

Experiment 3: Concave Grating Spectrometer

Ashmit Bathla, 210216.

10th Febuary 2025

```
In [1]: # import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import linregress
```

```
In [2]: # set default params for matplotlib :
plt.rcParams['font.size'] = 14
plt.rcParams['lines.linewidth'] = 2
plt.rcParams["figure.figsize"] = (8,5)
```

```
In [3]: # read all the data form the excel file :
data_dict = {f'Linear-Order-{i+1}': pd.read_excel('data.xlsx', sheet_name=i+1) for i in range(10)}
data_dict.update( {f'Angular-Order-{i-1}': pd.read_excel('data.xlsx', sheet_name=i-1) for i in range(1,11)} )
data_dict['Sodium'] = pd.read_excel('data.xlsx', sheet_name=-1)
```

```
In [4]: # list to store interpolated results :
sodium_wavelength = np.array([])
```

```
In [5]: # define function to plot the clabration curves and get the interpolated v
# the given data set

def calibrate_interpolate(x,y,order:int,linear:bool=False,p=None):
    """
    param x: indepenet variable of the data.
    param y: give data points of the function.
    param p: list of points to intepolate on.
    returns : interpolated values of y(p).
    """

    if( type(x) == list ):
        x = np.array( x )
    if( type(y) == list ):
        y = np.array( y )
    if linear :
        if( type(p) == list ):
            p = np.array(p)

    # get the linear interpolated results :
    if linear :
        q = np.interp( p , x , y )
```

```

# get the slope post linear regression :
regress = linregress( x , y )

# plot the curves :
plt.plot( x , y , linewidth = 3 )
plt.scatter( x , y , linewidths=2 , label = 'Recorded Data Point')
if linear :
    plt.scatter( p , q , color = 'r' , marker = '|' , s = 160 , label = 'Calibration Curve')
    plt.xlabel(r'Linear Distance from the Slit $X$ ($cm$)')
    plt.title(fr'$X$ vs $\lambda$ Calibration Curve of Hg for Order {order}')
else :
    plt.xlabel(r'Angular Distance from the Slit $\theta$ ($^\circ$)')
    plt.title(fr'$\theta$ vs $\lambda$ Calibration Curve of Hg for Order {order}')
plt.ylabel(r'Wavelength $\lambda$ ($nm$)')
plt.grid()
plt.legend()
plt.show()

# return the slope of the regression line
if linear :
    return q , regress[0]
else :
    return regress[0]

```

```

In [6]: print('-----Reading for the Sodium Lamp-----')
sodium_data = data_dict['Sodium']
sodium_data

```

-----Reading for the Sodium Lamp-----

```

Out[6]:

```

	Linear Distance (cm)	Color	Order
0	13.94	Yellow(1)	1
1	13.98	Yellow(2)	1
2	29.23	Yellow(1)	2
3	29.25	Yellow(2)	2

```

In [7]: print('-----Linear Readings for Order 1 Bands of Hg-----')
display( data_dict['Linear-Order-1'] )

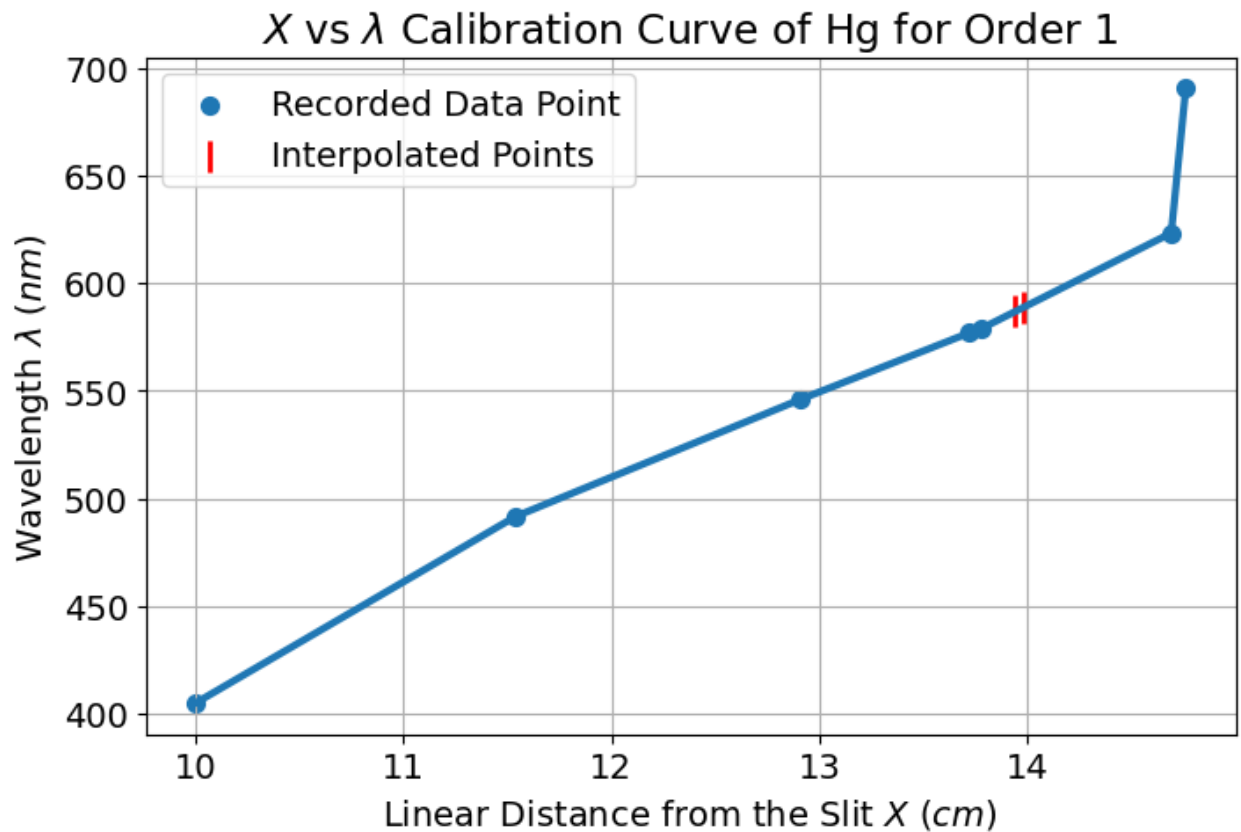
p = sodium_data['Linear Distance (cm)'].to_numpy()[1:2]
q , slope_order1 = calibrate_interpolate(
    x = data_dict['Linear-Order-1']['Linear Distance (cm)'].to_numpy() ,
    y = data_dict['Linear-Order-1']['Wavelength (nm)'].to_numpy() , order
    linear=True , p = p
)

sodium_wavelength = np.append( sodium_wavelength , q )
q

```

-----Linear Readings for Order 1 Bands of Hg-----

	Linear Distance (cm)	Color	Wavelength (nm)
0	10.00	Violet[Deep]	404.7
1	11.54	Blue/Green	491.6
2	12.91	Green	546.1
3	13.72	Yellow-Orange(1)	577.0
4	13.78	Yellow-Orange(2)	579.1
5	14.69	Red(1)	623.4
6	14.76	Red(2)	690.8



Out[7]: array([586.88901099, 588.83626374])

