

Department of Chemical Engineering
University of California, Santa Barbara

Chemical Engineering 180B
Analysis of CO₂ and He Compressibility at High Pressures

Benjamin Wolfe, Chase Hartsell, Ian Beresford
Group 4W
Conducted from January 18th - February 2nd, 2023

Abstract

Gaseous compressibility is a key consideration for accurate engineering calculations in many applications. In this report, the compressibility factors of CO₂ and helium were investigated using Burnett's isothermal pressure ratio method at 10°, 20°, and 30° C. The compressibility factor Z_{CO_2} was found to have a decreasing linear trend with increasing pressure, with the lowest value $Z = 0.6 \pm 0.02$ at 760 psi, and the system became more ideal with $Z = 1 \pm 0.08$ as pressure decreased to atmospheric. Helium exhibited ideal behavior, with Z_{He} being relatively ideal with $Z = 1 \pm 0.1$ regardless of pressure. Although no significant deviations from ideality were observed from helium gas, there is a slight positive deviation with temperature increase, showing a temperature dependence in the second virial coefficient $B(T)$ in helium gas, in addition to CO₂. The second virial coefficients, which deviated by 50-100% from literature values, increased non-linearly with temperature, and were negative for temperatures below 20° C for helium and at all temperatures measured for CO₂. CO₂'s decreasing compressibility factor with pressure highlights the implications of non-ideality in gases, and the calculations for CO₂ wouldn't be as accurate at same temperature and pressure as its ideal counterpart helium.

Introduction

With recent world events and supply chain issues, supply storage has become a major concern in the United States. Many vital instruments, like MRI machines in hospitals, require liquid helium for cooling purposes. However, with the recent turmoil in one of the world's major helium suppliers, helium has become scarce[1] and this shortage could potentially derail scientific and medical research. In lieu of this, finding new compressible gasses may prove to be important for future applications in this sense. The goal of this report is to find the compressibility factor Z for gases CO₂ and Helium, and to create a solid model for z at different temperatures. The pressure, molar volume, and temperature of a gas are related by the equation of state:

$$Z = \frac{PV}{nRT} \quad (1)$$

where R is the universal gas constant. The compressibility factor, Z , is a unit-less measure of gas ideality (ideal gas has $Z=1$), and it can be determined for different gasses at different conditions. Factors that affect gas ideality include inter-molecular forces and molecular size. Because of this, ideal behavior is observed at high temperatures and low pressures, when these two effects are minimized. To measure the compressibility factors of carbon dioxide and helium, we employed Burnett's method[2] and observed the change in pressure as a gas was expanded at a constant temperature. By relating the change in pressure after the r^{th} expansion to a consistent change in volume (represented by the apparatus constant, N), the deviations from ideal expectations can be measured.

$$\frac{p_{r-1}}{p_r} = N \frac{Z_{r-1}}{Z_r} \quad (2)$$

Because gas acts closest to ideal at low pressures, taking the limit as pressure goes to zero allows us to solve for the apparatus constant. Using this, a relation can be developed between the ratio in pressures after an expansion to the ratio in compressibility factors. We can then calculate Z from the initial and current pressure at any given expansion.

$$p_r N^r = \left(\frac{p_0}{Z_0} \right) Z_r \quad (3)$$

Because compressibility is also affected by temperature, a linear model can be generated to obtain Z as a function of pressure at each isotherm. The slope of this line is the temperature dependent B' ; the intercept, at zero pressure, is $Z=1$: an ideal gas.

$$Z = 1 + B'P \quad (4)$$

From the slope of compressibility factor B' , the second virial coefficient can be obtained from B' through equation[3]:

$$B(T) = B' * RT \quad (5)$$

However, due to unique molecular structures, the compressibility and second virial coefficient would deviate from ideal state as pressure increases. This is due to multiple factors from intermolecular forces. The attractive and repulsive behavior between the pure gas molecules, like the Van der Waals forces, would

affect how ideally a molecule like methane would behave under a set temperature and pressure[4]. The Van der Waal equation of state takes into account these forces, and the fact that molecules have finite volume, and the pressure and volume are adjusted using species-specific coefficients so that they better fit their species' behavior.

Experimental Methods

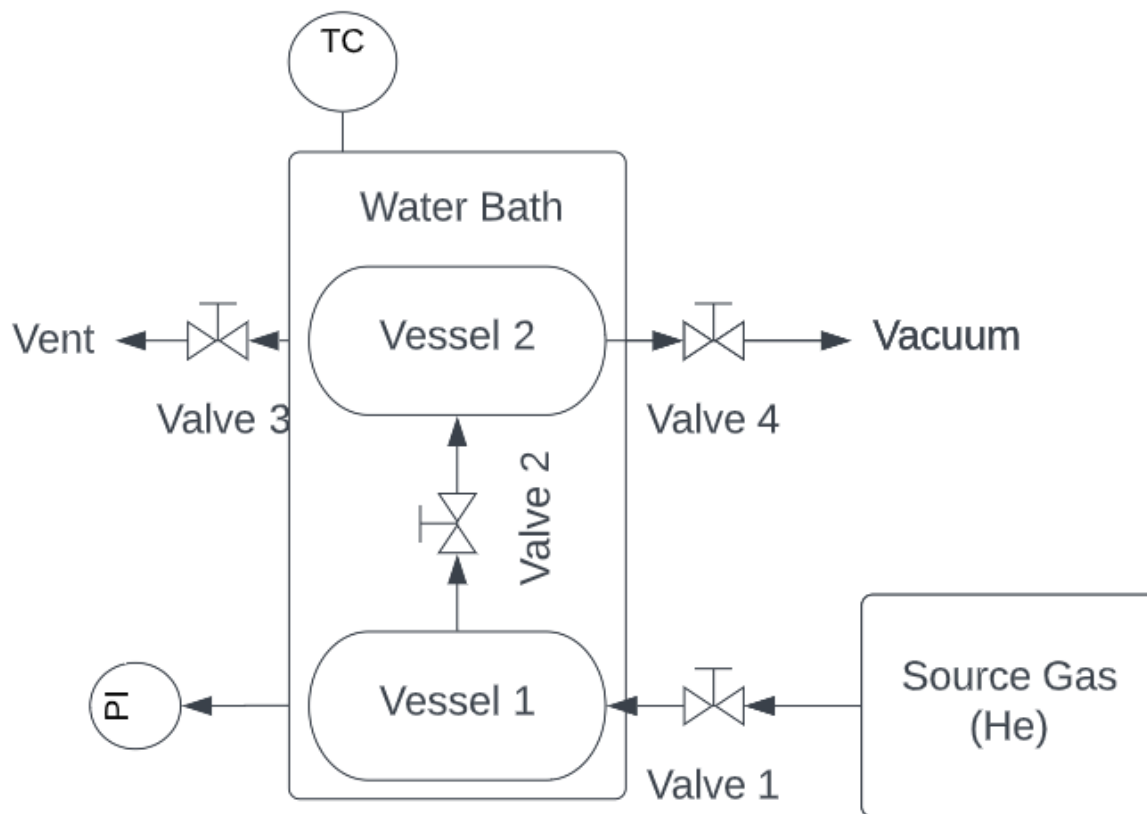


Figure 1: Pressure vessel apparatus where gas was expanded and the pressure was recorded after it reached equilibrium.

The apparatus (Figure 1) consists of a high purity gas source (He and CO₂, Airgas, 100%). The high purity gas source is connected to two interconnected vessels, one of which is attached with a pressure indicator (PI) and the other to vent and vacuum. All the connections connected to the vessels are regulated with a manual gate valve, with the whole apparatus taking place in a temperature controlled (TC) water bath.

The goal of this experiment is to use Burnett's method to find the compressibility factor Z of gases He and CO₂, at different pressure and temperature. A source gas, CO₂ or He, is introduced into the vacuumed vessel 1 via opening valve 1. Once the set pressure is reached in vessel 1, valve 1 is closed and valve 2 is opened slowly to allow gas to expand into the vacuumed vessel 2 isothermally. Once the pressure equalized, it is recorded and valve 2 is closed and the gas in vessel two is vented out via valve 3 and vacuumed via valve 4. After vessel 2 is vacuumed, valve 2 is reopened for another expansion. Expansions are repeated until the equilibrium pressure approaches atmospheric pressure, or the measurement is too low to accurately read. With the expansion data at different temperature, the compressibility factor can be calculated via equation 2 and 3, thus a pressure dependent compressibility value Z graph is made (Figures 2 & 3). The slope of the Z graph is the temperature dependent second virial coefficient, which is graphed at the three temperatures measured (Figure 4).

Results and Discussion

The Z factors as a function of pressure for helium and carbon dioxide (Figures 2 & 3), exhibit a linear dependence in this pressure range. Values for raw pressure readings used to calculate Z are located in Tables 1 & 2 in Appendix A. The compressibility factor for helium does not have a clear trend with increasing pressure. There is imprecise correlation between pressure and the Z factor given by the low R^2 values at three different temperatures, however the error bars suggest relative ideality with $Z = 1 \pm 0.1$ (Figure 2). Gas is most ideal at high temperatures and low pressures, however the 30° isotherm exhibits the most deviation from ideal behavior, even outside of the error range for most measurements.

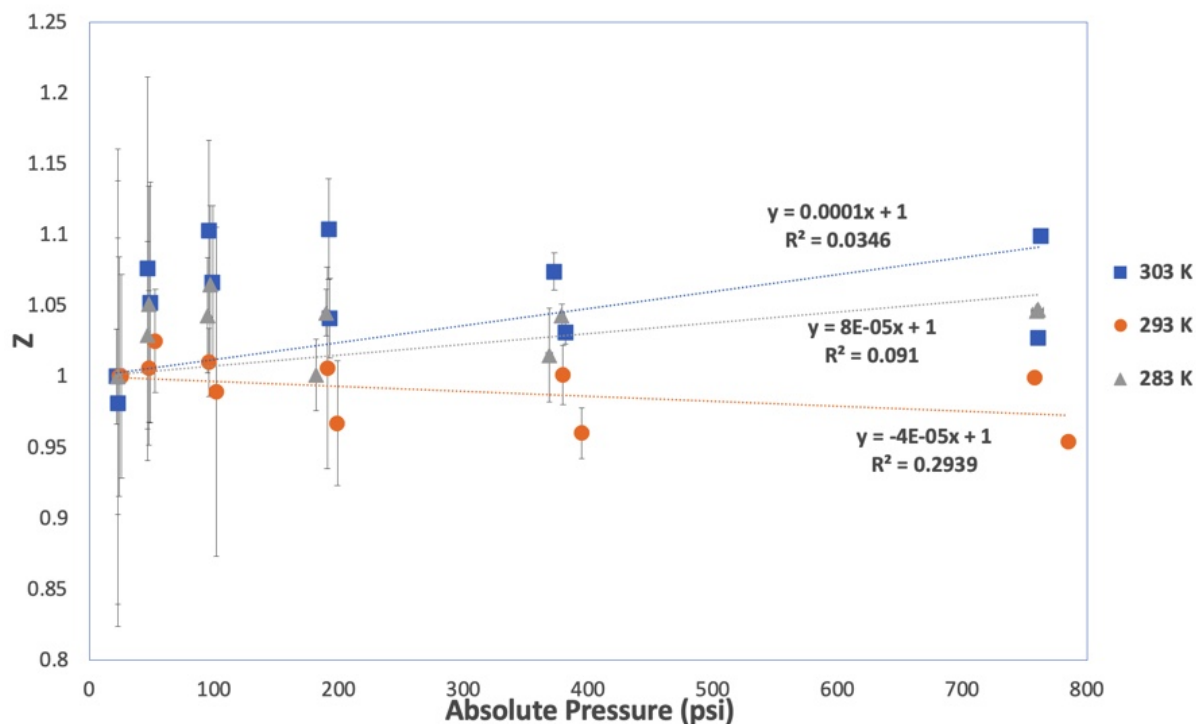


Figure 2: Compressibility of helium at 3 isotherms. Error bars represent standard error of calculations and pressure gauge readings.

The seemingly unpredictable points shown in Figure 2 can be explained by Helium's inert nature. Due to helium's monoatomic and stable nature from filled orbitals, helium atoms have no attraction towards each other, making it inert. Helium is also the second smallest atom, so the volume of the container is a very fair estimate for the volume of empty space, whereas the larger carbon dioxide makes a more significant effect on this assumption. This explains the ideal behavior seen in Figure 2. Even with the imprecise distribution, Z_{He} has a value of relatively one across all pressures. This is similar to Imbert's Z value of $1 + 0.0045 P(\text{MPa})$ [5]. The cause of the high spread in these values, especially at lower pressures, is because error in the pressure readings gets propagated through after each expansion. In addition to this, the gauge was accurate to $\pm 0.25\%$, and this systematic error also becomes more significant at lower pressure readings. Because helium was so close to ideal, these deviations look like a large spread with little correlation however the error bars suggest a linear pressure dependence at these conditions.

In addition to helium, a trend for Z_{CO_2} 's compressibility is also reasonable. However, the Z factors for carbon dioxide show a stronger pressure dependence, decreasing linearly in this temperature and pressure range (Figure 3). CO_2 's bonds provided the molecule with a net zero dipole, but compared to helium's inert nature, CO_2 experiences bigger Van der Waals forces due to the instantaneous dipole moment given by a more electronegative oxygen atom at each end. This means that when a molar charge is introduced into a constant volume, CO_2 gas molecules experience a stronger attractive force, exhibiting a lower pressure than ideal and thus having a lower Z value at these pressures. The trends shown in Figure 3 also match with Sage and Lacy (1955) and Burton Technical Bulletin (1990) data showing a

negative linear slope in the Z factor until 1500 psia at 100°F[6]. Experimental data also shows the same negative slope, with the similar Z values within 4%. Experimental data also displays ideal gas trends, with the higher temperature exhibiting the most ideal behavior.

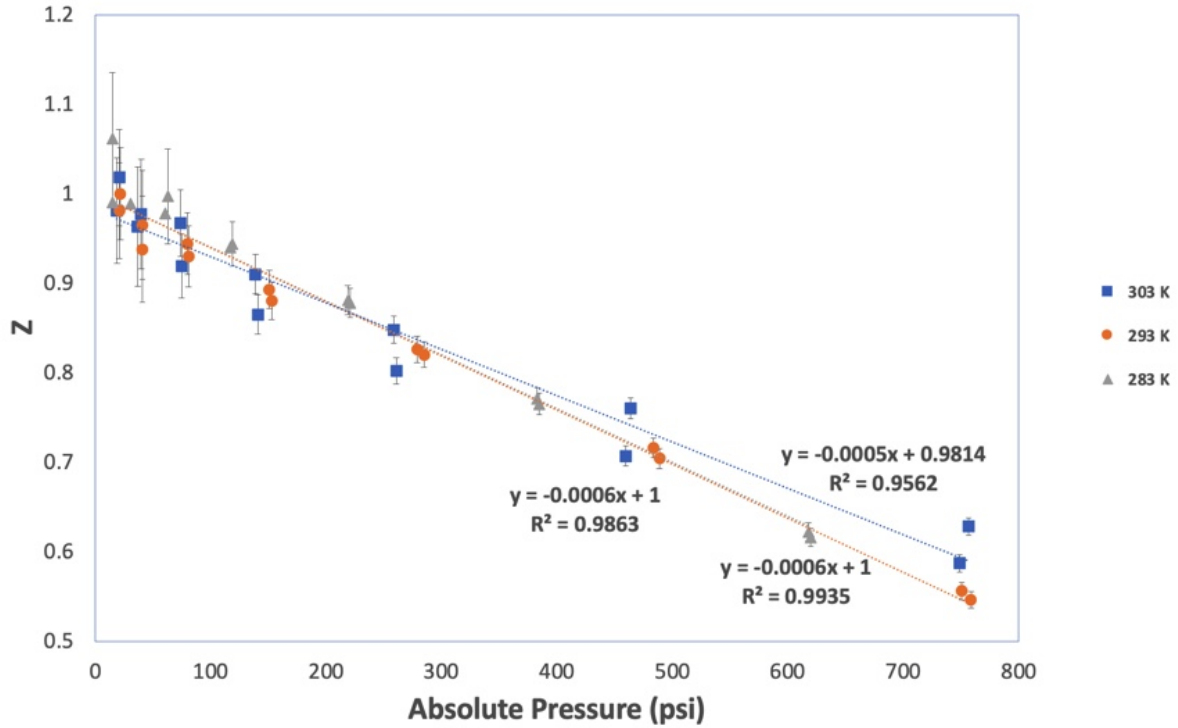


Figure 3: Compressibility of carbon dioxide at 3 isotherms. Error bars represent standard error of calculations and pressure gauge readings.

A key difference in the Z factor determination between helium and carbon dioxide gas is that carbon dioxide would readily liquefy under high pressure. From the CO₂ phase diagram (Figure 7), CO₂ liquefies at 30°C and 760 psi, and at 10°C it liquefies at 650 psi. This prevented us from charging the vessel with as much gas as we did for helium, and one less expansion was able to be precisely measured at 10°C.

After obtaining experimental data for the compressibility factors of each gas, a linear regression model was set up, with the intercept set at the ideal $Z = 1$. Using this isothermal variation of the pressure ratio method to determine Z factors, the slope of these lines (Figure 4) are the second virial coefficients, which are dependent on temperature[7]. Although the values for these coefficients were off by 50-100% for most trials (some were also off by a factor of 10), the respective Z that is obtained when using the linear model gives Z values that are within 10%.

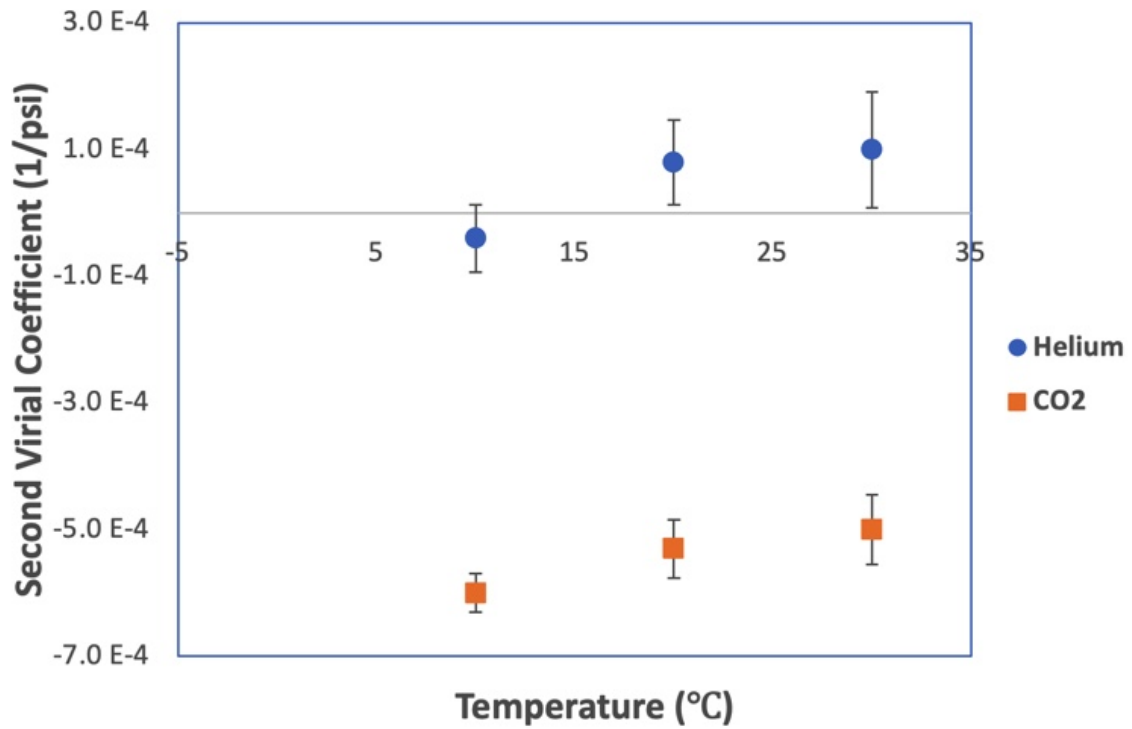


Figure 4: The second virial coefficients of helium and carbon dioxide at the three temperatures that were measured. Error bars represent standard error of the values calculated over two trials of six expansions at each isotherm.

These linear models can be used to interpolate between the pressure ranges and isotherms, giving a solid relation for Z at any temperature or pressure within the measured range.

Conclusion

The chemical make-up of a gas affects how compressible it is under high pressure. CO_2 's compressibility decreased with increasing pressure at all three isotherms. The compressibility factor Z_{CO_2} decreased down to almost 0.6 at 750 psi, meaning that at same pressure and volume, CO_2 contains 40% more moles than the relatively ideal helium according to equation 1. The decrease in Z_{CO_2} is a result of the Van der Waals attraction between the molecules, and as molecules being compressed, they would have less expulsion forces between each other as experienced on helium gas. In addition, CO_2 is a significantly larger molecule than monoatomic helium, and the fact that a vessel of CO_2 has less empty space between the gas particles than helium has implications on its compressibility. A linear model for the Z factor as a function of pressure was established in the pressure ranges measured, and the slope of these lines are the second virial coefficients, which is a temperature dependent quantity that can be used to interpolate for Z at temperatures and pressures that were not measured. The implications of gas compressibility can be widely utilized for many engineering applications, especially for storing and modeling gas at high pressures and low temperatures.

References

1. Helium shortages return. (2022). C&EN Global Enterprise, 100(6), 9–9. doi:10.1021/cen-10006-buscon4
2. Silberberg, I. H., Kobe, A. K., McKetta, J. J.(1959). "Gas Compressibilities with Burnett Apparatus." J. Chem. Eng. Data 4, 314-329
- 3.Hajjar, Raja Faris. "THE DETERMINATION OF THE SECOND VIRIAL COEFFICIENTS AND THE MOLECULAR CONSTANTS OF SIX HALOGEN-SUBSTITUTED METHANES BYU A GAS BALANCE METHOD." The Ohio State University, 1967.
- 4.Schamp, Jr, H W, Mason, E A, Richardson, A C.B., and Altman, A. COMPRESSIBILITY AND INTERMOLECULAR FORCES IN GASES: METHANE. <https://doi.org/10.1063/1.1705891>
5. Imbert, G et al. "The compressibility and the capacitance coefficient of helium-oxygen atmospheres." Undersea biomedical research vol. 9,4 (1982): 305-14.
6. Adisoemarta, P et al. "Measurement of Z-Factors for Carbon Dioxide Sequestration." Texas Tech University.(2004).
7. Berberan Santos, Mario N and Bodunov, Evgeny N. and Pogliani, Lionello. (2008). The van der Waals equation: Analytical and approximate solutions. Journal of Mathematical Chemistry. 43. 1437-1457. 10.1007/s10910-007-9272-4.
8. Van Ness, H. C., Classical Thermodynamics of Non-Electrolyte Solutions, Pergamon Press, pp. 45-51 (1964).
9. Burnett, E. S. "Compressibility Determinations Without Volume Measurements." Journal of Applied Mechanics. 3(4), A136-A140. (1936).
10. Witonsky, J. R., Miller, G. J. (1963). "Compressibility of Gasses. IV. The Burnett Method Applied." American Chemical Society.
12. J. M. H. Levelt Sengers, Max Klein, and John S. Gallagher. (1971). "Tables of Second Virial Coefficients and Thier First and Second Derivatives for the Stockmayer (m,6,3) Potential Function." National Bureau of Standard and Physical Chemistry.
13. Gallagher, S. J., Crovetto R. (1992). "The Thermodynamic Behavior of the Co2-H2O Sytsem from 400 to 1000K, up to 100 MPa and 30% Mole Fraction of CO2." National Institute of Standards and Technology, 443.
14. Barth, Ginger. (2006). "Methane Gas Volume Expansion Ratios and Ideal Gas Deviation Factors for the Deep-Water Bering Sea Basins." U.S. Geological Survey.

Appendix A: Raw Data

Figure 4 displays how N , the apparatus constant, was experimentally determined. N is the ratio of final volume to initial volume for each expansion, and it was constant for each expansion. Vessel 1 and 2 theoretically had the same volume, so the apparatus constant was expected to be two, and the average of both gas's calibration proved this to be believable within the error range.

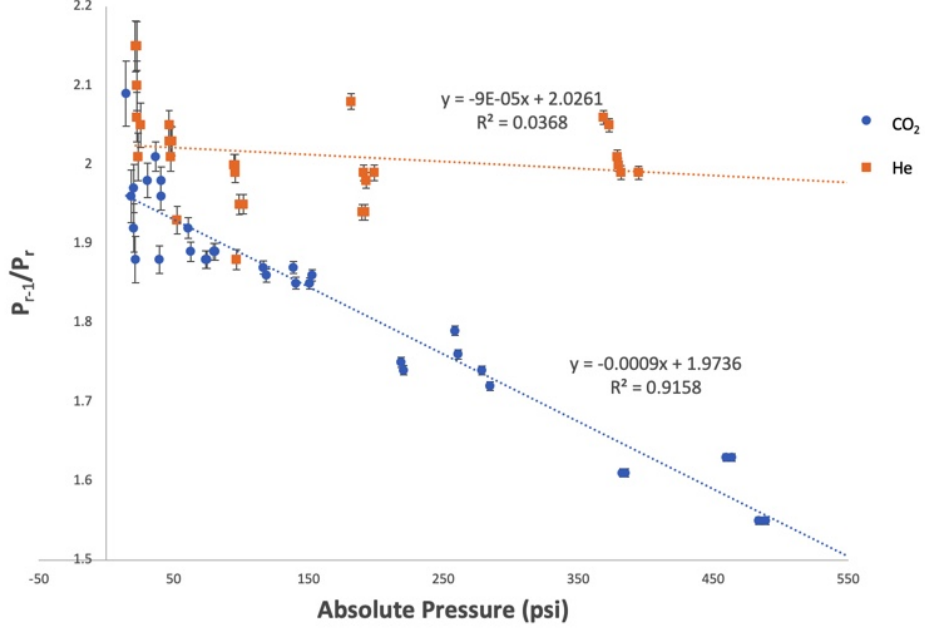


Figure 5: Calibration for the apparatus constant of the helium pressure vessel system. The limit as pressure approaches 0 (which is the y-intercept) determines the apparatus constant N i.e. the ratio of V_r/V_{r-1} . This was determined on the same apparatus with helium and carbon dioxide, and the expected value is 2.

To determine Z_0 for each run, regression was used to find the limit of our data as gauge pressure approached zero. This gives the ratio of the $\frac{P_0}{Z_0}$, so that successive Z factors can be calculated using equation 6 (Appendix B).

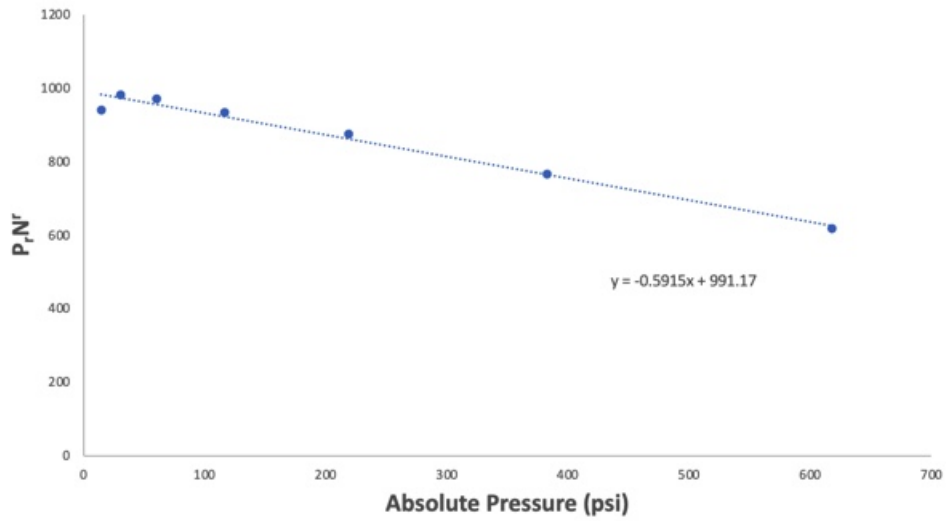


Figure 6: As P_r approaches low pressures, the value of Z_r approaches unity. The limit of this data set as pressure approaches atmospheric allows us to determine the value of Z_0 in equation 3. The data in this graph is from the 12th run (carbon dioxide at 283 K).

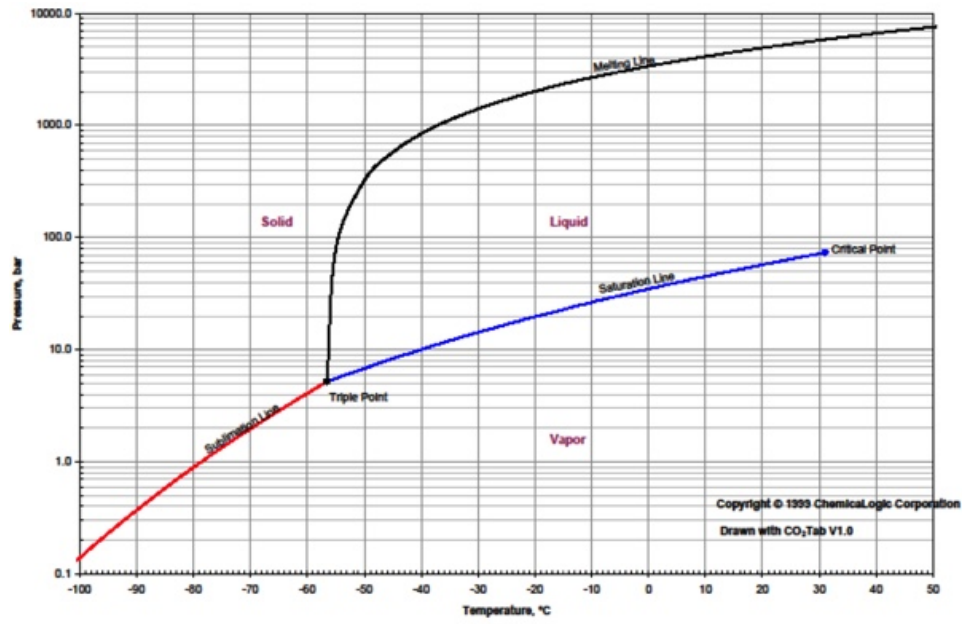


Figure 7: Phase Diagram of Carbon Dioxide.

Table 1: Raw pressure measurements for each successive expansion at a given temperature for He. Two series of five expansions were tested for each temperature.

Species	Run [experiment #]	Expansion # (r)	Temperature [°C]	P _r [psi, gauge]
He	1	0	30.4	746
He	1	1	30.2	367
He	1	2	29.9	178
He	1	3	29.9	84
He	1	4	30.1	34
He	1	5	30.1	8
He	2	0	30.1	748
He	2	1	30.1	358
He	2	2	30.1	177
He	2	3	30.1	81
He	2	4	30.1	32
He	2	5	30.1	7
He	3	0	20.1	770
He	3	1	20.1	380
He	3	2	20.1	184
He	3	3	20.1	87
He	3	4	20.1	38
He	3	5	20.1	11
He	4	0	20.1	743
He	4	1	20.1	365
He	4	2	20.1	176
He	4	3	20.1	81
He	4	4	20.1	33
He	4	5	20.1	9
He	5	0	10.2	746
He	5	1	10.3	354
He	5	2	10.4	175
He	5	3	10.4	80
He	5	4	10.5	32
He	5	5	10.6	8
He	6	0	10.2	745
He	6	1	10.2	364
He	6	2	10.3	167
He	6	3	10.2	82
He	6	4	10.3	33
He	6	5	10.3	8

Table 2: Raw pressure measurements for each successive expansion at a given temperature for CO_2 . Two series of six expansions were tested for each temperature.

Species	Run [experiment #]	Expansion # (r)	Temperature [°C]	P_r[psi, guage]
CO2	7	0	30.1	742
CO2	7	1	29.9	449
CO2	7	2	29.9	244
CO2	7	3	30.0	124
CO2	7	4	30.0	59
CO2	7	5	30.0	22
CO2	7	6	30.0	4
CO2	8	0	30.0	734
CO2	8	1	30.0	445
CO2	8	2	30.0	246
CO2	8	3	30.0	126
CO2	8	4	30.0	60
CO2	8	5	30.0	25
CO2	8	6	30.0	6
CO2	9	0	20.0	744
CO2	9	1	20.0	474
CO2	9	2	20.0	270
CO2	9	3	20.0	138
CO2	9	4	20.0	66
CO2	9	5	20.0	26
CO2	9	6	20.0	7
CO2	10	0	20.0	736
CO2	10	1	20.0	469
CO2	10	2	20.0	264
CO2	10	3	20.0	136
CO2	10	4	20.0	65
CO2	10	5	20.0	26
CO2	10	6	20.0	6
CO2	11	0	10.0	605
CO2	11	1	10.0	370
CO2	11	2	10.0	206
CO2	11	3	10.0	104
CO2	11	4	10.0	48
CO2	11	5	10.0	8
CO2	11	6	10.0	Unreadable
CO2	12	0	10.0	603
CO2	12	1	10.0	368
CO2	12	2	10.0	204
CO2	12	3	10.0	102
CO2	12	4	10.0	46
CO2	12	5	10.0	16
CO2	12	6	10.0	Unreadable

Table 3: Comparison of our second virial coefficient calculations with literature values

Species	Temp	Virial Coefficient: Lit values			
		Lit (cm ³ /mol)	us (regression line)	us cm ³ / mol	%error
CO2	30	111.3	-0.0005	-182.8	64.23%
CO2	20	125.6	-0.00053	-187.4	49.17%
CO2	10	136.7	-0.0006	-204.9	49.87%
He	30	3.8	0.0001	36.6	862.03%
He	20	-2.2	0.00008	28.3	1185.50%
He	10	-21.7	-0.00004	-13.7	-37.06%

Appendix B: Sample Calculations

For each subsequent expansion,

$$\left(\frac{p_{r-1}}{p_r}\right) = N \left(\frac{Z_{r-1}}{Z_r}\right) \quad (6)$$

this equation was used to determine the apparatus coefficient. As low pressures are approached, the gas behaves more ideally. Consequentially the ratio of $\frac{Z_{r-1}}{Z_r}$ approaches 1 and N will be the ratio of $\frac{p_{r-1}}{p_r}$ as seen in the limit.

$$\lim_{p \rightarrow 0} \left(\frac{p_{r-1}}{p_r}\right) = N \quad (7)$$

$$p_r N^r = \left(\frac{p_0}{Z_0}\right) Z_r \quad (8)$$

This equation was used to find the value of Z_0 . At low pressures, the value of Z_r approaches unity since it behaves ideally. The value of Z_0 can be determined from the the ratio of the initial pressure and $P_r N^r$.

Every value recorded for pressure was within .25% of its true value according to the gauge, as well as a .5 psi uncertainty between the tick marks, and an estimate was made to best determine the observed pressure after each expansion. With this error, it gets bigger as we take the ratio of the pressure at each given expansion (calculated by the 'Multiplication or division' method). This error is carried through to calculating Z and B with the same method.

Type of calculation	Example†	Standard deviation of y
Addition or subtraction	$Y=a+b-c$	$S_y=(S_a^2+S_b^2+S_c^2)^{1/2}$
Multiplication or division	$Y=a*b/c$	$S_y/y=\{(S_a/a)^2+(S_b/b)^2+(S_c/c)^2\}^{1/2}$
Exponentiation	$Y=a^x$	$S_y/y=x (S_a/a)$
Logarithm	$Y=\log_{10}a$	$S_y=0.434 (S_a/a)$
Antilogarithm	$Y=\text{antilog}_{10}a$	$S_y/y=2.303 S_a$

† a, b, and c are experimental variables whose standard deviations are S_a , S_b , and S_c , respectively.

Figure 8: Error propagation was calculated using the methods on this chart. When combining two values with error, the error is propagated through to the new value.

Appendix C: Program Files

```
4/25/22, 2:26 PM /Users/wesleyjohanson/Documents/College Classes_UC Santa Barbara/ChemE 180A Chemical Engineering Lab/z_factors/z_factors.py

1 #Wesley Johanson
2 # from re import I
3 import ChE
4 import numpy as np
5
6 #Files to Load data from
7 file1 = "CSV/z_factors_CSV_1.csv"
8 files = [file1]
9
10 #Data labels
11 rowIndex_of_labels = 1
12 dataStartsAtIndex = 3
13 labels = np.loadtxt(file1, unpack=True,
14 delimiter=',', dtype=str, skiprows=rowIndex_of_labels, max_rows=1)[: ,
15 rowIndex_of_labels]
16 savePlotAs = "plot.png"
17 folder = "Images/"
18
19 segmentDataCol = 0
20
21 # regressionVars = [0] #Variables to calculate Linear Regression R^2 values with
22 # respect to
23 customColors = None
24
25 plots = None
26 plots = [
27     [ 4, 5],
28     [ 3, 6],
29     [ 3, 8]
30 ]
31 error = [
32     [ 12, 13],
33     [ 9, 10],
34     [ 9, 11]
35 ]
36
37 segmentCols = [
38     0,
39     0,
40     0
41 ]
42
43 #For each dataset, determine whether to plot errBars or not
44 # errBarsDS = [
45 #     False, #Dataset 0 does not exist in this CSV
46 #     False,
47 #     False,
48 #     False,
49 #     False,
50 #     True,
51 #     False
52 # ]
53
54 #Booleans to determine which data set segments have error bars plotted
55 errBarsDS = {
56     1:False,
57     2:False,
58     3:False,
59     4:False,
60     5:False,
61     6:True
62 }
```

localhost:50193

1/2

Figure 9: Primary logic/script: python code

```

4/25/22, 2:26 PM /Users/wesleyjohanson/Documents/College Classes_UC Santa Barbara/ChemE 180A Chemical Engineering Lab/z_factors/z_factors.py
55     }
56
57 # error = None #OVERRIDE DELETE ME
58
59 yLabelOverride = None
60 # YLabels = [ "$Vz/Vz_{Max}$",
61 #             "friction factor",
62 #             "$V^{+}$",
63 #             "poop3"
64 #             ]
65
66 seg_mks = [ ".", "v", "+", "^", '3', '2', '3', '2' ]
67
68
69 _markers = [seg_mks for i in range(0,12)]
70 # print(_markers)
71
72 for file in files:
73     i = 0
74     for plotData in plots:
75         #Make Plot Obj
76         plot = ChE.ChEplot()
77         plot.loadCSV_str(file, labels, indepVars=1, skip=dataStartsAtIndex)
78         #Set Data
79         #FOR MORE FILES THIS WILL NEED TO CHANGE
80         plot.setDataLabel(labels)
81         #Plotting
82         plot.setDataColors(customColors)
83         plot.setFxn2Plot(plotData)
84         # plot.plotData_str(width=6,height=6, markers=_markers,
85 err=error[i],seg=segmentCols[i])
86         plot.plotData_str(width=6,height=6, markers=_markers,
87 err=error[i],seg=segmentDataCol, pltErrBar=errBarsDS)
88         #Statistics & Regression
89         # plot.plotLRegLines(width=0.5)
90         # plot.printAllRSquared()
91
92         #Plot Parameters
93         xAxisLabel = labels[plotData[0]] #Don't Change
94         yAxisLabel = yLabelOverride[i] if yLabelOverride else labels[plotData[1]]
95         plot.setAxisLabels(xAxisLabel, yAxisLabel, xpadding=5, ypadding=5)
96         plot.setTicProps()
97         # plot.setNumTics(delta_x=10, delta_y=10, x_subTics=3, y_subTics=3)
98         plot.showLegend()
99         # plot.changeFont()
100
101         #Presentation
102         # plot.showPlot()
103         temp = folder + str(i) + ' ' + savePlotAs
104         plot.savePlot(filename=temp,_dpi=600)
105         print(temp)
106         plot.close()
107         i += 1
108
109 print("Program Complete")

```

localhost:50193

2/2

Figure 10: Primary logic/script: python code

```

1 #Wesley Johanson
2 # from aifc import _Marker
3 # from re import I
4 from pprint import pprint
5 from re import I
6 import matplotlib as mpl
7 import matplotlib.pyplot as plt
8 import numpy as np
9 import matplotlib.font_manager as fm
10 from sklearn.linear_model import LinearRegression
11 from scipy import stats
12 import random
13 import sys
14
15 # sys.path.append("/Users/wesleyjohanson/Documents/Python_Modules")
16 class ChEplot:
17     def __init__(self):
18         self.figure=None
19         self.dataLabels=None
20         self.dataColors=None
21         #Counting Elements
22         self.numDataVars=None
23         self.numDataFns=None
24         self.numDataSets=None
25         self.data=None
26         self.fxns2plot=None
27
28
29
30
31     def loadCSV_str(self, filename: str, names: list, indepVars, skip=0):
32         """Loads each column of data from the CSV file into a row of a numpy
33         array stored in self.data
34
35         'names' is a list of names for the data sets in each col of the CSV"""
36         # if indepVars < 1 or indepVars > len(names): return
37         self.data = np.loadtxt(filename, unpack=True, delimiter=',', skiprows=skip,
38 dtype=str)
39         # if indepVars > self.numDataSets: self.data = none; return
40         self.dataLabels = names
41         self.numDataVars = indepVars
42         self.numDataSets = len(self.data)
43         self.numDataFns = self.numDataSets - self.numDataVars
44         # for col in range(0, len(self.data)):
45         #     new_x = []
46         #     new_y = []
47         #     for row in range(0, len(self.data[0])):
48         #         if self.data[col,row] != "":
49         #             new_x.append(float(self.data[col,row]))
50         #             new_y.append(float(self.data[0,row]))
51
52
53     #LEGACY
54     def setData(self, data: list, vars=1):
55         """Replaces Data and performs same operations as loadCSV"""
56         self.data = data

```

Figure 11: Plotting Class python code

```

4/25/22, 2:26 PM /Users/wesleyjohnson/Documents/College Classes/UC Santa Barbara/ChemE 180A Chemical Engineering Lab/z_factors/ChE.py
57 self.numDataVars = vars
58 self.numDataSets = len(self.data)
59 self.numDataFns = self.numDataSets - self.numDataVars
60
61
62 #Printers
63 def printAllData(self):
64     print(
65         "\n",self.figure
66         ,"\n",self.dataLabels
67         ,"\n",self.numDataVars
68         ,"\n",self.numDataFns
69         ,"\n",self.numDataSets
70         ,"\n",self.data
71         ,"\n",self.fxns2plot)
72     "print all data points in self.data"
73     print(self.data)
74
75 def printMeans(self):
76     "Prints the mean value of each row vector in self.data"
77     for i in range(0, self.numDataSets):
78         outputStr = "the mean of "
79         if self.dataLabels[i] is not None:
80             outputStr += self.dataLabels[i]
81             outputStr += "\t\tis " + str(np.mean(self.data[i]))
82
83 #Setters
84 def setDataLabel(self, names):
85     """
86     Stores a list of strings into the instance, where each str in the list
87     is the name of the corresponding column in the CSV file
88     """
89     self.dataLabels = names
90
91 def segmentData(self):
92     pass
93
94 def setLRegLineColors(self, colors=[]):
95     self.LRegLineColors = colors
96
97 def setIndepVars(self, vars):
98     "Vars are the first arrays in the self.data matrix"
99     self.setIndepVars = vars
100
101 #Plotters
102 @staticmethod
103 def randColor():
104     return "#"+''.join([random.choice('0123456789ABCDEF') for j in range(6)])
105
106 def plotData_str(self, width, height, markers=None, err=None, xScale="", yScale="",
seg=None, pltErrBar=False):
107     self.figure = plt.figure(figsize=(width, height))
108     L, B, W, H = [0.15, 0.1, 0.80, 0.85]
109     self.figure.axis = []
110     self.figure.axis.append(self.figure.add_axes([L, B, W, H]))
111     var = self.fxns2plot[0]
112     for fn in self.fxns2plot[1:]:
113         #Segment the data if neccessary

```

localhost:50187

2/7

Figure 12: Plotting Class python code

```

114     if seg is not None:
115         #Make a dictionary, for each dataset index, make a blank array
116         # print(self.data[seg])
117         segSet = {int(s) for s in self.data[seg]}
118         segSet = set(segSet)
119         segSet = list(segSet)
120         # print("segSet = ", segSet)
121         Xseg = { i:[] for i in segSet}
122         Yseg = { i:[] for i in segSet}
123         Xseg_err = { i:[] for i in segSet}
124         Yseg_err = { i:[] for i in segSet}
125         # print("Xseg = ", Xseg)
126
127         #add the data, remove blank string elements
128         for i in range(0,len(self.data[var])):
129             # print("AT LOC I = ", self.data[seg][i])
130             if (self.data[var][i] != "" and self.data[fn][i] != ""):
131                 Xseg[int(self.data[seg][i])].append(self.data[var][i])
132                 Yseg[int(self.data[seg][i])].append(self.data[fn][i])
133                 Xseg_err[int(self.data[seg][i])].append(float(self.data[err[0]]
134 [i]))
135                 Yseg_err[int(self.data[seg][i])].append(float(self.data[err[1]]
136 [i]))
137
138         # print("Xseg\n",Xseg)
139         # print("Yseg\n",Yseg)
140
141         for i in segSet:
142             #REMOVE ME
143             # if i !=3: continue
144             x = Xseg[i]
145             y = Yseg[i]
146             lbl = self.dataLabels[fn] + "_" + str(i)
147             new_x = [float(i) for i in x]
148             new_y = [float(i) for i in y]
149             x_error = Xseg_err[i]
150             y_error = Yseg_err[i]
151             #Add the datasets index to the label
152             # lbl = lbl + "_" + str(i)
153             #Set markers for plotting
154             if markers is not None:
155                 # "modified for the segmented data"
156                 mk = markers[fn][i]
157             else:
158                 mk = "."
159
160             #Colors
161             if self.dataColors is not None:
162                 # clr = self.dataColors[fn]
163                 clr = ChEplot.randColor()
164                 self.figure.axis[0].plot(new_x,new_y,mk,label=lbl,color=clr)
165             else: #I don't think this condition is possible anymore
166                 self.figure.axis[0].plot(new_x,new_y,mk,label=lbl)
167             #ERROR BARS MODIFY THIS FOR SEGMENTED DATA
168             if err is not None and pltErrBar[i] is True:
169                 # $plt.errorbar(new_x, new_y, xerr=x_error, yerr=y_error)
170                 #DELETE THIS LINE, TESTING
171                 # if i != 5: continue

```

Figure 13: Plotting Class python code


```

4/25/22, 2:26 PM /Users/wesleyjohanson/Documents/College Classes_UC Santa Barbara/ChemE 180A Chemical Engineering Lab/z_factors/ChE.py
170         self.figure.axis[0].errorbar(new_x, new_y, xerr=x_error,
yerr=y_error ,linewidth=0.9) #,ecolor=None) # fmt='none')
171     # else:
172     #     print("ERROR BARS ARE NOT PLOTTED FOR", err, "\n", pltErrBar[i])
173     #Scale
174     if xScale=="log":
175         self.figure.axis[0].set_xscale('log')
176     if yScale=="log":
177         self.figure.axis[0].set_yscale('log')
178
179 #No segmentation
180     else:
181         x = self.data[var]
182         y = self.data[fn]
183         lbl = self.dataLabels[fn]
184         #Remove Data points with blank strings that don't map to floats
185         for col in range(0, len(self.data)):
186             new_x = []
187             new_y = []
188             x_error = []
189             y_error = []
190             for row in range(0, len(self.data[0])):
191                 if self.data[col,row] != "" and self.data[var,row] != "":
192                     if err is not None:
193                         x_error.append(float(self.data[err[0], row]))
194                         y_error.append(float(self.data[err[1], row]))
195                         new_x.append(float(self.data[var,row]))
196                         new_y.append(float(self.data[col,row]))
197             #Set markers for plotting
198             if markers is not None:
199                 mk = markers[fn]
200             else:
201                 mk = "."
202             #Colors
203             if self.dataColors is not None:
204                 clr = self.dataColors[fn]
205                 self.figure.axis[0].plot(new_x,new_y,mk,label=lbl,color=clr)
206             else: #I don't think this condition is possible anymore
207                 self.figure.axis[0].plot(new_x,new_y,mk,label=lbl)
208             #ERROR BARS
209             if err is not None:
210                 plt.errorbar(new_x, new_y, xerr=x_error, yerr=y_error, )
211             #Scale
212             if xScale=="log":
213                 self.figure.axis[0].set_xscale('log')
214             if yScale=="log":
215                 self.figure.axis[0].set_yscale('log')
216
217     def plotLRegLines(self, width=0.5, style='-'):
218         var = [float(x) for x in self.data[0]]
219     # self.fxns2plot[0]
220     # fxns = self.fxns2plot[1:]
221     fxns = self.fxns2plot[1:]
222
223     # THIS IS DESIGNED FOR PLOTTING ALL COLS OF DATA WITH OLD INDVARS FUNCTIONS
224     # var = [float(x) for x in var]
225     # fxns = [float(x) for x in fxns]

```

localhost:50187

4/7

Figure 14: Plotting Class python code

```

4/25/22, 2:26 PM /Users/wesleyjohnson/Documents/College Classes/UC Santa Barbara/ChemE 180A Chemical Engineering Lab/z_factors/ChE.py
226     for fn in fxns:
227         y_0 = [float(x) for x in self.data[fn]]
228         # x_0 =
229         m, b = np.polyfit(var, y_0, 1)
230         x = np.linspace(min(var), max(var), num=len(var))
231         y = m * x + b
232         txt1 = "LinReg line for y = " + self.dataLabels[fn]
233         txt2 = "and x = " + self.dataLabels[var]
234         txt3 = "y = (%.4f" % m + ")x + (%.4f" % b + ")"
235         txt4 = "R^2 = %.4f" % ChEplot.rSquared(var, y_0)
236         print( f"{txt1:<35}{txt2:<30}{txt3:>20}{txt4:>20}")
237         if self.dataColors is None:
238             self.figure.axis[0].plot(x, y, \
239                                     linewidth=width, linestyle=style)
240         else:
241             self.figure.axis[0].plot(x, y, \
242                                     color=self.dataColors[fn], \
243                                     linewidth=width, linestyle=style)
244
245     # def calc(self):
246     #     pass
247
248     # def plotErrorBars(self):
249     #     x = self.data[self.fxns2plot[0]]
250     #     for y in self.fxns2plot[1:]:
251
252
253     #Statistics
254     @staticmethod
255     def rSquared(x, y):
256         x = x.reshape((-1,1))
257         y_reg = LinearRegression().fit(x,y)
258         return y_reg.score(x,y)
259
260     def printAllRSquared(self, precision=5, vars=None, fxns=None):
261         if vars is None:
262             vars = range(0, self.numDataSets)
263         if fxns is None:
264             fxns = range(0, self.numDataSets)
265
266         for fn in fxns:
267             for var in vars:
268                 if fn == var:
269                     continue
270                 rSquared = ChEplot.rSquared(self.data[0], self.data[fn])
271                 rStr = "R^2 = %1." + str(precision) + "f"
272                 rStr = rStr % rSquared
273                 if rStr is None: print("Error_0")
274                 if self.dataLabels is None: print("Error_1")
275                 print( f"{rStr:<25}{self.dataLabels[fn-self.numDataVars]:<20}{ 'with
respect to':<20}{self.dataLabels[var]:>10}")
276
277     #COMPLETE ME
278     def confInterv(self, n=1):
279         self.lowerBound_CI = []
280         self.upperBound_CI = []
281         for fn in range(self.numDataVars, self.numDataSets):
282             x_bar, stdDev = np.mean(self.data[fn]), np.std(self.data[fn])

```

localhost:50187

5/7

Figure 15: Plotting Class python code

```

4/25/22, 2:26 PM /Users/wesleyjohnson/Documents/College Classes_UC Santa Barbara/ChemE 180A Chemical Engineering Lab/z_factors/ChE.py
283 SE = stdDev / math.sqrt(self.num)
284 DoF = n
285 stats.t.ppf(q=0.05,)
286 scipy.stats.t.ppf(q=.05,df=22)
287
288
289 #Plot: Setters
290 def setFxn2Plot(self, fxns):
291     self.fxn2plot = fxns
292
293 def setDataStyles(self, styles: list):
294     self.lineStyles = styles
295
296 def setDataColors(self, colors=None):
297     #Assign Random Colors, if no custom color array is given
298     if colors == None:
299         colors = []
300         for i in range (len(self.data)):
301             color = "#"+''.join([random.choice('0123456789ABCDEF') for j in range(6)])
302             colors.append(color)
303
304     self.dataColors = colors
305     self.setLRegLineColors(colors)
306
307 def setAxisLabels(self, x: str, y:str, indepVar=0, xpadding=5, ypadding=5):
308     self.figure.axis[indepVar].set_xlabel(x,labelpad=xpadding)
309     self.figure.axis[indepVar].set_ylabel(y,labelpad=ypadding)
310
311 def setTicProps(self, _size=4, _width=1, _direction='in'):
312     self.figure.axis[0].xaxis.set_tick_params(which='major', size=_size, width=_width,
direction=_direction, top='on')
313     self.figure.axis[0].xaxis.set_tick_params(which='minor', size=_size, width=_width,
direction=_direction, top='on')
314     self.figure.axis[0].yaxis.set_tick_params(which='major', size=_size, width=_width,
direction=_direction, right='on')
315     self.figure.axis[0].yaxis.set_tick_params(which='minor', size=_size, width=_width,
direction=_direction, right='on')
316
317 def setNumTics(self, delta_x=0.1, delta_y=0.1, x_subTics=3, y_subTics=3):
318     self.figure.axis[0].xaxis.set_major_locator(mpl.ticker.MultipleLocator(delta_x))
319
320 self.figure.axis[0].xaxis.set_minor_locator(mpl.ticker.MultipleLocator(delta_x/x_subTics))
321 self.figure.axis[0].yaxis.set_major_locator(mpl.ticker.MultipleLocator(delta_y))
322 self.figure.axis[0].yaxis.set_minor_locator(mpl.ticker.MultipleLocator(delta_y/y_subTics))
323
324 #Plot: Features
325 def showLegend(self, x=0.01, y=0.01, width=1, height=1, _loc='lower left',
frame=True, _fontSize=10):
326     # plt.legend(bbox_to_anchor=(x,y, width, height), loc=_loc, frameon=frame,
fontSize=_fontSize)
327     plt.legend(frameon=frame, fontsize=_fontSize)
328
329 def changeFont(self, font='Alvenir', size=10, linewidth=0.9):
330     mpl.rcParams['font.family'] = font
331     plt.rcParams['font.size'] = size
332     plt.rcParams['axes.linewidth'] = linewidth
333

```

localhost:50187

6/7

Figure 16: Plotting Class python code

```

4/25/22, 2:26 PM /Users/wesleyjohnson/Documents/College Classes_UC Santa Barbara/ChemE 180A Chemical Engineering Lab/z_factors/ChE.py
334 #Presentation
335 def showPlot(self):
336     "Shows the figure"
337     plt.show()
338
339 #Save
340 def savePlot(self, filename="poop.png", _dpi=900, _transparent=False,
bbox_inches='tight'):
341     "Saves the Figure(graph) made to a file"
342     plt.savefig(filename, dpi=_dpi, transparent=_transparent,
bbox_inches=bbox_inches)
343
344 #Close the figure, plots, and data associated with the object
345 def close(self):
346     "Close the figure(window) that we were plotting in"
347     plt.close(self.figure)

```

localhost:50187

7/7

Figure 17: Plotting Class python code