Lab 1: Gas Constant Determination

Max Huggins – UCA Department of Chemistry

February 9, 2021

Abstract

The universal gas constant was found via the following reaction and the perfect gas law.

$$2\,HCl(aq) + Mg(s) \longrightarrow MgCl_{2(s)} + H_2(g) \quad (1)$$

$$PV = nRT \quad (2)$$

Here it can be found that any mass of magnesium is in a 1:1 mole ratio with the hydrogen gas formed and if pressure, temperature, and volume are also recorded for the formation of the gas then the gas constant can be determined. Four different values for R were determined. The first was obtained from low accuracy instruments, the second with higher accuracy instruments, the third from fitting volume as a function of moles using a class data set, and lastly an average value for R determined by the removal of outliers using a q-test. These values were found to be: $(9 \pm 4, 7.6 \pm 0.6, 6 \pm 4, 8.8 \pm .7)\frac{J}{molK}$ respectively.

1 Introduction and Objective

The purpose of this experiment was was to determine the universal gas constant through several methods using the perfect gas law. This law relies on two major assumptions:

- A gas is composed of molecules that are much smaller than the space between the molecules.
- Forces of attraction and repulsion between gas molecules are negligible.

And while these are not necessarily the case, they will provide quite good predictions for many scenarios. This experiment is also used as a method of expanding statistical analysis tools developed in previous coursework.

2 Procedure

2.1 Overview

The procedure for this lab is straightforward. A strip of magnesium metal is reacted with hydrochloric acid in excess. The hydrogen gas produced from the reaction was captured in a measuring vessel such as a graduated cylinder or buret. The temperature, atmospheric pressure, volume of hydrogen gas, and mass of magnesium were recorded in two trials. One trial with high accuracy measurement devices and the other with low accuracy measurement devices.

2.2 Details

This procedure was taken from the lab handout, *Determination of the Gas Constant and Propagation of Error.* Two apparatus were constructed using two 1000mL beakers, two test tubes (one large and one small), a transfer line with rubber stopper on the large test tube, a graduated cylinder a buret, and two thermometers. The test tubes were assembled such that the larger held the HCl and the smaller held the magnesium. The smaller one slid into the larger such that it would not react until the experimenter gave it a shake. The larger tube was capped off with a rubber stopper and had a transfer line to carry hydrogen gas through water in a beaker into either the graduated cylinder or buret depending on whether it was a high or low accuracy reading. A thermometer was placed inside the water where the gas would bubble through to record the temperature. Two trials were done with a high and low accuracy thermometer. The mass of magnesium was also measured on two separate scales for each trial. One recording an extra significant figure from the other. Once the experiment was initiated by tilting the test tubes to allow the magnesium and HCl to come in contact, the displacement of fluid was measured in the cylinders. A class data set was recorded and sent out for all of the high accuracy measurements.

3 Results and Analysis

All of my analysis was completed in Python using open-source software. Because of this I will not go too much into depth about the methods used and the prescriptions applied for each I will focus mainly on the data and the results. All of my scripting is attached in the appendix and I have tried to make my variable naming such that it is readable. The values I determined from the program are listed below:

1. $R_{low} = (9 \pm 4) J/molK$
2. $R_{high} = (7.6 \pm 0.6) J/molK$
3. $R_{slope} = (6.3 \pm 4) J/molK$
4. $R_{removedoutliers} = (8.8 \pm 0.7) J/molK$

Figure 1 is the plot from the V v. n graph. This was obtained by multiplying each n value by T/P, such that the slope of the linear regression function was simply the gas constant. Looking at this plot gives a clue as to why these numbers have such large uncertainties associated with them. I proceeded to write a function for a high and low value q-test on the averaged R values and found that one of the values was indeed an outlier. After that was removed, the value was much closer to the anticipated one. Next, a t-test was done to determine if the new value was statistically different. Using a 95% confidence interval, it was found that the value found was statistically different.

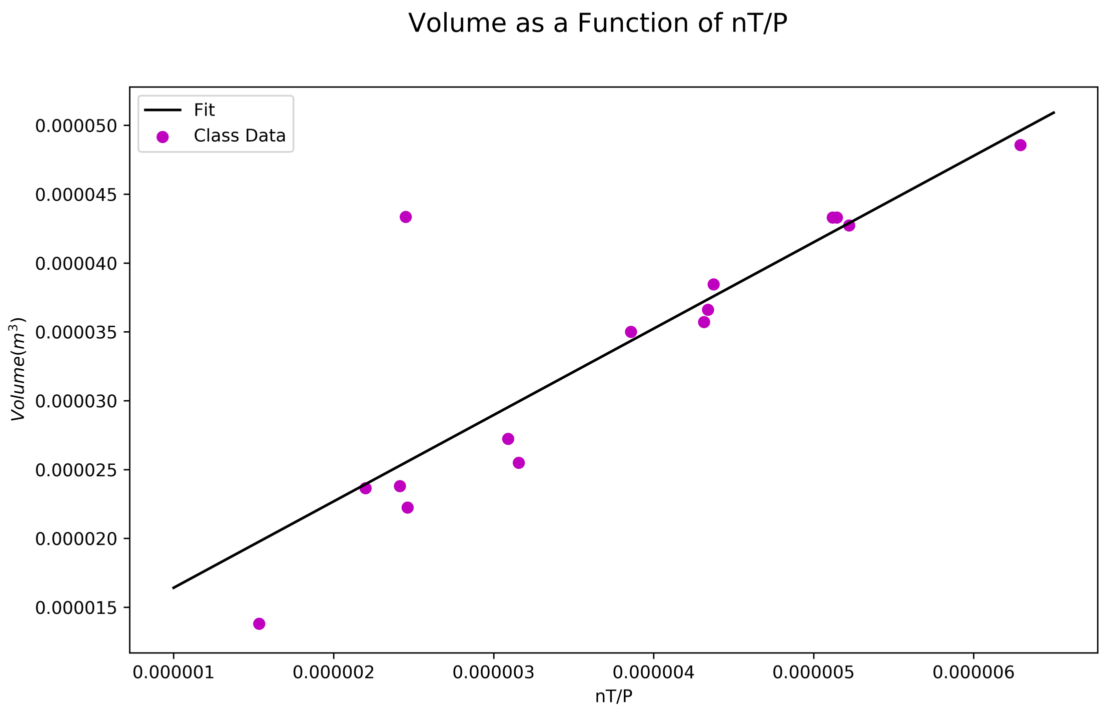

Figure 1: Plot of the volume versus nT/P for the class data with a linear regression fitted to it.

4 Conclusion

The objective in this experiment was to experimentally determine a value for the known gas constant, 8.31446261815324J/molK. The result acquired, however, is not similar enough to the known value to be statistically significant. However, it is no misunderstanding that most of these methods acquired values quite close to the known value. While one of the objectives may not have been met, another objective has. These various methods of statistical analysis have been used. To improve the accuracy of this experiment several methods can be used, but simply acquiring more data points is most likely to have the largest affect on accuracy and may aid in determining systematic errors.

5 Appendix

Here I have included the script written for this lab. I have also included the output of the code in the form of a photo from the kernel.

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jan 30 22:46:19 2020

@author: maxhu
#==========================================================================#
mass_mg_l = mass of magnesium low accuracy (g)
mass_mg_h = mass of magnesium high accuracy (g)

V_HCl_l = Volume of HCl low accuracy (mL)
V_HCl_h = Volume of HCl high accuracy (mL)

V_H_2_l = Volume of H2 low accuracy (mL)
V_H_2_h = Volume of H2 high accuracy (mL)

T_H2O_l = Temperature of H2O low accuracy (K)
T_H2O_h = Temperature of H2O high accuracy (K)

P_H2O_l = Vapor pressure of H2O at low accuracy temperature (inHg)
P_H2O_h = Vapor pressure of H2O at high accuracy temperature (inHg)

P_total = Total atmospheric pressure at that time of day (inHg)

inHg_to_Pa = There are 3386.38867 Pascals per inHg. (A conversion factor)

err_XX_l = error in species XX for low accuracy
err_XX_h = error in species XX for high accuracy

mm_Mg = Molar mass of Magnesium (g/mol)
mm_H2 = Molar mass of Hydrogen gas (g/mol)

#==========================================================================#
Objectives for this program are to:
    1) Determine the number of moles formed in the reaction of:
        Mg + 2HCl --> MgCL2 + H2
```

```

    2) Determine the gas constant, R, for two data sets. The high and low
    accuracy data sets for which I will denote the high accuracy R, R_h and
    low accuracy R, R_l.

    3) Determine the error propogated within each R. (err_R_h & err_R_l)

    4) Use class data to plot volume vs. number of moles for the class set.

    5) Determine slope of that line

    6) Propogate the error for the slope of the line.

    7) Determine the variance in the data set

    8) Perform a q-test on the data to look for outliers

    9) After removing outliers, calculate the mean R, the average deviation,
    std dev, and std dev of the mean.

    10) Determine if the class mean is statistically different from the
    accepted R using a 95% confidence interval. IOW perform a t-test.
#==============================================================================#
"""
#==============================================================================#
import pandas as pd
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
#==============================================================================#
mass_mg_l = .025
err_mass_mg_l = .001

mass_mg_h = .0510
err_mass_mg_h = .0001

V_HCl_l = 5.0
err_V_HCl_l = .1

V_HCl_h = 5.00
err_V_HCl_h  = .01

V_H2_l = 26.3
err_V_H2_l = .1

V_H2_h = 48.57
err_V_H2_h = .01

T_H2O_l = 293.25
err_T_H2O_l = .01

T_H2O_h = 299.740
err_T_H2O_h = .001

```

```python
P_H2O_l = 0.69432166
err_P_H2O_l = 0.0000001

P_H2O_h = 1.033603447
err_P_H2O_h = 0.0000001

P_total = 30.16
err_P_total = .01

inHg_to_Pa = 3386.38867 #Pascals per 1 inHg
atm_to_Pa = 101325 #Pascals per 1 atm
C_to_K = 273.15 #K in 1 C

mm_Mg = 24.305
err_mm_Mg = .001

mm_Hg = 2.01588
err_mm_Hg = .00001
#===========================================================================#
def error_propogation_for_R(variable_of_interest, list_of_variables,
                            list_of_errors):
    summer = 0
    for i in range(0,len(list_of_variables)):
        summer = summer + list_of_errors[i]**2 * list_of_variables[i]**2

    error_in_measurement = np.sqrt(summer)
    print('The uncertainty in {}: ,'.format(variable_of_interest),
        error_in_measurement)

    return error_in_measurement
#===========================================================================#
n_Mg_l = mass_mg_l / mm_Mg
n_Mg_h = mass_mg_h / mm_Mg

n_H2_l = n_Mg_l
n_H2_h = n_Mg_h

P_H2_l = P_total - P_H2O_l
P_H2_h = P_total - P_H2O_h

"""
Using:
    PV = nRT
    R = PV / nT
"""

R_l = (P_H2_l * inHg_to_Pa * (V_H2_l / 1000)*1e-3) / (n_H2_l * T_H2O_l)
R_h = (P_H2_h * inHg_to_Pa * (V_H2_h / 1000)*1e-3) / (n_H2_h * T_H2O_h)
print('R_l= ', R_l)
print('R_h= ', R_h)

R_l_variables_list = [P_total, P_H2O_l, V_H2_l, mass_mg_l, mm_Mg, T_H2O_l]
R_l_errors_list = [err_P_total, err_P_H2O_l, err_V_H2_l, err_mass_mg_l,
                   err_mm_Mg, err_T_H2O_l]
```

```python


R_h_variables_list = [P_total, P_H2O_h, V_H2_h, mass_mg_h, mm_Mg, T_H2O_h]
R_h_errors_list = [err_P_total, err_P_H2O_h, err_V_H2_h, err_mass_mg_h,
                   err_mm_Mg, err_T_H2O_h]

error_propogation_for_R('R_l', R_l_variables_list, R_l_errors_list)
error_propogation_for_R('R_h', R_h_variables_list, R_h_errors_list)

#==============================================================================#
data = pd.read_excel (
        r'C:\Users\maxhu\Documents\PChem\PchemII\Lab_1\Lab_1_Gas_Constant.xlsx'
        , sheet_name='Class_data')

df = pd.DataFrame(data, columns= ['mass Mg used (g)'])
temp = df.values.tolist()
mass_mg_h_class  = [val for sublist in temp for val in sublist]

df = pd.DataFrame(data, columns= ['Temp (Celcius)'])
temp = df.values.tolist()
T_H2O_h_class  = [val for sublist in temp for val in sublist]

df = pd.DataFrame(data, columns= ['Volume of H2 (mL)'])
temp = df.values.tolist()
V_H2_h_class  = [val*1e-3*1e-3 for sublist in temp for val in sublist]

df = pd.DataFrame(data, columns= ['P totals (atm)'])
temp = df.values.tolist()
P_total_class  = [val for sublist in temp for val in sublist]

df = pd.DataFrame(data, columns= ['P(H2O)  (atm)'])
temp = df.values.tolist()
P_H2O_h_class  = [val for sublist in temp for val in sublist]

df = pd.DataFrame(data, columns= ['P(H2) (atm)'])
temp = df.values.tolist()
P_H2_h_class  = [val for sublist in temp for val in sublist]
#==============================================================================#
"""
Using:
    PV = nRT
    V(n) = n * (RT / P)
    therefore, the slope is equal to RT / P
    if each n value is multiplied by the corresponding T / P value, then the
    slope is truly R.
    V(n') = n * (T/P) * R
    V(n') = n' * R
"""
n_H2_h_class = []
for i in mass_mg_h_class:
    n_H2_h_class.append(i / mm_Mg)

modified_n_values = []
```

```
for i in range(0,len(n_H2_h_class)):
    modified_n_values.append(n_H2_h_class[i] * ((T_H2O_h_class[i] + C_to_K) /
                                (P_H2_h_class[i] * atm_to_Pa)))
#==============================================================================#
def linear_regression_plot(x, b, m):
    std_dev = std_err * np.sqrt(len(modified_n_values))
    print('The R value from the slope is: ', m)
    print('The uncertainty in R is: ', std_dev)
    return m*x + b

slope, intercept, r_value, p_value, std_err = stats.linregress(
        modified_n_values,V_H2_h_class)

x_values = np.linspace(1e-6,6.5e-6)

size = 10
fig = plt.figure(1,figsize=(10,6))
fig.suptitle('Volume as a Function of nT/P', fontsize=15)
my_fig = fig.add_subplot(111)
plt.plot(x_values, linear_regression_plot(x_values, intercept, slope),
         color='black', label='Fit')
plt.scatter(modified_n_values,V_H2_h_class,color='m', label='Class Data')
plt.ylabel('Volume (m^3)', fontsize=size)
plt.xlabel('nT/P', fontsize=size)
plt.legend(loc='best', fontsize=size)
plt.savefig('Report_1.png', dpi=600)
plt.legend(loc='best')
plt.show
#==============================================================================#
q_ref = 0.396

def q_test(data, q_ref):
    data.sort()
    q_exp_lower = (data[1] - data[0]) / (data[len(data)-1] - data[0])
    q_exp_higher = (data[len(data)-1] -
                          data[len(data)-2]) / (data[len(data)-1] - data[0])

    if q_exp_lower > q_ref:
        print('Reject ', data[0])
        return data.remove(data[0])
    else:
        print('Do not reject ', data[0])

    if q_exp_higher > q_ref:
        print('Reject ', data[len(data)-1])
        return data.remove(data[len(data)-1])
    else:
        print('Do not reject ', data[len(data)-1])

R_class_values = []
for i in range(0,len(V_H2_h_class)):
    R_class_values.append((P_H2_h_class[i] * atm_to_Pa * V_H2_h_class[i]) /
                          (n_H2_h_class[i] * (T_H2O_h_class[i] + C_to_K)))

```

```python
q_test(R_class_values, q_ref)
array_for_R_values = np.array(R_class_values)

mean = np.average(array_for_R_values)
def ave_deviation(data):
    summer = 0
    for i in data:
        summer = summer + np.abs(i - mean)
    return (1 / len(data))*summer

average = np.average(array_for_R_values)
average_deviation = ave_deviation(R_class_values)
standard_deviation = np.std(array_for_R_values)
standard_deviation_mean = np.std(array_for_R_values) / np.sqrt(len(
        R_class_values))
print('The mean or average for R is: ', average)
print('The average deviation for R is: ', average_deviation)
print('The standard deviation is: ', standard_deviation)
print('The standard deviation of the mean is: ', standard_deviation_mean)

def t_test(data, known_value, standard_deviation_mean, t_acceptable):
    t = (mean - known_value) / (standard_deviation_mean)
    print('The t-value is: ', t)
    if t > t_acceptable:
        print('Since the calculated t is greater than the 95% confidence '
              ' interval t, the value is statistically different than '
              'the known R.')
    return t
t_acceptable = 1.94
t_test(average, 8.31446261815324, standard_deviation_mean, t_acceptable)
```

```
R_l=  8.70014699659116
R_h=  7.616797512822176
The uncertainty in R_l: , 3.950697349032194
The uncertainty in R_h: , 0.6459890483986712
The R value from the slope is:  6.2729528773347445
The uncertainty in R is:  4.149457172101669
Do not reject  7.718244388016872
Reject  17.688614030668557
The mean or average for R is:  8.777450957466343
The average deviation for R is:  0.5537030963600401
The standard deviation is:  0.7479920753441065
The standard deviation of the mean is:  0.19990929099711763
The t-value is:  2.315992103237356
Since the calculated t is greater than the 95% confidence  interval t, the value is statistically
different than the known R.
```

Figure 2: Output of the above code.