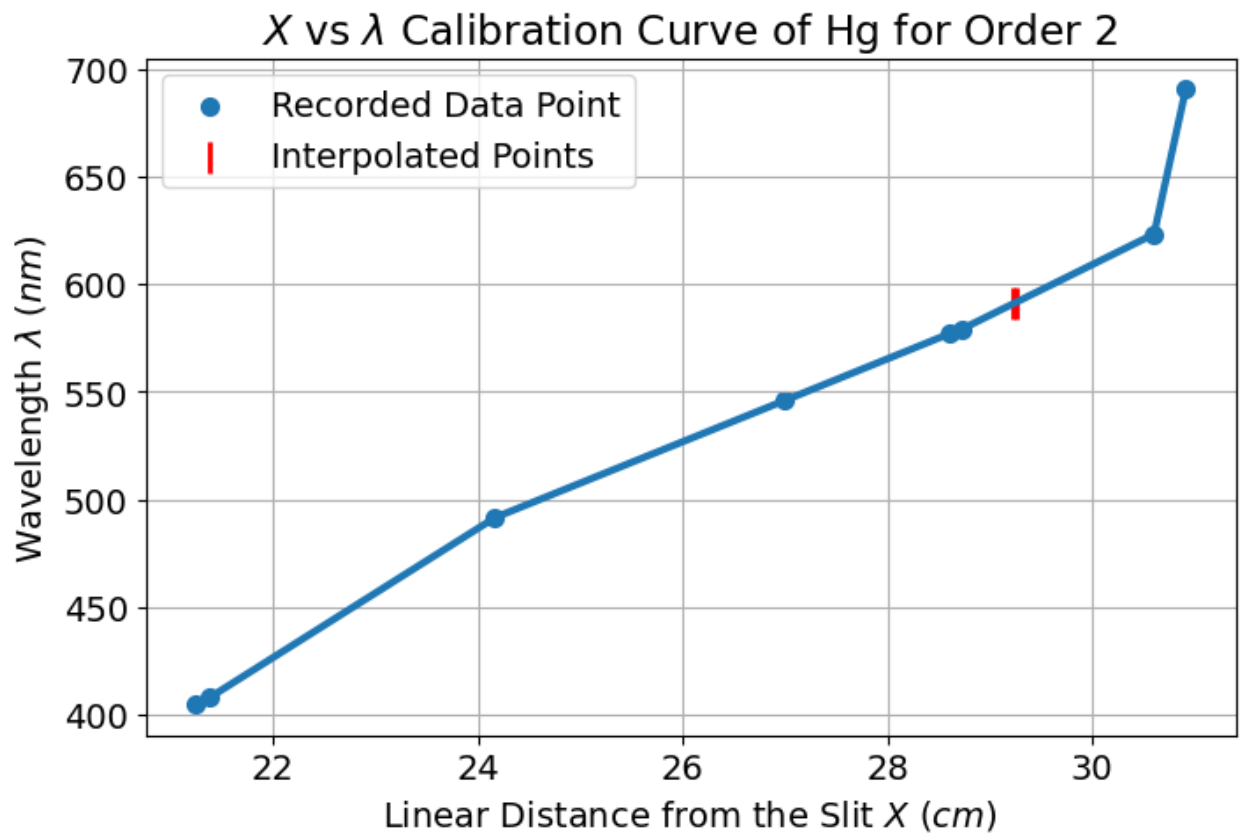
```
In [8]: print('-----Linear Readings for Order 2 Bands of Hg-----')
display( data_dict['Linear-Order-2'] )

p = sodium_data['Linear Distance (cm)'].to_numpy()[2:]
q , slope_order2 = calibrate_interpolate(
    x = data_dict['Linear-Order-2']['Linear Distance (cm)'].to_numpy() ,
    y = data_dict['Linear-Order-2']['Wavelength (nm)'].to_numpy(), order
    linear = True , p = p
)

sodium_wavelength = np.append( sodium_wavelength , q )
q
```

-----Linear Readings for Order 2 Bands of Hg-----

	Linear Distance (cm)	Color	Wavelength (nm)
0	21.24	Violet[Deep]	404.7
1	21.38	Violet	407.8
2	24.17	Blue/Green	491.6
3	27.00	Green	546.1
4	28.60	Yellow-Orange(1)	577.0
5	28.72	Yellow-Orange(2)	579.1
6	30.60	Red(1)	623.4
7	30.91	Red(2)	690.8



```
Out[8]: array([591.11755319, 591.58882979])
```

since $(d\lambda/dx)_{\text{order}n} \propto 1/n$ we expect :

$$\zeta = \frac{(d\lambda/dx)_{\text{order}1}}{(d\lambda/dx)_{\text{order}2}} = \frac{2}{1}$$

```
In [9]: slope_order1/slope_order2
```

