10.40 Thermodynamics **Fall 2003**

Problem Set 10

Problem 1

Using the Peng-Robinson EOS to calculate the fugacity coefficient of pure carbon dioxide, estimate the vapor pressure and saturated liquid and vapor molar volumes at these temperatures – 10, 20, and 30°C. How do your predictions compare with data for pure CO_2 (see Table below)? You might want to refer to Example 8.4 on p. 276-277, Example 9.7 on p. 350-351, and Figure 8.12. If you use Matlab or a similar mathematical software package please document your results and describe the approach you followed in reaching a solution.

Experimental Data for Pure CO_2

Temp (°C)	Pressure (bar)	Liquid Volume (cm^3/mole)	Vapor Volume (cm^3/mole)
10	45.02	51.11	325.6
20	57.29	56.91	226.6
30	72.14	74.18	127.5

Solution:

For pure CO_2 at vapor-liquid equilibrium (VLE):

$$\ln \phi_{CO_2}^{(L)} = \ln \phi_{CO_2}^{(V)}$$

Given an appropriate EOS, $\ln \phi_i^{(\alpha)}$ can be determined using equations given in Chapter 9 of Tester & Modell. For this problem, we are asked to use the Peng-Robing EOS to estimate vapor pressures at a number of temperatures. Normally we would have to derive the expression for $\ln \phi_i^{(\alpha)}$ from (9-143), but luckily it is given for us for the PR EOS in Example (9.7):

$$RT \ln \phi_i = \left(A - A^o\right) + RT\left(Z - 1\right)$$

Also, from Example (8.4) for the PR EOS we are given that:

$$\left(A - A^o\right) = RT \ln \frac{V^o}{V - b} + \frac{a}{2\sqrt{2}b} \ln \frac{V + b\left(1 - \sqrt{2}\right)}{V + b\left(1 + \sqrt{2}\right)}$$

$$V^o = \frac{RT}{P} \text{ (ideal gas volume)}$$

In order to continue, we need to determine the volume of the liquid and vapor phase to use in the above equations. We can find these by converting the PR EOS to its cubic form and calculating the roots of the polynomial. From Appendix E:

$P = \frac{RT}{V-b} - \frac{a}{V^2}$ becomes …

$$V^3 + \left(b - \frac{RT}{P}\right)V^2 + \left(\frac{a}{P} - 3b^2 - \frac{2bRT}{P}\right)V + \left(b^3 + \frac{b^2RT}{P} - \frac{ab}{P}\right) = 0$$

This cubic polynomial can be solved using a variety of methods, including the methods described in Appendix E, the *roots( )* function in Matlab (returns all 3 roots of the equation), or the *fsolve( )* function in Matlab (not recommended since only one root is returned, no guarantee that root will be physical or real). It should be noted that the 3 roots will be either 1.) all real and different, corresponding to the vapor and liquid volumes and a third, non-physical volume in the unstable region of the PVT diagram, 2.) all real and identical, corresponding to the critical point, or 3.) 1 real root, corresponding to the volume of a phase not in VLE. Your program should check to make sure that your roots correspond to two phases in VLE to prevent errors.

We now have 5 variables ($\phi_{CO2}^{(L)}$,$\phi_{CO2}^{(V)}$, V_L, V_V, and P_{VP}) but only 4 equations (the VLE equation, and equation for each $\phi_i^{(\alpha)}$, and the PR EOS). There are no other equations we can use, so we are forced to use an interative process to guess the P_{VP} that will satisfy the condition that the fugacity of the two phases be equal (this is tantamount to making a 5th equation: $P_{VP} = x$, and finding an x that will solve all the equations). The following algorithm is used:

0.) Given a Temperature, guess an initial P_{VP}
1.) Calculate V_L and V_V using PR EOS and solving for cubic roots
2.) Calculate ln $\phi_{CO2}^{(L)}$
3.) Calculate ln $\phi_{CO2}^{(V)}$
4.) Compare $\phi_{CO2}^{(L)}$ and Calculate ln $\phi_{CO2}^{(V)}$
    a. If difference is greater than allowed tolerance, guess new P_{VP} and return to step 1.
    b. If difference is less than allowed tolerance, record P_{VP} and quit

There are several methods we can use to guess a new P_{VP} when the difference between fugacities is not zero. One of the simplest (other than random guessing) is Newton's method:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

where x_{i+1} is the next guess, x_i is the previous guess, $f(x_i)$ is some function of x_i, and $f'(x_i)$ is the derivative of that function. For our problem:

$x = P_{VP}$

$$f(x) = RT\ \ln\left(\phi^L\right) - RT\ \ln\left(\phi^V\right) = \mu^L - \mu^V$$

$$f'(x) = \left(\frac{\partial\left(\mu^L - \mu^V\right)}{\partial P}\right)_T = V_L - V_V$$

$$\therefore P_{VP,new} = P_{VP} - \left(\frac{\mu^L - \mu^V}{V_L - V_V}\right)$$

Therefore, we can use information from our previous guess to make our next guess. This algorithm is very robust, given a good initial guess, and converges quickly.

Another option would be to use a substitution method, where $x_{i+1} = F(x_i)$. In this case, one can show that since:

$$Z^{\alpha} = \frac{PV^{\alpha}}{RT}$$

$$P = -\left[\frac{\left(A - A^{o}\right)\Big|_{L} - \left(A - A^{o}\right)\Big|_{V}}{V_L - V_V}\right]$$

Using this relation, the next P_{VP} guess would based on the previous guess (see Appendix E).

An example of a Matlab Program that incorporates the above algorithm is shown at the end of the problem statement. Successful execution of this (or any) program is very dependent on the initial P_{VP} guess. However, this program has a safety check built into that determines if the initial guess is bad and most of the time is able to correct it and result in a solution. The results of the P_{VP} as determined by the PR EOS are shown below. Note that at lower temperatures, the agreement is pretty good, but more error is introduced at the critical temperature is approached.

Experimental Data for Pure CO_2 vs. Peng-Robinson Estimates

	Experimental			Peng_Robinson		
Temp (^{o}C)	Vapor Pressure (bar)	Liquid Volume (cm^3/mole)	Vapor Volume (cm^3/mole)	Vapor Pressure (bar)	Liquid Volume (cm^3/mole)	Vapor Volume (cm^3/mole)
10	45.02	51.11	325.6	44.95	53.57	323.4
20	57.29	56.91	226.6	57.32	62.43	223.4
30	72.14	74.18	127.5	72.07	87.90	129.1

```
% VaporPressure.m
% 10.40 Problem Set #9
% Problem 1

% This program calculates the vapor pressure at Vapor-Liquid equilibrium of
% CO2 for a given temperature.  At equilibrium, the fugacities of the vapor
% and liquid phases are equal.  The program iteratively calculates
% the difference between the fugacity coefficient of the vapor and liquid
% phases at successive pressures until the difference is zero.  The
% Peng-Robinson EOS is used in these calculations:
%      R T              a
% P = ----- - ----------------------
%     V - b   V (V + b) + b (V - b)

% DECLARED CONSTANTS
% ==================
% Temperature (K)
T = 10 + 273.15;

% Initial Pressure (bar) Guess:
P = 50;             % Pressure (bar)
delta_P = 5;   % change in pressure for next iteration (bar)
P = P + delta_P;

% Set tolerance and difference in fugacity coefficients
delta_phi = 1;
tol = 1E-8;
count = 0;  %counter

Tc = 304.2;     % Critical Temperature (K)
Pc = 73.76;      % Critical Pressure (bar)
w = 0.225;      % accentric factor (unitless)
R = 83.14;      % Universal Gas Constant (cm^3 bar)/(mol K)

% CALCULATED CONSTANTS
% ====================
% These terms are used to calculate the constants in the PR EOS
ac = 0.45724 * R^2 * Tc^2 / Pc;             % VdW atttractive term
K = 0.37464 + 1.54226 * w - 0.26992 * w^2;
alpha = (1 + K * (1 - sqrt(T/Tc)))^2;
a = ac * alpha;             % PR attractive term
b = 0.07780 * R * Tc / Pc;  % PR repulsive term

%  Set output format
format short g;

% ITERATE
% =======
% Perform iterations until fugacity coefficients are equal
while (abs(delta_phi) > tol)
   % CALCULATE VAPOR AND LIQUID VOLUMES
   % ==================================
   % Use PR_Zroots function, returns roots of PR cubic EOS
   % in terms of Z (3x1 vector)
   Z = PR_Zroots(a, b, P, T, R);

   % Check that roots are all real before proceeding
   flag = 0;   %Flag for pressure adjusting
   while (isreal(Z) == 0)
      % Some roots are non-real, we are not on VLE curve
      disp('Non-Real roots in initial guess: Adjusting pressure');

      real = 0;   % Calculate real root
      for k = 1:length(Z)
         real = real + isreal(Z(k))*Z(k);
      end
```

```
    % if Z < Zc, pressure guess is too high, and vice versa
    % Note Zc = 0.307 is crit. volume for PR EOS, not experimental
    if real < 0.307
       sign = -1;  % Pressure will be reduced
    else
       sign = 1;   % Pressure will be increased
    end

    % If flag changes sign, or this is first time through, then reset i and halve delta_P
    if((flag - sign ~= 0))
       delta_P = (1/2)*delta_P;    % halve the step
       i = 0;      % reset the counter, we're switching directions
       flag = sign; % flag can now check if sign changes next time
    end

    P = P + flag*(2^i)*delta_P; % Adjust the pressure
    i = i+1;                    % Increment counter

    Z = PR_Zroots(a, b, P, T, R); %Recalculate Z
  end % roots are real while loop

  % Calculate volumes: V (cm^3/mol) is a 3x1 vector of the volume roots
  V = R*T*Z/P;
  Vv = max(V);    % vapor volume is largest root
  Vl  = min(V);   % liquid volume is smallest root

  % CALCULATE FUGACITY COEFFICIENTS
  % ===============================
  % Calculate RT*ln(phi)= (A-Ao) + RT(Z-1)
  % From Example (8.4):
  A_Ao_liq = R*T*log((R*T/P)/(Vl-b)) + a/(2*sqrt(2)*b)*log((Vl+b*(1-sqrt(2)))/(Vl+b*(1+sqrt(2))));
  A_Ao_vap = R*T*log((R*T/P)/(Vv-b)) + a/(2*sqrt(2)*b)*log((Vv+b*(1-sqrt(2)))/(Vv+b*(1+sqrt(2))));

  % Calculate difference
  delta_phi = A_Ao_liq - A_Ao_vap + P*(Vl-Vv);
  count = count + 1;
  disp([delta_phi count]);

  % CALCULATE NEW PRESSURE
  % ======================
  % Use Newton's Method
  delta_P = delta_phi/(Vv - Vl);
  P = P + delta_P;
end % fugacities are equal while loop

% OUTPUT RESULTS
results =[T P Vl Vv];
disp('Temperature (K)  Pressure(bar) Vvap(cm3/mol) Vliq(cm3/mol)');
disp('===============  ============= ============= =============');
disp(results);
```

```
% PR_Zroots.m
%
% Chad Augustine
% 10.40
% PS 10
%
% This program finds the zeros of the Peng-Robinson cubic
% equation of state in its compressibility factor (Z) form.
% These zeros can correspond to the liquid molar volume,
% vapor molar volume, and an unstable volume for a system
% in VL equil., or they can correspond to a single real
% volume if T > Tc or the conditions are such that VLE
% doesn't exist.

% This program works for pure components or for mixtures,
% as long as the parameters a and b are calculated for the
% mixture of interest.

% The program returns the three roots of the cubic equation.
% The user can then decide how to use these roots.

% ================================
% The Peng Robinson cubic equation of state in the compressibility
% factor format has the form:
%
%   3          2          2            3    2
%  Z  + (B - 1) Z  + (A - 3 B  - 2 B) Z + B  + B  - A B = 0
%
% where
%  A = a*P/(R*T)^2
%  B = b*P/(R*T)

% INPUT
% a = attractive term parameter of PR EOS, scalar
% b = repulsive term paramter of PR EOS, scalar
% P = system pressure, scalar
% T = system temperatuer, scalar
% R = universal gas constant in same units at P and T

% OUTPUT
% Z = roots of cubic equation, 3x1 vector

function[Z] = PR_Zroots(a, b, P, T, R);

% ================================
% Can rewrite cubic equation as
% Z^3 p1(Z^2) + p2(Z) + p3 = 0
% where...
p1 = b*P/(R*T) - 1;
p2 = a*P/((R*T)^2) - 3*(b*P/(R*T))^2 - 2*b*P/(R*T);
p3 = (b*P/(R*T))^3 + (b*P/(R*T))^2 - (a*P/(R^2*T^2)*b*P/(R*T));

% Calculate roots of cubic using root function
Z = roots([1 p1 p2 p3]);

return;  %Return control to calling function
```