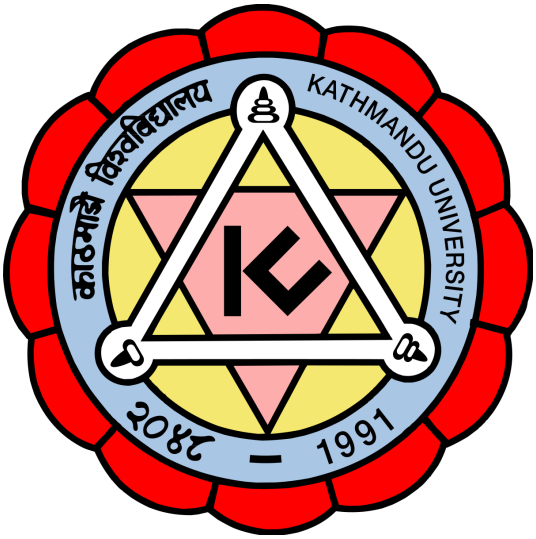
ASSIGNMENT

A lab report submitted on partial fulfillment of the course
Modelling and Simulation (**CHEG305**)
by:

Name: **ASHAL ADHIKARI**
Roll no: **02**



Submitted to:

Dr. Kundan Lal Shrestha
Associate Professor

Er. Sagar Ban
Teaching Assistant

DEPARTMENT OF CHEMICAL SCIENCE AND ENGINEERING
SCHOOL OF ENGINEERING

KATHMANDU UNIVERSITY

JUNE 2023

Assignment : Part A.

1). Discretize the Poisson equation $\nabla^2 u = q$.

ans: We have,

$$\nabla^2 u = q.$$

So,

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = q$$

as we know, the taylor series expansion of 'u_{i+1}' and 'u_{i-1}' are:

$$u_{i+1} = u_i + \Delta x \left.\frac{\partial u}{\partial x}\right|_i + \frac{\Delta x^2}{2} \frac{\partial^2 u}{\partial x^2} + O(\Delta x^3) \quad \ldots\ldots (i)$$

$$u_{i-1} = u_i - \Delta x \left.\frac{\partial u}{\partial x}\right|_i + \frac{\Delta x^2}{2} \frac{\partial^2 u}{\partial x^2} - O(\Delta x^3) \quad \ldots\ldots (ii)$$

So,

adding (i) and (ii),

$$u_{i+1} + u_{i-1} = 2u_i + \Delta x^2 \frac{\partial^2 u}{\partial x^2} + O(\Delta x^4).$$

$\rightarrow 0$ [assuming $\Delta x \to 0$]

Now,

$$\frac{\partial^2 u}{\partial x^2} = \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2}$$

Similarly,

$$\frac{\partial^2 u}{\partial y^2} = \frac{u_{j+1} - 2u_j + u_{j-1}}{\Delta y^2}$$

Then,

Since, $q(x,y)$ is a function of both x and y,

So,

we have,

$$\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} + \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta y^2} = q$$

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2} = q_{i,j}$$ //.

2). Incorporate Neumann and Dirichlet boundary condition.

ans: Dirichlet boundary condition specifies the value of the function at the boundary i.e.

for $y'' + y = 0$,

boundary conditions are:

$y(a) = \alpha$ and $y(b) = \beta$.

It is easily implemented in Python and in finite difference method by setting the value of 'α' and 'β' at ~~[a, b]~~. 'a' and 'b'. ie.

if: (in laplace equation step)

$p = 0$ at $x = 0$ and $p = 2$ at $x = 2$, at interval $[0, 2]$,

we could use:

```
p[:, 0] = 0
p[:, -1] = 2
```

where, the lines of code set first column of every row to 'zero' and line column of every row to 'two'.

Now. Neumann boundary condition specifies the derivative at the boundary i.e. for $y'' + y = 0$,

boundary conditions are:

$$\left.\frac{dy}{dx}\right|_{x=a} = \alpha \quad \text{and} \quad \left.\frac{dy}{dx}\right|_{x=b} = \beta.$$

for the implementation of Neumann condition in Python, [∵ taking an example from Laplace equation]: if,

$$\frac{\partial p}{\partial y} = 0 \quad \text{at } y = 0 \text{ and } 1, \quad \text{at interval } y[0,1]:$$

we could write,

~~p[0,:~~

```
p[0,:] = p[1:]
p[-1,:] = p[-2,:]
```

So, the first line of code implements $\partial p/\partial y = 0$ at $y = 0$ by setting the first row of 'p' to be equal with second row of 'p'. Since, in finite difference,

$$\frac{\partial p}{\partial y} = \frac{p_i - p_{i-1}}{\Delta y},$$

this would result in giving the value 'zero'. Similarly, the second line of code implements $\partial p/\partial y = 0$ at $y = 1$.

Question No. 4

(b) A mixture of He and N_2 gas is contained in a pipe at 298 K and 1 atm total pressure which is constant throughout. At one end of the pipe at point 1 the partial pressure p_{A1} of He is 0.60 atm and at the other end 0.2 m p_{A2} = 0.20 atm. Calculate the flux of He at steady state if D_{AB} of the He-N_2 mixture is 0.687×10^{-4} m^2/s. **{Geankoplis 6.1-1}**

In [5]:
```
T = 298             # K
x = 0.2             # m
p_A1 = 0.60         # atm
p_A2 = 0.20         # atm
D_AB = 0.687e-4     # m^2/s
R = 8.206e-5        # atm*m3/mol*K

Ca = p_A1 / (R * T)
Cb = p_A2 / (R * T)


flux_A = - D_AB * (Cb - Ca) / x
print(f"Flux of Helium is = {flux_A:.5f} mol/(m\N{superscript two} s)")
```

```
Flux of He is = 0.00562 mol/(m² s)
```

(c) Ammonia gas (A) is diffusing through a uniform tube 0.10 m long containing N_2 gas (B) at 1.0132×10^5 Pa press and 298 K. At point 1, $p_{A1} = 1.013 \times 10^4$ Pa and at point 2, p_{A2} =0.507 $\times 10^4$ Pa. The diffusivity $D_{AB} = 0.230 \times 10^{-4}$ m^2/s. (a) Calculate the flux J_A^* at steady state. (b) Repeat for J_B^* . **{Geankoplis 6.2-1}**

In [18]:
```
p_A1 = 1.013e4            # Pa
p_A2 = 0.507e4            # Pa

DAB =  0.230e-4           # m2/s
R = 8.314                 #m3*Pa/mol*K

x = 0.10                  # m
T = 298                   #K

Ca = p_A1 / (R * T)
Cb = p_A2 / (R * T)

flux_A = - DAB * (Cb - Ca) / x
flux_B = - flux_A

print(f"Flux of Nitrogen is = {flux_A:.6f} mol/(m\N{superscript two} s)")
print(f"Flux of Ammonia is = {flux_B:.6f} mol/(m\N{superscript two} s)")
```

```
Flux of Nitrogen is = 0.000470 mol/(m² s)
Flux of Ammonia is = -0.000470 mol/(m² s)
```

(d) Water in the bottom of a narrow metal tube is held at a constant temperature of 293 K. The total pressure of air (assumed dry) is 1.01325×10^5 Pa and the temperature is 293 K. Water evaporates and diffuses through the air in the tube and the diffusion path z_2—z_1 is 0.1524 m long. Calculate the rate of evaporation at steady state in kg mol/s·m^2. The diffusivity of water vapor at 293 K and 1 atm pressure is 0.250×10^{-4} m^2/s. Assume that the system is isothermal.**{Geankoplis 6.2-2}**

In [23]:
```
P_total = 1.013e5           # Pa
T = 293                     # K

P_1 = 1.333e4               # Pa
P_2 = 6.666e3               # Pa
x = 0.1524                  # m
DAB = 2.50e-4               # m^2/s
R = 8314                    #m3*Pa/kmol*K

C_w = P_total / (R * T)

Evap = - DAB * C_w / x

print(f"Evapouration rate is = {Evap:.7f} kmol/(m\N{superscript two} s)")
```

```
Evapouration rate is = -0.0000682 kmol/(m² s)
```

(e) Ammonia gas is diffusing through N_2 under steady-state conditions with N_2 nondiffusing since it is insoluble in one boundary. The total pressure is 1.013×10^5 Pa and the temperature is 298 K. The partial pressure of NH_3 at one point is 1.333×10^4 Pa and at the other point 20 mm away it is 6.666×10^3 Pa. The D_{AB} for the mixture at 1.013×10^5 Pa and 298 K is 2.30×10^{-5} m^2/s. Calculate the flux of NH_3 in kg mol/s·m^2.**{G.Ex.6.2-3}**

In [24]:
```
P_total = 1.013e5           # Pa
T = 298                     # K

P_1 = 1.333e4               # Pa
P_2 = 6.666e3               # Pa
x = 20e-3                   # m
DAB = 2.30e-5               # m^2/s
R = 8314                    #m3*Pa/kmol*K

Ca = P_1 / (R * T)
Cb = P_2 / (R * T)

flux_NH3 = - DAB * (Cb - Ca) / x

print(f"Flux of Ammonia is = {flux_NH3:.7f} kmol/(m\N{superscript two} s)")
```

```
Flux of Ammonia is = 0.0000031 kmol/(m² s)
```

In [ ]:
```

```

4).

ans: Statistics, Data mining, Data analytics and Data Science are all related fields that deal with collecting, analyzing and interpreting data.

- Statistics is a practice of collecting and analyzing numerical data in large quantities and draw conclusions from it. It deals by determining reliable conclusions from the data given using statistical tools like hypothesis testing and regression analysis.
- Data mining is the process of discovering patterns in large datasets. It involves using MLAs, clustering techniques and statistical models to identify trends and relationship in datas.
- Data analytics is the practice of extracting insights from data to make informed business decisions. It involves statistics, computer scienu and business intelligence to analyze data and generate actionable recommendations.
- Data science is a multidisciplinary field that combines elements of statistics, mathematics, computer science to extract knowledge and insights from structured and unstructured data.

5)

ans: Supervised learning is a machine learning platform in which an algorithm is trained on labelled data with the goal of making accurate predictions on new, unseen data. The labeled data consists of input features and corresponding output labels.

Regression is a supervised learning task that involves predicting a numerical value for the input feature. It is useful for problem that requires numerical data as output such as housing prices from input features as no. of bedrooms, square footage area, location etc.

Classification is another type of supervised learning that predicts a categorical label or class. Example would be to find the weather pattern from input features such as temperature, pressure, wind speed and relative humidity etc.

6).

ans: Unsupervised learning is another ML platform where the NN model learns patterns of the input data without any output labeled data. The model tries to train the ~~algo~~ algorithm to find patterns in the data without knowing what the output of the data should be.

Clustering is an example of unsupervised learning where the algorithm groups similar datapoints together based on their features. For eg. clustering can be used to group customers based on their purchasing habits

Anomaly detection is another unsupervised learning where the algorithm identifies unusual or rare datapoints that don't fit within the expected pattern. This learning can be used for fraud detection and error identification in data and other unusual behaviour detecting scenarios.

7).

ans: Reinforcement learning is a type of machine learning that involves learning from experience through trial and error interactions with the environment. In this approach, an agent learns to make decisions based on the rewards or punishment it receives for its actions.

Markov decision process is an mathematical framework used to model decision making in situations where outcomes are partly random and partly under the ~~und~~ control of the decision maker. For eg. an autonomous car navigating through traffic can be modelled where the car's action (brake, turn) affects its future state (position, velocity) based on current state (position, velocity).

Q-learning is a type of RL algorithm that uses a table of Q-values to guide the behaviour of the agent. Q-values represent the expected reward for taking particular action in a given state. The algorithm updates this values as the agent interacts with the environment. For eg. it is used in a robot learning to navigate through a maze can use Q-learning to determine which direction to move in at each step based on the rewards received for reaching the goal or punishment for hitting a wall.

Monte-Carlo methods are a class of RL algorithms that learns by averaging the returns obtained from episodes of interactions with the environment. Unlike Q-learning, Monte-Carlo methods don't use value function to estimate the expected reward of each action in a given state. For eg. a game-playing agent can use Monte-Carlo methods to estimate the value of each possible move by playing out many games from that position and averaging the outcomes.

