refractiveIndex

November 27, 2017

1 Measuring The Refractive Index of Air

Micheal Jones, University of Southern Maine, Dept. of Physics

```
In [1]: import numpy as np
        from io import StringIO
        import matplotlib.pyplot as plt
        from scipy.optimize import curve_fit
        % matplotlib inline
        from matplotlib import rc
        rc('text', usetex=True)
        from pylab import rcParams
        rcParams['figure.figsize'] = 10,8
```

With the necessary libraries imported above, I will load my data in the next cell and assign my variable names by slicing the imported data in the following cell.

```
In [2]: data = np.loadtxt('michelsonData.txt', skiprows=1)
```

```
In [3]: p_0 = data[:,0]
        p_f = data[:,1]
        deltaP = data[:,2]
        deltaN = data[:,3]
        l = 0.208
        extP = 763.56
```

Next I will use a loop to calculate and store the values I want to use for plotting on the x and y axis. In the following cell I will then define a function I will use to fit my data and one I will use to evaluate and disply my uncertainty values.

```
In [4]: x = np.zeros(len(deltaP))
        y = np.zeros(len(deltaN))
        i = 0
        for i in range(0,40):
            x[i] = deltaP[i]/extP
            y[i] = deltaN[i]/(2*l)
```

```
In [5]: def linFit(t, m):
            return m*t

        def mError(n, p):
            return (0.000251)*np.sqrt((1.0/n)**2 + (1.0/p)**2 + 0.00037)
```

In the cells below I use SciPy to evaluate my best fit parameters and the deviation in my slope based on the data, then I call the error function defined in the above cell to calculate the error in the value I am actually after, the refractive index of air, or n, and come up with values for x and y error bars using another loop.

```
In [6]: fitParam, fitCovariance = curve_fit(linFit, x, y)
        zeroInterceptSigma = np.sqrt([fitCovariance[0,0]])
```

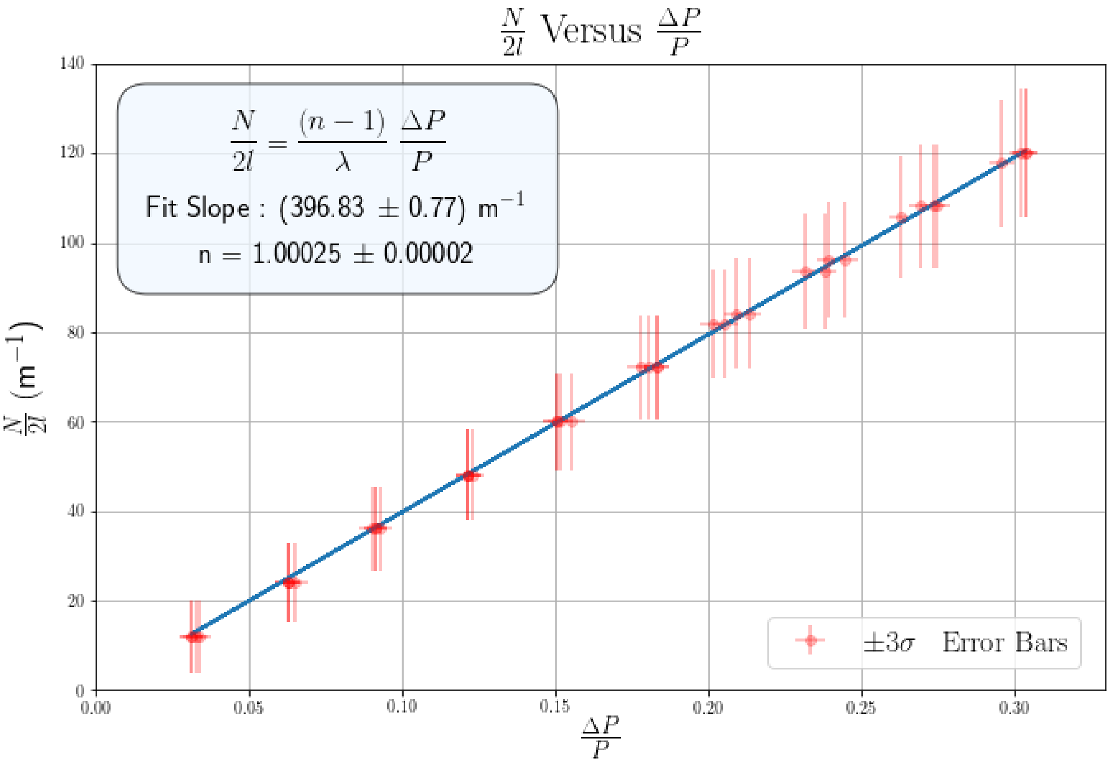
```
In [8]: deltaM = np.zeros(len(deltaP))
        deltaY = np.zeros(len(deltaP))
        deltaX = np.zeros(len(deltaP))
        j = 0
        for j in range(0,len(deltaP)):
            deltaM[j] = mError(deltaN[j], deltaP[j])
            deltaY[j] = ((deltaN[j]+1.0)/(2*(l-0.004)))-y[j]
            deltaX[j] = ((deltaP[j]+1.0)/(extP-0.1))-x[j]

        #print(np.mean(deltaM))
```

```
1.61738655193e-05
```

Now I plot the data in accordance with my formatting preferences and display the figure.

```
In [8]: plt.figure(
            figsize=(11,7)
            )
        plt.plot(
            x, linFit(
                x, fitParam[0]
                ),
            linewidth = 1.75
            )
        plt.errorbar(
            x, y,
            fmt = 'ro',
            xerr = 3*(deltaX),
            yerr = 3*(deltaY),
            linewidth = 2.0,
            label = r"$\displaystyle \pm 3\sigma\;\;$ \textrm{Error Bars}",
            alpha = 0.25
            )
        plt.title(
            r'$\frac{N}{2l}$ \textrm{Versus} $\frac{\Delta P}{P}$',
            fontsize = 24
            )
        plt.ylabel(
            r'$\frac{N}{2l}$ \textrm(m$^{-1}$)',
            fontsize = 21
            )
        plt.xlabel(
            r'$\frac{\Delta P}{P}$',
            fontsize = 21
            )
        plt.grid(True)
        plt.legend(
            bbox_to_anchor=(0.99, 0.14),
            fontsize= 18
            )
        props = dict(
            boxstyle='round, pad=1.0',
            facecolor='aliceblue',
            alpha=0.75
            )
        plt.text(
            0.017, 90.79,
            (r'$$ \frac{N}{2l} = \frac{(n-1)}{\lambda} \ \frac{\Delta P}{P}$$'
             "\n Fit Slope : (%.2f\t $\pm$ %.2f) m$^{-1}$ \n n = %.5f $\pm$ %.5f"
            % (fitParam[0], zeroInterceptSigma[0], 1.00025, np.mean(deltaM))),
            fontsize= 18,
            bbox=props,
            weight='bold',
            linespacing=1.7,
            multialignment='center'
            )
        plt.xlim(
            0,0.33
            )
        plt.ylim(
            0,140
            )
        plt.savefig(
            'michelsonGraphZeroInt.png',
            figsize=(6,4),
            dpi=400
            )
        plt.show()
```


$\frac{N}{2l}$ Versus $\frac{\Delta P}{P}$
$\frac{N}{2l} = \frac{(n-1)}{\lambda}\frac{\Delta P}{P}$
Fit Slope : (396.83 ± 0.77) m^{-1}
n = 1.00025 ± 0.00002
±3σ Error Bars
$\frac{N}{2l}$ (m^{-1})
$\frac{\Delta P}{P}$
0
20
40
60
80
100
120
140
0.00
0.05
0.10
0.15
0.20
0.25
0.30