



IJEAST

INTERNATIONAL JOURNAL
OF ENGINEERING APPLIED SCIENCE
AND TECHNOLOGY



VOLUME : 7 ISSUE : 01 Print / Issue Publication Date: 07-Jul-2022



ISSN : 2455-2143



DOI : 10.33564/IJEAST.2022.v07i01.041

Indexed In



WWW.IJEAST.COM

editor@ijeast.com



EQUATIONS OF STATE IN CHEMICAL ENGINEERING USING PYTHON

Siddhi Shukla

Student,

Dept. Of Chemical Engineering IIST, Indore

Dr. Samatha Singh

Associate Professor

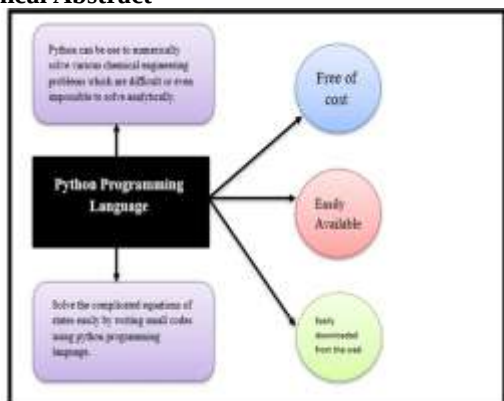
Dept. Of Chemical Engineering IIST Indore

Abstract- The python programming language is very useful for all kinds of scientific and engineering tasks by plotting data and analyzing various problems as well. We can use google Colab for python to numerically solve various chemical engineering problems which are difficult or even impossible to solve analytically. One of the best advantages of python programming language is that it is free of cost, and can be easily downloaded from the web.

Solving equations of state allows us to find the specific volume of a gaseous mixture of various chemicals at a specified temperature and pressure. Without using the equation of state, it would be difficult for us to design a chemical plant virtually. By knowing the specific volume, we can determine the size and cost of the chemical plant, including the diameter of pipes, distillation towers and chemical reactors, the horsepower of compressors and pumps. This paper illustrates how to solve the complicated equations of states easily by writing small codes using python programming language.

Keywords: Python programming language, Google Colab, Specific Volumes of gaseous mixture.

Graphical Abstract



I. INTRODUCTION

MATHEMATICAL STATEMENT OF EQUATIONS OF STATE

The ideal gas equation of state, which relates the Pressure, Specific Volume and Temperature is given by following formula:

$$pV = nRT \text{ or } p\hat{v} = RT \text{ where } \hat{v} = \frac{V}{n} \dots\dots\dots (1.1)$$

The term p is absolute pressure, V is volume, n is the number of moles, R is the universal gas constant, T is the absolute temperature, and $\hat{v}$ is the specific volume.

Also, the generalized formula of the ideal gas law was the van der waals equation of state which is given by the following formula:

$$p = \frac{RT}{v-b} - \frac{a}{v^2} \dots\dots\dots (1.2)$$

Here, “a” is the interaction force between two molecules and “b” is the excluded volume which means a second molecule cannot use the same space already used by the first molecule. Since the above van der waals equation is not a good approximation at high pressures therefore we define Redlich -Kwong equation of state which is the modification of van der waals equation of state given by the following formula:



$$p = \frac{RT}{v-b} - \frac{a}{v(v+b)} \quad (1.3)$$

Where,

$$a = 0.42748 \left(\frac{RT_c^2}{p_c} \right) \alpha, b = 0.08664 \left(\frac{RT_c}{p_c} \right), \text{ and } T_r = \frac{T}{T_c}, \alpha = \frac{1}{T_r^{0.5}} \quad (1.4)$$

Here, T_c is the critical temperature, p_c is the critical pressure, T_r is the reduced temperature which is the absolute temperature per critical temperature, and α is the particular to the Redlich-Kwong equation of state.[1]

Similarly, Redlich-Kwong-Soave equation of state is introduced which is the modified version of Redlich-Kwong equation of state which is given by the following equation:

$$p = \frac{RT}{v-b} - \frac{a}{v(v+b)} \quad (1.5)$$

Now the term a is given by a different formula,

$$a = [1 + m(1 - T_r^{0.5})]^2, m = 0.480 + 1.574\omega + 0.176\omega^2 \quad (1.6)$$

Here the term ω is the acentric factor which is tabulated quantity for different substances. Therefore, the value of ω can be performed for each chemical and reduced temperature.

Peng-Robinson equation, which is another variation of

equation of states and is given by the following equation:

$$p = \frac{RT}{\hat{v}-b} - \frac{a}{\hat{v}(\hat{v}+b)+b(\hat{v}-b)} \quad (1.7)$$

All of the above equations can be rearranged into a cubic function of specific volume and is given by the following equation:

$$\hat{v}^3(p) - \hat{v}^2(RT) + \hat{v}(a - pb^2 - RTb) - ab = 0 \quad (1.8)$$

II. SOLVING EQUATIONS OF STATE USING PYTHON PROGRAMMING LANGUAGE (SINGLE EQUATION IN ONE UNKNOWN)

Nonlinear algebraic equations like equation of state can be solved using Python, too. First, we have to define the problem statement to solve by defining a function; then, we check it; finally, we issue a command along with correct syntax to solve it.[2]

For example, we take the following equation and solve this with the help of python programming language

$$f(x) = x^2 - 2x - 8$$

There are various simple steps which we have to remember while solving the problem statement. Step 1: Firstly, Import the fsolve program from SciPy and define the function.

```

[1] #Equations of state using python programming language
from scipy.optimize import fsolve
def f(x):
    return x**2-2*x-8
    
```

Figure 1: Import the fsolve program.



NOTE: The indentation after the declaration `def f(x)` which is mandatory in python to define the blocks of statements.

Step 2: Now, Check the function and issue a command in

which we find the value of function at $x=1.1$ and we get the required result i.e., -8.99 that means the function $f(x)$ is correct.

```

Equation of state using python
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[5] #Equations of state using python programming language
    from scipy.optimize import fsolve
    def f(x):
        return x**2-2*x-8
    print (f(1.1))

-8.99
  
```

Figure 2: Check the function and issue a command.

NOTE: In the above code we used a value of “x” that means every term in the function was important.

Step 3: We use the `f solve` function in python to find the value of x that makes the function i.e., $f(x)=0$.

```

Equation of state using python
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[5] #Equations of state using python programming language
    from scipy.optimize import fsolve
    def f(x):
        return x**2-2*x-8
    print (f(1.1))

-8.99

[6] x=fsolve(f,0)
    print('The root is %5.3f'%x)

The root is -2.000.
  
```

Figure 3: Find the values of “x”.

Here, `%5.3f` will mean that the width of the floating-point number should be 5.3. This command solves the following problem for x : $f(x) = 0$ starting from the initial guess of 0 and we get the required result i.e., -2.000.

To summarize the steps, In Step 1 we defined the problem we wish to solve using python and evaluated the problem for some “x”, Step 2 we check the function and issue a command to find the value of the function and in Step 3 we instructed python to solve the problem. Hence the problem become easier by applying all the correct syntax and commands in python programming language.[3]

Examples of solving Chemical Engineering Problems with the help of Python Programming Language

Example 1: Find the specific volume of n-octane at 600°K and 20 atm using the Redlich-Kwong equation of state.

Step 1: Firstly, define the function that will calculate the $f(x)$, here specificvol (v), given the temperature, pressure, and thermodynamic properties.



```

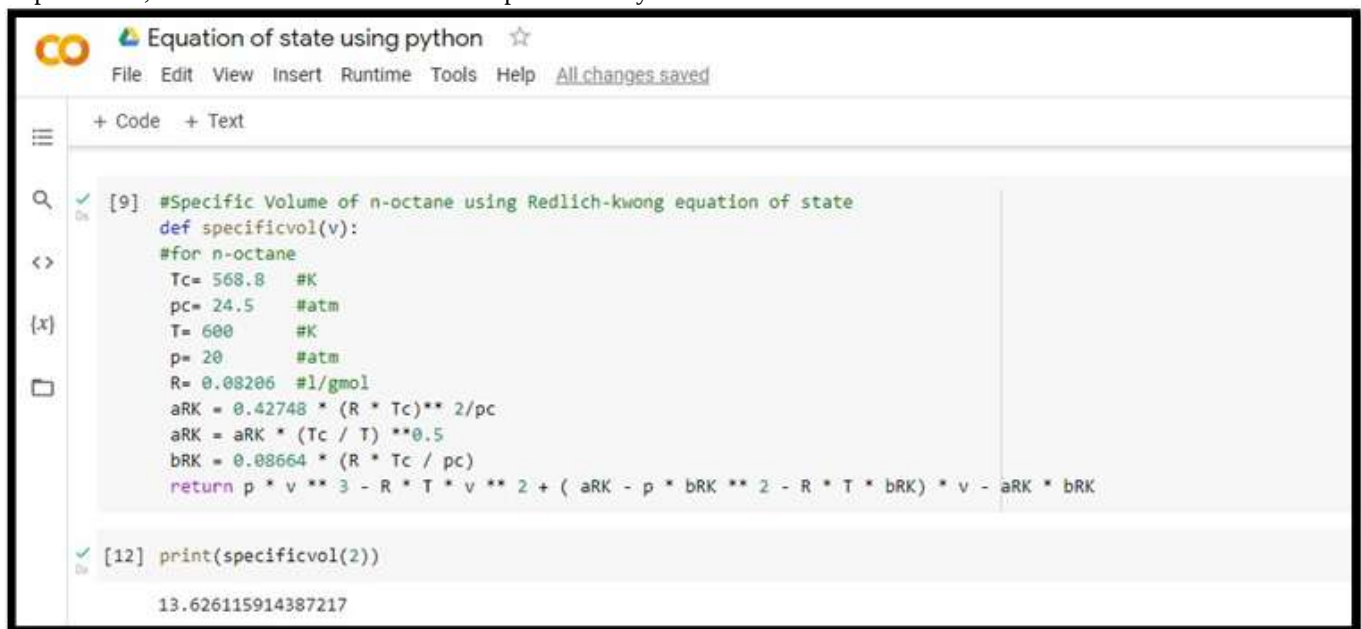
[7] #Specific Volume of n-octane using Redlich-kwong equation of state
def specificvol(v):
    #for n-octane
    Tc= 568.8 #K
    pc= 24.5 #atm
    T= 600 #K
    p= 20 #atm
    R= 0.08206 #l/gmol
    aRK = 0.42748 * (R * Tc)** 2/pc
    aRK = aRK * (Tc / T) **0.5
    bRK = 0.08664 * (R * Tc / pc)
    return p * v ** 3 - R * T * v ** 2 + ( aRK - p * bRK ** 2 - R * T * bRK) * v - aRK * bRK
    
```

Figure 4: Using Redlich-kwong equation of state.

Here, the function specificvol(v) defines the problem which we wish to solve.

issuing the print command and we get the required result.

Step 2: Now, we have to test the function specificvol by



```

[9] #Specific Volume of n-octane using Redlich-kwong equation of state
def specificvol(v):
    #for n-octane
    Tc= 568.8 #K
    pc= 24.5 #atm
    T= 600 #K
    p= 20 #atm
    R= 0.08206 #l/gmol
    aRK = 0.42748 * (R * Tc)** 2/pc
    aRK = aRK * (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

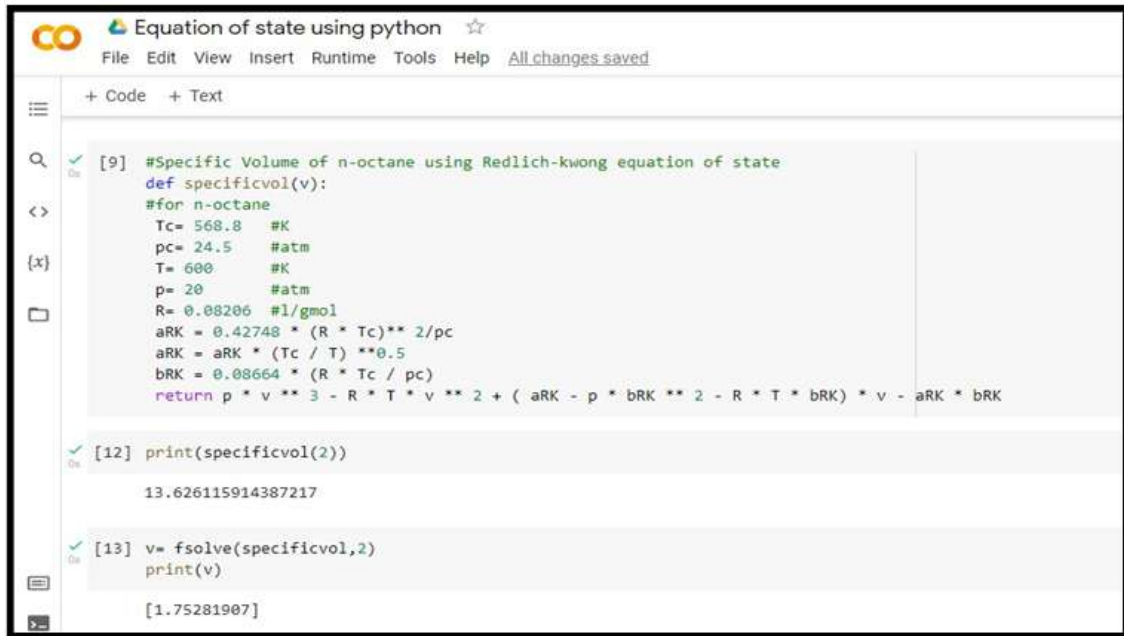
[12] print(specificvol(2))

13.626115914387217
    
```

Figure 5: Testing the function.

The specificvol function causes python to compute the value of the function specificvol when v=2 and we get 13.626115914387217 which is correct.

Step 3: Next, we have to issue the command which solve the function specificvol.



```
[9] #Specific Volume of n-octane using Redlich-kwong equation of state
def specificvol(v):
    #for n-octane
    Tc= 568.8 #K
    pc= 24.5 #atm
    T= 600 #K
    p= 20 #atm
    R= 0.08206 #l/gmol
    aRK = 0.42748 * (R * Tc)** 2/pc
    aRK = aRK * (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

[12] print(specificvol(2))

13.626115914387217

[13] v= fsolve(specificvol,2)
print(v)

[1.75281907]
```

Figure 6: Issue the command to solve the function.

Hence, we get the required result 1.75281907. In specificvol function the 2 is the initial guess of the answer.

Example 2: Rearrange the python code to compute the compressibility factor for a number of pressure values. The following equation defines the compressibility factor:

$$Z = \frac{pv}{RT}$$

When the gas is ideal, the pressure is low and the

compressibility factor will be close to 1.00. Compressibility factor will increase with increase in pressure. The following code solves for the Redlich-Kwong, Redlich-Kwong-Soave, and Peng-Robinson equations of state and thus plots the compressibility factor versus pressure. [4]

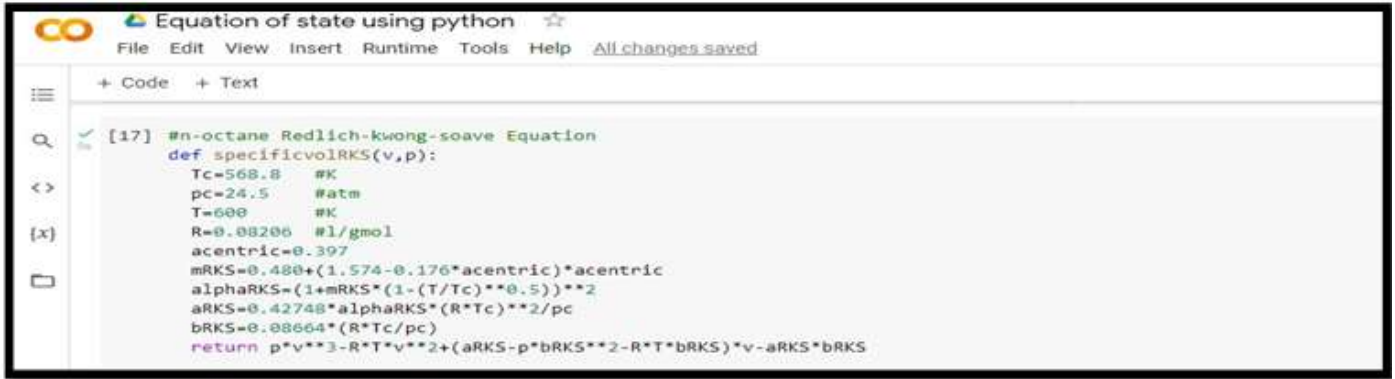


```
[16] #python code to compute the compressibility factor for a number of pressure values.
from scipy.optimize import fsolve
import numpy as np
import matplotlib.pyplot as plt

#n-octane Redlich kwong Equation
def specificvolRK(v,p):
    Tc=568.8 #K
    pc=24.5 #atm
    T=600 #K
    R=0.08206 #l/gmol
    aRK=0.42748*(R*Tc)**2/pc
    aRK=aRK*(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
```

Figure 7: Compute the compressibility factor.

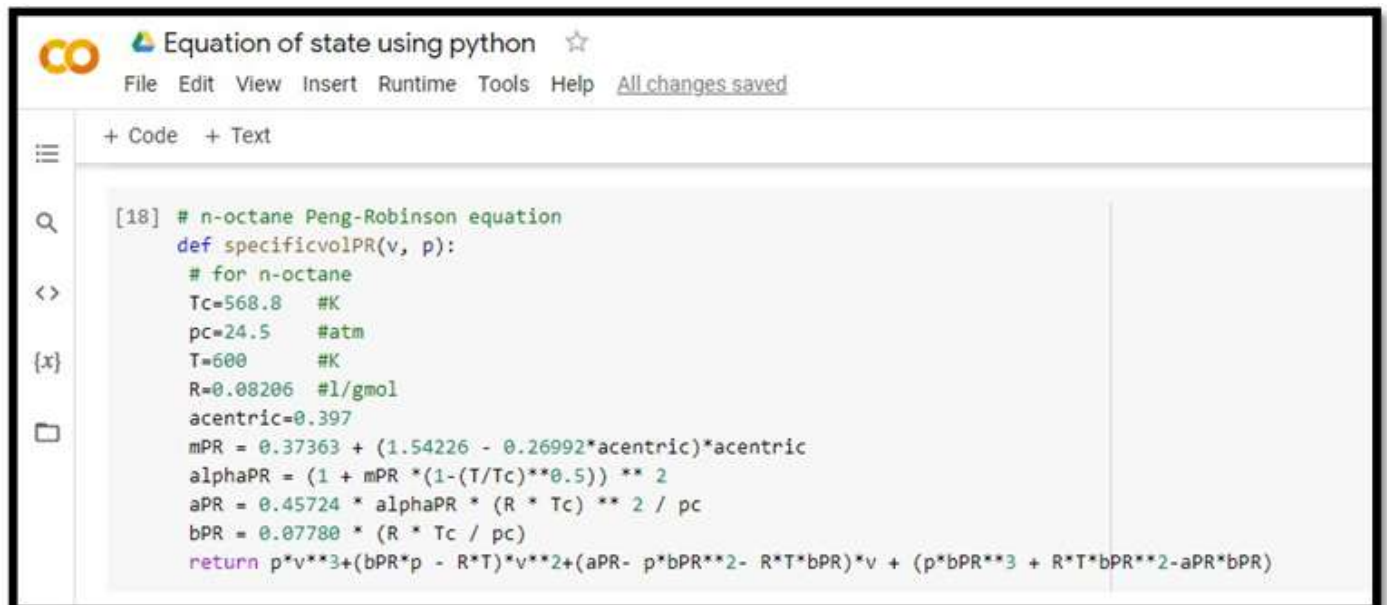
The above python code is for Redlich-Kwong equation of state.



```
[17] #n-octane Redlich-kwong-soave Equation
def specificvolRKS(v,p):
    Tc=568.8 #K
    pc=24.5 #atm
    T=600 #K
    R=0.08206 #l/gmol
    acentric=0.397
    mRKS=0.480+(1.574-0.176*acentric)*acentric
    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
```

Figure 8: Python code for Redlich-Kwong -Soave equation of state.

The above python code is for Redlich-kwong-soave equation of state.



```
[18] # n-octane Peng-Robinson equation
def specificvolPR(v, p):
    # for n-octane
    Tc=568.8 #K
    pc=24.5 #atm
    T=600 #K
    R=0.08206 #l/gmol
    acentric=0.397
    mPR = 0.37363 + (1.54226 - 0.26992*acentric)*acentric
    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- p*bPR**2- R*T*bPR)*v + (p*bPR**3 + R*T*bPR**2-aPR*bPR)
```

Figure 9: Python code for Peng-Robinson equation of state.

The above python code is for Peng-Robinson equation of state.



```
[22] T = 600
R = 0.08206
pressure = np.arange(1, 27, 5)
print(pressure)
print(pressure[0])
print(pressure[5])
zcompRK = np.zeros(6,dtype=float)
zcompRKS = np.zeros(6,dtype=float)
zcompPR = np.zeros(6,dtype=float)
print(zcompRK)

[ 1  6 11 16 21 26]
1
26
[0. 0. 0. 0. 0. 0.]
```

Figure 10: Python code for Compressibility Factor equation of state.

The above python code is for Compressibility Factor equation of state.



```

[24] for i in range (0, 6, 1):
    p = pressure[i]
    guess = R*T/p
    v = fsolve(specificvolRK, guess, p)
    z = p * v / (R * T)
    zcompRK[i] = z
    v = fsolve(specificvolRKS,v,p)
    z = p * v / (R * T)
    zcompRKS[i] = z
    v = fsolve(specificvolPR,v,p)
    z = p * v / (R * T)
    zcompPR[i] = z
    
```

Figure 11: Python code for Compressibility Factor equation of state.



```

print(zcompRK)
print(zcompRKS)
print(zcompPR)
plt . plot (pressure,zcompRK,'o-g',label='Redlich-Kwong')
plt . plot (pressure,zcompRKS,'x-b',label='Redlich-Kwong-Soave')
plt . plot (pressure,zcompPR,'s-r',label='Peng-Robinson')
plt . legend(loc='best')
plt . xlabel('Pressure (atm)')
plt . ylabel('Z')
plt . title ('n-Octane')
plt . show()

[0.98800398 0.92532925 0.85709802 0.78108721 0.69313965 0.58397867]
[0.98850164 0.9286727 0.86413637 0.79330263 0.71354567 0.62832486]
[0.98391114 0.8976994 0.79742609 0.66797414 0.88760226 0.10613573]
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: MatplotlibDeprecationWarning: Unrecognized location 'best'. Falling back on 'best'; valid locations:
  best
  upper right
  upper left
  lower left
  lower right
  right
  center left
  center right
  lower center
  upper center
  center
This will raise an exception in 3.3.
import sys
    
```

Figure 12: Python code for Compressibility Factor equation of state.

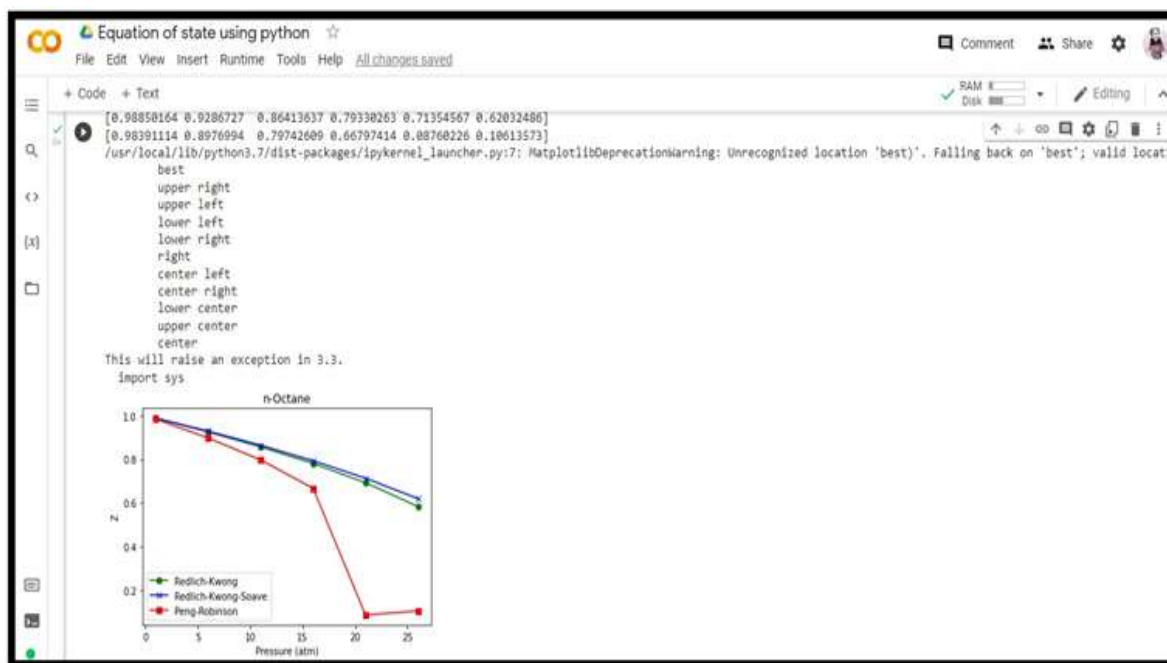


Figure 13: Python code for Compressibility Factor equation of state.