8).

ans: The importances of such libraries are:

- NumPy : It is a library that provides support for arrays and matrices. It provides advanced mathematical operations on arrays and matrices in quick time which is essential for many data analysis.

- SciPy : It is another Python library that is built upon NumPy to provide additional functionality for scientific computation. It provides modules for optimization, integration, interpolation, linear algebra which is widely used to solve complex problems such as ODEs and PDEs.

- Matplotlib : It is the Python library used to create static, animated or interactive visualizations of data. This library provides variety of graphs, charts and plots that help to communicate data insights in an intuitive manner.

- Pandas : ~~Buit~~ Built upon the NumPy library, Pandas can be used to provide easy-to-use data structures to store and manipulate tabular data. It also provides user to manipulate data in various ways by applying various filter, aggregrate, flexible and grouping tools.

9).

ans: Some of the Machine Learning Core Libraries in Python are:

- Scikit-learn
- Tensorflow
- Pytorch
- Theano
- NumPy
- Pandas

10)

ans: Both the normalization and standardization are common rescaling data in ML. While normalization scales the value of the features to fall in range between 0 and 1, standardization transforms the data to have zero mean and unit variance.

Formulas is:

for normalization:

$$X_{norm} = \frac{x - min(x)}{max(x) - min(x)}$$

for standardization:

$$X_{std} = \frac{x - mean(x)}{sd(x)}.$$

Now,

Normalization	Standardization
Normalization scales the data to the range of 0 and 1.	Standardization scales the data to have mean of 0 and unit variance.
It is more useful for data that doesn't follows Gaussian distribution.	It is useful for data that follow Gaussian distribution.
It is affected by outliers since it accounts for minimum and maximum values of dataset.	It ~~isn't~~ has no affect of outliers and is more robust.
It is more easy to implement and understand.	It requires data properties and statistical knowledge to implement.
$X_{norm} = \frac{X - min(X)}{max(X) - min(X)}$	$X_{std} = \frac{X - mean(X)}{sd(X)}$.

11).

ans:

Correlation	Causation
It is a statistical relationship between two variables to measure change in one variable's associateness with change in other variable.	It is a statistical relationship between two variable to relate one variable directly causes a change in the other variable.
Correlation is simpler to measure and measured using correlation coefficient.	Causation ~~are~~ requires more than observing correlation and use randomized controlled experiments.
Correlation doesn't count for factors that affect both variables to measure the change in variable.	Causation keeps all the other factors constant that ~~causes~~ affect variables to measure the change.

12).

ans:

Over-fitting	Under-fitting.
Overfitting is an data processing problem in ML where model captures noises in the training data as output rather than the patterns.	Underfitting is is described when the dataset is few in number for model to find any relevant patterns for learning process.
The model performs well in training data but poorly in unseen data.	The model performs poorly in both training and unseen data.
This occurs when model is too flexible and has too many parameters relative to complexity of the data.	This occurs when model is not flexible enough and has few parameters relative to complexity of the data.
Signs of overfitting: - high training accuracy but low test accuracy. - high variance.	Signs of underfitting: - low training and test accuracy. - high bias.
It is addressed by: - Reducing model complexity - Increasing amount of training data with large feature range. - Adding regularization terms to the loss function.	It is addressed by: - Increasing model complexity - Increasing amount of training data - Reducing regularization terms to the loss function.

13).

ans: So,

for ML:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

where,

TP = Model correctly predicting positive class. (True positives).

FP = Model incorrectly predicts the positive class when it is negative class (False positives).

FN = Model incorrectly predicts the negative class when it is true class (False negatives).

TN = Model correctly predicts negative class. (True negatives).

14).

ans: The key parameters for Scikit-learn MLP can be used as:

```
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier ( hidden_layer_sizes = (13,13,13), activation = "relu",
                      solver = "adam", max_iter = 500, learning_rate_init = 0.001).
```

15).

ans: Soln:

Given,

$T_c = 405.5\ K$ $\quad\quad$ $\omega = 0.25$

$P_c = 111.3\ atm$ $\quad\quad$ $R = 0.082\ L \cdot atm / K \cdot mol.$

$a = 4.2527$ $\quad\quad$ $T = 450\ K$

$b = 0.02590$ $\quad\quad$ $P = 56\ atm.$

for RK EOS,

we know,

$$pv^3 - v^2(RT) + v(a - pb^2 - RTb) - ab = 0.$$

$$\text{or. } 56v^3 - 36.9v^2 + 3.2594v - 0.110 = 0.$$

now,

solving, we get:

~~$V = 0.5615\ L$~~

$V = 0.5792\ L/gmol.$

16).

