# PengRobinson_Me_throttle_example

December 1, 2025

## 1 Departure functions from Peng-Robinson equation of state

These are codes to calculate the enthalpy departure function from the complete, generalized Peng-Robinson equation of state as given in SIS 5th edition equations 6.4-2 and 6.7-1 to 6.7-4.

We use it to calculate analyze an isenthalpic expansion of methane through a throttling process.

Eric Furst
November 2024

Version updates:
Revisions in layout, calculation of PR EOS (11/2025)

---

As ususal, we import the *numpy* and *matplotlib* libraries. We'll also use the *SciPy* constants library.

[2]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy import constants
import scipy.optimize as opt

R = constants.R # Set the gas constant to R
```

### 1.1 Problem statement

Methane gas undergoes a rapid, adiabatic expansion through a continous throttling process from upstream conditions 13˚C and 184 bar to a lower pressure at -43˚C .

Calculate the downstream gas pressure.

To solve this problem, we will calculate the enthalpy of methane such that it satisfies an isenthalpic condition. In terms of the departure function values, this is

$$\Delta \underline{H} = \Delta \underline{H}^{\mathrm{IG}} + \Delta[\underline{H} - \underline{H}^{\mathrm{IG}}] = 0$$

or

$$\Delta \underline{H}^{\mathrm{IG}} + [\underline{H} - \underline{H}^{\mathrm{IG}}]_2 - [\underline{H} - \underline{H}^{\mathrm{IG}}]_1 = 0$$

The specific departure functions we will use are derived from the generalized Peng-Robinson equation of state.

## 2 Generalized Peng-Robinson Equation of State

The Generalized Peng-Robinson equation of state is

$$P = \frac{RT}{\underline{V} - b} - \frac{a(T)}{\underline{V}(\underline{V} + b) + b(\underline{V} - b)} \qquad \text{(Eq. 6.4-2)}$$

with

$$b = 0.07780\frac{RT_c}{P_c} \qquad \text{(Eq. 6.7-2)}$$

$$a(T) = a(T_c)\alpha(T) = 0.45724\frac{R^2T_c^2}{P_c}\alpha(T) \qquad \text{(Eq. 6.7-1)}$$

$$\sqrt{\alpha} = 1 + \kappa\left(1 - \sqrt{\frac{T}{T_c}}\right) \qquad \text{(Eq. 6.7-3)}$$

$$\kappa = 0.37464 + 1.54226\omega - 0.26992\omega^2 \qquad \text{(Eq. 6.7-4)}$$

The acentric factor $\omega$ and the crticial temperatures and pressures are given in SIS table 6.6-1.

Calculating the pressure $P$ given $\underline{V}$ and $T$ is straightforward, but to calculate the molar volume given $P$ and $T$, we need to solve the cubic equation of state of the form

$$Z^3 + \alpha Z^2 + \beta Z + \gamma = 0 \qquad \text{(Eq. 6.4-4)}$$

where $Z$ is the compressibility factor

$$Z = \frac{P\underline{V}}{RT}$$

For the Peng-Robinson EOS,

$$\alpha = -1 + B$$
$$\beta = A - 3B^2 - 2B$$
$$\gamma = -AB + B^2 + B^3$$

and

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

[3]:
```
"""
Generalized Peng-Robinson EOS
```

```
PR_pressure returns the pressure given V, T, Pc, Tc, omega
PR_volume returns all molar volumes (real roots of EOS) given P, T, Pc, Tc,␣
 ↪omega
"""

from scipy import constants
from numpy.polynomial import Polynomial

R = constants.R # Set the gas constant to R

def calc_b(Pc,Tc):
    return 0.07780*R*Tc/Pc

def calc_a(T,Pc,Tc,omega):
    kappa = 0.37464 + 1.54226*omega - 0.26992*omega**2
    sqrtalpha = 1 + kappa*(1-np.sqrt(T/Tc))
    return 0.45724*R**2*Tc**2/Pc*sqrtalpha**2

# Calculate the presure given V, T for PR EOS
def PR_pressure(V,T,Pc,Tc,omega):
    a = calc_a(T,Pc,Tc,omega)
    b = calc_b(Pc,Tc)

    P = R*T/(V-b) - a/(V*(V+b)+b*(V-b))
    return P

# Solve for the compressibility factor given P, T for PR EOS in m^3/mol
# Note that we can return multiple real roots (up to three)
# The largest and smallest will be the vapor and liquid, respectively
def PR_compressibility(P, T, Pc, Tc, omega):
    # Calculate a, b, A, and B
    a = calc_a(T, Pc, Tc, omega)
    b = calc_b(Pc, Tc)
    A = a*P/R**2/T**2
    B = b*P/R/T

    # Definitions of alpha, beta, gamma in SIS Table 6.4-3 for PR EOS
    alpha = -1 + B
    beta = A - 3*B**2 -2*B
    gamma = -A*B + B**2 + B**3

    # polynomial with coefficients in increasing order: c0 + c1 x + c2 x**2 + ..
 ↪.
    p = Polynomial([ gamma, beta, alpha, 1 ])

    roots = p.roots()        # returns all (possibly complex) roots of Z
    real_roots = roots.real[abs(roots.imag) < 1e-12]  # filter real ones
```

```
    return real_roots
```

### 2.1 Data for methane

For methane, we need the critical pressure and temperature and Pitzer's acentric factor:

$P_c = 4.6$ MPa
$T_c = 190.6$ K
$\underline{V}_c = 0.099\,\mathrm{m}^3/\mathrm{kmol} = 9.9 \times 10^{-5}\,\mathrm{m}^3/\mathrm{mol}$
$\omega = 0.008$

[4]:
```
# Critical values for Methane and the accentric factor
# pressure in Pa, temperature in K
Pc = 4.6e6
Tc = 190.6
omega = 0.008

# Known values in problem
T1 = 286        # Inlet temperature in K
P1 = 18.4e6     # Inlet pressure in Pa
T2 = 230        # Outlet temperature in K
```

## 3 Enthalpy calculation

### 3.1 Start by calculating $Z$ for the inlet condition

[5]:
```
TR1 = T1/Tc      # Reduced temperature
PR1 = P1/Pc      # Reduced pressure

# Use the PR_vol function to calculate the molar volume
Z1 = np.max(PR_compressibility(P1,T1,Pc,Tc,omega))
```

[6]:
```
print("Z1 = {:.2f}".format(Z1))
```

```
Z1 = 0.77
```

### 3.2 Calculate the departure function for the inlet condition

The enthalpy departure function in $(P, T)$ control variables for the Peng-Robinson equuation of state is

$$[\underline{H} - \underline{H}^{\mathrm{IG}}] = RT(Z-1) + \frac{T\frac{da}{dT} - a}{2\sqrt{2}b} \ln\left[\frac{Z + (1+\sqrt{2})B}{Z + (1-\sqrt{2})B}\right] \qquad \text{(Eq. 6.4-29)}$$

where we need to evaluate the compressibility factor,

$$Z = \frac{P\underline{V}}{RT}$$

the scaled value of $b$,

$$B = \frac{bP}{RT}$$

and the deriviative of $a(T)$ with respect to temperature given in SIS (p. 264),

$$\frac{da}{dT} = -0.45724\frac{R^2T_c^2}{P_c}\kappa\sqrt{\frac{\alpha}{TT_c}}$$

[7]:
```
# With Z already calculated, we define a function to calculate the departure function

"""
Enthalpy departure function from the Peng-Robinson equation of state (eq. 6.4-29)
For convenience, we recalculate b, kappa, sqrtalpha, and a(T) here, but we could use the earlier functions, too.

Uses globals Tc, Pc, R
"""
def calc_depH(T,P,Z):
    b = 0.07780*R*Tc/Pc
    B = b*P/R/T
    kappa = 0.37464 + 1.54226*omega - 0.26992*omega**2
    sqrtalpha = 1 + kappa*(1-(T/Tc)**0.5)
    a = 0.45724*R**2*Tc**2/Pc*sqrtalpha**2
    dadT = -0.45724*R*R*Tc*Tc/Pc*kappa*sqrtalpha/(T*Tc)**0.5

    depH = R*T*(Z-1)+(T*dadT-a)/(2*2**0.5*b)*np.log((Z+(1+2**0.5)*B)/(Z+(1-2**0.5)*B))
    return depH
```

[8]:
```
depH1 = calc_depH(T1,P1,Z1)

print(f"*** depH1 = {depH1:.0f} J/mol")
```

```
*** depH1 = -3134 J/mol
```

[9]:
```
# I'm calculating this number to compare with corresponding states later...
print(f"{-depH1/Tc/4.184:.2f}")
```

```
3.93
```

### 3.3 Calculate the change in enthalpy in the ideal state

$$\Delta\underline{H}^{\mathrm{IG}} = \int_{T_1}^{T_2} C_p^*(T)dT$$

where we will use the ideal state heat capacities summarized in Appendix A.II,

$$C_p^*(T) = A + BT + CT^2 + DT^3$$

resulting in

$$\Delta \underline{H}^{\mathrm{IG}} = A(T_2 - T_1) + \frac{1}{2}B(T_2^2 - T_1^2) + \frac{1}{3}C(T_2^3 - T_1^3) + \frac{1}{4}D(T_2^4 - T_1^4)$$

[10]:
```
# Heat capacity data for N2, O2, CO2, CH4, C2H6
params = np.array([
    [28.83, -0.157, 0.808, -2.871, 1800],
    [25.46, 1.519, -0.715, 1.311, 1800],
    [22.243, 5.977, -3.499, 7.464, 1800],
    [19.875, 5.021, 1.268, -11.004, 1500],
    [6.895, 17.255, -6.402, 7.280, 1500]
    ])

i = params[3,:]  # equation parameters for methane

deltaHIG = i[0]*(T2-T1) + i[1]*10**-2*(T2**2-T1**2)/2 +␣
 ↪i[2]*10**-5*(T2**3-T1**3)/3 + i[3]*10**-9*(T2**4-T1**4)/4
```

[11]:
```
print("Delta HIG = {:.0f} J/mol".format(deltaHIG))
```

```
Delta HIG = -1875 J/mol
```

Now solve

$$[\underline{H} - \underline{H}^{\mathrm{IG}}]_2 = [\underline{H} - \underline{H}^{\mathrm{IG}}]_1 - \Delta \underline{H}^{\mathrm{IG}}$$

We are looking for a value of the departure function:

[12]:
```
print("depH2 must equal {:.0f} J/mol".format(depH1-deltaHIG))
```

```
depH2 must equal -1259 J/mol
```

Do this by guessing an outlet pressure, calculating $[\underline{H} - \underline{H}^{\mathrm{IG}}]_2$ and iterating (guessing new values of $P_2$) until the condition above is met.

[ ]:
```
# Guess:
P2 = 4.145e6  # guess a pressure

# Use the PR_compressibility function to calculate the molar volume and␣
 ↪compressibility factor
Z2 = np.max(PR_compressibility(P2,T2,Pc,Tc,omega))

depH2 = calc_depH(T2,P2,Z2)
#print(P2/Pc)
print("depH2 = {:.0f} J/mol, deltaH = {:.0f} J/mol <-- should be zero".
 ↪format(depH2,deltaHIG+depH2-depH1))
```

```
depH2 = -1259 J/mol, deltaH = 0 J/mol <-- should be zero
```

---

## 4 Change in molar entropy

Now we can calcuate the change in molar entropy of the methane,

$$\Delta \underline{S} = \Delta \underline{S}^{\mathrm{IG}} + \Delta[\underline{S} - \underline{S}^{\mathrm{IG}}]$$

Since the throttling process is adiabatic, this change represents the rate of entropy generation, $\dot{\underline{S}}_{\mathrm{gen}}$.

The entropy departure function for the Peng-Robinson equation is

$$[\underline{S} - \underline{S}^{\mathrm{IG}}] = R\ln(Z-B) + \frac{da/dT}{2\sqrt{2}b}\ln\left[\frac{Z+(1+\sqrt{2})B}{Z+(1-\sqrt{2})B}\right] \qquad \text{(Eq. 6.4-30)}$$

and the entropy change of the real fluid in an ideal state with $P$ and $T$ control variables is

$$\Delta \underline{S}^{\mathrm{IG}} = \int_{T_1}^{T_2} \frac{C_p^*(T)}{T} dT - R\ln\frac{P_2}{P_1}$$

with $C_p^*(T)$ given above,

$$\Delta \underline{S}^{\mathrm{IG}} = A\ln\frac{T_2}{T_1} + B(T_2 - T_1) + \frac{1}{2}C(T_2^2 - T_1^2) + \frac{1}{3}D(T_2^3 - T_1^3) - R\ln\frac{P_2}{P_1}$$

[27]:
```
deltaSIG = i[0]*np.log(T2/T1) + i[1]*10**-2*(T2-T1) + i[2]*10**-5*(T2**2-T1**2)/
↪2 + i[3]*10**-9*(T2**3-T1**3)/3 - R*np.log(P2/P1)
```

[28]:
```
"""
Entropy departure function from the Peng-Robinson equation of state (eq. 6.4-30)
For convenience, we recalculate b, kappa, sqrtalpha, and a(T) here, but we could
use the earlier functions, too.

Uses globals Tc, Pc, R
"""
def calc_depS(T,P,Z):
    b = 0.07780*R*Tc/Pc
    B = b*P/R/T
    kappa = 0.37464 + 1.54226*omega - 0.26992*omega**2
    sqrtalpha = 1 + kappa*(1-np.sqrt(T/Tc))
    a = 0.45724*R**2*Tc**2/Pc*sqrtalpha**2
    dadT = -0.45724*R*R*Tc*Tc/Pc*kappa*sqrtalpha/np.sqrt(T*Tc)

    depS = R*(Z-B)+dadT/(2*2**0.5*b)*np.log((Z+(1+2**0.5)*B)/(Z+(1-2**0.5)*B))
    return depS

depS1 = calc_depS(T1, P1, Z1)
depS2 = calc_depS(T2, P2, Z2)
```

[29]:
```
print("T1: {:.1f} K, P1: {:0.2e} Pa, Z1: {:.2f}".format(T1,P1,Z1))
print("T2: {:.1f} K, P2: {:0.2e} Pa, Z2: {:.2f}\n".format(T2,P2,Z2))
print("delSIG= {:.2f}, depS2= {:.1f}, depS1= {:.1f}\n".
  ↪format(deltaSIG,depS2,depS1))
print("Change in molar entropy: {:.1f} J/mol.K".format(deltaSIG + depS2 -␣
  ↪depS1))
```

```
T1: 286.0 K, P1: 1.84e+07 Pa, Z1: 0.77
T2: 230.0 K, P2: 4.14e+06 Pa, Z2: 0.79

delSIG= 5.11, depS2= 4.9, depS1= 1.6

Change in molar entropy: 8.5 J/mol.K
```

[ ]: