In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [8]:
```
# Part 1

distance_micron = np.array([36.5,37.0,36.5,35.9,37.6,37.0,37.0,37.0,3
8.0,38.5])
#Enter my measurements into an array for easy calculation

average_micron = np.average(distance_micron)
#take the average of the distances measured

lamdas_microns = 2.0 * average_micron / 100.0
#convert the distances measured into wavelengths using formula 5 on th
e lab sheet

lamda_Scaled = 10000 * lamdas_microns
#convert the micron measurement to angstroms

print lamda_Scaled, "Angstroms"
#known value = 6328 angstroms



# Error Part 1

distance_error = np.array([1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
1.0, 1.0])
#The error of each of our measurements was +- 1 micron

error_summed = np.sum(distance_error)
#square and sum each individual error (1^2 = 1 so I skipped the squari
ng)

total_distance_error = (np.sqrt(error_summed)/len(distance_error))
# The forumla for the error is square root of the sum
# of the absolute errors squared, divided by N. N = 10 here.

print total_distance_error, " microns = the total distance measurement
 error"
# now we have the error in the distances averaged, time to get the err
or in the wavelength
# the formula for lambda is 2 * distance / the number of counts
# inorder to determine lambda's error we need to do some calculations

distance_errorx2 = total_distance_error * 2
#first we double the error of the total distance as it is doubled in t
he formula

# Next we understand that counting error is the squareroot of the numb
er of counts, 100.

# with those tools we can use error propagation to find the error of l
```

```
ambda in microns
error_total_lambda = lamdas_microns * np.sqrt(((distance_errorx2/(2 *
average_micron))**2) + ((10.0/100.0)**2))
# This formula above basicly uses RMS error techniques to find the tot
al error in the equation

error_total_lambda_Angstroms = error_total_lambda * 10000
#next we scale from microns to angstroms

print lamda_Scaled, " +/- ", error_total_lambda_Angstroms, " Angstrom
s"
#finally we can print our final value out
```

```
7420.0 Angstroms
0.316227766017  microns = the total distance measurement error
7420.0  +/-  744.690539755  Angstroms
```

In [9]:

```
#Part 2 data analysis
distance_i = np.array([10.9, 10.9, 10.9, 10.9, 10.9, 10.9])
distnace_f = np.array([39.1, 39, 39, 39, 40, 38.7])
# create arrays of initial distance and final distances in microns

delta_distance = distnace_f - distance_i
#find the change in distance in microns

laser_lambda_avg = ((5889.9 + 5898.9)/2.0) * 0.0001
# calculate the average wavelength of the laser in microns

lambda_delta = (laser_lambda_avg ** 2) / (2 * delta_distance)
# This calculates delta lambda in microns

lambda_final = lambda_delta * 10**4
#Calculate the final values for lambda in Angstroms

print np.average(lambda_final)
#print the mean of the lambdas

# part 2 error
# This is just standard deviation of the mean

mean = np.average(lambda_final)
#take the mean of the lambda numbers

lambda_final_calculated = (lambda_final - mean) ** 2
# subtract the mean from the numbers and square

mean2 = np.average(lambda_final_calculated)
# take the mean of these new variables

final_error = np.sqrt(mean2)
# squareroot the new mean for the final error

print mean, " +/- ", final_error, "Angstroms"
```

```
61.5425525407
61.5425525407  +/-  0.869677089199 Angstroms
```

In [12]:
```
# part 3

Pressure_measured = np.array([21.0, 20.5, 21.0, 20.7, 20.7, 20.7,
20.7, 20.5])
Pressure_difference = Pressure_measured - 24.6
#compute the difference from the room pressure and the measured pressu
re/

fringes = np.array([15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0])
L = np.array([3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0])
Cm_converstion = 6.5 * 10.0**(-5)
#just defining some arrays here of data we took

n_air = 1 + ((fringes * Cm_converstion)/(2 * L * Pressure_difference))
 * 24.6
# This does the calculation for the index of air

delta_n_air = n_air - 1
#subtract 1 to find delta air

delta_average = np.average(delta_n_air)
# Take the average of the delta index numbers

print delta_average

# part 3 error
#This is just standard deviation of the mean

mean_n = np.average(delta_n_air)
#take the mean of all the n air numbers

n_final_calculated = (delta_n_air - mean_n) ** 2
# subtract the mean from the numbers and square

mean_n2 = np.average(n_final_calculated)
# take the mean of these new variables

final_error_n = np.sqrt(mean_n2)
# squareroot the new mean for the final error

print mean_n, " +/- ", final_error_n,
```

```
-0.00103385416667
-0.00103385416667  +/-  4.8688863909e-05
```

In [ ]: