Exercise 6

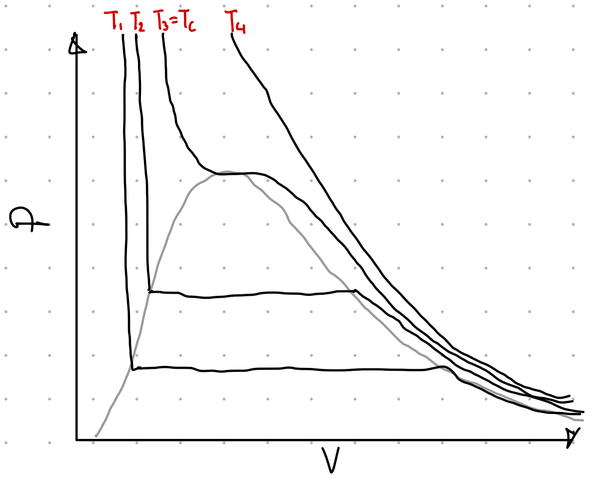
1 | Pv-diagram

a) Sketch a Pv-diagram with isotherms $T_1 < T_c$, $T_2 < T_c$, $T_3 = T_c$, and $T_4 > T_c$. Here T_c is the critical temperature, P is the pressure and v is the molar volume.

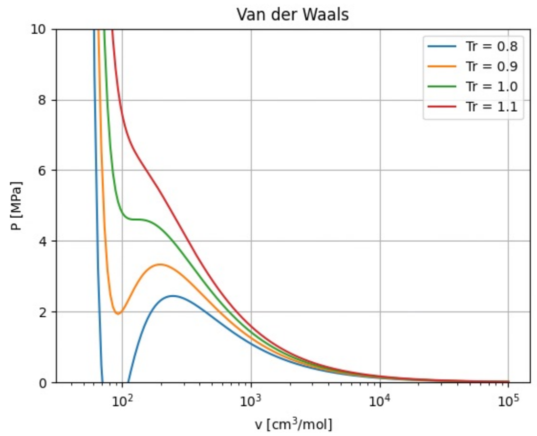
b) Use Python to generate a Pv-diagram with a few isotherms. Use the van der Waals, and Soave-Redlich-Kwong or (and) Peng-Robinson equations of state. Study how the isotherms behave for (i) high and low pressures, (ii) large and small molar volumes, and (iii) above and below the critical temperature. Discuss the mathematical properties of the EOS in context with the phase diagram you made in the task above.

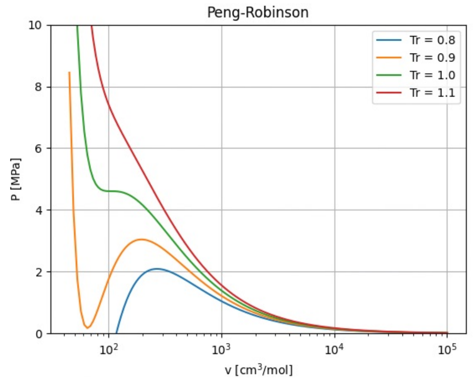
For methane the critical pressure is P_c = 4.604 [MPa], the critical temperature is T_c = 190.6 [K] and the acentric factor is ω = 0.011 [-].

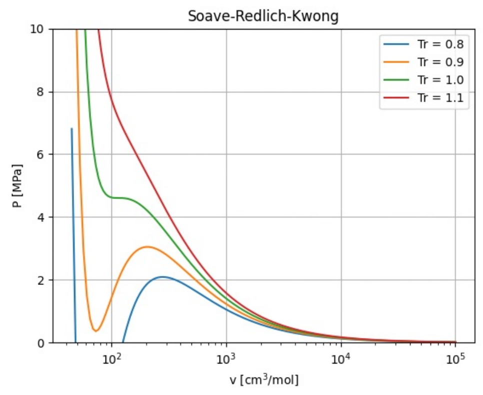
a) Sketching from picture given in lectures:



b) The python code used is attached on the next pages







i) For high pressures, the isotherms increase along an asymptote.
For low pressures V goes towards infinity. The isotherms "merges" at high and low pressures
In the "mid-range", there are more than one solution to the EOS.

ii) Small molar volumes, high pressures asymptotic limit where $p \to \infty$
Large molar volumes, pressure tends to zero

iii) Above the critical temperatures, the isotherms behave "linearily" and are 1-to-1
Below critical temperatures, the isotherm will have a 3-root region, where there is a "wave" causing there to be 3 roots for each pressure.

```python
import numpy as np
import matplotlib.pyplot as plt

#   Defining the given data and the gas constant
Pc = 4.604  # [MPa]
Tc = 190.6  # [K]
omega = 0.011  # [-] Accentric factor
R = 8.314  # [J/mol*K] or [MPa*cm3/mol*K]

# Wish to have 4 isotherms, choosing similar ones to the ones in
Tr = np.array([0.8, 0.9, 1, 1.1])  # [-]
T = Tr * Tc  # [K] The temperatures used in the EOS

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

#   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

#   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


# Defining the EOS as functions of molar volume and temperature
def P_vW(V, T):
    P = (R * T / (V - b_vW)) - a_vW / V ** 2

    return P


def P_pr(V, T, Tr_1):
    alpha_pr = (1 + kappa_pr*(1 - np.sqrt(Tr_1))) ** 2
    ac_pr = 0.45724 * (R * Tc)**2 / Pc
    a_pr = ac_pr * alpha_pr

    P = (R * T / (V - b_pr)) - a_pr / (V * (V + b_pr) + b_pr * (V - b_pr))

    return P


def P_srk(V, T, Tr_1):
    alpha_srk = (1 + kappa_srk * (1 - np.sqrt(Tr_1))) ** 2
    ac_srk = 0.42748 * (R * Tc) ** 2 / Pc
    a_srk = ac_srk * alpha_srk

    P = (R * T / (V - b_srk)) - a_srk / (V * (V + b_srk))

    return P


# Creating the plots:
V_list = np.linspace(45, 1e5, 25000)
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
for i in range(len(Tr)):
    isothermT = T[i]
    Tr1 = Tr[i]
    P_list = P_vW(V_list, isothermT)
    #P_list = P_pr(V_list, isothermT, Tr1)
    #P_list = P_srk(V_list, isothermT, Tr1)
    ax.plot(V_list, P_list, label=f"Tr = {Tr[i]}")

ax.set_xscale('log')
plt.ylim(0, 10)
plt.grid(True)
plt.xlabel(r"v [cm$^3$/mol]")
plt.ylabel("P [MPa]")
plt.title("Van der Waals")
#plt.title("Peng-Robinson")
#plt.title("Soave-Redlich-Kwong")
plt.legend()
plt.show()
```

2 | Roots of cubic EOS

The Peng-Robinson equation of state (EOS) can be written as a cubic equation in terms of the compressibility factor, Z, as

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

where

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

Here, v denotes the molar volume, P the pressure, R the gas constant, and T the temperature. Furthermore, a and b are parameters of the EOS.

a) Consider CO_2 at $T = 216.104\,\mathrm{K}$ and $P = 1.0\,\mathrm{MPa}$. At this condition, $A = 0.1517$ and $B = 0.0148$. Dertermine the roots of (1) using the analytical procedure given in appendix A. Compare the results with NumPy's function *roots*.

b) Consider CO_2 at $T = 216.104\,\mathrm{K}$ and $P = 3.0\,\mathrm{MPa}$. At this condition, $A = 0.4552$ and $B = 0.0445$. Dertermine the roots of (1) using the analytical procedure given in appendix A. Compare the results with NumPy's function *roots*.

c) If the program returns imaginary numbers in the above problems, use *imag* and *real* from the NumPy library to eliminate the imaginary numbers and only provide the real roots as output (see appendix B).

d) The saturation pressure of CO_2 is 0.5 MPa at $T = 216.104$ K. Which phase(s) is/are stable and what is/are the molar volume(s) of the stable phase(s) for CO_2 in task a) and b)?

a) Using the analytical approach, implemented in python

$Q = 0{,}06737$

$R = -0{,}01649$

Get that $R^2 < Q^3 \Rightarrow$ Three roots

$\Theta = 2{,}802$

$\boxed{Z_1 = 0{,}0197 \quad Z_2 = 0{,}8442 \quad Z_3 = 0{,}1213}$

Using a Numerical approach:

$\boxed{Z_1 = 0{,}8442 \quad Z_2 = 0{,}1213 \quad Z_3 = 0{,}0197}$

The answers are almost exactly equal (down to e^{-16})

b) + c)

This problem gave imaginary numbers in the numerical solution, so the np.imag and np.real was used.

Analytical solution:

$Q = -0{,}0186$

$R = 0{,}0160$

$U = -0{,}3179$

$V = 0{,}0586$

$\boxed{Z_1 = 0{,}0592}$

Numerical solution

$\boxed{Z_1 = 0{,}0592}$

The solutions differ by $3e^{-17}$

d) As we have $P > P^{sat}$ in both a) and b), the stable phase is the liquid phase => Use the smallest v (and Z)

We have that $Z = \frac{Pv}{RT} \Rightarrow v = \frac{ZRT}{P}$

Solving for the smallest solution (root) of each problem:

For a) $\underline{\underline{v = 35{,}516 \ cm^3/mol}}$

For b) $\underline{\underline{v = 35{,}459 \ cm^3/mol}}$

The code used in the problem:

```python
import numpy as np

# Defining the parameters
A = 0.1516
#A = 0.4552
B = 0.0148
#B = 0.0445
T = 216.104     # [K]
P = 1           # [MPa]
#P = 3           # [MPa]
R_gas = 8.314   # [J/mol*K] or [MPa*cm3/mol*K]

# Calculating the parameters in the analytical approach:
alpha = B - 1
beta = A - 3 * B ** 2 - 2 * B
gamma = -A * B + B ** 3 + B ** 2

print("-----The analytical method-----")
Q = (alpha ** 2 - 3 * beta) / 9
R = (2 * alpha ** 3 - 9 * alpha * beta + 27 * gamma) / 54

print(f"Q = {Q}")
print(f"R = {R}")

if R ** 2 < Q ** 3:
    theta = np.arccos(R / np.sqrt(Q ** 3))
    print(f"theta = {theta}")
    Z1 = -2 * np.sqrt(Q) * np.cos(theta / 3) - alpha / 3
    Z2 = -2 * np.sqrt(Q) * np.cos((theta + 2 * np.pi) / 3) - alpha / 3
    Z3 = -2 * np.sqrt(Q) * np.cos((theta - 2 * np.pi) / 3) - alpha / 3
    print(f"Z1 = {Z1}")
    print(f"Z2 = {Z2}")
    print(f"Z3 = {Z3}")
else:
    U = -np.sign(R) * (np.abs(R) + np.sqrt(R ** 2 - Q ** 3)) ** (1/3)
    if Q == 0:
        V = 0
    else:
        V = Q / U
    print(f"U = {U}")
    print(f"V = {V}")
    x1 = (U+V) - alpha/3
    print(f"x1 = {x1}")

print("\n-----The numerical method-----")

p = [1, alpha, beta, gamma]
r = np.roots(p)
print(r)
index = np.imag(r) == 0
r_real = np.real(r[index])
for i in range(len(r_real)):
    print(f"Z{i+1} = {r_real[i]}")

print("-----The molar volume-----")
v = min(r_real)*R_gas*T/P
print(f"v = {v} cm3/mol")
```