The first three commands bring in the needed routines – fsolve, numpy, and pylab (for plotting the graph). Then there are three definitions of functions that define the equations which are governing the specific volume, Eq. (1.5) and (1.6). The main program sets the temperature, provides a vector of 6 pressures, equidistant from 1 to 27; pressure = [1, 6, 11, 16, 21, 26]. The index goes from 0 to 5. The vectors of compressibility's are also set with 6 values for every equation of state, starting with 0. Then a loop calculation is made for "i" from 0 to 5.[5]
 For each pressure in turn, the guess for the Redlich-Kwong equation of state is the result from the ideal gas law. The result from the Redlich-Kwong equation of state is used for the guess when solving the Redlich-Kwong-Soave equation of state, and that solution is used as the guess for the Peng-Robinson equation of state. For each "i", after the compressibility is found it is put in a vector for that equation of state. Finally, the results are plotted, with three curves on one plot.[6]

III. CONCLUSION

This paper summarizes the intention to solve equations of state using the Google Colab platform using Python computational language. The Google Colab platform was chosen because it is free of cost, does not require the installation, setup, and configuration of Python packages and their libraries in the students' personal computers. Google Colab is a multiuser and collaborative environment, ideal for remote classes. The notebooks in the Google Colab can be shared between instructor and students or between the students, which eases the communication and track of

student's progress.

This paper illustrates how to solve the complicated problems and equations of states easily by writing small codes using Python programming language.

IV. REFERENCES

- [1] P. T. Cummings et al., "Open-source molecular modeling software in chemical engineering focusing on the Molecular Simulation Design Framework," *AIChE Journal*, vol. 67, no. 3, Mar. 2021, doi: 10.1002/aic.17206.
- [2] S. Olofsson, L. Hebing, S. Niedenführ, M. P. Deisenroth, and R. Misener, "GPdoemd: A Python package for design of experiments for model discrimination," *Computers and Chemical Engineering*, vol. 125, pp. 54–70, Jun. 2019, doi: 10.1016/j.compchemeng.2019.03.010.
- [3] J. Gostick et al., "PoreSpy: A Python Toolkit for Quantitative Analysis of Porous Media Images," *Journal of Open-Source Software*, vol. 4, no. 37, p. 1296, May 2019, doi: 10.21105/joss.01296.
- [4] D. A. C. Beck, J. M. Carothers, V. R. Subramanian, and J. Pfandner, "Data science: Accelerating innovation and discovery in chemical engineering," *AIChE Journal*, vol. 62, no. 5, pp. 1402–1416, May 2016, doi: 10.1002/aic.15192.
- [5] V. Yadav, S. Karmakar, P. P. Kalbar, and A. K. Dikshit, "PyTOPS: A Python based tool for TOPSIS," *SoftwareX*, vol. 9, pp. 217–222, Jan. 2019, doi: 10.1016/j.softx.2019.02.004.
- [6] R. H. Landau and M. J. Pérez, "Introduction to Python



for Science and Engineering Series in Computational
Physics Introduction to Numerical Programming: A
Practical Guide for Scientists and Engi-neers Using
Python and C/C++ Titus A. Beu Computational
Problems for Physics: With Guided Solutions Using
Python.” [Online]. Available: [https://
www.crcpress.com/](https://www.crcpress.com/)

IJEAST

INTERNATIONAL JOURNAL
OF ENGINEERING APPLIED SCIENCE
AND TECHNOLOGY

ABOUT IJEAST

International Journal of Engineering Applied Science and Technology (IJEAST) is a peer-reviewed, open access journal that publishes high-quality research papers in the field of Engineering, Applied Science and Technology.

IJEAST aims to provide a platform for researchers, academicians, and professionals to share their innovative ideas, research findings, and practical experiences with the global scientific community.

FOCUS AREAS

- Engineering
- Applied Science
- Technology
- Innovation & Development
- Interdisciplinary Studies



PEER REVIEWED

All submissions are rigorously peer reviewed to ensure quality.



OPEN ACCESS

Free and unrestricted access to research for all.



GLOBAL REACH

Connecting researchers and professionals worldwide.



TIMELY PUBLICATION

We ensure a swift and efficient publication process.



For more information, visit our website
www.ijeast.com



INTERNATIONAL JOURNAL
OF ENGINEERING APPLIED SCIENCE
AND TECHNOLOGY

✉ editor@ijeast.com

🌐 www.ijeast.com

📍 India



2455-2143