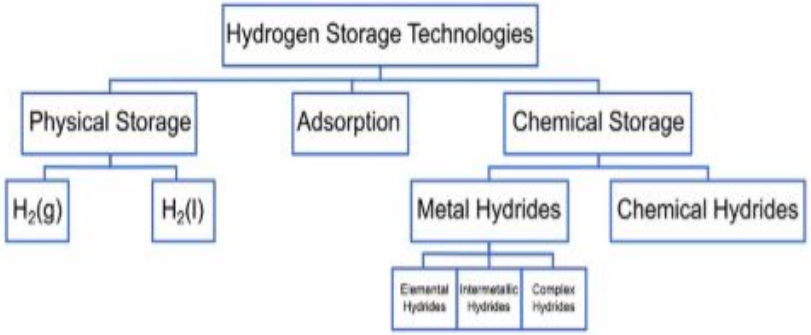
Application of Python to Analyze CO_2 Hydrogenation Reaction

By Stephen Galvin

Introduction

- Hydrogen energy is a potential solution to aid in decreasing carbon emissions
- However, storage of such a small, flammable and low energy dense molecule can be dangerous, and inefficient [1]
- One method to store hydrogen is to create methanol, from CO2 hydrogenation [1,3]
- The methanol is then used in a fuel cell [2] or a decomposition or steam reform reaction is used to turn back into H2 [1]



Hydrogen storage via methanol falls within the 'Chemical Hydrides' category [1]

Experiment

- In most cases, a Cu/ZnO/Al2O3 catalyst is used to direct the CO2 hydrogenation reaction [1,2]
- 4 different catalyst were tested in the batch reactor, we will focus on one of these catalyst, the commercial catalyst nano sized
- The catalyst was tested on 4 different days, but we will focus on one the first day only

On the left is the sonication setup, on the right is the batch reactor where the focal point data is collected. This setup is at the Advanced Energy Center (AERTC)

Goal

- To create a code that can be used for multiple runs, by only needs to input new data at the start of the code, as well as use the ‘find function’ to change some variable names
- Plots the data as the code progresses so that the user can catch any artifacts that may occur in real data due to human error or malfunction
- To find the moles of CO2 converted, while also attempting to find the rate law of the reaction in term of CO2 concentration

Type of Data Collected

- Need to create a code that can read through the reaction data in order to calculate how much CO2 was converted during reaction, and verify the reaction rate
- Data is provided by the Parr reactor program as a txt file

1	2	3	4	0	0	0	0	0	0	1 30 50
26.1	0.0	15.1	24.4	0.0	25.6	0.0	0.0	0.0	0.0	9:36:59
26.1	0.0	15.1	24.4	0.0	0.0	0.0	0.0	0.0	0.0	9:37: 0
26.1	0.0	15.1	24.4	0.0	0.0	0.0	0.0	0.0	0.0	9:37: 1
26.1	0.0	15.1	24.4	0.0	0.0	0.0	0.0	0.0	0.0	9:37: 2
26.1	0.0	15.1	24.4	0.0	0.0	0.0	0.0	0.0	0.0	9:37: 3
26.1	0.0	15.1	24.4	0.0	0.0	0.0	0.0	0.0	0.0	9:37: 4
26.1	0.0	15.1	24.4	0.0	0.0	0.0	0.0	0.0	0.0	9:37: 5
26.1	0.0	15.1	24.4	0.0	0.0	0.0	0.0	0.0	0.0	9:37: 6
26.1	0.0	15.1	24.4	0.0	0.0	0.0	0.0	0.0	0.0	9:37: 7
26.1	0.0	15.1	24.4	0.0	0.0	0.0	0.0	0.0	0.0	9:37: 8
26.1	0.0	15.1	24.4	0.0	0.0	0.0	0.0	0.0	0.0	9:37: 9
26.1	0.0	15.1	24.4	0.0	0.0	0.0	0.0	0.0	0.0	9:37:10
26.1	0.0	15.1	24.3	0.0	0.0	0.0	0.0	0.0	0.0	9:37:11
26.1	0.0	15.1	24.3	0.0	0.0	0.0	0.0	0.0	0.0	9:37:12
26.1	0.0	15.1	24.3	0.0	0.0	0.0	0.0	0.0	0.0	9:37:13
26.1	0.0	15.1	24.3	0.0	0.0	0.0	0.0	0.0	0.0	9:37:14
26.1	0.0	15.1	24.3	0.0	0.0	0.0	0.0	0.0	0.0	9:37:15
26.1	0.0	15.1	24.3	0.0	0.0	0.0	0.0	0.0	0.0	9:37:16
26.1	0.0	15.1	24.3	0.0	0.0	0.0	0.0	0.0	0.0	9:37:17
26.1	0.0	15.1	24.3	0.0	0.0	0.0	0.0	0.0	0.0	9:37:18
26.1	0.0	15.1	24.3	0.0	0.0	0.0	0.0	0.0	0.0	9:37:19
26.1	0.0	15.1	24.3	0.0	0.0	0.0	0.0	0.0	0.0	9:37:20
26.1	0.0	15.1	24.3	0.0	0.0	0.0	0.0	0.0	0.0	9:37:21
26.1	0.0	15.1	24.3	0.0	0.0	0.0	0.0	0.0	0.0	9:37:23
26.1	0.0	15.1	24.3	0.0	0.0	0.0	0.0	0.0	0.0	9:37:24

The raw text file data

Reading the Txt File

- Using pandas to open the text file, each column is saved under a variable name, which is then used later in the code
- Pandas concat to put separate files together
- A time scale is created using numpy linspace

```
#import packages
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from scipy.optimize import curve_fit
```

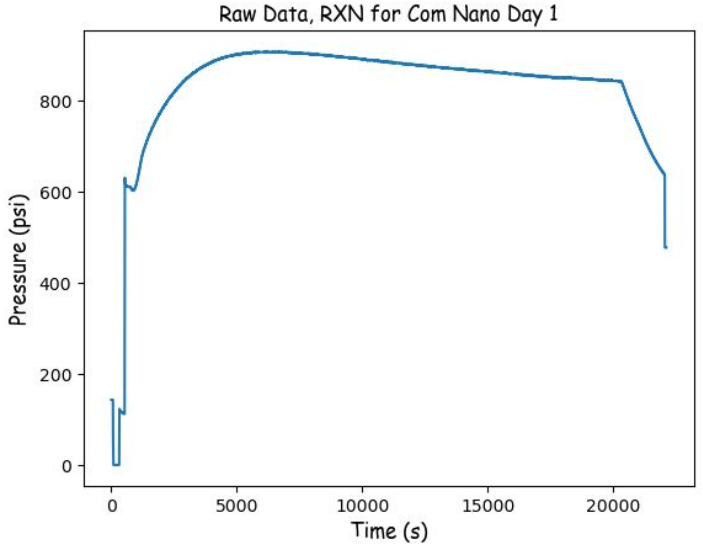
```
#Opens the data files for day 1 RXN data
#Com_Nano data files, 2 because turned off the program

print('If there is an error regarding size of dataframe, user may need to create two random column headers, since time is div
dataFolder = "C:/Users/steph/OneDrive/Desktop/AERTC Thesis/ReactionDataMethanol/"

#Two files, one during the heating and recation, one the following day or so to collect the final pressure
Rxn_Com_Nanos_Day1_P1 = pd.read_csv(dataFolder+'Week_02_25/Com_Nano_Rxn_02_24.txt', delim_whitespace = True)
Rxn_Com_Nanos_Day1_P2 = pd.read_csv(dataFolder+'Week_02_25/Com_Nano_Rxn_02_24_end.txt', delim_whitespace = True)

#Put the two programs together
Rxn_Com_Nanos_Day1 = pd.concat([Rxn_Com_Nanos_Day1_P1, Rxn_Com_Nanos_Day1_P2], axis = 0, ignore_index=True)

Heater_Com_Nanos_Day1 = Rxn_Com_Nanos_Day1['1']
#2 is the 150 rpm motor, it is not kept track of, but is constant
Pressure_Rxn_Com_Nanos_Day1 = Rxn_Com_Nanos_Day1['3']*10
Temperature_Rxn_Com_Nanos_Day1 = Rxn_Com_Nanos_Day1['4']
#Create a time scale in seconds, use any length in order to get the right time scale
x_Rxn_Com_Nanos_Day1 = np.linspace(0,len(Temperature_Rxn_Com_Nanos_Day1)-1,len(Temperature_Rxn_Com_Nanos_Day1))
```

```
If there is an error regarding size of dataframe, user may need to create two random column headers, since time is divided b
y space
```

Raw Data Plot

- To understand what the raw data looks like, and to see if there are any large artifacts, the raw data is plotted
- Pressure is added, 125 psi of CO2, then 500 psi of H2

```
In [3]:  #Plot the raw data for pressure
         fontdict = {'fontname': 'comic sans ms', 'fontsize': 12}
         plt.plot(x_Rxn_Com_Nanos_Day1,Pressure_Rxn_Com_Nanos_Day1)
         plt.title('Raw Data, RXN for Com Nano Day 1', fontdict)
         plt.xlabel('Time (s)', fontdict)
         plt.ylabel('Pressure (psi)', fontdict)

Out[3]: Text(0, 0.5, 'Pressure (psi)')
```



Looping to Divide the Data

- The next slide will show 3 loops, each to find points where important events occur:

1) The time when pressure is at its max value
2) When the temperature slope begins to decrease starting at the max pressure value. At this time, temperature is increasing due to the exothermic reaction, therefore a negative slope indicates reactor heat off
3) When the reactor heat is turned on

Looping to Divide the Data

```
#find the x value when the rxn starts, therefore pressure starts to decrease
for i in x_Rxn_Com_Nanos_Day1:
    if Pressure_Rxn_Com_Nanos_Day1[i] >= Pressure_Rxn_Com_Nanos_Day1.max(): #Tried setting 'equal to', but that set every value equal to the max, had to re-run code
        break

#Create another array starting from the start of rxn, so it does not pick up any other slopes
x_Rxn_Start_Com_Nanos_Day1 = np.linspace(int(i),int(len(Pressure_Rxn_Com_Nanos_Day1) - 1), int(len(Pressure_Rxn_Com_Nanos_Day1) - i)) #would not take the numbers unless they were i

#Find the x value when the slope of the temperature starts to drop dramatically, due to reactor shut off
for n in (x_Rxn_Start_Com_Nanos_Day1):
    slope = (Temperature_Rxn_Com_Nanos_Day1[int(n) + 60] - Temperature_Rxn_Com_Nanos_Day1[int(n)]) / (x_Rxn_Com_Nanos_Day1[int(n) + 60] - x_Rxn_Com_Nanos_Day1[int(n)])
    if slope <= -0.1: #The slope of pressure drop is much larger when the reactor is turned off
        break

#Also a loop that finds the time when the reactor heat is turned on
for j in (x_Rxn_Com_Nanos_Day1[0:(len(x_Rxn_Com_Nanos_Day1) - 60)]):
    slope = (Heater_Com_Nanos_Day1[int(j) + 60] - Heater_Com_Nanos_Day1[int(j)]) / (x_Rxn_Com_Nanos_Day1[int(j) + 60] - x_Rxn_Com_Nanos_Day1[int(j)])
    if slope > 0.2: #The slope is higher right when the reactor is turned on
        break

#Linspace likes integers over floats, make next step easier
i = int(i)
n = int(n)
j = int(j)

print('The starting time in when the pressure reaches max is {} seconds'.format(i))
print('The ending time when the reactor is turned off is {} seconds'.format(n))
print('The reaction was left to run for aprroximately {:.0f} minutes'.format((n-i)/60))
print('Once the reactor temperature was set, it took {:.0f} minutes for the reaction to start from max pressure'.format((i-j)/60))
```

```
The starting time in when the pressure reaches max is 6199 seconds
The ending time when the reactor is turned off is 20274 seconds
The reaction was left to run for aprroximately 235 minutes
Once the reactor temperature was set, it took 90 minutes for the reaction to start from max pressure
```

Plotting the Data Before T and P Correction

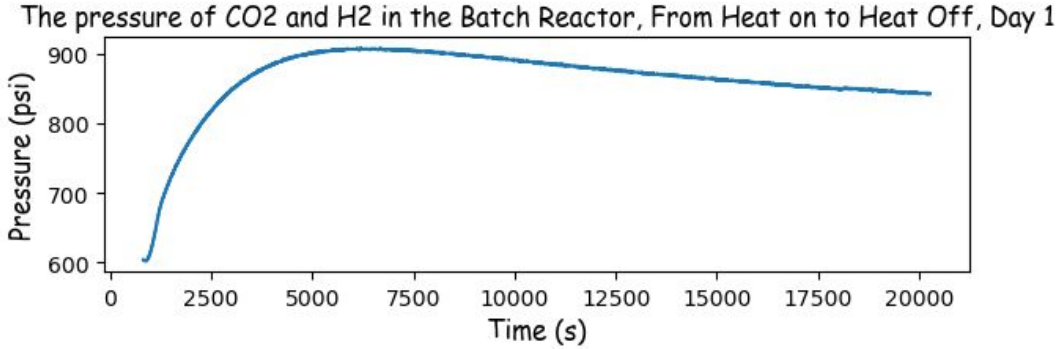
- After breaking the data apart, it was obvious that as the pressure was decreasing (the reaction was taking place), the temperature was in fact increasing as well
- Considering the ideal gas law, since the pressure was decreasing but the temperature was increasing, the pressure increasing due to the exothermic heat created must be accounted for
- Therefore, it must be assumed that all temperature increase after the pressure reaches max is due to the heat produced

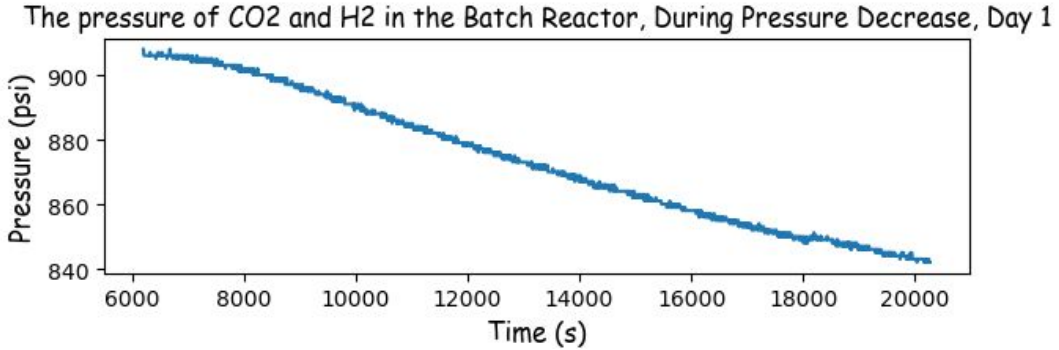
Plotting the Data Before T and P Correction

```
#Now plot the data from the reaction start to the time when the reactor is turned off
plt.subplot(2,1,1)
time_heat = np.linspace(int(x_Rxn_Com_Nanos_Day1[j]), int(x_Rxn_Com_Nanos_Day1[n]) - 1, (int(x_Rxn_Com_Nanos_Day1[n]) - int(x_Rxn_Com_Nanos_Day1[j])))
plt.plot(time_heat,Pressure_Rxn_Com_Nanos_Day1[j:n])
plt.xlabel('Time (s)', fontdict)
plt.ylabel('Pressure (psi)', fontdict)
plt.title('The pressure of CO2 and H2 in the Batch Reactor, From Heat on to Heat Off, Day 1', fontdict)

plt.subplot(2,1,2)
time_rxn = np.linspace(int(x_Rxn_Com_Nanos_Day1[i]), int(x_Rxn_Com_Nanos_Day1[n]) - 1, (int(x_Rxn_Com_Nanos_Day1[n]) - int(x_Rxn_Com_Nanos_Day1[i])))
plt.plot(time_rxn, Pressure_Rxn_Com_Nanos_Day1[i:n])
plt.xlabel('Time (s)', fontdict)
plt.ylabel('Pressure (psi)', fontdict)
plt.title('The pressure of CO2 and H2 in the Batch Reactor, During Pressure Decrease, Day 1', fontdict)

#This rxn is exothermic, and therefore there was a slight temperature increase, despite pressure decreasing
print('Temperature difference from start of pressure drop to heat off was {:.0f} degrees celsius'.format(Temperature_Rxn_Com_Nanos_Day1[n] - Temperature_Rxn_Com_Nanos_Day1[i]))

plt.tight_layout()
```

Plotting the Data Before T and P Correction

Temperature difference from start of pressure drop to heat off was 11 degrees celsius





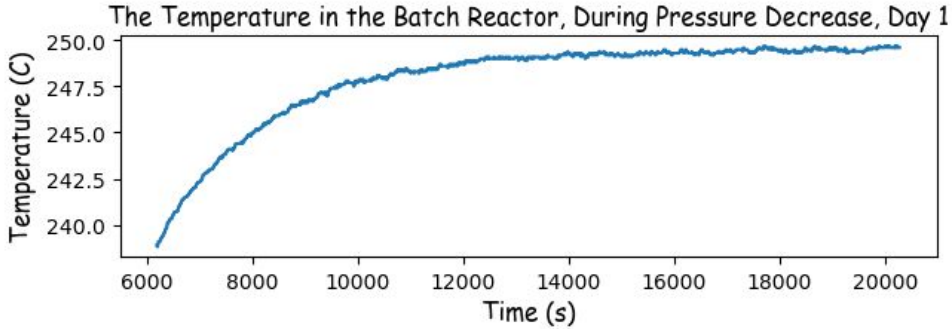
Using PV=nRT to Figure Expected Pressure Increase

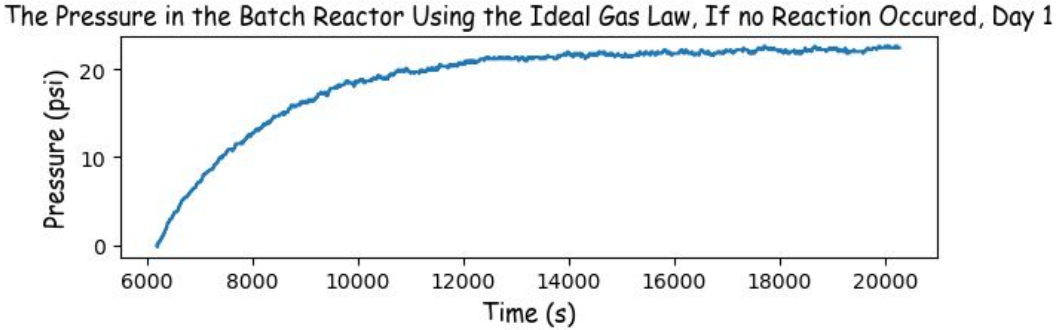
- Plots the temperature during the time from pressure drop to heat off
- Using PV=nRT, finds the amount of psi you could expect if there was no reaction occurring

```
#Plot of the temperature during the pressure change
plt.subplot(2,1,1)
plt.plot(time_rxn, Temperature_Rxn_Com_Nanos_Day1[i:n])
plt.xlabel('Time (s)', fontdict)
plt.ylabel('Temperature (C)', fontdict)
plt.title('The Temperature in the Batch Reactor, During Pressure Decrease, Day 1', fontdict)

#Find expected pressure if there was no pressure drop from max pressure, assuming moles of CO2 did not change before

#For volume, we assume hexadecane will stay constant under changing temperature at the moment
P_exp = ((CO2_n0 + H2_n0)*8.314*(Temperature_Rxn_Com_Nanos_Day1[i:n].add(273.15))/0.000225)/6894.76
print('The initial moles of CO2 on day 1: {:.2f} mols'.format(CO2_n0))
print('The initial moles of H2 on day 1: {:.2f} mols'.format(H2_n0))

#plot the actual pressure vs the expected pressure if there was no reaction
plt.subplot(2,1,2)
P_change = (P_exp - P_exp[i]) #P_exp - P_exp[i] will give the absolute difference in pressure
plt.plot(time_rxn, P_change)
plt.xlabel('Time (s)', fontdict)
plt.ylabel('Pressure (psi)', fontdict)
plt.title('The Pressure in the Batch Reactor Using the Ideal Gas Law, If no Reaction Occured, Day 1', fontdict)

plt.tight_layout()

#Initial moles when temperature is room temperature and inlet pressure was 625 psi, not important at the moment
CO2_n0 = (125*6894.76*0.000225)/(8.314*(25+273.15))
H2_n0 = (500*6894.76*0.000225)/(8.314*(25+273.15))
```

Using PV=nRT to Figure Expected Pressure Increase

The initial moles of CO2 on day 1: 0.08 mols
The initial moles of H2 on day 1: 0.31 mols





Plotting and Calculating Moles of CO2 Reacted

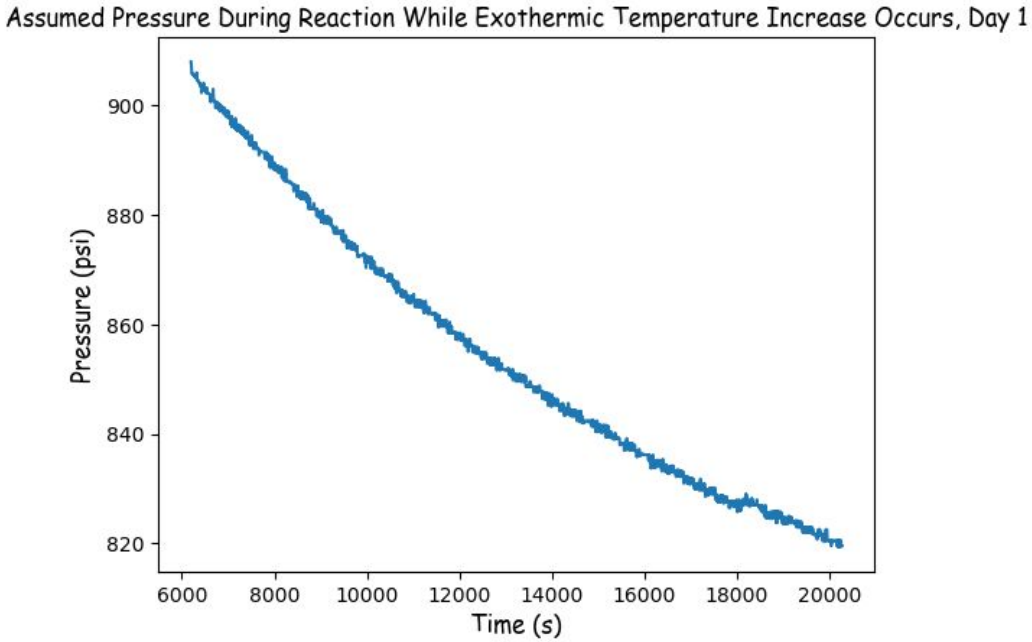
- Pressure during the reaction will now be converted into pascals, while the temperature during the reaction will be converted into kelvin. The pressure expected is easily input in this section, at initial and final time
- This will allow for the amount of moles of CO2 consumed to be calculated using the ideal gas law
- Then, a for loop is utilized in order to subtract the pressure element wise in the pandas series to get the expected pressure drop

Plotting and Calculating Moles of CO2 Reacted

```
#Convert psi change to moles overall converted, not accounting for temperature change
#The equation for CO2 hydrogenation: CO2 + 3H2 <--> CH3OH + H2O

#Assume ideal conditions, and correct for the pressure change due to temperature increase during pressure drop:
P = ((Pressure_Rxn_Com_Nanos_Day1[i] - P_change[i]) - (Pressure_Rxn_Com_Nanos_Day1[n] - P_change[n - 1]))*6894.76 #Final - initial, convert to m^3 f
T = (Temperature_Rxn_Com_Nanos_Day1[n] - Temperature_Rxn_Com_Nanos_Day1[i]) + 273.15 #Inital - Final, to avoid negative, convert to K
#V is constnat, equals 0.000225 m^3, because 300 mL reactor and 75 mL is hexadecane

#PV = nRT, could try compresssability or vander waals, but mostly hydrogen and therefore pretty ideal
CO2_n = ((P*0.000225)/(8.314*T))/4 #Divide by 4, as 4 mols react, but only 1 is CO2
print('The moles of CO2 reacted according to the pressure change is {:.3f} mols'.format(CO2_n))
#1:1 ratio of CO2 to methanol
methanol_vol = CO2_n*32.04/0.7913
print('The volume of methanol created is {:.3f} mL'.format(methanol_vol))

#Subtract the expected pressure change, element wise
Pressure_Edit = []
for d in np.linspace(int(x_Rxn_Com_Nanos_Day1[i]), int(x_Rxn_Com_Nanos_Day1[n]) - 1, (int(x_Rxn_Com_Nanos_Day1[n]) - int(x_Rxn_Com_Nanos_Day1[i]))):
    Pressure_Edit.append(Pressure_Rxn_Com_Nanos_Day1[d] - P_change[d])

plt.plot(time_rxn, Pressure_Edit)
plt.xlabel('Time (s)', fontdict)
plt.ylabel('Pressure (psi)', fontdict)
plt.title('Assumed Pressure During Reaction While Exothermic Temperature Increase Occurs, Day 1', fontdict)
```

Plotting and Calculating Moles of CO2 Reacted

```
The moles of CO2 reacted according to the pressure change is 0.015 mols
The volume of methanol created is 0.588 mL

Text(0.5, 1.0, 'Assumed Pressure During Reaction While Exothermic Temperature Increase Occurs, Day 1')
```

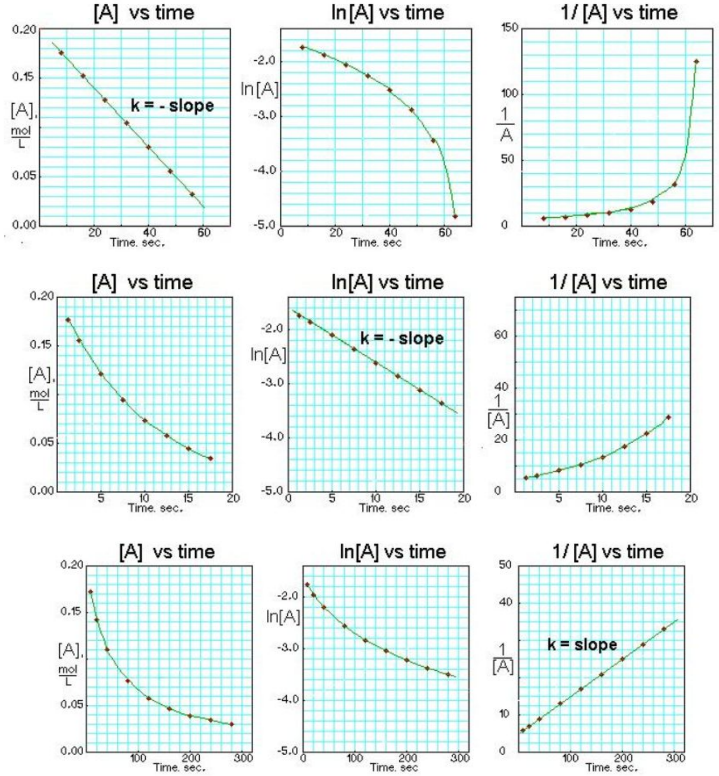


Reaction Order in Relation to CO2

- Convert the moles of CO2 converted into the concentration of CO2 converted, over the entire pressure drop with the pressure correction due to temperature increase factored in
- Use linear regression to see if the reaction order with respect to CO2 can be calculated, using the ‘curve_fit’ function
- ‘.reset_index’ is needed so that both the edited pressure series and the temperature series are at the same index values. The same length is not enough to create same length series to plot with the typical reaction time variable

Background on Integrated Rate Laws

- In order to find the integrated rate law, the concentration of a reactant must be plotted 3 different ways:
  - [reactant]
  - Ln[reactant]
  - 1 / [reactant]
- As shown in the images on the right, if the trends follow the top row of plots, it is 0th order, if it follows the second row it is 1st order, and the last row would follow 2nd order [4].
- This is why linear regression will be used in order to fit the data



Images from Purdue University [4]

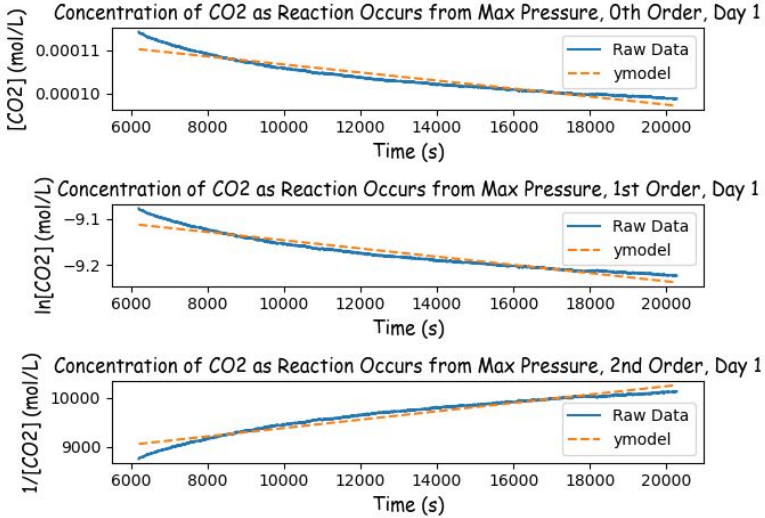
Reaction Order in Relation to CO2

```
#Need to get concentration change over time though
Pressure_Edit = pd.Series(Pressure_Edit)
CO2_conc = (((Pressure_Edit*0.000225)/(8.314*Temperature_Rxn_Com_Nanos_Day1[i:n].reset_index(drop=True)))/4)/0.225  #0.225 L

#define a function of a linear line, to see if any of these plot are linear
def f_lin(x,a0,a1): #for the curve_fit we must put x first
    return a0+ a1*x

#Plot to zeorth order plot-----------------------------------------------------------------------------------------
plt.subplot(3,1,1)
plt.plot(time_rxn,CO2_conc)
plt.xlabel('Time (s)', fontdict)
plt.ylabel('[CO2] (mol/L)', fontdict)
plt.title('Concentration of CO2 as Reaction Occurs from Max Pressure, 0th Order, Day 1', fontdict)

par,cov = curve_fit(f_lin,time_rxn,CO2_conc)   #Linear function, x, y data
print('\nThe slope of the line equation is {}'.format(par[1]))
ymodel = f_lin(time_rxn, *par) #* means more than one value attached to said variable
plt.plot(time_rxn, ymodel, linestyle = 'dashed')
plt.legend(['Raw Data','ymodel'])

#Find the residuals
res = CO2_conc - ymodel #Diff between actual and predicted values
#Find the R^2 value
ss_res = sum(res**2)
ss_tot = sum((CO2_conc - np.mean(CO2_conc))**2)
r2 = round(1-(ss_res/ss_tot),4)
print('The r^2 value for zeroth order is {}\n'.format(r2))
```

```
#Plot to first order plot------------------------------------------------------------------------------------------
plt.subplot(3,1,2)
plt.plot(time_rxn,np.log(CO2_conc)) #np.log is base 10 log, or ln
plt.xlabel('Time (s)', fontdict)
plt.ylabel('ln[CO2] (mol/L)', fontdict)
plt.title('Concentration of CO2 as Reaction Occurs from Max Pressure, 1st Order, Day 1', fontdict)

par,cov = curve_fit(f_lin,time_rxn,np.log(CO2_conc))   #Linear function, x, y data
print('\nThe slope of the line equation is {}'.format(par[1]))
ymodel = f_lin(time_rxn, *par) #* means more than one value attached to said variable
plt.plot(time_rxn, ymodel, linestyle = 'dashed')
plt.legend(['Raw Data','ymodel'])

#Find the residuals
res = np.log(CO2_conc) - ymodel #Diff between actual and predicted values
#Find the R^2 value
ss_res = sum(res**2)
ss_tot = sum((np.log(CO2_conc) - np.mean(np.log(CO2_conc)))**2)
r2 = round(1-(ss_res/ss_tot),4)
print('The r^2 value for first order is {}\n'.format(r2))

#Plot to second order plot-----------------------------------------------------------------------------------------
plt.subplot(3,1,3)
plt.plot(time_rxn, 1/CO2_conc) #np.log is base 10 log, or ln
plt.xlabel('Time (s)', fontdict)
plt.ylabel('1/[CO2] (mol/L)', fontdict)
plt.title('Concentration of CO2 as Reaction Occurs from Max Pressure, 2nd Order, Day 1', fontdict)

par,cov = curve_fit(f_lin,time_rxn, 1/CO2_conc)   #Linear function, x, y data
print('\nThe slope of the line equation is {}'.format(par[1]))
ymodel = f_lin(time_rxn, *par) #* means more than one value attached to said variable
plt.plot(time_rxn, ymodel, linestyle = 'dashed')
plt.legend(['Raw Data','ymodel'])

#Find the residuals
res = 1/CO2_conc - ymodel #Diff between actual and predicted values
#Find the R^2 value
ss_res = sum(res**2)
ss_tot = sum(((1/CO2_conc) - np.mean(1/CO2_conc))**2)
r2 = round(1-(ss_res/ss_tot),4)
print('The r^2 value for first order is {}\n'.format(r2))


plt.tight_layout()
```

Reaction Order in Relation to CO2

```
The slope of the line equation is -9.318278383164568e-10
The r^2 value for zeroth order is 0.9227


The slope of the line equation is -8.9083942805856e-06
The r^2 value for first order is 0.9322


The slope of the line equation is 0.08524754609710072
The r^2 value for first order is 0.9411
```



Results and Discussion

- None of the plots that are used to determine the rate law fit properly to the linear regression line that was plotted using the expected data
- It can be possible that the relation to reaction order is 2nd order, however the R^2 value is seemingly too far from 1 to say for certain
- The reaction order may be too difficult to find with our data, considering the catalyst is playing a major role in the reaction rate and is overshadowing the concentration change of CO_2
- However, moles of CO_2 was able to be calculated while accounting for the expected pressure increase

Future Works and Conclusions

- This code will be used for the other types of catalyst that were tested and as well as for the other reaction days, after the H2 and CO2 recharges, allowing to test the durability of the catalyst
- This code could be used for other batch reactor type testing for exothermic reactions, in order to find the reaction rate
- Additions to the code will be made, focusing on comparing the amount of CO2 converted amongst different catalysts, as well as methanol created
- Find a way to factor in the slight change in density and therefore volume of hexadecane, as well as assess other corrections that can be added

Citations

1) Andersson, J., & Grönkvist, S. (2019). Large-scale storage of hydrogen. *International Journal of Hydrogen Energy*, *44*(23), 11901–11919. https://doi.org/10.1016/j.ijhydene.2019.03.063

2) *Fuel cells*. Methanol Institute. (2022, August 16). https://www.methanol.org/fuel-cells/#:~:text=Methanol%20is%20an%20excellent%20hydrogen,hydrogen%20for%20fuel%20cell%20cars.

3) Qaderi, J. (2020). A brief review on the reaction mechanisms of CO2 hydrogenation into methanol. *International Journal of Innovative Research and Scientific Studies*, *3*(2), 33–40. https://doi.org/10.53894/ijirss.v3i2.31

4) Purdue Univeristy, 'Rate Laws from Graphs of Concentration Versus Time (Integrated Rate Laws)', https://www.chem.purdue.edu/gchelp/howtosolveit/Kinetics/IntegratedRateLaws.html