Exercise 7

Problem 1: Compute molar volumes from cubic EOS

a) Use the van der Waals, the Soave-Redlich-Kwong and the Peng-Robinson equations of state (EOS) to determine the molar volumes of the *saturated phases* for Ethylene at 260 K (the saturation pressure P^{sat} is 3.035 MPa).

For this task, recall that cubic EOS can be written on the form $Z^3 + \alpha Z^2 + \beta Z + \gamma = 0$, where Z is the compressibility factor, and α, β and γ are generalized parameters.

b) Compare the model-predicted molar volumes with the tabulated values $v_L = 0.0713$ L/mol and $v_V = 0.456$ L/mol. Here, subscripts L and V denote liquid phase and vapor phase, respectively. Give the errors in %.

Solved in python, the code is attached on the last pages
The critical values were found in the table provided in the lectures:

Compounds	T_c [K]	P_c [MPa]	ω [-]
Methane	190.6	4.604	0.011
Ethylene	282.4	5.032	0.085
Methanol	512.6	8.096	0.566
Oxygen	154.6	5.043	0.022
Argon	150.9	4.898	-0.004
Ethanol	516.4	6.384	0.637

Equation used	Calculated vL [cm3/mol]	Calculated vV [cm3/mol]	Error vL [%]	Error vV [%]
van der Waals	117.91	502.86	65.37	10.28
Peng-Robinson	72.47	437.96	1.64	3.96
Soave-Redlich-Kwang	81.88	454.21	14.83	0.39

Problem 2: Cubic EOS and fugacity

The following values are tabulated for CO_2: $T_c = 304.2$ K, $P_c = 7.382$ MPa, and $\omega = 0.228$. P_c is the critical pressure, T_c is the critical temperature, and ω is the acentric factor.

Peng-Robinson EOS presented in terms of compressibility factor Z:

$$Z^3 + (B-1)Z^2 + (A - 3B^2 - 2B)Z - AB + B^3 + B^2 = 0 \quad (1)$$

where

$$Z = \frac{Pv}{RT} \quad (2)$$

$$A = \frac{aP}{(RT)^2}, \quad B = \frac{bP}{RT}$$

$$a = a_c\alpha, \quad b = 0.07779607\frac{RT_c}{P_c}, \quad a_c = 0.45723553\frac{(RT_c)^2}{P_c},$$

$$\alpha = [1 + \kappa(1 - \sqrt{T_r})]^2, \quad \kappa = 0.37464 + 1.54226\omega - 0.26992\omega^2$$

Fugacity from Peng-Robinson EOS:

$$f = P\exp\left[Z - 1 - \ln(Z - B) - \left(\frac{A}{2\sqrt{2}B}\right)\ln\left(\frac{Z + (1+\sqrt{2})\,B}{Z + (1-\sqrt{2})\,B}\right)\right] \quad (3)$$

a) At $T = 216.1$ K and $P = 1.5$ MPa, use the Peng-Robinson EOS to compute

(i) The roots of (1)

(ii) Molar volumes, v (cm^3/mol), from (2)

(iii) Fugacities, f (MPa), from (3)

Also, respond the following

- Based on the calculations, determine whether there is one or two phases. Explain!

b) What is the saturation pressure for isotherm $T = 216.1$ K?

Both a) and b) was solved in python, see attached code

a)

```
------ Problem 2 a) ------
For P = 1.5 MPa:

Z1 = 0.7405512379926831
Z2 = 0.207501038849022
Z3 = 0.029695197039820768

-----The molar volumes-----
v_liquid = 35.56802407710533 cm3/mol
v_gas = 887.0102538108262 cm3/mol

-----The fugacities-----
f_liquid = 0.4733555314657436 MPa
f_gas = 1.1949095088950967 MPa
```

- The fugacities are not equal => We do not have p^{sat} => One stable phase

  As $f_{liquid} < f_{gas}$, $p < p^{sat}$ => <u>The stable phase is the liquid phase</u>

b) At p^{sat}, $f_{liquid} = f_{gas}$, using iteration:

```
------ Problem 2 b) ------
Psat = 0.4978925885157031 MPa
f_liq = 0.46404688259615434 MPa
f_vap = 0.46417252539575526 MPa
```

<u>$p^{sat} \approx 0{,}498$ MPa</u>

Problem 1

```python
import numpy as np

# ------ Problem 1 a) and b) ------

#   Defining the given data and the gas constant
Pc = 5.032  # [MPa]
Tc = 282.4  # [K]
omega = 0.085  # [-] Accentric factor
R = 8.314  # [J/mol*K] or [MPa*cm3/mol*K]
Psat = 3.035# [MPa]
T = 260     # [K]
Tr = T/Tc
vL = 0.0713 * 10**3     # [cm3]
vG = 0.456  * 10**3     # [cm3]

#   Defining the constants in the van der Waals EOS
a_vW = 27 / 64 * (R * Tc)**2/Pc
b_vW = R * Tc / (8 * Pc)
A_vW = a_vW*Psat/(R*T)**2
B_vW = b_vW*Psat/(R*T)


#   Defining the constants in the Peng-Robinson EOS
kappa_pr = 0.377464 + 1.54226 * omega - 0.26992*omega**2
b_pr = 0.07780 * R * Tc / Pc
alpha_pr = (1 + kappa_pr*(1 - np.sqrt(Tr))) ** 2
ac_pr = 0.45724 * (R * Tc)**2 / Pc
a_pr = ac_pr * alpha_pr
A_pr = a_pr*Psat/(R*T)**2
B_pr = b_pr*Psat/(R*T)

#   Defining the constants in the Soave-Redlich-Kwong EOS
kappa_srk = 0.480 + 1.574 * omega - 0.176 * omega ** 2
b_srk = 0.08664 * R * Tc / Pc
alpha_srk = (1 + kappa_srk * (1 - np.sqrt(Tr))) ** 2
ac_srk = 0.42748 * (R * Tc) ** 2 / Pc
a_srk = ac_srk * alpha_srk
A_srk = a_srk*Psat/(R*T)**2
B_srk = b_srk*Psat/(R*T)


# Defining the parameters for the equations
alfa_vW = - (B_vW + 1)
beta_vW = A_vW
gamma_vW = - A_vW*B_vW

alfa_pr = B_pr - 1
beta_pr = A_pr - 3*B_pr**2 - 2*B_pr
gamma_pr = - A_pr*B_pr + B_pr**3 + B_pr**2

alfa_srk = -1
beta_srk = A_srk - B_srk - B_srk**2
gamma_srk = - A_srk*B_srk

# Solving the equations

# van der Waals
p_vW = [1, alfa_vW, beta_vW, gamma_vW]
r_vW = np.roots(p_vW)
```

```
index_vW = np.imag(r_vW) == 0
Z_list_vW = np.real(r_vW[index_vW])
v_liquid_vW = min(Z_list_vW)*R*T/Psat
v_gas_vW = max(Z_list_vW)*R*T/Psat
err_vL_vW = abs((v_liquid_vW - vL)/vL * 100)
err_vG_vW = abs((v_gas_vW - vG)/vG * 100)

# Peng Robinson
p_pr = [1, alfa_pr, beta_pr, gamma_pr]
r_pr = np.roots(p_pr)
index_pr = np.imag(r_pr) == 0
Z_list_pr = np.real(r_pr[index_pr])
v_liquid_pr = min(Z_list_pr)*R*T/Psat
v_gas_pr = max(Z_list_pr)*R*T/Psat
err_vL_pr = abs((v_liquid_pr - vL)/vL * 100)
err_vG_pr = abs((v_gas_pr - vG)/vG * 100)

# Soave-Redlich-Kwang
p_srk = [1, alfa_srk, beta_srk, gamma_srk]
r_srk = np.roots(p_srk)
index_srk = np.imag(r_srk) == 0
Z_list_srk = np.real(r_srk[index_srk])
v_liquid_srk = min(Z_list_srk)*R*T/Psat
v_gas_srk = max(Z_list_srk)*R*T/Psat
err_vL_srk = abs((v_liquid_srk - vL)/vL * 100)
err_vG_srk = abs((v_gas_srk - vG)/vG * 100)

# Printing the results
print('{:<20} {:>25} {:>25} {:>15} {:>15}'.format("Equation used",
            "Calculated vL [cm3/mol]", "Calculated vV [cm3/mol]", "Error vL [%]", "Error vG [%]"))
print(104*'-')
print('{:<20} {:>25} {:>25} {:>15} {:>15}'
      .format("van der Waals", f"{v_liquid_vW:.2f}",
              f"{v_gas_vW:.2f}", f"{err_vL_vW:.2f}", f"{err_vG_vW:.2f}"))
print('{:<20} {:>25} {:>25} {:>15} {:>15}'.
      format("Peng-Robinson", f"{v_liquid_pr:.2f}",
             f"{v_gas_pr:.2f}", f"{err_vL_pr:.2f}", f"{err_vG_pr:.2f}"))
print('{:<20} {:>25} {:>25} {:>15} {:>15}'
      .format("Soave-Redlich-Kwang", f"{v_liquid_srk:.2f}",
              f"{v_gas_srk:.2f}", f"{err_vL_srk:.2f}", f"{err_vG_srk:.2f}"))
```

Problem 2

```
import numpy as np

# ------ Problem 2 a) ------

# Defining given data
Pc = 7.382  # [MPa]
Tc = 304.2  # [K]
omega = 0.228  # [-] Accentric factor
R = 8.314  # [J/mol*K] or [MPa*cm3/mol*K]

# The conditions in the assignment (except pressure because of 2b)
T = 216.1   # [K]


# Defining the constants in the Peng-Robinson expression
Tr = T/Tc
kappa = 0.377464 + 1.54226 * omega - 0.26992*omega**2
b = 0.07779607 * R * Tc / Pc
alpha = (1 + kappa*(1 - np.sqrt(Tr))) ** 2
ac = 0.45723553 * (R * Tc)**2 / Pc
a = ac * alpha

def find_Z_roots(P):
    A = a*P/(R*T)**2
    B = b*P/(R*T)

    # Using the method from the appendix of the last exercise,
    # the polynomial is rewritten to Z^3 + alfa*Z^2 + beta*Z + gamma
    alfa = B - 1
    beta = A - 3*B**2 - 2*B
    gamma = - A*B + B**3 + B**2

    # Solving
    p = [1, alfa, beta, gamma]
    r = np.roots(p)
    index = np.imag(r) == 0
    Z_list = np.real(r[index])
    return Z_list

def fugacity(Z, P):
    A = a * P / (R * T) ** 2
    B = b * P / (R * T)
    ln_expr = (Z + (1 + np.sqrt(2))*B) / (Z + (1 - np.sqrt(2))*B)
    exp_expr = Z - 1 - np.log(Z - B) - (A/(2*np.sqrt(2)*B))*np.log(ln_expr)
    f = P * np.exp(exp_expr)
    return f

def print_answers(Z_list, P):
    print(f"For P = {P} MPa:\n")
    for i in range(len(Z_list)):
        print(f"Z{i+1} = {Z_list[i]}")

    print("\n-----The molar volumes-----")
    v_liquid = min(Z_list)*R*T/P
    v_gas = max(Z_list)*R*T/P
    print(f"v_liquid = {v_liquid} cm3/mol")
    print(f"v_gas = {v_gas} cm3/mol")

    f_l = fugacity(min(Z_list), P)
```

```
    f_v = fugacity(max(Z_list), P)
    print("\n-----The fugacities-----")
    print(f"f_liquid = {f_l} MPa")
    print(f"f_gas = {f_v} MPa")

Z_roots = find_Z_roots(1.5)
print_answers(Z_roots, 1.5)

# ------ Problem 2 b) ------
Psat = 0
f_vap = 0
f_liq = 0

pressures = np.linspace(0.0001, 7, 5000)
for pressure in pressures:
    Z_values = find_Z_roots(pressure)
    f_l = fugacity(min(Z_values), pressure)
    f_v = fugacity(max(Z_values), pressure)
    if (f_l - f_v) < 0:
        Psat = pressure - ((7 - 0.0001)/5000)/2
        Z_values = find_Z_roots(Psat)
        f_l = fugacity(min(Z_values), Psat)
        f_v = fugacity(max(Z_values), Psat)
        f_vap = f_v
        f_liq = f_l
        break

print("\n------ Problem 2 b) ------")
print(f"Psat = {Psat}")
print(f"f_liq = {f_liq}")
print(f"f_vap = {f_vap}")
```