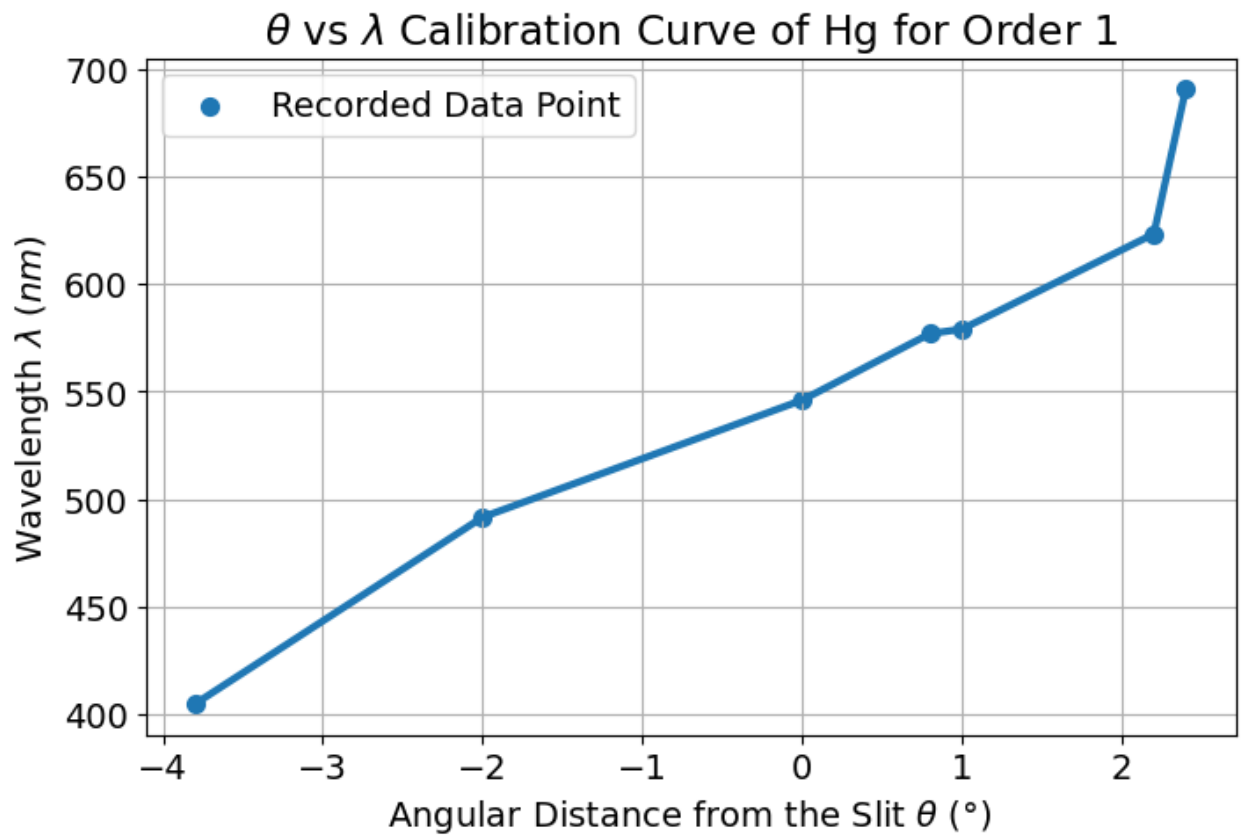
```
Out[9]: 2.011690000313097
```

we get $\zeta = 2.011690000313097$ which is highly accurate

```
In [10]: print('-----Angular Readings for Order 1 Bands of Hg-----')
          data_dict['Angular-Order-1']
```

```
slope_order1 = calibrate_interpolate(
    x = -data_dict['Angular-Order-1']['Angular Distance (*)'].to_numpy()
    y = data_dict['Angular-Order-1']['Wavelength (nm)'].to_numpy(), order
)
```

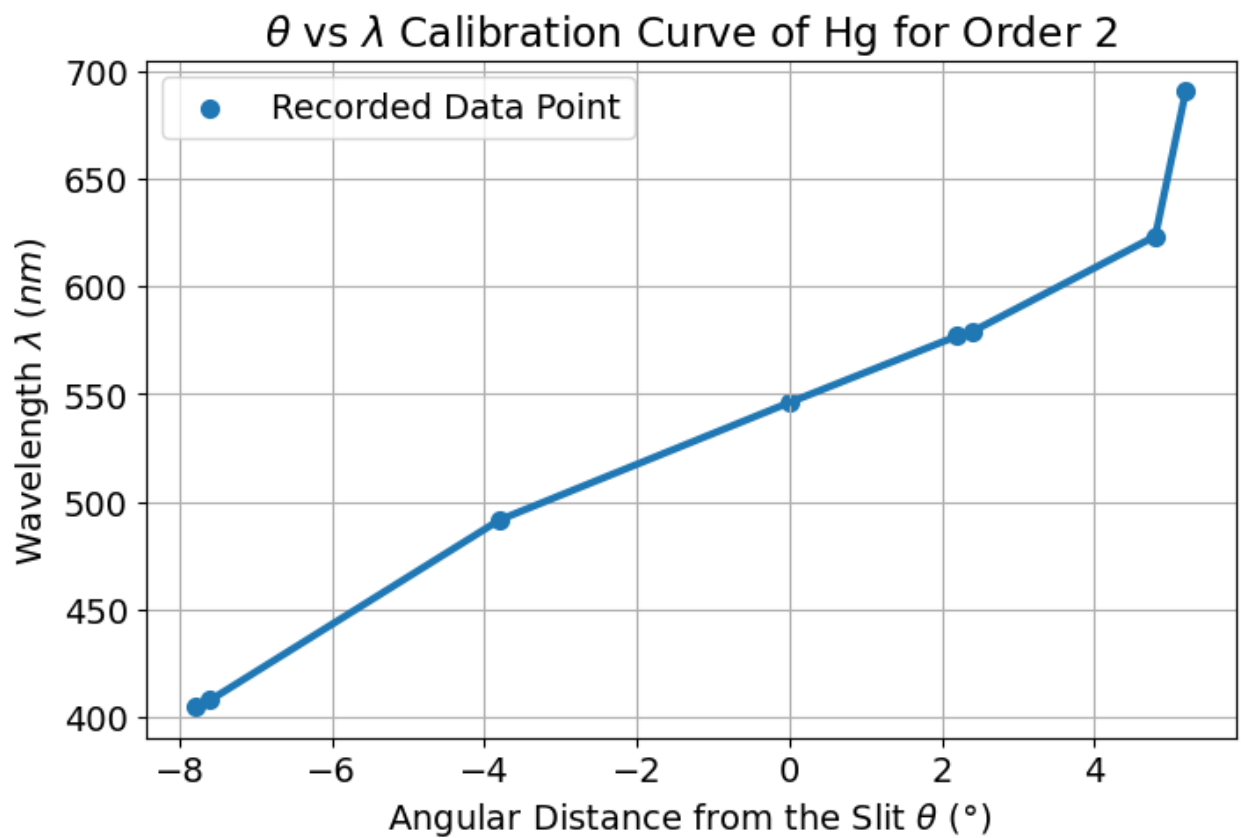
-----Angular Readings for Order 1 Bands of Hg-----



```
In [11]: print('-----Angular Readings for Order 2 Bands of Hg-----')
data_dict['Angular-Order-2']

slope_order1 = calibrate_interpolate(
    x = -data_dict['Angular-Order-2']['Angular Distance (*)'].to_numpy()
    y = data_dict['Angular-Order-2']['Wavelength (nm)'].to_numpy(), order
)
```

-----Angular Readings for Order 2 Bands of Hg-----



here too $(d\lambda/d\theta)_{\text{order } n} \propto 1/n$ we expect :

$$\zeta = \frac{(d\lambda/d\theta)_{\text{order1}}}{(d\lambda/d\theta)_{\text{order2}}} = \frac{2}{1}$$

In [12]: `slope_order1/slope_order2`

Out[12]: 0.7409141340876566

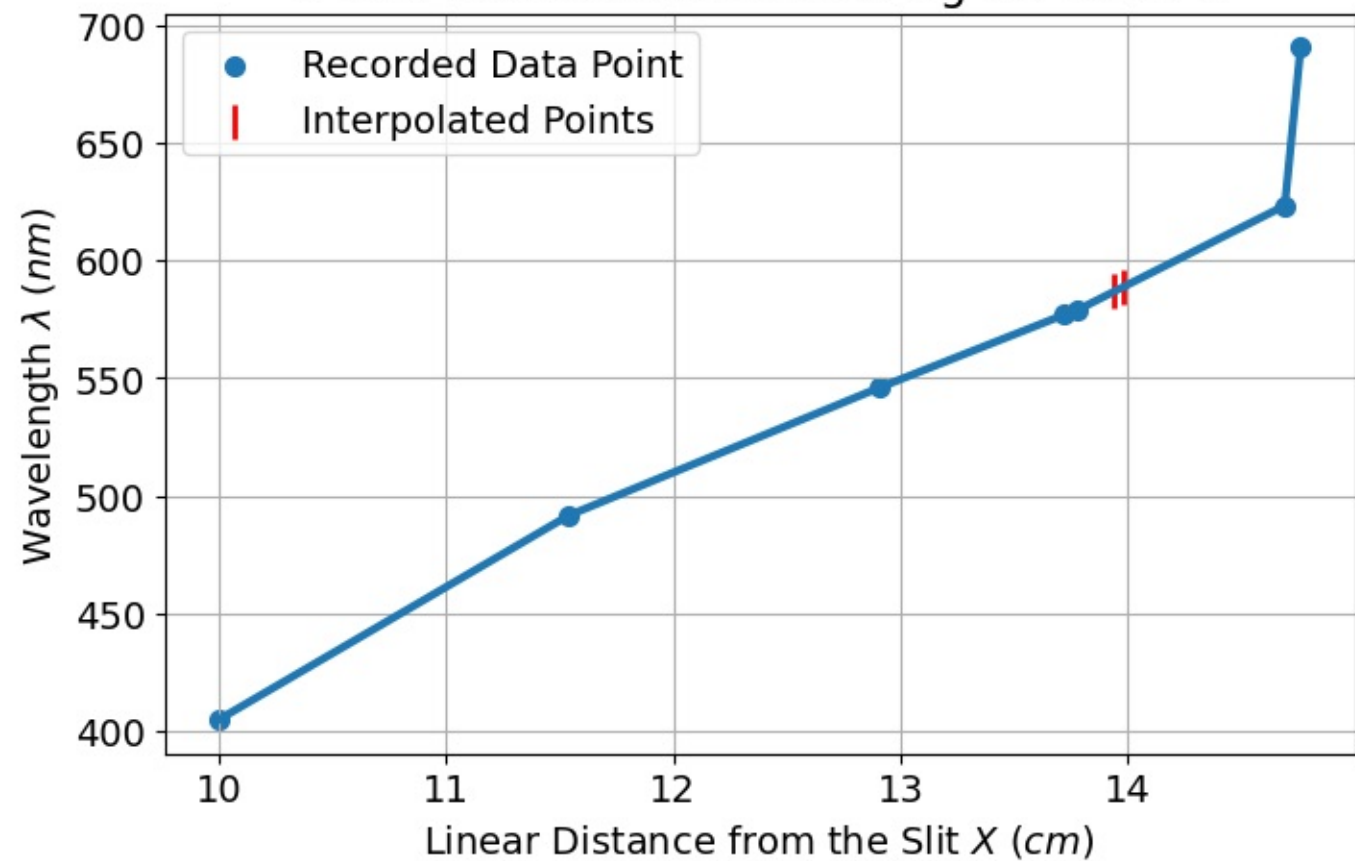
we get $\zeta = 0.7409141340876566$ which is somewhat inaccurate

In [13]: `sodium_data['Calculated Wavelength (nm)'] = sodium_wavelength
sodium_data`

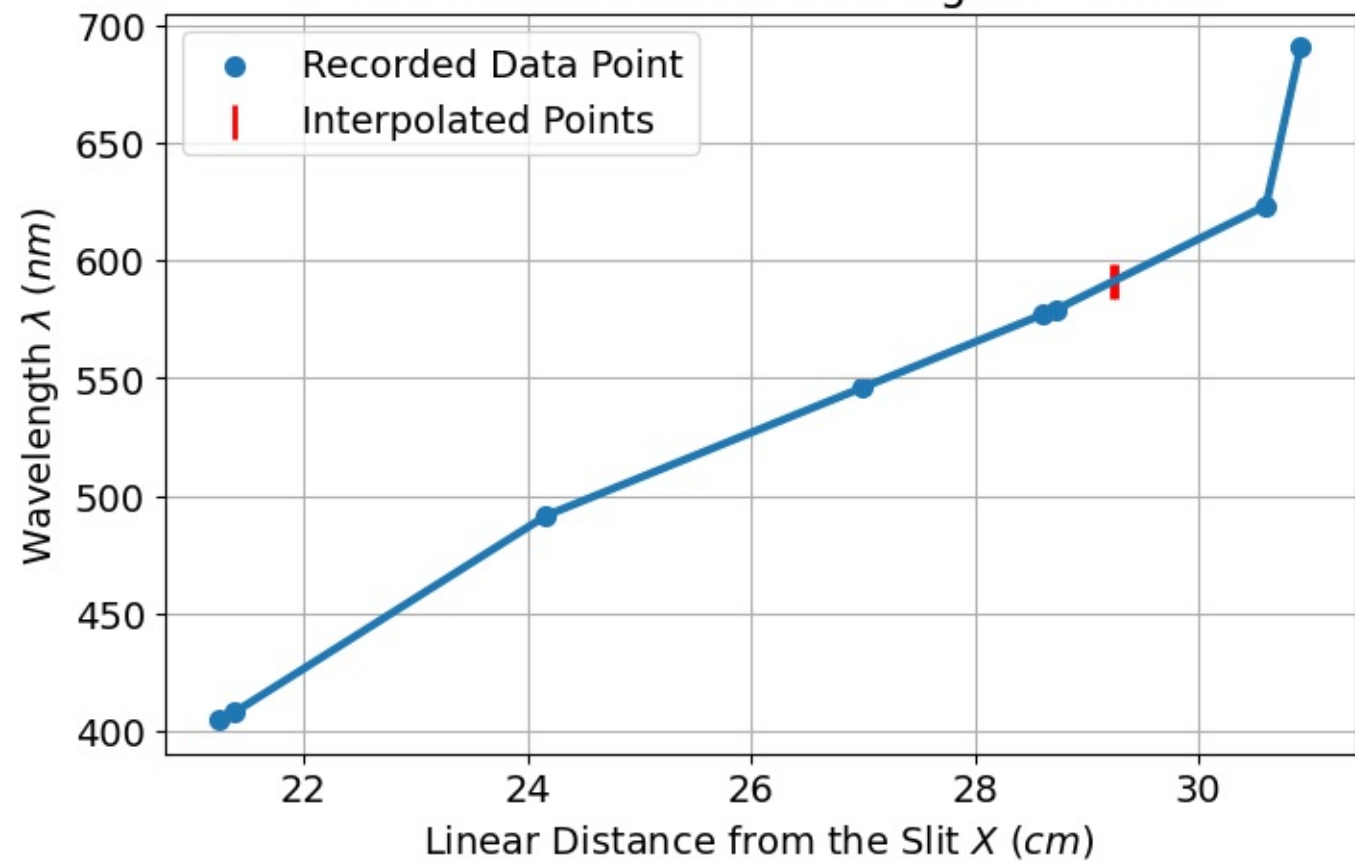
Out[13]:

	Linear Distance (cm)	Color	Order	Calculated Wavelength (nm)
0	13.94	Yellow(1)	1	586.889011
1	13.98	Yellow(2)	1	588.836264
2	29.23	Yellow(1)	2	591.117553
3	29.25	Yellow(2)	2	591.588830

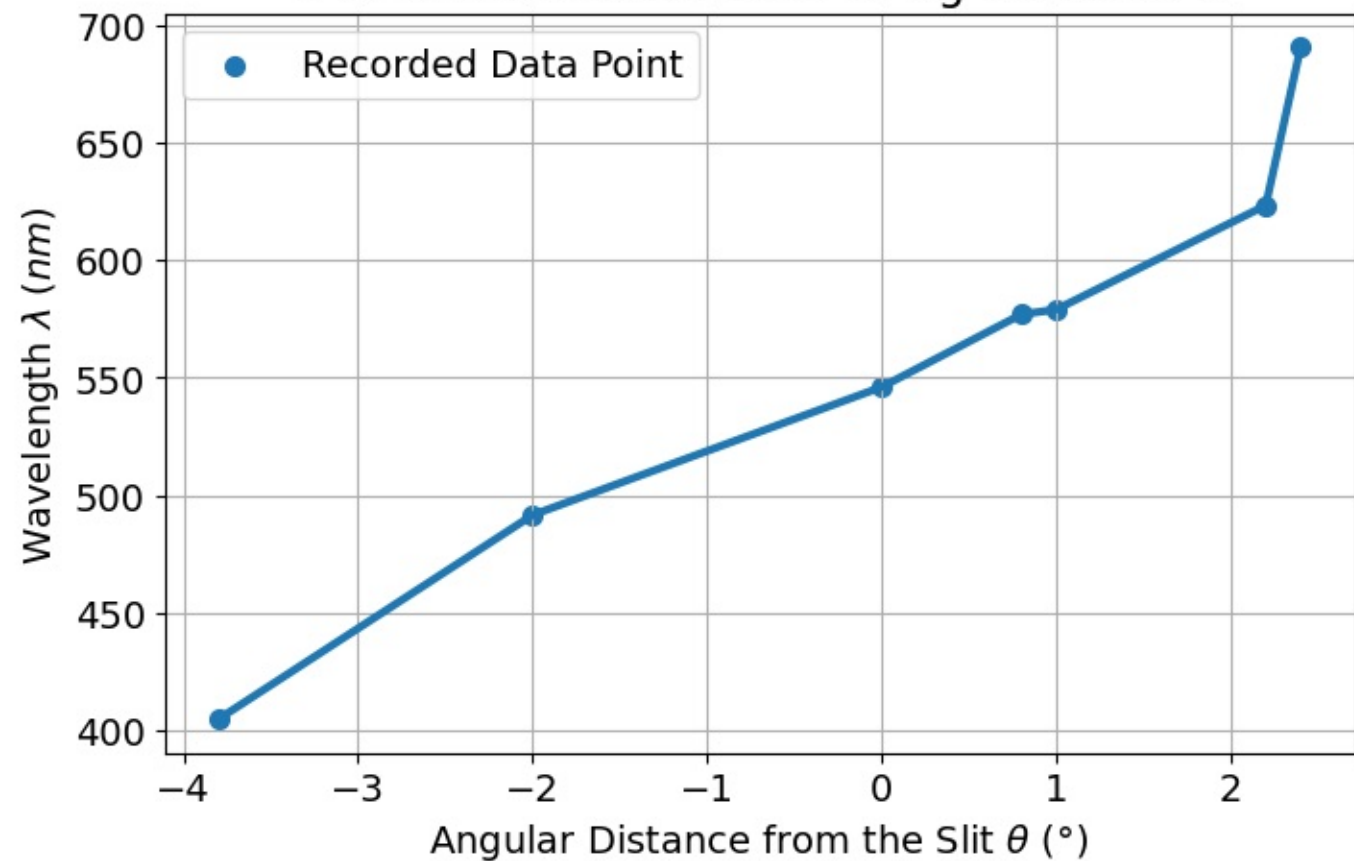
X vs λ Calibration Curve of Hg for Order 1



X vs λ Calibration Curve of Hg for Order 2



θ vs λ Calibration Curve of Hg for Order 1



θ vs λ Calibration Curve of Hg for Order 2

