Homework Assignment 1

Given Information

Industrial applications routinely require knowledge of the phase diagram of a compound. Here you will consider the case of vapor-liquid equilibrium (VLE) using either the Peng-Robinson or Soave-Redlich-Kwong equation of state. You will make use of thermodynamic data available online from the (United States) National Institute of Standards and Technology (NIST) at webbook.nist.gov/chemistry/fluid/

The data is available in tabular form (for plotting in python, Matab or whatever program you prefer) by unchecking the option for the Java applet.

Question 1
Select one of the compounds available in the NIST webbook. Look up T_c, p_c, and ω - the critical temperature, critical pressure, and acentric factor. These are the only inputs you will need for your EOS calculations. Report your compound and the three inputs to your EOS.

Compound of choice: **Hydrogen Sulfide**

Critical Temperature $T_c = 373.1$ K
Critical Pressure $p_c = 9$ MPa
Acentric Factor $\omega = 0.1$

Equation of State used: **Soave-Redlich-Kwong**. Pressure is given by:

$$p = \frac{R.T}{v-b} - \frac{a}{\sqrt{T}.v.(v+b)}$$

where,

$$a = 0.42748.\frac{R^2.{T_c}^{2.5}}{p_c}$$

$$b = 0.08664.\frac{R.T_c}{p_c}$$

For the given input values, $a = 8.828862025790215$ [SI Units] and $b = 2.9863131126216268 \times 10^{-5}$ [SI Units]

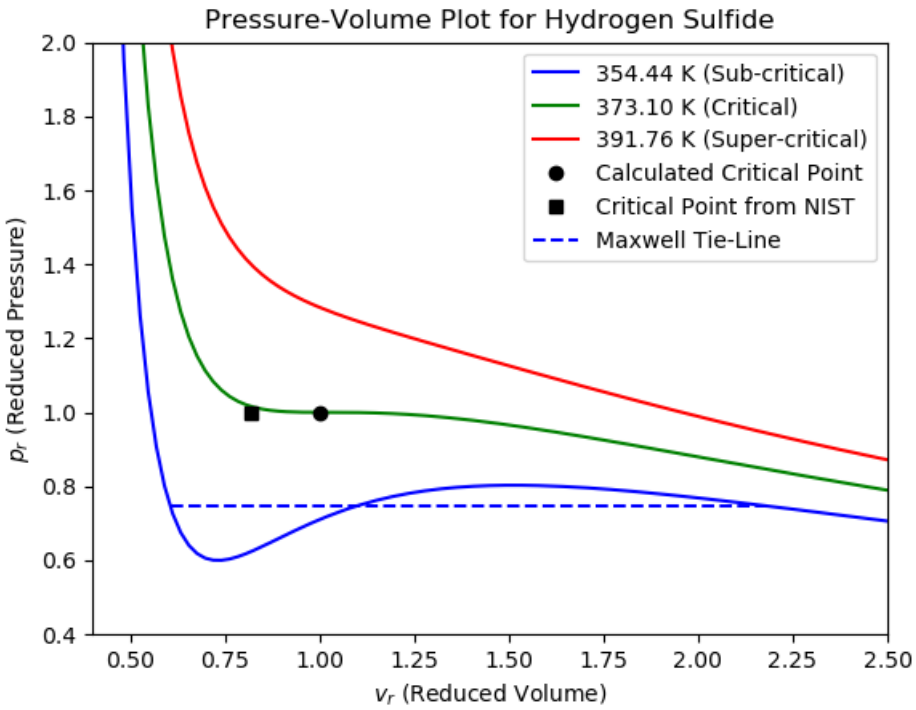
Question 2
Use python, Matlab, or your favorite alternative to calculate the critical molar volume. Compare the value predicted by your EOS and the value reported by NIST.

Using Python, Critical Molar Volume was calculated from the EOS of choice:

Calculated Critical Molar Volume $v_c = 0.112788538874284$ [L/mol]
Actual Critical Molar Volume [NIST Data] $v_{c,actual} = 0.09221700000000001$ [L/mol]
Error = 18.239 %

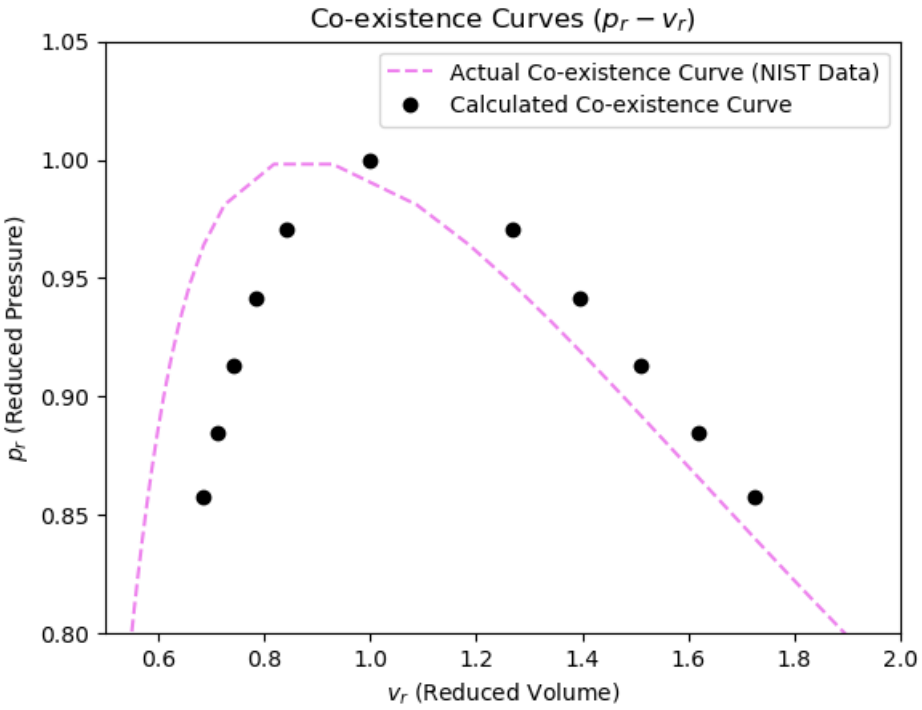
Question 3
Plot three isotherms of your EOS - one above, one at, and one below the critical temperature. For the sub-critical isotherm, include the Maxwell tie line (which you will have to determine numerically). Here and in all subsequent plots, report your results in terms of reduced quantities, i.e. the reduced pressure $p_r = p/p_c$, reduced molar volume $v_r = v/v_c$, and reduced temperature $T_r = T/T_c$. Clearly label your axes and provide legends where needed. Ask yourself two questions: could a classmate understand what I have plotted without having me here to explain? Have I provided enough information that my instructor could reproduce my plot?

Choosing a Sub-critical and Super-critical temperature, we plot the 3 required isotherms for Hydrogen Sulfide in $p_r - v_r$ space. We also show the actual and calculated critical molar volumes along with the Maxwell tie-line which represents the phase change correction as EOS does not account for it.



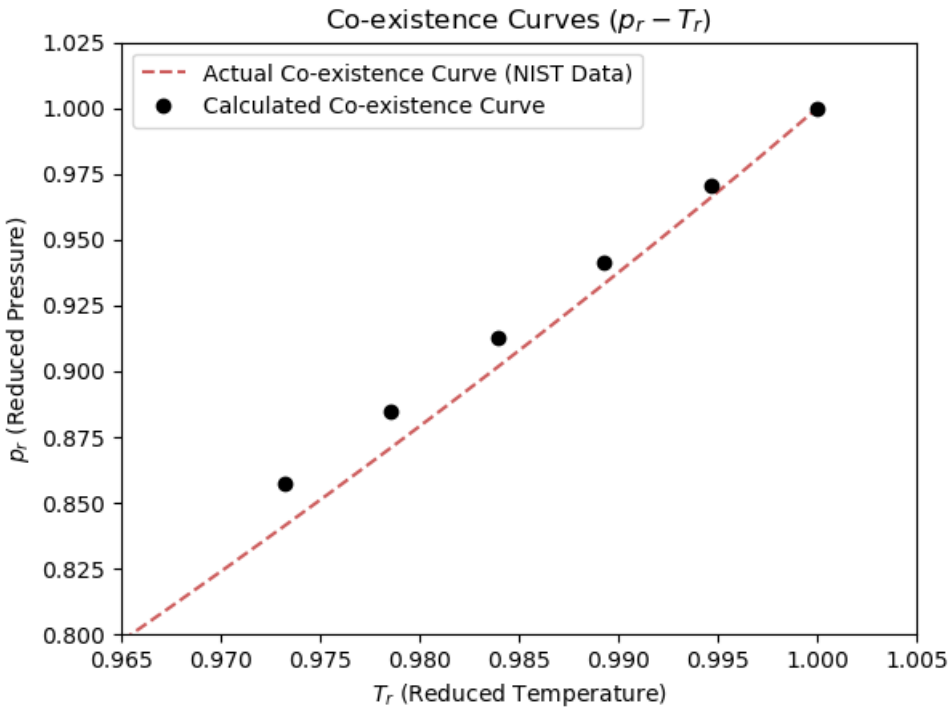
Question 4
Repeat your calculation of the coexistence pressure for five additional temperatures. Plot the boundaries of the coexistence region predicted by your EOS in $p_r - v_r$ space. Calculate v_r using the critical molar volume predicted by your EOS. In the same plot, plot the boundaries of the coexistence data reported by NIST (using different symbols). Include a legend.

As per the instructions provided, coexistence region plot:



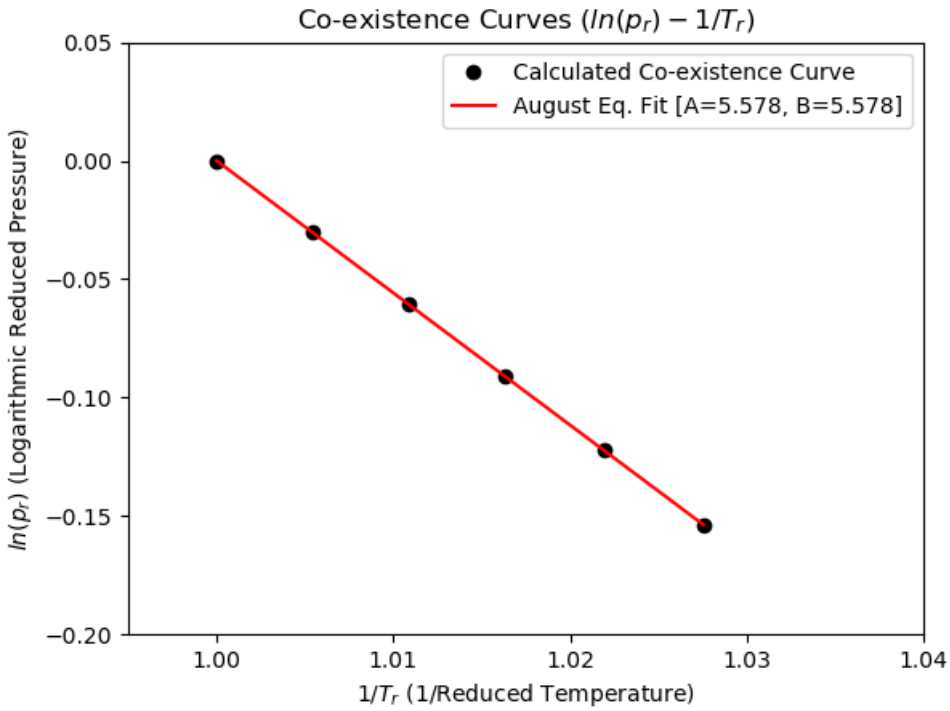
Question 5
Plot the coexistence curve predicted by your EOS in $p_r - T_r$ space. In the same figure, plot the coexistence curve from NIST. Include a legend.

As per the instructions provided, coexistence region plot:



Question 6
Plot the coexistence curve predicted by your EOS as $ln(p_r)$ versus $1/T_r$. Fit a line to the data and report the result of the fit. Use your fit coefficients to estimate the enthalpy of vaporization. Compare your result to the value reported by NIST.

As per the instructions provided, coexistence region plot:



We fit the line using August Equation, which is given by:

$$ln\left(\frac{p}{p_c}\right) = ln(p_r) = A - \frac{B}{T}$$

Clausius-Clapeyron Equation states that:

$$\frac{\mathrm{d}(ln(pr))}{\mathrm{d}T} = \frac{\Delta h^{LV}}{R.T^2}$$

Refactoring terms to take the derivative in terms of $1/T$:

$$\frac{\mathrm{d}(ln(pr))}{\mathrm{d}(1/T)} = \frac{\Delta h^{LV}}{R}$$

This means that the slope of our $ln(p_r) - 1/T_r$ multiplied with the molar gas constant and T_c would give us the enthalpy of vaporization. Calculating based on the B value obtained from the fit:

$$\Delta h^{LV} = 17,305.121\ [J/mol]$$

Enthalpy of vaporization calculated from NIST data gives us:

$$\Delta h^{LV}_{NIST} = 18,357.126\ [J/mol]$$

The error in the value we calculated is 5.73%

Appendix: Python Script

```
import matplotlib.pyplot as plt
import numpy as np
import scipy.optimize as spo
import scipy.integrate as spi
from sympy import var,solve

# Question 1: Reporting compound and inputs for EOS

Fluid = "Hydrogen␣Sulfide"

# Source: NIST Website
T_c = 373.1 # Critical Temp [K]
p_c = 9.0000 * (10**6) # Critical Pressure [Pa]
w = 0.100 # Acentric factor
R = 8.3144621 # Gas constant [SI Units]
M = 34.08088*(10**-3) # Molecular weight [mol/kg]
rho_c = 10.844*10**3 # Density [mol/m^3]

# Redlich-Kwong a,b values
a = 0.42748 * ((R**2)*(T_c**2.5))/p_c
b = 0.08664 * (R*T_c)/p_c
print("Redlich-Kwong␣Parameters:\na␣=␣",a,"\nb␣=␣",b,"\n")

# Redlich-Kwong Equation
def pressure(T,v):
    p = (R*T)/(v-b)-a/((T**0.5)*v*(v+b))
    return p

# Molar Volume Solver
def molar_volumes(p,T):
    v_r = var('v_r')
    solution = solve((R*T)/(v_r-b)-a/((T**0.5)*v_r*(v_r+b))-p)
    solution = [abs(n) for n in solution]
    return solution

# Question 2: Calculating Critical Molar Volume

v_c = molar_volumes(p_c,T_c)[0]
print("Critical␣Molar␣Volume:␣",v_c * 1000,"[L/mol]")

v_c_NIST = 0.092217 * 0.001 # Critical Molar Volume [Units: m3/mol]
print("Critical␣Molar␣Volume␣(NIST):␣",v_c_NIST * 1000,"[L/mol]")

v_c_error = (v_c-v_c_NIST)/v_c
print("Error:␣",v_c_error,"%\n")

volume_set = np.linspace(0.00004,0.0005,191)
print("Molar␣Volume␣Array:␣",volume_set,"\n")
```

```
# Question 3: Plotting 3 isotherms for R-K EOS

# Calculating the pressure for the molar volume array


def pressure_set(T, v_set):
    pressure = R * T / (v_set - b) - a / ((T ** 0.5) * v_set * (v_set + b))
    pressure = [abs(n) for n in pressure]
    return pressure

Temperature_set = [T_c*0.95,T_c,T_c*1.05]
Pressure_set = [pressure_set(Temperature_set[0],volume_set),pressure_set(
    ↪ Temperature_set[1],volume_set),pressure_set(Temperature_set[2],volume_set)]

def F_integral(T,p_trial,v_L,v_V):
    F_p,F_p_error = spi.quad(lambda v: ((R*T)/(v-b)-a/((T**0.5)*v*(v+b))-p_trial),
        ↪ v_L,v_V)
    return F_p

def tie_line_parameters(T,p_trial,dp,error):
    v_roots = molar_volumes(p_trial,T)
    F_p = F_integral(T, p_trial, min(v_roots), max(v_roots))

    while F_p > error:
        v_roots = molar_volumes(p_trial, T)
        F_p = F_integral(T,p_trial,min(v_roots),max(v_roots))
        F_p_dp = F_integral(T,p_trial+dp,min(v_roots),max(v_roots))
        F_dash = (F_p_dp - F_p)/dp
        p_trial -= F_p/F_dash

    return [p_trial,v_roots]

p_star,v_star = tie_line_parameters(Temperature_set[0], 3.5*1000000, 10, 0.1)

plt.figure(1)
plt.clf()

# Plotting the 3 isotherms for the values is Temperature_set
plt.plot([v/v_c for v in volume_set],[p/p_c for p in Pressure_set[0]],color='blue
    ↪ ',label="{:0.2f} K (Sub-critical)".format(Temperature_set[0]))
plt.plot([v/v_c for v in volume_set],[p/p_c for p in Pressure_set[1]],color='green
    ↪ ',label="{:0.2f} K (Critical)".format(Temperature_set[1]))
plt.plot([v/v_c for v in volume_set],[p/p_c for p in Pressure_set[2]],color='red',
    ↪ label="{:0.2f} K (Super-critical)".format(Temperature_set[2]))

# Plotting critical points
plt.plot(1,1,'o',color='black',label="Calculated Critical Point") # As in reduced
    ↪ pressure and volume p = p/p_c and v = v/v_c | p = v = 1
plt.plot(v_c_NIST/v_c,1,'s',color='black',label="Critical Point from NIST")
```

```
# Plotting maxwell tie line
plt.plot([v/v_c for v in v_star], [p_star/p_c]*3, '--', color='blue', label="
    ↪ Maxwell Tie-Line")

# Context additions to the plot
plt.title("Pressure-Volume Plot for "+Fluid)
plt.xlabel(r"$v_r$"+" (Reduced Volume)")
plt.ylabel(r"$p_r$"+" (Reduced Pressure)")
ax = plt.gca()
ax.set_xlim([0.4, 2.5])
ax.set_ylim([0.4, 2])
plt.legend()
plt.savefig("PVPlot_"+Fluid.replace(" ", "")+".png")

# Question 4: Coexistence Pressure calculation for additional terms

coexistence_temp = np.append((np.flip([T_c-2*(i+1) for i in range(0,5)])),T_c)

coexistence_p_v = [tie_line_parameters(coexistence_temp[i], 3.5*1000000, 10, 0.1)
    ↪ for i in range(0,5)]

# Creating a set of v and p points to plot the coexistence curve
v_star_set = np.append(np.append([min(coexistence_p_v[i][1]) for i in range(0,5)],
    ↪ v_c),[max(coexistence_p_v[i][1]) for i in range(0,5)])
p_star_set = np.append(np.append([coexistence_p_v[i][0] for i in range(0,5)],p_c)
    ↪ ,[coexistence_p_v[i][0] for i in range(0,5)])

print("Coexistence values:")
print("Volume: ",v_star_set)
print("Pressure: ",p_star_set)

v_star_set_NIST = [<NIST data set too large to insert>]
p_star_set_NIST = [<NIST data set too large to insert>]

plt.figure(2)
plt.clf()
plt.plot([v/v_c for v in v_star_set_NIST],[p/p_c for p in p_star_set_NIST],'--',
    ↪ color='violet',label='Actual Co-existence Curve (NIST Data)')
plt.plot([v/v_c for v in v_star_set],[p/p_c for p in p_star_set],'o',color='black
    ↪ ',label='Calculated Co-existence Curve')

plt.title("Co-existence Curves $(p_r-v_r)$")
plt.xlabel(r"$v_r$"+" (Reduced Volume)")
plt.ylabel(r"$p_r$"+" (Reduced Pressure)")
plt.legend()
ax = plt.gca()
ax.set_xlim([0.5, 2])
ax.set_ylim([0.8, 1.05])
plt.savefig("CoexistenceCurve_"+Fluid.replace(" ", "")+".png")
```

```
# Question 5: Plotting coexistence curve in P_r-T_r space

coexistence_temp_NIST = [<NIST data set too large to insert>]

plt.figure(3)
plt.clf()
plt.plot([T/T_c for T in coexistence_temp_NIST],[p/p_c for p in p_star_set_NIST
    ↪ [0:74]],'--',color='indianred',label='Actual Co-existence Curve (NIST Data)
    ↪ ')
plt.plot([T/T_c for T in coexistence_temp],[p/p_c for p in p_star_set[0:6]],'o',
    ↪ color='black',label='Calculated Co-existence Curve')
plt.title("Co-existence Curves $(p_r-T_r)$")
plt.xlabel(r"$T_r$"+" (Reduced Temperature)")
plt.ylabel(r"$p_r$"+" (Reduced Pressure)")
plt.legend()
ax = plt.gca()
ax.set_xlim([0.965, 1.005])
ax.set_ylim([0.800, 1.025])
plt.savefig("CoexistenceCurve_PvsT_"+Fluid.replace(" ", "")+".png")

# Question 6: Plotting coexistence curve in ln P_r - 1/T_r space

# Curve fitting using SciPy

# August Equation
def august_equation(x,A,B):
    return A - B * x

p_opt,p_cov = spo.curve_fit(august_equation,[T_c/T for T in coexistence_temp],[np.
    ↪ log(p/p_c) for p in p_star_set[0:6]])

plt.figure(4)
plt.clf()
#plt.plot([T_c/T for T in coexistence_temp_NIST],[np.log(p/p_c) for p in
    ↪ p_star_set_NIST[0:74]],'--',color='mediumturquoise',label='Actual Co-
    ↪ existence Curve (NIST Data)')
plt.plot([T_c/T for T in coexistence_temp],[np.log(p/p_c) for p in p_star_set
    ↪ [0:6]],'o',color='black',label='Calculated Co-existence Curve')
plt.plot([T_c/T for T in coexistence_temp],[p_opt[0] - p_opt[1] * T_c/T for T in
    ↪ coexistence_temp],'r-',label='August Eq. Fit [A=%5.3f, B=%5.3f]' % tuple(
    ↪ p_opt))

plt.title("Co-existence Curves $(ln(p_r)-1/T_r)$")
plt.xlabel(r"$1/T_r$"+" (1/Reduced Temperature)")
plt.ylabel(r"$ln(p_r)$"+" (Logarithmic Reduced Pressure)")
plt.legend()
ax = plt.gca()
ax.set_xlim([0.995, 1.04])
ax.set_ylim([-0.2, 0.05])
plt.savefig("CoexistenceCurve_lnPvsinvT_"+Fluid.replace(" ", "")+".png")
```

```
# Enthalpy of vaporization
delta_h = p_opt[0] * R * T_c

print('Enthalpy of vaporization: ',delta_h)

# Repeating calculations and plotting for NIST data
p_opt_NIST,p_cov_NIST = spo.curve_fit(august_equation,[T_c/T for T in
    ↪ coexistence_temp_NIST],[np.log(p/p_c) for p in p_star_set_NIST[0:74]])

plt.figure(5)
plt.clf()
plt.plot([T_c/T for T in coexistence_temp_NIST],[np.log(p/p_c) for p in
    ↪ p_star_set_NIST[0:74]],'--',color='mediumturquoise',label='Actual Co-
    ↪ existence Curve (NIST Data)')
plt.plot([T_c/T for T in coexistence_temp_NIST],[p_opt_NIST[0] - p_opt_NIST[1] *
    ↪ T_c/T for T in coexistence_temp_NIST],'r-',label='August Eq. Fit [A=%5.3f,
    ↪ B=%5.3f]' % tuple(p_opt_NIST))

plt.title("Actual Co-existence Curves $(ln(p_r)-1/T_r)$ (NIST Data)")
plt.xlabel(r"$1/T_r$"+" (1/Reduced Temperature)")
plt.ylabel(r"$ln(p_r)$"+" (Logarithmic Reduced Pressure)")
plt.legend()
ax = plt.gca()
ax.set_xlim([0.995, 1.04])
ax.set_ylim([-0.2, 0.05])
plt.savefig("ActualCoexistenceCurve_lnPvsinvT_"+Fluid.replace(" ", "")+".png")

# Enthalpy of vaporization (NIST)
delta_h_NIST = p_opt_NIST[0] * R * T_c

print('Actual Enthalpy of vaporization: ',delta_h_NIST)

print('Error: ',(delta_h_NIST-delta_h)/delta_h_NIST)
```