Soln.

for RKS equation:

$$P = \frac{RT}{V-b} - \frac{a}{V(V+b)}$$

$$a = 0.42748 \left(\frac{R^2 \cdot T_c^2}{P_c} \right) \cdot \alpha$$

$$b = 0.08664 \left(\frac{R \cdot T_c}{P_c} \right)$$

$$\alpha = [1 + m (T - T_r^{0.5})]^{0.5}$$

$$m = 0.48 + 1.574\omega + 0.176\omega^2$$

$$T_r = \frac{T}{T_c}.$$

So $V = 0.5849$ L/gmol.

for PR equation:

$PV^3 + (bP - RT)V^2 + (aPb^2 - 2RTb)V + (Pb^3 + RTb^2 - ab) = 0.$

where,

$$a = \frac{0.45724 R^2 T_c^2}{P_c} \alpha.$$

$$b = 0.07780 \frac{RT_c}{P_c}.$$

$$\alpha = 1 + (0.37464 + 1.54426\omega - 0.26992\omega^2 (1 - T_R^{0.5}))^2.$$

So.

$V = 0.57441$ L/gmol.

FOR RK equation

In [5]:
```
import numpy as np
from scipy.optimize import fsolve
import matplotlib.pyplot as plt
```

In [2]:
```
Tc = 405.5          # K
pc = 113.5          # atm
T = 450             # K
R = 8.314e-2        # L * atm / K * mol

omega = 0.25
p = 56              # atm
```

In [3]:
```
# function that defines the RK equation and returns value of Volume
def RK(v):

    aRKw = 0.42748 * (R * Tc) ** 2 / pc
    aRK = aRKw * (Tc / T) ** 0.5
    bRK = 0.08664 * (R * Tc / pc)
    return p * v ** 3 - R * T * v ** 2 + (aRK - p * bRK ** 2 - R * T * bRK) * v - aRK * bRK
```

In [4]:
```
a = fsolve(RK, 0.7)
print(a)
```

```
[0.57920022]
```

In [ ]:
```
```

In [6]:
```
import numpy as np
from scipy.optimize import fsolve
import matplotlib.pyplot as plt
```

In [7]:
```
Tc = 405.5          # K
pc = 113.5          # atm
T = 450             # K
R = 8.314e-2        # L * atm / K * mol

omega = 0.25
p = 56              # atm
```

For RKS Equation

In [8]:
```
# function that defines the RKS equation and returns value of Volume
def RKS(v):

    mRKS = 0.480 + (1.574 - 0.176 * omega) * omega
    alphaRKS = (1 + mRKS * (1 - (T / Tc) ** 0.5)) ** 2
    aRKS = 0.42748 * alphaRKS * (R * Tc) ** 2 / pc
    bRKS = 0.08664 * (R * Tc / pc)
    return p * v ** 3 - R * T * v ** 2 + (aRKS - p * bRKS ** 2 - R * T * bRKS) * v - aRKS * bRKS
```

In [9]:
```
a = fsolve(RKS, 0.9)
print(a)
```

```
[0.58493115]
```

For PR Equation

In [10]:
```
# function that defines the PR equation and returns value of Volume
def PR(v):

    mPR = 0.37363 + (1.54226 - 0.26992 * omega) * omega
    alphaPR = (1 + mPR * (1 - (T / Tc) ** 0.5)) ** 2
    aPR = 0.45724 * alphaPR * (R * Tc) ** 2 / pc
    bPR = 0.07780 * (R * Tc / pc)
    return p * v ** 3 + (bPR * p - R * T) * v ** 2 + (aPR - 3 * p * bPR ** 2 - 2 * R * T * bPR) * v + (p * bPR ** 3 + R * T * bPR ** 2 - aPR * bPR)
```

In [11]:
```
a = fsolve(PR, 0.5)
print(a)
```

```
[0.57441405]
```

In [ ]:
```
```

The SRK EOS and PR EOS takes the intermolecular forces and the molecular sizes into account but the RK EOS doesn't. It takes into account the non-ideal nature of the fluid. Hence, it is necessary to implement SRK and PR EOS.

17).

ans: Soln:

Now,

$a = 4.4734$

$b = 0.02588.$

Now,

$$56V^3 - 36.9V^2 + 3.4809V - 0.11577 = 0.$$

Let,

$$f(V) = 56V^3 - 36.9V^2 + 3.4809V - 0.11577.$$

So,

$$f'(V) = 168V^2 - 73.8V + 3.4809.$$

Now,

n	x_0	$f(x_0)$	$f'(x_0)$	$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$
1	0.5	-0.6003	8.5809	0.57
2	0.57	0.2497	15.9934	0.5543
3	0.5543	0.0141	14.1966	0.5534
4	0.5534	0.0001	14.0848	0.5533.

$\therefore$ V (for RK method using Newton Raphson Method) = 0.5533 L/gmol.

In [9]:
```
import numpy as np
from scipy.optimize import fsolve
```

In [10]:
```
yin = 0.02
Tin = 600              # K
Q = 0.06555            # m3/s
Ctot = 0.0203          # kmol/m3
alpha = 26900          # m2/m3
V = 6e-4               # m3
M = 30                 # kg/kmol
Cpg = 1070             # J/kg*K
delHrxn = -2.84e8      # J/kmol
```

In [11]:
```
def model(x):

    y = x[0]
    T = x[1]

    k1 = 6.7e10 * np.exp(-12556 / T)
    K1 = 65.5 * np.exp(961 / T)
    r = (0.05 * k1 * y) / (T * (1 + K1 * y) ** 2)

    eq1 = Q * Ctot * (yin - y) - alpha * V * r
    eq2 = Q * Ctot * M * Cpg * (Tin - T) - alpha * V * r * (-delHrxn)

    return [eq1, eq2]
```

In [12]:
```
initial = np.array([0.6, 300])
sol = fsolve(model, initial)

print(f"The mol fraction of CO is = {sol[0]:.4f}\n")
print(f"The reactor temperature is = {sol[1]:.4f} K\n")
```

```
The mol fraction of CO is = 0.0154

The reactor temperature is = 559.5129 K
```

In [ ]:
```

```

In [13]:
```
import numpy as np
from scipy.integrate import odeint
```

In [14]:
```
yin = 0.02
Q = 0.06555           # m3/s
Ctot = 0.0203         # kmol/m3
alpha = 26900         # m2/m3
V = 6e-4              # m3
M = 30                # kg/kmol
Cpg = 1070            # J/kg*K
delHrxn = -2.84e8     # J/kmol
```

In [15]:
```
V = np.linspace(0, 1, 100)
Tin = [600, 550, 500]

for i in range(3):
    def model_f(x, V):

        y = x[0]
        T = x[1]

        k1 = 6.7e10 * np.exp(-12556 / T)
        K1 = 65.5 * np.exp(961 / T)
        r = (0.05 * k1 * y) / (T * (1 + K1 * y) ** 2)

        dydV = (- alpha * r) / (Q * Ctot)
        dTdV = (- alpha * r * (-delHrxn)) / (Q * Ctot * M * Cpg)

        return [dydV, dTdV]

    initial = [yin, Tin[i]]
    sol = odeint(model_f, initial, V)

    print(f"For inlet Temperature (T) = {Tin[i]} K,\n")
    print(f"The mol fraction of CO is = {sol[-1,0]:.2f}\n")
    print(f"The reactor temperature is = {sol[-1,1]:.2f} K\n\n\n\n")
```

```
For inlet Temperature (T) = 600 K,

The mol fraction of CO is = 0.00

The reactor temperature is = 423.05 K



For inlet Temperature (T) = 550 K,

The mol fraction of CO is = 0.00

The reactor temperature is = 405.65 K



For inlet Temperature (T) = 500 K,

The mol fraction of CO is = 0.01

The reactor temperature is = 412.48 K
```

In [ ]: