\ c \end{bmatrix} = \begin{bmatrix} \sum x_i^2 & \sum x_i \\ \sum x_i & N \end{bmatrix}^{-1} \begin{bmatrix} \sum x_i y_i \\ \sum y_i \end{bmatrix} \tag{1}$$

$$\sigma^2 = \frac{1}{N-2} \sum_i [y_i - (mx_i + c)]^2 \tag{2}$$

$$\sigma_m^2 = \frac{N\sigma^2}{N\sum_i x_i^2 - \left(\sum_i x_i\right)^2} \tag{3}$$

$$\sigma_c^2 = \frac{\sigma^2 \sum_i x_i^2}{N\sum_i x_i^2 - \left(\sum_i x_i\right)^2} \tag{4}$$

4. DATA ANALYSIS & MODELING

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

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

spec_pixel	spec_intensity (ADU)	nist_wave (nm)	nist_intensity	element
132.3649283	236.48	388	60-300	He
288.9587617	249.46	447	25-200	He
412.1908462	98.46	492	20	He
437.7643945	271.16	501	100	He
677.5521828	3106.25	588	120-500	He
908.0143522	564.74	668	200	He
1021.697778	867.6	706	20-100	He
673.712	1696.45	585.25	200	Ne
700.126	617.38	594.48	50	Ne
722.492	181.31	602.99	100	Ne
747.974	1339.68	614.3	100	Ne
775.796	290.88	621.73	100	Ne
820.02	3428.72	640.22	200	Ne
860.822	1265.04	650.65	150	Ne
886.09	419.28	659.9	100	Ne
913.44	767.24	667.82	50	Ne
982.83	635.18	692.94	1000	Ne
1013.238	834.96	705.91	100	Ne
1134.814	80.86	748.87	300	Ne

spec_pixel	spec_intensity	ref_wave
23.50092286	30772.15763	326.105
184.1577234	37829.79081	361.051
229.5492333	40120.66215	365.015
477.9985327	11444.21891	404.656
639.6619601	30172.47648	435.833
949.4331504	35609.38425	479.992
1257.08592	5622.686548	505.882
1372.254406	10136.48802	546.074
1638.903321	10132.41605	587.562

5. DISCUSSION

What kind of astronomical sources

6. CONCLUSION

7. AUTHOR CONTRIBUTIONS

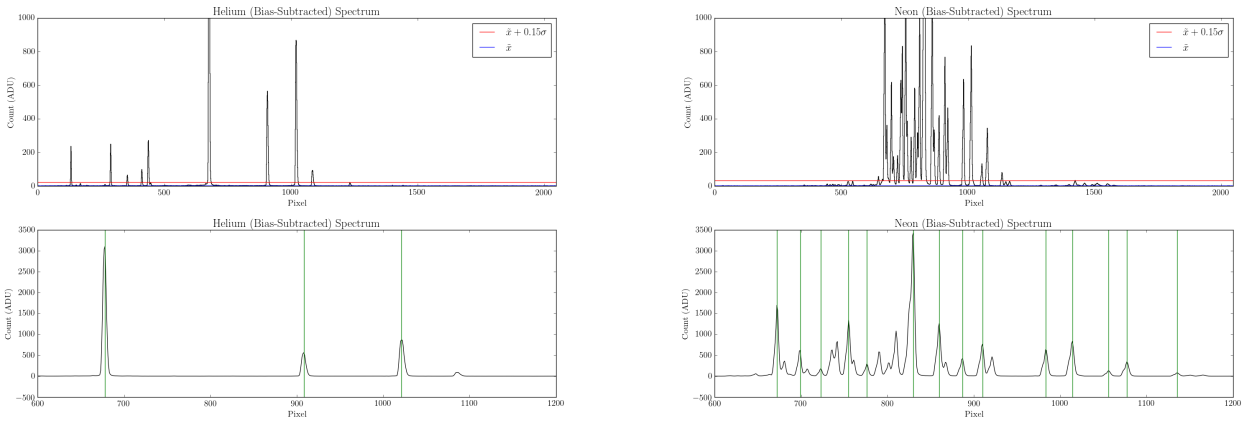

Figure 1. Centroid finding routine.

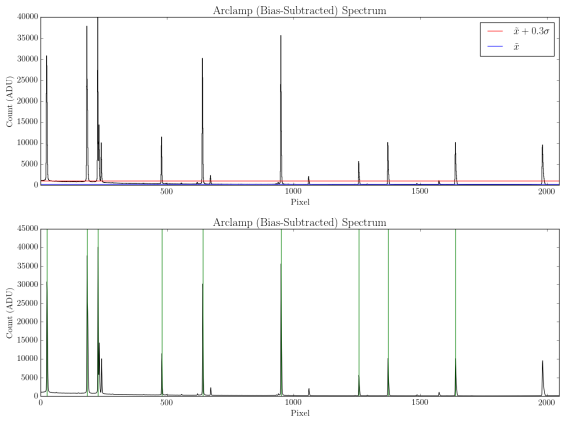

Figure 2. Centroid finding routine.

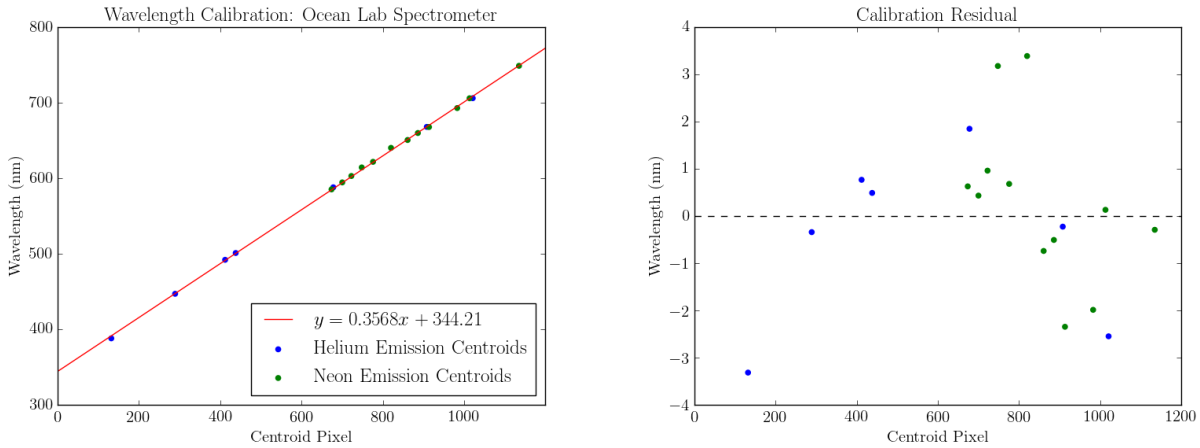

Figure 3. Wavelength calibration for the Ocean Lab spectrometer.

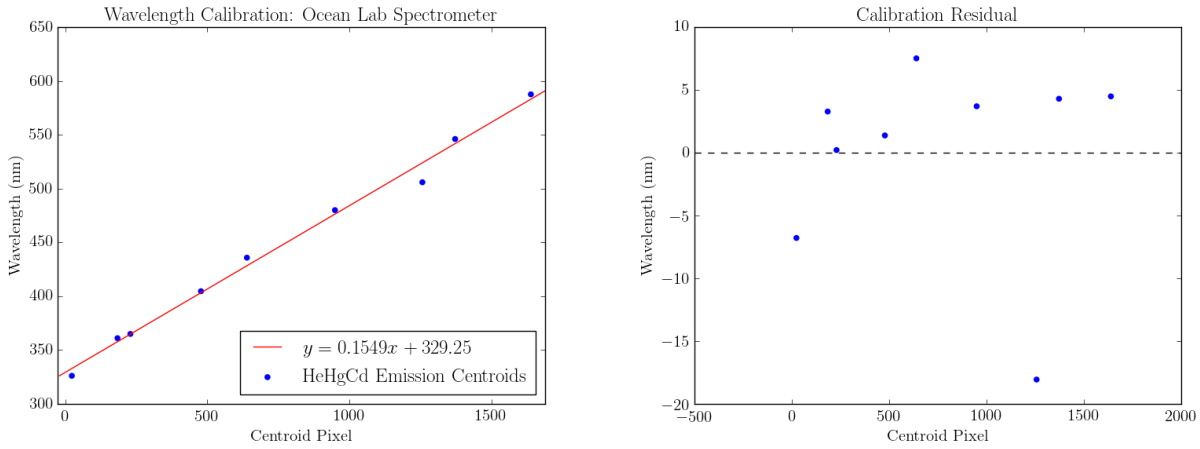

Figure 4. Wavelength calibration for the KAST spectrometer.

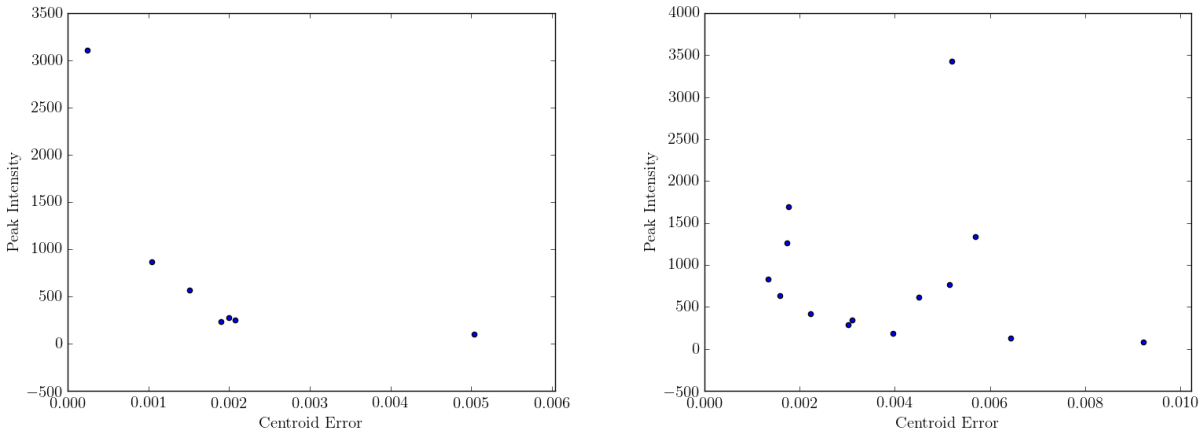

Figure 5. Error vs. Intensity.

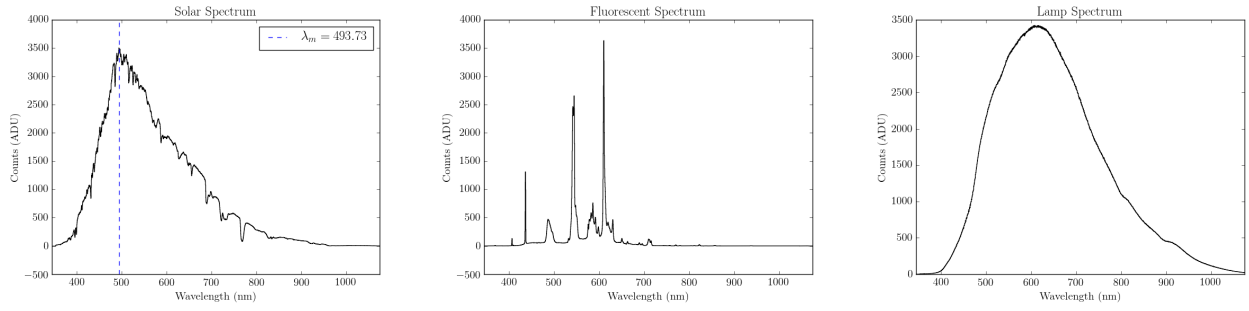

Figure 6. Three spectra taken on the Ocean Lab Spectrometer: the Sun, a fluorescent light bulb, and an incandescent lamp. Wavelength scale determined by calibration with emission peaks in Helium and Neon lamps.

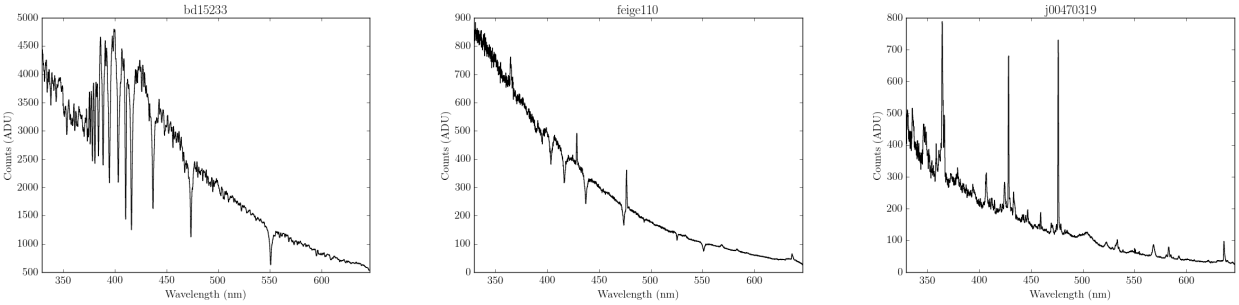

Figure 7. Three astronomical spectra taken on the KAST spectrograph.

8. APPENDIX

8.1. *Centroid Identification Routine*

```
def emission(data, **kwargs):
    """
    Input:  'data': 2D array [x,y]
            'thres': standard deviation threshold
    Output: 'emission': an array of arrays which contain the index of
                each emission which lies above some threshold cut
    """
    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)

"""
Produces float values for the centroids, calculating their error and width
Credit: Julian
```

```
"""
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]]))
```

8.2. *Wavelength Calibration*

```
def linear_regression(x, y):
    """
    Input:  x, y: 1D arrays
    Output: [m, c], [m_err, c_err]: slope and intercept best fit and error
    """
    N = len(x)
    x, y = np.array(x), np.array(y)

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

    fit = np.dot(np.linalg.inv(A), a)

    sig_sq = np.sum(y - (fit[0]*x + fit[1]))**2/(N + 2)
    m_err = np.sqrt(N*sig_sq/(N*np.sum(x**2) - (np.sum(x))**2))
    c_err = np.sqrt(sig_sq*np.sum(x**2)/(N*np.sum(x**2) - (np.sum(x))**2))
    err = np.array([m_err, c_err])

    return fit, err
```

8.3. *KAST Data Reduction*

```
"""
Credit: Russell
"""
def readData(folder):
    '''
    Reads all the fits files in a folder and creates a 3D array
    '''
    files = os.listdir(folder)
    array3D = []
    for ff in files:
        arr = fits.getdata(folder+ff,ignore_missing_end=True)
        array3D.append(arr)
    return np.array(array3D)

def combineFrame(data_array):
```

```
16      '''
17      Avergaes the 3D into 2D array
18      '''
19      return np.median(data_array,axis=0)
20
21  def slit_size(data = 'fits_file'):
22      '''
23      takes a fits file and finds the slit sized used
24      '''
25      dt = fits.open(data)
26      header = dt[0].header
27      print(header['SLIT_N'])
28
29
30  def norm_flat(bias_folder,flat_folder):
31      '''
32      Input the bias and flat folders and creates
33      bias,flats, and normalized flats
34      '''
35      flat3d = readData(flat_folder)
36      bias3d = readData(bias_folder)
37
38      flat2d = combineFrame(flat3d)
39      bias2d = combineFrame(bias3d)
40
41      norm_flat = (flat2d-bias2d)/np.median(flat2d-bias2d)
42      return norm_flat,flat2d,bias2d
43
44  def science_image(science_folder,flat_folder,bias_folder):
45      '''
46      Creates a science image and outputs the 2D array to be used to make spectra
47      '''
48      # create flats and bias
49      normflat, flat, bias = norm_flat(bias_folder,flat_folder)
50
51      science_3d = readData(science_folder) # reducing down to 2d
52      science_2d = combineFrame(science_3d)
53      science_final = (science_2d - bias)/normflat # subtracting bias and normalizing
54      # Plots the science image
55      plt.imshow(science_final,origin='lower',interpolation='nearest',\
56          cmap='gray',vmin=10,vmax=1000)
57      plt.show()
58      return science_final
59
60  def science_spectra(science_2d,start_slice,end_slice,file_name='file_name'):
61      '''
62      Creates a spectra by slicing up the science image at the input spatial
63      pixels, saves array, and outputs plot
64      '''
65      spectra = np.mean(science_2d[start_slice:end_slice,:],axis=0)
66      spectra = spectra[spectra>0]
67      x = np.arange(0,len(spectra))
68      plt.plot(x,spectra,color='black')
69      plt.show()
```

```
70      np.save('%s_spectra'%file_name,science_2d)
71      return spectra
```