Scilab Textbook Companion for Physical And Chemical Equilibrium For Chemical Engineers by N. de Nevers[1]

Created by
Ravi Shankar Patel
B. Tech
Chemical Engineering
IT-BHU
College Teacher
Prakash Kotecha
Cross-Checked by
Prakash Kotecha

July 31, 2019

[1]Funded by a grant from the National Mission on Education through ICT, http://spoken-tutorial.org/NMEICT-Intro. This Textbook Companion and Scilab codes written in it can be downloaded from the "Textbook Companion Project" section at the website http://scilab.in

Book Description

Title: Physical And Chemical Equilibrium For Chemical Engineers

Author: N. de Nevers

Publisher: John And Wiley & Sons Inc., New York

Edition: 1

Year: 2002

ISBN: 978-0471071709

Scilab numbering policy used in this document and the relation to the above book.

Exa Example (Solved example)

Eqn Equation (Particular equation of the above book)

AP Appendix to Example(Scilab Code that is an Appednix to a particular Example of the above book)

For example, Exa 3.51 means solved example 3.51 of this book. Sec 2.3 means a scilab code whose theory is explained in Section 2.3 of the book.

Contents

List of Scilab Codes

Chapter 1

Introduction to Equilibrium

Scilab code Exa 1.1 `Units Conversion Factors and Notations`

```

clear;
//clc();

// Example 1.1
// Page: 9
printf("Example-1.1   Page no.-9\n\n");

//***Data***//
m_i = 10;//[g]
m_w = 990;//[g]
M_i = 342.3;//[g]
M_w = 18;//[g]
// The mass fraction is
// ( mass fraction of sucrose ) = x_i (by mass) =
   m_i/(sum of all substances)
x_i = m_i/(m_i+m_w);
x_i = x_i*100;// [in percentage]
// This is also the weight fraction.
// The mole fraction is
// ( mole fraction of sucrose ) = x_j (by mole) =
   n_i/(sum of number moles of all the substances)
n_i = m_i/M_i;// number of moles of sucrose
n_w = m_w/M_w;// number of moles of water
x_j = n_i/(n_i+n_w);
x_j = x_j*100;// [in percentage]
// The molality , a concentration unit is widely used
    in equilibrium calculations , is defined as
// m (molality) = (moles of solute)/(kg of solvent)
m = n_i/m_w*1000;//[molal]
// For solutions of solids and liquids (but not
   gases) ppm almost always means ppm by mass, so
x_ppm = x_i*10^(6)/100;//[ppm]
printf(" sucrose concentration in terms of the mass
   fraction is %f%%\n",x_i);
printf(" sucrose concentration in terms of the mole
   fraction is %f%%\n",x_j);
printf(" sucrose concentration in terms of the
   molality is      %f molal\n",m);
printf(" sucrose concentration in terms of the ppm
   is            %f ppm",x_ppm);
```

Scilab code Exa 1.2 Density dependency of concentration

```
clear;
//clc();

// Example 1.2
// Page: 9
printf("Example-1.2  Page no.-10\n\n");

//***Data***//
T = 20;//[C]
d = 1.038143/1000*10^(6);//[kg/m^(3)]
m_i = 10;//[g] mass of sucrose
M_i = 342.3;//[g/mol] molecular weight of sucrose
```

```
// In the previous example i.e. example 1.1 the mass
    was chosen to be 1.00 kg, so that
m = 1.00;//[kg]
V = m/d*1000;//[L]
// The mass concentration is
// m_1 ( mass concentration of sucrose ) = (mass of
   sucrose)/(volume of solution)
m_1 = m_i/V;//[g/L]
// The mole concentration is
// m_2 ( mole concentration of sucrose ) = (moles of
    sucrose)/(volume of solution)
m_2 = (m_i/M_i)/V;//[mol/L]
printf(" Mass concentration of the solution is %f g/
   L\n",m_1);
printf(" Mole concentration of the solution is %f
   mol/L\n",m_2);
// By the definition of the molarity, molarity is
   mole concentration of the solute
// so molarity
m = m_2;
printf(" Molarity of the solution is            %f
   mol/L",m_2);
```

Chapter 2

Basic Thermodynamics

Scilab code Exa 2.1 Calculation of the thermodynamics properties of steam

```
clear;
//clc();

// Example 2.1
// Page: 27
printf("Example-2.1  Page no.-27\n\n");

//***Data***//
m = 1;//[lbm] Mass of the steam
T_1 = 300;//[F] Initial temperature
P_1 = 14.7;//[psia] Initial pressure
P_sorronding = 14.7;//[psia]
Q = 50;//[Btu] Amount of the energy added to the
    system as heat

// This is a closed system and we can apply the
    following equations
// delta_U_system = sum(dQ_in_minus_out) + sum(
    dW_in_minus_out)                              (
    A)
// dS_system = (m*ds)_system = sum((dQ)/T)
```

```
    _in_minus_out + dS_reversible
                              (B)

// From the steam tables, we look up the properties
    of steam at temperature 300F and pressure 14.7
    psia and find
u_initial = 1109.6;//[Btu/lbm] Internal energy of
    the steam
h_initial = 1192.6;//[Btu/lbm] Enthalpy of the steam
s_initial = 1.8157;//[Btu/(lbm*R)] Entropy of the
    steam

// The work here is done by the system, equal to
//  -delta_w = P*A_piston*delta_x = P*m*delta_v

// Substituting this in the equation (A) and
    rearranging, we have
//  m*delta_(u + P*v) = m*delta_h = delta_Q
// From which we can solve for the final specific
    enthalpy
h_final = h_initial + Q;//[Btu/lbm]

// Now, by the linear interpolation we find that at
    h = 1242.6 Btu/lbm and P = 1 atm, temperature of
    the steam is given
T_2 = 405.7;//[F] Final temperature

// At this final temperature and pressure we have
    the steam properties
u_final = 1147.7;//[Btu/lbm]
s_final = 1.8772;//[Btu/(lbm*R)]

// Thus, increase in the internal energy, enthalpy
    and entropy are
delta_u = u_final - u_initial;//[Btu/lbm]
delta_s = s_final - s_initial;//[Btu/(lbm*R)]
delta_h = Q;//[Btu/lbm]

```

```
// The work done on the atmosphere is given by
w = delta_h - delta_u;//[Btulbm]

printf("The increase in internal energy of the steam
    by adding the heat is %0.2f Btu/lbm\n",delta_u);
printf("The increase in enthalpy of the steam by
   adding the heat is        %0.2f Btu/lbm\n",
   delta_h);
printf("The increase in entropy of the steam by
   adding the heat is         %0.4f Btu/lbm\n",
   delta_s);
printf("Work done by the piston, expanding against
   the atmosphere is        %0.2f Btu/lbm",w)
```

Scilab code Exa 2.2 Work done in reversible adiabatic steady flow steam turbine

```
clear;
//clc();

// Example 2.2
// Page: 28
printf("Example-2.2  Page no.-28\n\n");

//***Data***//
T_in = 600;//[F] Input steam temperature
P_in = 200;//[psia] Input steam pressure
P_exit = 50;//[psia]

// Because this is a steady-state, steady-flow
   process, we use
// (work per pound) = W/m = -( h_in - h_out )

// From the steam table we can read the the inlet
   enthalpy and entropy as
h_in = 1322.1;//[Btu/lbm]
```

```
s_in = 1.6767;//[Btu/(lb*R)]

// Now, we need the value of h_out

// For a reversible adiabatic steady-state, steady-
    flow process, we have
// sum(s*m_in_minus_out) = ( s_in - s_out ) = 0

// Which indicates that inlet and outlet entropies
    are same
// We can find the outlet temperature by finding the
     value of the temperature in the steam table
// For which the inlet entropy at 50 psia is the
    same as the inlet entropy, 1.6767 Btu/(lb*R).
// By linear interpolation in the table we find
T_in = 307.1;//[R]

// and by the linear interpolation in the same table
     we find that
h_out = 1188.1;//[Btu/lb]

// Thus, we find
W_per_pound = -(h_in - h_out);//[Btu/lb]

printf(" The work output of the turbine of steam is
    %0.1f Btu/lb",-W_per_pound);
```

Scilab code Exa 2.3 Estimation of Compressibility factor

```
clear;
//clc();

// Example 2.3
// Page: 38
printf("Example-2.3  Page no.-38\n\n");
```

```

//***Data***//
T = 500;//[F]
P = 680;//[psi]
// It is reported in the book in the table A.1(page
   417) that for water
// We know that T_r = T/T_c and P_r = P/P_c, so
T_c = 647.1*1.8//[R]
P_c = 220.55*14.51;//[psia]
w = 0.345;
T_r = (T+459.67)/T_c;
P_r = P/P_c;
z_0 = 1+P_r/T_r*(0.083-0.422/T_r^(1.6));
z_1 = P_r/T_r*(0.139-0.172/T_r^(4.2));
z = z_0+w*z_1;
printf("The compressibility factor of steam at the
   given state is %0.3f",z);
// Based on the steam table (which may be considered
    as reliable as the experimental data) the value
   of z is 0.804.
```

Chapter 3

The Simplest Phase Equilibrium Examples and Some Simple Estimating Rules

Scilab code Exa 3.1 Mole fraction of water vapour in air

```
clear;
//clc();

// Example 3.1
// Page: 52
printf("Example-3.1   Page no.-52\n\n");


//***Data***//
T = 20;//[C]
P = 1;//[atm]
// From Raoult's law   y_i*P = x_i*p_i
// Rearranging
//y_i = x_i*p_i/P;
// Here we have ternary mixture of nitrogen, oxygen,
    and water. If we let the subscript i stand for
   water, we can say that
```

```
//  x_water = 1-x_N2-x_O2;
// but we know from experience that the mole
   fractions of dissolved N2 and O2 are quite small,
    so that we are safe in saying that
x_N2 = 0;
x_O2 = 0;
x_water = 1-x_N2-x_O2;
// From any steam table we may look up the value of
   the vapour pressure of water at 20C, finding
p_water = 0.023;//[atm]
// So
y_water = x_water*p_water/P;
printf("The mole fraction of water vapour in air in
   equilibrium with water is %f",y_water);
```

Scilab code Exa 3.2 Roults and Henrys Law

```
clear;
//clc();

// Example 3.2
// Page: 53
printf("Example-3.2  Page no.-53\n\n");

//***Data***//
T = 20;//[C]
P = 1;//[atm]
// From previous example i.e. example 3.1
y_water = 0.023;
//so that
//y = y_N2+y_O2
y = 1-y_water;
// The oxygen is 0.21 mole fraction of this mix, so
   that
y_O2 = y*0.21;
```

```
// It is reported in the book in table A.3 (page
   419) that Henry's law cinstant for oxygen in
   water at 20C is
H_O2 = 40100;//[atm]
// From Henry's law, we have
// y_i = x_i*H_i/P
// rearranging
// x_i = y_i*P/H_i
// so
x_O2 = y_O2*P/H_O2;
// By the same logic we find that
y_N2 = y*0.79;
//and Henry's law constant for nitrogen in water at
   20C is
H_N2 = 80400;//[atm]
// hence
x_N2 = y_N2*P/H_N2;
// Now expressing the dissolved oxygen concentration
    in terms of the volume of the oxygen at STP viz.
    taken as 1 atm and 20C
// c = (concentration of dissolved oxygen in
   equilibrium with air at 1 atm and 20C)
c = x_O2*998.2/18;//[(mole O2)/(L solution)]
// V = (volume of O2, STP)/(L solution)
V = c*24.06;//[(L O2, STP)/(L solution)]
V = V*1000;//[(ml O2, STP)/(L solution)]
printf("Concentration of oxygen dissolved in water
   at equilibrium is %f (mL O2, STP)/(L solution)",V
   );
```

Scilab code Exa 3.3 Composition of Air and Water

```
clear;
//clc();

```

```
// Example 3.3
// Page: 52
printf("Example-3.3  Page no.-56\n\n");


//***Data***//

P = 1.0;//[atm]
p_w = 0.023;//[atm] Vapor pressure of pure water
H_o = 40100;//[atm] Vapor pressure of pure oxygen
H_n = 80400;//[atm] Vapor pressure of pure nitrogen
// From Raoult's law, we have
// ( y_i*P ) = ( x_i*p_i )
// So
//For water
// ( y_w*P ) = ( x_w*p_i )
// For oxygen
//( y_o*P ) = ( x_o*p_i )
// And for nitrogen
//( y_n*P ) = ( x_n*p_i )

// Also
// ( y_w + y_o + y_n ) = 1
// ( x_w + x_o + x_n ) = 1

// In air, the mole fraction of nitrogen and oxygen
   are 0.79 and 0.21 respectively. So,
// y_o/y_n = 0.21/0.79

// We will take the help of matrix method to solve
   these six equations for six unknowns
A = [0.023 0 0 -1 0 0;0 40100 0 0 -1 0;0 0 80400 0 0
    -1;0 0 0 1 1 1;1 1 1 0 0 0;0 0 0 0 0.79 -0.21];
B = [0;0;0;1;1;0];
X = A^(-1)*B;

printf(" The composition in liquid and vapor phase
   are summarized in the following table:\n\n");
```

```
printf("     y_water         \t  %f\n",X(4));
printf("     y_oxygen        \t  %f\n",X(5));
printf("     y_nitrogen      \t  %f\n",X(6));
printf("     x_water         \t  %f\n",X(1));
printf("     x_oxygen        \t  %e\n",X(2));
printf("     x_nitrogen      \t  %e",X(3));
```

Scilab code Exa 3.4 Some Simple Applications of Raoults and Henrys laws

```
clear;
//clc();

// Example 3.4
// Page: 57
printf("Example-3.4  Page no.-57\n\n");


//***Data***//
T = 20;//[C]
x_b = 0.80;
x_t = 0.20;
// Here we calculate the vapour pressures of benzene
   and toluene at 20C using the Antoine equation
// log10(p) = A-B/(T+C)
// here pressure p is in torr and temperature T is
   in C
// From the reported table A.2 (page 418) in the
   book, the constant A,B,C in the above equation
   for benzene have the vaues as
A_b = 6.90565;
B_b = 1211.033;
C_b = 220.79;
// So, for benzene
p_b = 10^(A_b-B_b/(T+C_b));
// now from the reported table A.2 (page 418) in the
```

```
   book, the constant A,B,C in the above equation
   for toluene have the vaues as
A_t = 6.95334;
B_t = 1343.943;
C_t = 219.337;
// So, for toluene
p_t = 10^(A_t-B_t/(T+C_t));
// Now we can compute that for benzene
// y_b*P = x_b*p_b
// let y_b*P = p_1 , so
p_1 = x_b*p_b;
// and correspondingly for toluene
// y_t*P = x_t*p_t
// let y_t*P = p_2 , so
p_2 = x_t*p_t;
// Now adding these two values of benzene and
   toluene, we have
// y_b*P+y_t*P = (y_b+y_t)*P
// i.e.
// P = (p_1+p_2)/(y_b+y_t)
// But we know that (y_b+y_t) must be equal to one i
   .e.
y = 1.00;// y =(y_b+y_t) sum of the mole fractions
   of the benzene and toluene in the gaseous phase
// Hence total pressure is
P = (p_1+p_2)/y;
// Now the mole fraction of either species in the
   gaseous phase will be ratio of the partial
   pressure of the species to the total pressure
// so
y_b = x_b*p_b/P;
y_t = x_t*p_t/P;
printf(" Vapour pressure of the mixture in the
   gaseous phase is %f torr\n",P);
printf(" Mole fraction of the benzene in the vapour
   phase is    %f\n",y_b);
printf(" Mole fraction of the toluene in the vapour
   phase is    %f",y_t);
```

Scilab code Exa 3.5 Some Simple Applications of Raoults and Henrys laws

```
clear;
//clc();

// Example 3.5
// Page: 57
printf("Example-3.5  Page no.-57\n\n");


//***Data***//
T = 20;//[C]
x_benzene = 1.00;
p_i = 75.2;//[torr] vapour pressure of the benzene
P = 760;//[torr] Pressure of the atmosphere

// So
y_benzene = (x_benzene*p_i)/P;

printf(" Mole fraction of the benzene in air that is
    saturated with benzene is %0.1f",y_benzene);
```

Scilab code Exa 3.6 Some Simple Applications of Raoults and Henrys laws

```
clear;
//clc();

// Example 3.6
// Page: 58
printf("Example-3.6  Page no.-58\n\n");

```

```

//***Data***//

P = 760;//[mm Hg]
x_b = 0.8;// Mole fraction of benzene in liquid
    phase
x_t = 0.2;// Mole fraction of toluene in liquid
    phase

// We will take the help of trial and error method
    to solve this problem
// From the table A.2 ( page 418 ), Antoine equation
     constants for benzene are
A_b = 6.90565;
B_b = 1211.003;
C_b = 220.79;

// and that for the toluene are
A_t = 6.95334;
B_t = 1343.943;
C_t = 219.337;

T = 82;//[C]
err = 1

while err > 10^(-3)
    p_b = 10^(6.90565 - 1211.003/(T + 220.79));
    p_t = 10^(6.95334 - 1343.943/(T + 219.337));
    y_b = x_b*p_b/P;
    y_t = x_t*p_t/P;
    err = abs((y_b + y_t) - 1);
    T = T + 0.01;
end

printf(" The temperature at which the given benzene-
    toluene mixture will have vapor pressure of 1 atm
     is %0.3f deg C",T);
```

Scilab code Exa 3.7 Some Simple Applications of Raoults and Henrys laws

```
clear;
//clc();

// Example 3.7
// Page: 60
printf("Example-3.7  Page no.-60\n\n");


//***Data***//

V = 0.25;//[L] Volume of water
T_1 = 0;//[C] Initial temperature of water
T_2 = 20;//[C] Final temperature of water

// From the example 3.3 the mol fractions of oxygen
   and notrogen in water at temperature 20 deg C are
x_o = 5.12*10^(-6);// mole fraction of oxygen
x_n = 9.598*10^(-6);// mole fraction of nitrogen


// Now we will calculate the mole fraction of oxygen
    and nitrogen in water at 0 deg C in the same
   manner as in example 3.3
// From the table A.3( page 419), Henry's constant
   of oxygen and nitrogen are
H_o = 2.55*10^(4);//[atm]
H_n = 5.29*10^(4);//[atm]

// And vapor pressure of water at 0 deg C is
p_w = 0.006;//[atm]

// Now using the same set of equations as in example
```

```
    3.3, by changing only H_o, H_n and p_w and
   solving by matrix method we have

A = [0.006 0 0 -1 0 0;0 25500 0 0 -1 0;0 0 52900 0 0
    -1;0 0 0 1 1 1;1 1 1 0 0 0;0 0 0 0 0.79 -0.21];
B = [0;0;0;1;1;0];
X = A^(-1)*B;

// Here the mole fraction of oxygen and nitrogen in
   water will be X(2) and X(3) respectively
// So oxygen rejected is
M_o_rej = V*( X(2) - x_o )/0.018;//[mole] oxygen
// Now At STP volume of the rejected oxygen is given
    as
V_o = M_o_rej*24200;//[ml] oxygen

// And rejected nitrogen is
M_n_rej = V*( X(3) - x_n )/0.018;//[mole] nitrogen
// In terms of volume
V_n = M_n_rej*24200;//[ml]

printf(" At equilibrium at 20 deg C the rejected
   amount of oxygen will be   %0.2f ml\n",V_o);
printf(" At equilibrium at 20 deg C the rejected
   amount of nitrogen will be %0.2f ml\n",V_n);
printf(" And total amount of the air rejected from
   the water will be        %0.2f ml",(V_o + V_n));
```

Scilab code Exa 3.8 Some Simple Applications of Raoults and Henrys laws

```
clear;
//clc();

// Example 3.8
// Page: 61
```

```
printf("Example-3.8  Page no.-61\n\n");


//***Data***//

P_1 = 5;//[atm]
y_n = 0.79;// Mole fraction of nitrogen in
   atmosphere
P_2 = 1.0;//[atm]
M = 55;//[kg] Mass of the diver
x_w = 0.75;// Fraction of water in human body
T = 37;//[C] Body temperature of the diver

// At 37 deg temperature, the Henry's constant for
   N2 from the table A.3 ( page 419 ) by the linear
   interpolation is
H_n = 10.05*10^(4);//[atm]

// Now, moles of nitrogen rejected will be
// M_rej = (moles of body fluid)*( x_N2,5 atm - x_N2
   ,1 atm)
// So
M_rej = (M*1000*x_w/18)*( P_1*y_n/H_n - P_2*y_n/H_n)
   ;//[mol]
// At STP the volume of the rejected nitrogen will
   be
V_n = M_rej*24.2;//[L]

printf(" Amount of rejected nitrogen will be %0.2f
   Litre",V_n);
```

Chapter 4

Minimization of Gibbs Free energy

Scilab code Exa 4.1 `Gibbs free energy calculation`

```
clear;
//clc();

// Example 4.1
// Page: 67
printf("Example-4.1   Page no.-67\n\n");

//***Data***//
T = 671.7;//[R] Equilibrium temperature
m_steam = 1;//[lbm] Condensing amount of the steam
// Using values from the steam table [1], we find
   that
delta_h_condensation = -970.3//[Btu/lbm] Enthalpy
   change of the steam
delta_s_condensation = -1.4446;//[Btu/(lbm*R)]
   Entropy change of the steam

// Gibb's free energy change of the steam is
delta_g_condensation = delta_h_condensation - T*
```

```
    delta_s_condensation;//[Btu/lbm]

printf("Gibb''s free energy change of the steam is
    %0.1f Btu/lbm",delta_g_condensation);
```

Scilab code Exa 4.2 Gibbs free energy diagram for the graphite diamond system

```
clear;
//clc();

// Example 4.2
// Page: 77
printf("Example-4.2  Page no.-77\n\n");

//***Data***//

// let we denote graphite by 'g' and diamond by 'd'
// Gibb's free energies of graphite and diamond are
    given by
g_g = 0.00;//[kJ/mol]
g_d = 2.90;//[kJ/mol]

// Specific volumes of graphite and diamond are
    given by
v_g = 5.31*10^(-1);//[kJ/(mol*kbar)]
v_d = 3.42*10^(-1);//[kJ/(mol*kbar)]

// Now from the equation 4.32 ( page 74) given in
    the book, we have
// (dg/dP) = v , at constant temperature
// where 'v' is specific volume
// let us denote (dg/dP) by 'D' ,so
D_g = v_g;//[J/(mol*Pa)] For graphite
D_d = v_d;//[J/(mol*Pa)] For diamond

```

```
// Now we can take our plot from P = 0( =1 ),
   however, total pressure is 1 atm.
// If we consider specific volumes of the given
   species to be constant with changing the pressure
    then g-P curve will be a straight line
// So the equation of the line for graphite is
// g = D_g*P + g_g
// and that for diamond
// g = D_d*P + g_d

P = [0:1:30]';

plot2d(P,[ D_d*P+g_d D_g*P+g_g ],style=[color("
   darkgreen"),color("red")]);

xlabel("Pressure, P, kbar");
ylabel("Gibb''s free energy per mol, g, kJ/mol");

printf(" Gibb''s free energy-pressure diagram for
   graphite-diamond system at 25 degC is as shown in
    the graphic window. ");
hl=legend(['Diamond, slope = 0.342 (kJ/mol)/kbar';'
   Graphite, slope = 0.532 (kJ/mol)/kbar']);
```

Scilab code Exa 4.3 Gibbs free energy for chemical reactions

```
clear;
//clc();

// Example 4.3
// Page: 80
printf("Example-4.3  Page no.-80\n\n");

//***Data***//
// We have the system which consists of isobutane
```

```
   and normal butane and isomerisaation is taking
   place between them
// The equilibrium constant for this reaction is
   given by
// K = (mole fraction of isobutane)/(mole fraction
   of n-butane) = x_iso/x_normal

// For this reaction , at 25C,
K = 4.52;

// and
//  x_iso + x_normal = 1
// so
// K = x_iso/(1-x_iso)

// solving for x_iso
deff('[y]=f(x_iso)','y = x_iso/(1-x_iso)-K');
x_iso = fsolve(0,f);

printf(" Mole fraction of isobutane isomer in
   equilibrium is %0.2f",x_iso);
```

Chapter 5

Vapor Pressure The Clapeyron Equation And Single Pure Chemical Species Phase Equilibrium

Scilab code Exa 5.1 Application of Clapeyron Equation

```
clear;
//clc();

// Example 5.1
// Page: 89
printf("Example-5.1  Page no.-89\n\n");

//***Data***//
T=212;// [F]

//**********//
//From the steam table , we have
delta_h=970.3;//[Btu/lbm]
delta_v=26.78;//[ft^(3)/lbm] and

```

```
// changing the units
delta_h1=delta_h*778;//[ft*lbf/lbm]
delta_v1=delta_v*144;//[ft*in^(2)/lbm]
T=671.7;//[R]

// We have dP/dT = delta_h/(T*delta_v)
//Thus
dP_by_dT=delta_h1/(T*delta_v1);//[psi/R]

printf("The value of dP/dT is %f psi/R",dP_by_dT);
//Using the nearest adjacent steam table entries for
    vapour pressure, wee have
//dP_by_dT = delta_P_by_delta_T=(15.291-14.125)
    /(214-210)=0.2915 psi/R
```

Scilab code Exa 5.2 `Application of Clausius Clapeyron Equation`

```
clear;
//clc();

// Example 5.2
// Page: 90
printf("Example-5.2  Page no.-90\n\n");

//***Data***//
p_2=0.005;// [psia]
R=1.987/18;//[1/R]

//*******//

//From the steam tables at the tripple point, we
    find
T_1=460+32.018;//[R]
p_1=0.0887;//[psia]

```

```
//delta_h(solid to gas) = delta_h(sublimation) =
   1218.7;//[Btu/lbm]
delta_H=1218.7;//[Btu/lbm]

//Assuming that the enthalpy change of vaporization
   is independent of temperature (a fairly good
   approximation in this case)
//we start with Eq. 5.10 and rearrange:
//1/T_2 = 1/T_1-(log(p_2/p_1))*R/delta_H
//So

T_2=1/(1/T_1-(log(p_2/p_1))*R/delta_H);//[R]
//Changing the temperature in farenheit
T_2F=T_2-460;//[F]

printf("The temperature is %f R",T_2F);
//BY linear interpolation in the steam tables, one
   finds -23.8 F. Because of imprecision of linear
   interpolation, these values are approximately
   equal.
```

Scilab code Exa 5.3 Application of Clausius Clapeyron Equation

```
clear;
//clc();

// Example 5.3
// Page: 91
printf("Example-5.3  Page no.-91\n\n");

//***Data***//

T_3=1155.2;//[R]
T_2=652.9;//[R]
T_1=787.5;//[R]
```

```
p_2=10;//[psia]
p_1=100;//[psia]

//******//
//Here we can write Eq. 5.9 as reported in the book
    in the form most often seen.
// log(p)=A-B/T
//Where A and B are constants to be determined from
    the pair of T and p values above.

//we simply write
//log(10)=A-B/652.9;
//log(100)=A-B/787.5;
// We have to solve the above two simultaneous
    equations having two vaiables A and B.

M=[1 -1/652.9;1 -1/787.5];
C=[log(10);log(100)];
X=inv(M)*C;
A=X(1);
B=X(2);

// By straightforward algebra we find the values of
    A and B. Thus, for 1155.2 R we have
p_3=exp(A-B/T_3);

printf("Vapuor pressure of water at given
    temperature is %f psia\n\n",p_3);
// p_3=3499 psia.
printf("It has been reported in the book that from
    table 5.1 we see that the correct value is 3000
    psia. Thus, there is an error of 16%% in the
    predicted pressure.");
```

Scilab code Exa 5.4 `The Accentric Factor`

```
clear;
//clc();

// Example 5.4
// Page: 94
printf("Example-5.4  Page no.-94\n\n");

//***Data***//
// At Tr = 0.7, we read
Pr=0.023;
// and thus accentric factor is given by

w=-log10(0.023)-1;

printf("The accentric factor based on the given data
    is %f",w);
//It has been reported in the book that table A.1
   shows that the value based on the best data is
   0.645.
```

Scilab code Exa 5.5 `Estimation of NBP using Antoine equation`

```
clear;
//clc();

// Example 5.5
// Page: 94
printf("Example-5.5  Page no.-94\n\n");

//***Data***//
//From Antoine equation we have
// log(p) = A-B/(T+C)
//Solving above equation for T, we have
// T = B/(A-log(p))-C
//Inserting the values of the constants for the
```

```
   water which are reported in the given book in the
    table A.2 (page 419),
// and the value of 1.00 atm expressed in torr, we
   find that

A=7.96681;
B=1668.21;
C=228.0;
p=760;//[torr]

//Thus
T=B/(A-log10(p))-C;

printf("NBP of water using antoine equation and
   table A.2 is %f C",T);

//This does not prove the overall accuracy of the
   Antoine equation, but does show that whoever
   fitted the constants to the experimental data for
    water made them represent the NBP (100C) very
   well.
```

Scilab code Exa 5.6 Applying the Clapeyron equation to other kind of equilibrium

```
clear;
//clc();

// Example 5.6
// Page: 96
printf("Example-5.6  Page no.-96\n\n");

//***Data***//
T_2=-22;//[C]
// converting temperature in farenheit
T_2F=T_2*9/5+32;//[F]
```

```

//Expressing T_2 in Rankine
T_2R=460+T_2F;//[R]

//******//

//delta_h = delta_h(fusion)
delta_h=143.35*778;//[ft*lbf/lbm]

//delta_v = v_water-v_ice
delta_v=0.01602-0.01747;//[ft^(3)/lbm]

// changing the unit
delta_v1=delta_v*144;//[ft*in/lbm]

//and
T_1=460+32;//[R]
dP_by_dT=delta_h/(T_1*delta_v1);//[psi/R] at 32F
delta_T=T_2R-T_1;

//This gives the rigorously correct slope of the
   liquid-solid curve at 32F on a P-T diagram.
//Here we use P instead of p  because neither phase
   is a gas, so this is not a vapour pressure.
//If we further assume that the solid-liquid curve
   is a straight line, which is equivalent to
   assuming that delta_h/(T*deta_v)is a constant
   over the region of interest, then we can estimate
    the pressure at -22C = -7.6F by
// delta_P = integrate(dP_by_dT)*dT = (dP_by_dT)*
   delta_T
//So

delta_P=(dP_by_dT)*delta_T;//[psi]

// From this we can estimate the final pressure as
delta_P=delta_P+0.09;//[psi]

```

```
printf("Freezing preesure of water at given
   temperature is %f psi",delta_P);
// In this case, the experimental pressure is well
   known, because this temperature corresponds to
   the tripple point between liquid and water,
// ice I(the common variety), and ice III, a variety
    that does not exist at pressure below about
   30000 psia (see figure 1.10 in the book).
// The measured value is 30000 psia, which shows
   that our assumption of a straight line on a P-T
   plot (delta_h/(T*delta_v)=constant) is only
   approximately correct.
```

Chapter 6

Partial Molal Properties

Scilab code Exa 6.1 Tagent Slopes

```
clear;
//clc();

// Example 6.1
// Page: 108
printf("Example-6.1   Page no.-108\n\n");

//***Data***//
T = 20;//[C]
m_1 = 0;//[molal]
m_2 = 1;//[molal]
// The data given in the figure 6.2 , as reported in
    book, can be repersented with excellent accuracy
    by a simple data fitting equation
//V = 1.0019+0.054668*m-0.000418*m^(2);
// Where 'V' is( solution volume, liters per 1000g
   of water ) and 'm' is the molality of ethanol in
   water
//The partial molal volume is obtained by
   differentiating the expression of the 'V' with
   respect to 'm'
```

```
// v_ethanol = dV/dm = 0.054668-2*0.000418*m
// So that at zero molality
m = 0;//[molal]
// the partial molal volume is
v_1 = 0.054668-2*0.000418*m;//[L/mol]
// and at
m = 1;//[molal]
v_2 = 0.054668-2*0.000418*m;//[L/mol]
v_1 = v_1*1000;//[cm^(3)/mol]
v_2 = v_2*1000;//[cm^(3)/mol]
printf("Partial molal volume of ethanol in water at
    zero molality is  %f cm^(3)/mol\n",v_1);
printf(" Partial molal volume of ethanol in water at
     unity molality is %f cm^(3)/mol",v_2);
```

Scilab code Exa 6.2 Volume Change on Mixing

```
clear;
//clc();

// Example 6.2
// Page: 109
printf("Example-6.2  Page no.-109\n\n");

//***Data***//
n_eth = 1;//[mol]
W_water = 1;//[kg]
Temp = 20;//[C]
// For pure ethanol at 20C
v_ethanol = 58.4;//[cm^(3)/mol]
v_ethanol = v_ethanol/1000;//[L/mol]
v_water = 1.0019;//[L/1000g]
// Molality of ethanol in water is
m = n_eth/W_water;//[molal]
// We have the equation used in the previous example
```

```
   as
V_final_mix = 1.0019+0.054668*m-0.000418*m^(2);
// Where 'V' is( solution volume, liters per 1000g
   of water ) and 'm' is the molality of ethanol in
   water
// V is the final volume of the solution
// The volume expansion on moxing is
V_exp = V_final_mix-v_ethanol-v_water;//[L]
V_exp = V_exp*1000;//[cm^(3)]
printf("Volume change on mixing etanol and water is
   %0.3f cubic cm",V_exp);
// We see that there is a net contraction on mixing
   of the volume of the ethanol added.
```

Scilab code Exa 6.3 Volume change on mixing

```
clear;
//clc();

// Example 6.3
// Page: 109
printf("Example-6.3  Page no.-109\n\n");

//***Data***//
// All the data are same as in the previous example
// The equation 6.5 reported in the book is
// delta_V_mixing = V_solution _final-V_(solution
   and material to be mixaed) = integrate(v_i-v_i_0)
   dn
// Here the integrated average value of v_i over the
    molality range from 0 to 1 is
v_i_average = 0.05425;//[L/mol]
// and
v_i_0 = 0.0584;//[L/mol]
delta_n = 1.00;//[mol]
```

```
delta_V_mixing = (v_i_average-v_i_0)*delta_n;//[L]
delta_V_mixing = delta_V_mixing*1000;//[cm^(3)]
printf("Volume change on mixing etanol and water is
   %f cm^(3)",delta_V_mixing);
// Which is same as the solution in example 6.2
```

Scilab code Exa 6.4 Tangent Intercept Concept

```
clear;
//clc();

// Example 6.4
// Page: 113
printf("Example-6.4   Page no.-113\n\n");

//***Data***//
m = 1;//[molal] Molality of the solution with
   respect to ethanol
M_water = 18;//[g/mol] molecular weight of water

// First we convert molality to mole fraction
x_ethanol = m/(m + 1000/M_water);

// For the low range of data point on figure 6.5(
   page 112), we can fit an equation
// (Specific volume ) = 0.018032 + 0.037002*
   x_ethanol - 0.039593*x_ethanol^(2) + 0.21787*
   x_ethanol^(3)
// This is applicable for (0 < x_ethanol < 0.04 ),
   which is the case we have

// So
v_tan = 0.018032 + 0.037002*x_ethanol - 0.039593*
   x_ethanol^(2) + 0.21787*x_ethanol^(3);//[L/mol]

```

```
// Now we will find the derivative of the specific
   volume with respect to x_ethanol at the known
   point x_ethanol
// (dv/dx_ethanol) =  0.037002 - 2*0.039593*
   x_ethanol + 3*0.21787*x_ethanol^(2)
// Hence
v_derv_tan = 0.037002 - 2*0.039593*x_ethanol +
   3*0.21787*x_ethanol^(2);//[L/mol]

// By simple geometry from the figure 6.6(page 113)
   of the book we find
// a = v_tan + (1-x_tan)*(dv/dx_1)_tan
// b = v_tan - x_tan*(dv/dx_1)_tan

// We have a = v_ethanol and b = v_water
x_tan = x_ethanol;
// So
v_ethanol = v_tan + (1-x_tan)*(v_derv_tan);//[L/mol]
v_water = v_tan - x_tan*(v_derv_tan);//[L/mol]

printf(" Partial molar volume of the ethanol in the
   given solution is %f L/mol\n",v_ethanol);
printf(" Partial molar volume of the water in the
   given solution is %f L/mol",v_water);
```

Scilab code Exa 6.5 Tangent Intercept concept

```
clear;
//clc();

// Example 6.5
// Page: 115
printf("Example-6.5  Page no.-115\n\n");

//***Data***//
```

```
printf("This is a theoratical question and there are
    no any numerical components. For the derivation ,
    refer to page no 115 of the book.");
```

Scilab code Exa 6.6 Idea of Tangent Intercept

```
clear;
//clc();

// Example 6.6
// Page: 115
printf("Example-6.6   Page no.-115\n\n");

//***Data***//
printf("This is a theoratical question and there are
    no any numerical components. Refer to page no
   115 of the book.");
```

Scilab code Exa 6.7 Partial Mass Properties

```
clear;
//clc();

// Example 6.7
// Page: 117
printf("Example-6.7   Page no.-117\n\n");

//***Data***//
x_sulph = 0.6;
x_water = 0.4;
Temp = 200;//[F]
// In the given figure 6.8 in the book, drawing the
   tangent to the 200F curve at 60 wt% H2SO4, we
   find that it intersects the 0%(pure water) axis
   at      25 Btu/lbm, and the 100% H2SO4 axis at
   -100Btu/lbm. i.e.
h_water_per_pound = 25;//[Btu/lbm]
h_sulph_per_pound = -100;//[Btu/lbm]
// also molecular weight of water and sulphuric acid
    are
M_water = 18;//[lbm/lbmol]
M_sulph = 98;//[lbm/lbmol]
// Using equation 6.20 given in the book we have
h_water = h_water_per_pound*M_water;//[Btu/lbmol]
h_sulph = h_sulph_per_pound*M_sulph;//[Btu/lbmol]
printf("Partial molar enthalpy of water in the
   mixture is  %f Btu/lbmol\n",h_water);
printf(" Partial molar enthalpy of H2SO4 in the
   mixture is %f Btu/lbmol",h_sulph);
```

Scilab code Exa 6.8 Differential Heat of Mixing

```
clear;
//clc();

// Example 6.8
// Page: 119
printf("Example-6.8  Page no.-119\n\n");

//***Data***//
x_sulph = 0.6;
x_water = 0.4;
M_i = 18;//[lbm/lbmol]
Temp = 200;//[F]
// From Equation 6.11 as given in the book, we have
// dQ/dm_in = h_i-h_in
// where h_i is partial molal enthalpy which is
   taken from the example 6.7 and h_in is the pure
   species molar enthalpy which is read from the
   figure 6.8.
// So at 200F we have
h_i = 25;//[Btu/lbm]
h_in = 168;//[Btu/lbm]
// hence
dQ_by_dm_in = h_i-h_in;;//[Btu/lbm]
// Now
dQ_by_dn_in = M_i*dQ_by_dm_in;//[Btu/lbmol]
printf("The amount of heat removed to keep the
   temperature constant is %f Btu/lbm of water added
   ",dQ_by_dm_in);
// The negative sign shows that this mixing is
   exothermic; we must remove 143 Btu/lbm of water
   added.
```

Scilab code Exa 6.9 Integral Heat of Mixing

```
clear;
//clc();

// Example 6.9
// Page: 119
printf("Example-6.9  Page no.-119\n\n");

//***Data***//
m_sulph = 0.6;
m_water = 0.4;
m = m_sulph+m_water;
Temp = 200;//[F]
// Here at 200F we can read the solution enthalpy
   h_solution and pure H2SO4 enthalpy h_sulph such
   that
h_solution = -50;//[Btu/lbm]
h_sulph = 53;//[Btu/lbm]
```

```
// By energy balance, using h_0_water from example
   6.7 in the book i.e.
h_0_water = 168;//[Btu/lbm]
// We find
delta_Q = m*h_solution-(m_sulph*h_sulph+m_water*
   h_0_water);//[Btu]
printf("The amount of heat added or removed is %f
   Btu\n\n",delta_Q);
// We must remove the given amount of to hold the
   temperature constant.
printf("However the book has some mistake in
   calculation and reporting -172 Btu")
```

Scilab code Exa 6.10 Integral Heat of Mixing

```
clear;
//clc();

// Example 6.10
// Page: 120
printf("Example-6.10  Page no.-120\n\n");

//***Data***//
x_sulph = 0.6;
x_water = 0.4;
Temp = 200;//[F]
// At the 200F we have
h_water = 25;//[Btu/lbm]
h_sulph = -100;//[Btu/lbm]
// From equation 6.16 (as reporated in the book),
   rewritten for masses instead of moles we have
h_solution = h_water*x_water+h_sulph*x_sulph;// [Btu
   /lbm]
printf("Enthalpy of the solution is %f Btu/lbm",
   h_solution);
```

Scilab code Exa 6.11 Application of Gibbs Duhem equation

```
clear;
//clc();

// Example 6.11
// Page: 121
printf("Example-6.11  Page no.-121\n\n");

//***Data***//
x_b = 0;
x_a = 1;
// We have
//dv_a/dx_a = 3*x_b^(2)+2*x_b
// We have the equation
// dv_b/dx_a = -(dv_a/dx_a)/(x_b/x_a)
// So
// dv_b/dx_a = -(x_a/x_b)*(3*x_b^(2)+2*x_b)
dv_b_by_dx_a = x_a*(-3*x_b-2);
printf("Value of the dv_b/dx_a at x_b =0 is %0.0f",
    dv_b_by_dx_a);
```

Scilab code Exa 6.12 Application of the gibbs Duhem equation

```
clear;
//clc();

// Example 6.12
// Page: 122
printf("Example-6.12  Page no.-122\n\n");

```

```
//***Data***//
x_b = 0;
x_a = 1;
// We have
//dv_a/dx_a = 3*x_b^(2)+2*x_b+1
// We have the equation
// dv_b/dx_a = -(dv_a/dx_a)/(x_b/x_a)
// So
// dv_b/dx_a = -(x_a/x_b)*(3*x_b^(2)+2*x_b+1)
//dv_b_by_dx_a = -x_a*(3*x_b+2+1)/x_b;
printf("Value of the dv_b/dx_a at x_b = 0 is minus
   infinity");
```

Chapter 7

Fugacity Ideal Solutions Activity Activity Coefficient

Scilab code Exa 7.1 The fugacity of pure gases

```
clear;
//clc();

// Example 7.1
// Page: 134
printf("Example-7.1  Page no.-134\n\n");

//***Data***//
T = 220+459.67;//[R] Temperature in Rankine
P = 500;//[psia] Pressure
R = 10.73;//[(psi*ft^(3)/(lbmol*R))] Gas constant

// We will follow the method 'a' as the book has
    given the multiple methods to solve this problem
// From  the equation 7.10 given in the book(page
    132), we have
// (f/P) = exp((-1/(R*T))*intgrate(a*dp)) , with
    intgration limits from zero to 'P'
// Where 'a' is known as volume residual
// Let us say , I = intgrate(a*dp)

// From the table 7.A(page 134) given in the book,
   the average value of alpha(a) is
a = 4.256;//[ft^(3)/lbmol]
// so
I = integrate('a*p^(0)','p',0,P);

// Now
f = P*exp((-1/(R*T))*I);//[psia]
printf("Fugacity of propane gas at the given
   condition is %f psia",f);
```

Scilab code Exa 7.2 `Compressibility factor and volume residual`

```
clear;
//clc();

// Example 7.2
// Page: 138
printf("Example-7.2  Page no.-138\n\n");

//***Data***//
T = 100 + 460;//[R] Temperature of the system in
   Rankine
P = 1;// [psia]
R = 10.73;//[(psi*ft^(3)/(lbmol*R))] Gas constant

// From the steam table, the specific volume of the
   water at 101.7 F, which is nearly equal to 100 F,
    and 1 psia is
v = 0.016136*18;//[ft^(3)/lbmol]
z = (P*v)/(R*T);

// and volume residual is given by
```

```
a = ((R*T)/P)*(1-z);//[ft^(3)/lbmol]

printf(" Compresssibility factor the liquid water at
    the given condition is %f\n ",z);
printf("Volume residual for the liquid water at the
   given condition is    %0.1f cubic feet/lbmol",a)
```

Scilab code Exa 7.3 Fugacity of pure liquid

```
clear;
//clc();

// Example 7.3
// Page: 138
printf("Example-7.3  Page no.-138\n\n");

//***Data***//

T = 100+460;//[R] Temperature
P = 1000;//[psia] Pressure
R = 10.73;//[(psi*ft^(3)/(lbmol*R))] Gas constant

// From the figure 7.3(page 138) we see that as P
   tends to zero, (f/P) tends to 1, so f tends to 0.
    Therefore, f_a tends to zero also in the diagram
// fugacity at point b is calculated by the equation
// (f/P)_b = exp((-1/(R*T))*integrate(a*dp)), with
   integration limit of p, 0 and P = 0.9503
// We have
f_b = 0.95;//[psia]

// We also can write
f_c = f_b;//[psia]

// To find the value of f_d, we use the equation
// integrate(d(logf))_T = integrate((v/(R*T))*dp)_T
    ..........................................(1)

// here 'v' is practically constant(for a liquid),
   and
v = 0.016136*18;//[ft^(3)/lbmol]

// and from the figure 7.3, we have
P_d = 1000;//[psia]
P_c = 1;//[psia]

// integrating the left hand side of the equation
   with the integration limits f_c and f_d and
   solving, we have
f_d = f_c*exp((v/(R*T))*integrate('p^(0)','p',P_c,
   P_d));

printf("Fugacity of the pure liquid water at the
   given condition is %0.1f psia",f_d);
```

Scilab code Exa 7.4 `Activity and activity coefficient`

```
clear;
//clc();

// Example 7.4
// Page: 145
printf("Example-7.4  Page no.-145\n\n");

//***Data***//

T = 78.15;//[C]
P = 1.0;//[atm]
// Here we name ethanol as the species 'a', and
   water as the species 'b', and name the vapor as
   phase 1 and the liquid as the phase 2.
// Thus vapor pressures of the pure species at the
   given temperature are
p_a_0 = 0.993;//[atm] Pure ethanol vapor pressure at
    78.15C
p_b_0 = 0.434;//[atm] Pure water vapor pressure at
   78.15C

// Also composition of the azeotrope is
x_a = 0.8943;// Amount of ethanol in the liquid
   phase
x_b = 0.1057;// Amount of water in liquid phase

// Also, for an azeotrope mixture
y_a = x_a;// Amount of ethanol in vapor phase
y_b = x_b;// Amount of water in the vapor phase

// For ideal gas , fugacity is equal to the total
   pressure of the system, i.e.
// f_i_0 = P  , (where P is the system pressure)
// For pure liquid system, fugacity of a species is
   independent of the total pressure of the system
   and is equal to the pure species vapor pressure
   at  this temprature, i.e.
// f_i_0 = p_i

// Now, fugacity of each species in gaseous phase
   and liquid phase will be equal
// so, writing the expression for both liquid and
   gas phase fugacity and equatinh them, we have
// f_a_2 = f_a_1 = (y*Y*P)_a_1 = (x*Y*p)_a_2
   ........................................(1)
// f_b_2 = f_b_1 = (y*Y*P)_b_1 = (x*Y*p)_b_2
   ........................................(2)

// We observe that this system has four values of 'Y
   ', one for each of the two species in each of two
    phases.
```

```
// Mixtures of the ideal gases are all ideal
   solutions and the value of 'Y' for all the
   species in ideal gas phase are unity, so for
   above two equations
Y_a_1 = 1.0;
Y_b_1 = 1.0;

// Now putting the values these gaseous phase 'Y's
   in their respective equations 1 and 2, and
   solving for the liquid phase 'Y's, we have
Y_a_2 = ((y_a*P)/(x_a*p_a_0));
Y_b_2 = ((y_b*P)/(x_b*p_b_0));

// From equations 1 and 2, the fugacity of each
   species in each phase is given by
f_a_1 = (y_a*Y_a_1*P);//[atm]
f_b_1 = (y_b*Y_b_1*P);//[atm]
// and from the definition we have
f_a_2 = f_a_1;//[atm]
f_b_2 = f_b_1;//[atm]

// As we have defined above about the pure species
   fugacity , so
// For vapor phase
f_a_1_0 = P;//[atm]
f_b_1_0 = P;//[atm]

// For liquid phase
f_a_2_0 = p_a_0;//[atm]
f_b_2_0 = p_b_0;//[atm]

printf(" The results are summarized in the following
    table:\n\n \tPhase\t\t\t\t Etahnol,i=a\t\t\t\t
   Water,i=b\n\n");
printf(" \tVAPOR, PHASE 1\n");
printf("  \t  f_i_1 , atm   \t\t\t %f \t\t\t\t %f\n",
   f_a_1,f_b_1);
printf("  \t  f_i_1_0 , atm \t\t\t %f \t\t\t\t %f\n",
    f_a_1_0,f_b_1_0);
printf("  \t  Y_i_1(assumed)  \t\t %f \t\t\t\t %f\n\n",Y_a_1,Y_b_1);
printf(" \tLIQUID, PHASE 2\n");
printf("  \t  f_i_2 , atm   \t\t\t %f \t\t\t\t %f\n",
    f_a_2,f_b_2);
printf("  \t  f_i_2_0 , atm \t\t\t %f \t\t\t\t %f\n",
    f_a_2_0,f_b_2_0);
printf("  \t  Y_i_2(assumed)  \t\t %f \t\t\t\t %f\n"
    ,Y_a_2,Y_b_2);
```

Scilab code Exa 7.5 Fugacities from gas PvT data

```
clear;
//clc();

// Example 7.5
// Page: 149
printf("Example-7.5  Page no.-149\n\n");

//***Data***//

T = 220+460;//[R] Temperature in rankine
P = 1000;//[psia] Pressure
y_methane = 0.784; // Mol fraction of methane in the
     given mixture
y_butane = (1-y_methane);// Mol fraction of n-butane
     in the given mixture
R = 10.73;//[(psia*ft^(3)/(lbmol*R))] gas constant

// In this problem, we need the partial molar volume
     residual.
// We find its value at 100 psia by plotting the
    volume resduals at 100 psia as a function of mole
     fraction, as shown in figure 7.9( page 150 )
```

```
// drawing the tangent to the data points at x_methane = 0.784 and reading its intercept on the 100 mol% methane axis as 0.6 ft^(3)/lbmol
// similarily volume residual is determined for all other pressures and plot them vs pressure, as shown in Figure 7.10 (page 151).
// From this plot we find the integral we need by numerical integration ( trapazoid rule ) as 290 ft ^(3)/lbmol.

// Thus, for methane
// f_i/(P*y_i) = exp((-1/(R*T))*integrate(a_i*dp)) with integral limits 0 to P = 1000 psia
// Let I = intefrate(a_i*dp))  and  J = f_i/(P*y_i), so
Im = 290;//[ft^(3)/lbmol]

// and
Jm = exp((-1/(R*T))*Im);

// hence
f_methane = Jm*P*y_methane;//[psia] fugacity of methane

// doing the same process for butane, we find
Ib = 5859;//[ft^(3)/lbmol]
// so, for butane we find
Jb = exp((-1/(R*T))*Ib);
// hence
f_butane = Jb*P*y_butane;//[psia] fugacity of butane

printf(" Fugacity of the methane in the gaseous mixture is %0.0f psia\n",f_methane);
printf(" Fugacity of the butane in the gaseous mixture is  %0.1f psia",f_butane);
```

Scilab code Exa 7.6 Fugacities from gas PvT data

```
clear;
//clc();

// Example 7.6
// Page: 153
printf("Example-7.6   Page no.-153\n\n");

//***Data***//

T = 220+460;//[R] Temperature in rankine
P = 1000;//[psia] Pressure
x_methane = 0.784; // Mol fraction of methane in the
    given mixture
x_butane = (1-x_methane);// Mol fraction of n-butane
    in the given mixture

// From the example 7.5, we found directly from the
   PvT data that for methane
// (f_i/(P*x_i)) = 0.961 = (v_i*Y_i) = phi_cap_i
// So, we can write that
v_i_into_Y_i = 0.961;
phi_cap_i = 0.961;

// From Starling's tables of hydrocarbon properties
   we read that for pure methane at this T and P,
// (F_i/P) = v_i = phi_i , from which it follows
v_i = 0.954;
phi_i = v_i;
Y_i = phi_cap_i/v_i;

printf(" The value of v_i is %f\n",v_i);
printf(" The value of Y_i is %f\n",Y_i);
printf(" The value of phi_cap_i is %f",phi_cap_i);
```

Scilab code Exa 7.7 Fugacities from an EOS for gas mixtures

```
clear;
//clc();

// Example 7.7
// Page: 154
printf("Example-7.7  Page no.-154\n\n");

//***Data***//

T_r = 0.889;
P_r = 1.815;

// Using the properties of n-butane from appendix A
   .1 and the equation 7.W, we find that
// (f/P) = v = phi = exp((P_r/T_r)*f(T_r,w))
// Say,    f(T_r,w) = f_f
f_f = -0.48553;
// so
v = exp((P_r/T_r)*f_f);
phi = v;
printf(" The value of v=phi for n-butane at given
   condition is %f",v);
```

Chapter 8

Vapor Liquid Equilibrium VLE at Low Pressures

Scilab code Exa 8.1 Calculation of K factors

```
clear;
//clc();

// Example 8.1
// Page: 163
printf("Example-8.1   Page no.-163\n\n");

//***Data***//
x_acetone = 0.05;// Mole fraction of Acetone in
   liquid
x_water = (1-x_acetone);
// Using the values from table 8.1(page 162) as
   reported in the book we have
y_acetone = 0.6381;// Mole fraction of Acetone in
   vapour
y_water = (1-y_acetone);
// We know that
// K_i = y_i/x_i
// So 'K' factors are
```

```
K_acetone = y_acetone/x_acetone;
K_water = y_water/x_water;
// and relative volatility is
a = K_acetone/K_water;
printf("The K factor of acetone is %f\n",K_acetone);
printf(" The K factor of water is    %f\n",K_water);
printf(" The relative volatility is  %f",a)
```

Scilab code Exa 8.2 Liquid phase activity coefficient

```
clear;
//clc();

// Example 8.2
// Page: 165
printf("Example-8.2  Page no.-165\n\n");

//***Data***//
P = 1;//[atm]
Temp = 74.8;//[C]
// Here we need to know the vapour pressure p_i
   corresponddng ti the temperatures of each of the
    values in the table.
// We can estimate them using Antoine equation  by
   the help of the values given in table A.2(page
   418) in the book
// log10(p_i) = A-B/(T+C)
// for acetone the constants are given as
A_a = 7.02447;
B_a = 1161;
C_a = 224;
// So p_acetone is given by
p_acetone = 10^(A_a-B_a/(Temp+C_a));//[mmHg]
// similarily for water the constants are given as
A_w = 7.94917;
```

```
B_w = 1657.462;
C_w = 227.02;
// So p_water is given by
p_water = 10^(A_w-B_w/(Temp+C_w));//[mmHg]
// expressing the pressures in atm
p_acetone = p_acetone/760;//[atm]
p_water = p_water/760;//[atm]
// Now from table 8.1 given the book
y_acetone = 0.6381;
x_acetone = 0.05;
y_water = (1-y_acetone);
x_water =(1-x_acetone);
// Hence the liquid-phase activity coefficients for
   acetone and water are given as
Y_acetone = y_acetone*P/(x_acetone*p_acetone);
//and
Y_water = y_water*P/(x_water*p_water);
printf("Liquid-phase activity coefficient for
   acetone is %f\n",Y_acetone);
printf(" Liquid-phase activity coefficient for water
    is   %f\n",Y_water);
```

Scilab code Exa 8.3 Non ideal solution behaviour

```
clear;
//clc();

// Example 8.3
// Page: 167
printf("Example-8.3  Page no.-167\n\n");

//***Data***//

x_a = 0.05;// mole fraction of acetone in liquid
   phase
```

```
x_w = (1-x_a);// mole fraction of the water in the
    liquid phase
P = 1.00;//[atm] Total pressure in vapor phase

// Let us assume that the solution is ideal
// We will take the help of trial and error methad
    and find a temperature at which sum of the
    computed ideal solution vapor phase mole fraction
     is 1.00
// For our first try let the temperatute is
T_1 = 80;//[C]
// Now from Table A.2( page 418), the Antoine
    equation constant for acetone are
A_a = 7.02447;
B_a = 1161;
C_a = 224;
// and that for water
A_w = 7.94917;
B_w = 1657.462;
C_w = 227.02;

// Now from Antoine equation
// log10(p) = A - B/(T+C)
// So, vapor pressure for pure acetone at 80 C (in
    atm)is
p_a_1 = (1/760)*10^(A_a - B_a/(T_1+C_a));//[atm]
// and that of water is
p_w_1 = (1/760)*10^(A_w - B_w/(T_1+C_w));//[atm]

// Now from Raoult's law
// y_i*P = x_i*p_i
// so, vapor phase composition at this temperature
    is
y_a_1 = (x_a*p_a_1)/P;
y_w_1 = (x_w*p_w_1)/P;

// Sum of these two compostion is
y_1 = (y_a_1 + y_w_1);
```

```
// Since, y_1 is not equal to 1.00, so assumed
   temperature is wrong

// Now we will assume our temperature as
T_2 = 96.4060;//[C]

// Again, from Antoine equation
// log10(p) = A - B/(T+C)
// So, vapor pressure for pure acetone at 80 C (in
   atm)is
p_a_2 = (1/760)*10^(A_a - B_a/(T_2+C_a));//[atm]
// and that of water is
p_w_2 = (1/760)*10^(A_w - B_w/(T_2+C_w));//[atm]

// Now from Raoult's law
// y_i*P = x_i*p_i
// so, vapor phase composition at this temperature
   is
y_a_2 = (x_a*p_a_2)/P;
y_w_2 = (x_w*p_w_2)/P;

// Sum of these two compostion is
y_2 = (y_a_2 + y_w_2);
// Value of y_2 is equal to 1.00, so our assumption
   is right
// These are the values when the solution would
   behave as ideal, but this is not the actual scene
// The experimental values of the boiling point and
   vapor phase composition are listed in the table
   8.1(page 162) given in book, which are
T_e = 74.8;//[C] Boiling temperature
y_a_e = 0.6381;// vapor phase composition of acetone

printf(" Comparison of experimental values to those
   computed by the ideal solution assumption,
   x_acetone = 0.05 and P = 1.00 atm\n\n");
printf(" \t\t\t Experimental Values from Table 8.1 \
   t\t\t\tValues calculated assuming idea solution\n
   \n");
printf(" Equilibrium(boiling) \t\t%0.1f \t\t\t\t\t\t
   \t\t\t %0.1f \n temperature T deg C\n\n",T_e,T_2)
   ;
printf(" Mole fraction acetone \t\t%f \t\t\t\t\t\t
   \t %f \n in the vapor phase (y_a)",y_a_e,y_a_2);
```

Scilab code Exa 8.4 Two liquid phase

```
clear;
//clc();

// Example 8.4
// Page: 177
printf("Example-8.4  Page no.-177\n\n");

//***Data***//
n_water = 80;//[mol]
n_butanol = 20;//[mol]
n_total = n_water+n_butanol;//[mol]
// Here from the figure 8.12 given in the book we
   can find the mole fraction of the water in each
   phase
// Let x_feed be the moles of water(species a) fed/
   total moles fed.
x_feed = 0.8;
x_a_1 = 0.65;
x_a_2 = 0.98;
// By material balence for water
// n_total*x_feed = n_1*x_a_1+n_2*x_a_2 ,
// here n_1 and n_2 are no. of mole in each phase
// So (n_1+n_2) = n_total
// Thus
// n_total*x_feed = n_1*x_a_1+(n_total-n_1)*x_a_2
// solving further
// n_1/n_total = (x_feed-x_a_2)/(x_a_1-x_a_2)
// and hence
n_1 = (x_feed-x_a_2)/(x_a_1-x_a_2)*n_total;//[mol]
n_2 = (n_total-n_1);//[mol]
// so
n_a_1 = 0.65*n_1;//[mol]
// and
n_a_2 = 0.98*n_2;//[mol]
printf(" Total moles of water present in the first
    phase is  %f mol\n",n_a_1);
printf(" Total moles of water present in the second
    phase is %f mol",n_a_2);
```

Scilab code Exa 8.5 `Two liquid phase`

```
clear;
//clc();

// Example 8.5
// Page: 178
printf("Example-8.5  Page no.-178\n\n");

//***Data***//
// At equilibrium on dew-point the conditions are
    given as
P = 1;//[atm]
y_water = 0.60;
// From the figure 8.12d, if we start at 130C and 60
     mol% water and cool.
//  We meet he dew-point line at 99C, and at the
    same temperature the bubble-point curve shows
x_water_1 = 0.22;
// Doing the same procedure with y_water = 0.90, we
    get the dew-point at the rightmost side at 98C
// In this case, the bubble-point line is the
```

```
    steeply sloping one at hte right , from wich we
    read
x_water_2 = 0.99;
// Similarily with y_water = 0.73 , we get that two
    dew-point meet at 92C.
// Vapour of this composition is in equilibrium with
     both liquid phases , as sketched in hte figure
    8.12d.
// Vapour with any other composition is in
    equilibrium with only one liquid i.e.
// if y_water < 0.73 , then
// x_water <0.65
// and if y_water > 0.73 , then
// x_water >0.98
printf(" The equilibrium amount of water in liquid
    at bubble-point for the dew-point composition
    y_water=60 mol%% is %f mol%% water\n",x_water_1);
printf(" The equilibrium amount of water in liquid
    at bubble-point for the dew-point composition
    y_water=90 mol%% is %f mol%% water",x_water_2);
```

Scilab code Exa 8.6 Activity coefficient of water and n butanol

```
clear;
//clc();

// Example 8.6
// Page: 178
printf("Example-8.6   Page no.-178\n\n");

//***Data***//

P = 1.00;//[atm] assumed total vapor pressure
// In psia unit
P1 = 14.7;//[psia]
```

```
// From the figure 8.12d ( page 176 ) in book, the
   mole fractions of water in all the three phases
   and temperature are known and given as
x_1_water = 0.65;
x_1_butanol = (1-x_1_water);
x_2_water = 0.98;
x_2_butanol = (1-x_2_water);
y_water = 0.73;
y_butanol = (1-y_water);
T = 92;//[C]

// At this temperature we have to estimate the vapor
    pressure of pure water and n-butanol with the
   help of Antoine equation
// log10(p) = A - B/(T+C)
// From Table A.2( page 418), the Antoine equation
   constants for water are
A_w = 7.94917;
B_w = 1657.462;
C_w = 227.02;

// and that for n-butanol are
A_b = 7.838;
B_b = 1558.190;
C_b = 196.881;

// Thus vapor pressure of water and n-butanol are
   respectively
p_water = (14.7/760)*10^(A_w - B_w/(T+C_w));
p_butanol = (14.7/760)*10^(A_b - B_b/(T+C_b));

// fugacity of the water and n-butanol are given as
// f_i = (y*Y*P)_i
// Where Y is the gas phase activity coefficient and
    its value is 1.00 in ideal gas mixture, so
f_water = (y_water*P);
f_butanol = (y_butanol*P);
// The fugacity will be same in both the phase 1 and
```

```
    2

// Now, liquid-phase activity coefficients are given
    by
// Y_i = (y_i*P)/(x_i*p_i)
// so,
Y_water_1 = (y_water*P1)/(x_1_water*p_water);
Y_butanol_1 = (y_butanol*P1)/(x_1_butanol*p_butanol)
   ;

// For phase 2
Y_water_2 = (y_water*P1)/(x_2_water*p_water);
Y_butanol_2 = (y_butanol*P1)/(x_2_butanol*p_butanol)
   ;

printf(" Four activity coefficients and fufacities
   are shown in the following table:\n\n");
printf("\t Phase \t x_water \t f_water(atm) \t
   Y_water \t x_butanol \t f_butanol(atm) \t
   Y_butanol\n\n");
printf(" \t   1 \t %f \t %f \t %f \t %f \t %f \t\t
   %f \n",x_1_water,f_water,Y_water_1,x_1_butanol,
   f_butanol,Y_butanol_1);
printf(" \t   2 \t %f \t %f \t %0.2f \t\t %f \t %f \
   t\t %f ",x_2_water,f_water,Y_water_2,x_2_butanol,
   f_butanol,Y_butanol_2);
```

Scilab code Exa 8.7 Zero Solubility and Steam distillatation

```
clear;
//clc();

// Example 8.7
// Page: 179
printf("Example-8.7  Page no.-179\n\n");
```

```

//***Data***//

P = 1;//[atm] Total pressure in the vapor phase

// Since the two liquids are not soluble in each
   other so Raoult's law will apply separately for
   these two phases.
// From Raoult's law we have
// (y_i*P) = (x_i*p_i)
// Here two phases are in pure stages so x_i=1 for
   both phases
// So
// y_i = (p_i/P)
// Writing this equation for each species, adding
   the equations, and solving for P, we find
// P = summation( y_i*P ) = summation( p_i/P*P ) =
   summation(p_i)

// The total pressure is the sum of the individual
   pure species vapor pressure
// To find the boiling point temperature we perform
   a trial and error
// Let us assume the boiling point temperature
T = 89;//[C]
// Antoine equation constants for water is given by
A_w = 7.94917;
B_w = 1657.462;
C_w = 227.02;

// and that for n-butanol are
A_b = 7.838;
B_b = 1558.190;
C_b = 196.881;

// Antoine equation is given by
// log10(p) = A - B/(T+C)
// Thus vapor pressure of water and n-butanol are
```

```
   respectively
p_water = (1/760)*10^(A_w - B_w/(T+C_w));
p_butanol = (1/760)*10^(A_b - B_b/(T+C_b));

// Now, vapor phase composition are
y_water = p_water/P;
y_butanol = p_butanol/P;
// summing these , we get
y = y_water + y_butanol;

// Value of y is nearly equal to one so our
   assumption of the temperature is correct
// So the boiling point of the mixture is 'T'

printf(" Boiling point of the two phase system is %0
   .0 f deg C\n",T);
printf(" In vapor phase , mole fraction of the water
   is %0.2 f",y_water);
```

Scilab code Exa 8.8 The little EOS

```
clear;
//clc();

// Example 8.8
// Page: 184
printf("Example-8.8  Page no.-184\n\n");

//***Data***//
Temp = 68;//[F]
P = 1;//[atm]
// Changing the temperature in 'K' and pressure in '
   bar' we have
Temp = 273.15+(Temp-32)*5/9;//[K]
P = P*1.01325;//[bar]
```

```
// For water from the table A.1(page 417)
T_c = 647.1;//[K]
P_c = 220.55;//[bar]
// Now
T_r = Temp/T_c;
P_r = P/P_c;
w = 0.345;
//Now applying the result for the little EOS from
   the example 7.1(page 135 ), we have
// f/P = exp(P_r/T_r*f(T_r))
// From the chapter 2 of this book, we have
f_T_r = (0.083-0.422/T_r^(1.6))+w*(0.139-0.172/T_r
   ^(4.2));
// So
f_by_P = exp(P_r/T_r*f_T_r);
printf("The value of the f/P for water vapour in the
    hypothetical state is %0.2f",f_by_P);
```

Scilab code Exa 8.9 Dew Point Calculations

```
clear;
//clc();

// Example 8.9
// Page: 189
printf("Example-8.9  Page no.-189\n\n");

//***Data***//

// Here we will denote ethanol as species 'a' and
   water as the species 'b'
x_a = 0.1238;
x_b = (1-x_a);
T = 85.3;//[C] Given boiling temperature

```

```
// We have
// x_a + x_b = 1    and    y_a + y_b = 1

// The Antoine equation constants for ethanol from
   the table A.2(page 418) given in the book, are
A_a = 8.04494;
B_a = 1554.3;
C_a = 222.65;

// and that for water
A_b = 7.96681;
B_b = 1668.21;
C_b = 228.0;

// Thus vapor pressure of ethanol and water are
   respectively
p_a = (1/760)*10^(A_a - B_a/(T+C_a));
p_b = (1/760)*10^(A_b - B_b/(T+C_b));

// Also the activity coefficients are given by
// Y_a = 10^((B^(2)*A*x_b^(2))/(A*x_a+B*x_b)^(2))
       and
// Y_b = 10^((A^(2)*B*x_b^(2))/(A*x_a+B*x_b)^(2))
// here A and B are Van Laar coefficients and their
   values for ethanol-water system is reported in
   the book at page 186 (last two lines), so
A = 0.7292;
B = 0.4104;

// hence
Y_a = 10^((B^(2)*A*x_b^(2))/(A*x_a+B*x_b)^(2));
Y_b = 10^((A^(2)*B*x_a^(2))/(A*x_a+B*x_b)^(2));

// Now taking into account of nonideality of the
   gaseous phase, the modified Raoult's law gives
// (y_a/x_a) = (Y_a*p_a)/P    and    (y_b/x_b) = (
   Y_b*p_b)/P

```

```
// we will take a simple method
// solving the above two equation for y_a and y_b
   and adding them, we get
P = (Y_a*p_a*x_a)+(Y_b*p_b*x_b);//[atm]

// So,
y_a = (Y_a*p_a*x_a)/P;
// and
y_b = (Y_b*p_b*x_b)/P;

printf(" Boiling pressure of the liquid at 85.3 deg
   C is %0.4f atm\n",P);
printf(" Mole fraction of ethanaol in vapor phase is
         %0.4f\n",y_a);
printf(" Mole fraction of water in the vapor phase
   is     %0.4f",y_b);
```

Scilab code Exa 8.10 `Pressure specified dew point`

```
clear;
//clc();

// Example 8.10
// Page: 191
printf("Example-8.10  Page no.-191\n\n");

//***Data***//

// This problem is similar to the example 8.9 except
    that, we are provided pressure instead of
   temperature and different liquid composition
// Here again, we will denote ethanol as species 'a'
    and water as the species 'b'
x_a = 0.2608;
x_b = (1-x_a);
```

```
P = 1.00;//[atm] Given boiling pressure

// We have
// x_a + x_b = 1   and   y_a + y_b = 1

// The Antoine equation constants for ethanol from
   the table A.2(page 418) given in the book, are
A_a = 8.04494;
B_a = 1554.3;
C_a = 222.65;

// and that for water
A_b = 7.96681;
B_b = 1668.21;
C_b = 228.0;

// Thus vapor pressure of ethanol and water are
   respectively
// p_a = (1/760)*10^(A_a - B_a/(T+C_a))
// p_b = (1/760)*10^(A_b - B_b/(T+C_b))
// Adding these two equation, we get
// ( p_a + p_b ) = (1/760)*10^(A_a - B_a/(T+C_a)) +
   (1/760)*10^(A_b - B_b/(T+C_b))
   .....................................(1)

// Also the activity coefficients are given by
// Y_a = 10^((B^(2)*A*x_b^(2))/(A*x_a+B*x_b)^(2))
       and
// Y_b = 10^((A^(2)*B*x_b^(2))/(A*x_a+B*x_b)^(2))
// here A and B are Van Laar coefficients and their
   values for ethanol-water system is reported in
   the book at page 186 (last two lines), so
A = 0.7292;
B = 0.4104;

// hence
Y_a = 10^((B^(2)*A*x_b^(2))/(A*x_a+B*x_b)^(2));
Y_b = 10^((A^(2)*B*x_a^(2))/(A*x_a+B*x_b)^(2));
```

```

// Now we will solve for T running the loop
// Let us assume the startup temperature
T = 80;
err = 1;

while err > 10^(-3)
    P_a = (10^(8.04494 - 1554.3/(222.65 + T)))/760;
    P_b = (10^(7.96681 - 1668.21/(228 + T)))/760;
    y_a = Y_a*P_a*x_a/P;
    y_b = Y_b*P_b*x_b/P;
    err = abs((y_a + y_b) - 1);
    T = T + 0.01;
end

printf(" Boiling temperature of the liquid at 1 atm
   pressure is %0.4f atm\n",T);
printf(" Mole fraction of ethanaol in vapor phase is
        \t%0.4f\n",y_a);
printf(" Mole fraction of water in the vapor phase
   is    \t%0.4f",y_b);
```

Scilab code Exa 8.11 Temperature specified bubble point

```
clear;
//clc();

// Example 8.11
// Page: 192
printf("Example-8.11  Page no.-192\n\n");

//***Data***//

// This problem is identical to that of the example
   8.9 except difference in the boiling temperature
```

```
    and liquid composition
// Here we will again denote ethanol as species 'a'
    and water as the species 'b'
y_a = 0.6122;
y_b = (1-y_a);
T = 80.7;//[C] Given boiling temperature

// We have
// x_a + x_b = 1    and    y_a + y_b = 1

// The Antoine equation constants for ethanol from
    the table A.2(page 418) given in the book, are
A_a = 8.04494;
B_a = 1554.3;
C_a = 222.65;

// and that for water
A_b = 7.96681;
B_b = 1668.21;
C_b = 228.0;

// Thus vapor pressure of ethanol and water are
    respectively
p_a = (1/760)*10^(A_a - B_a/(T+C_a));
p_b = (1/760)*10^(A_b - B_b/(T+C_b));

// Also the activity coefficients are given by
// Y_a = 10^((B^(2)*A*x_b^(2))/(A*x_a+B*x_b)^(2))
        and
// Y_b = 10^((A^(2)*B*x_b^(2))/(A*x_a+B*x_b)^(2))
// here A and B are Van Laar coefficients and their
    values for ethanol-water system is reported in
    the book at page 186 (last two lines), so
A = 0.7292;
B = 0.4104;

// Now taking into account of nonideality of the
    gaseous phase, the modified Raoult's law gives
```

```
// (y_a/x_a) = (Y_a*p_a)/P    and    (y_b/x_b) = (
   Y_b*p_b)/P

// Now can take the help of trial and error method
   to solve the above equations
// In this method, we will assume the different
   values of P and will calculate the values of x_a
   and x_b from the above two equations, till their
   sum    comes to unity
x_a = 0.6122;// Initial assumption of liquid phase
   composition of ethanol
x_b = 0.3;// Initial assumption of liquid phase
   composition water
P = 0.80;//[atm]
err = 1;

while err > 2* 10^(-2)
    P = P + 0.01;
    Y_a = 10^((B^(2)*A*x_b^(2))/(A*x_a+B*x_b)^(2));
    Y_b = 10^((A^(2)*B*x_a^(2))/(A*x_a+B*x_b)^(2));

    err = abs((x_a + x_b) - 1);
    x_a = y_a*P/(Y_a*p_a);
    x_b = y_b*P/(Y_b*p_b);
end



printf(" Boiling pressure of the liquid at 80.7 deg
   C is %0.4f atm\n",P);
printf(" Mole fraction of ethanaol in liquid phase
   is    %0.4f\n",x_a);
printf(" Mole fraction of water in the liquid phase
   is    %0.4f",x_b);
```

Scilab code Exa 8.12 Pressure specified bubble point

```
clear;
//clc();

// Example 8.12
// Page: 193
printf("Example-8.12   Page no.-193\n\n");

//***Data***//

// This problem is similar to the example 8.10
   except that, we are provided vapour phase
   composition
// Here again, we will denote ethanol as species 'a'
    and water as the species 'b'
y_a = 0.1700;
y_b = (1-y_a);
P = 1.00;//[atm] Given boiling pressure

// We have
// x_a + x_b = 1   and   y_a + y_b = 1

// The Antoine equation constants for ethanol from
   the table A.2(page 418) given in the book, are
A_a = 8.04494;
B_a = 1554.3;
C_a = 222.65;

// and that for water
A_b = 7.96681;
B_b = 1668.21;
C_b = 228.0;

// Thus vapor pressure of ethanol and water are
   respectively
// p_a = (1/760)*10^(A_a - B_a/(T+C_a))
// p_b = (1/760)*10^(A_b - B_b/(T+C_b))

// Also the activity coefficients are given by
// Y_a = 10^((B^(2)*A*x_b^(2))/(A*x_a+B*x_b)^(2))
     and
// Y_b = 10^((A^(2)*B*x_b^(2))/(A*x_a+B*x_b)^(2))
// here A and B are Van Laar coefficients and their
   values for ethanol-water system is reported in
   the book at page 186 (last two lines), so
A = 0.7292;
B = 0.4104;

// Now taking into account of nonideality of the
   gaseous phase, the modified Raoult's law gives
// (y_a/x_a) = (Y_a*p_a)/P    and    (y_b/x_b) = (
   Y_b*p_b)/P

// Now we can take the help of trial and error
   method to solve the above equations
// In this method, we will assume the different
   values of T and will calculate the values of x_a
   and x_b from the above two equations, till their
   sum    comes to unity

x_a = 0.0100;// Initial assumption of liquid phase
   composition of ethanol
x_b = 0.9;// Initial assumption of liquid phase
   composition water
T = 80;//[C] Initial guess of the temperature
err = 1;

while err >  1/16*10^(-2)
    P_a = (10^(8.04494 - 1554.3/(222.65 + T)))/760;
    P_b = (10^(7.96681 - 1668.21/(228 + T)))/760;

    Y_a = 10^((B^(2)*A*x_b^(2))/(A*x_a+B*x_b)^(2));
    Y_b = 10^((A^(2)*B*x_a^(2))/(A*x_a+B*x_b)^(2));

    x_a = y_a*P/(Y_a*P_a);
    x_b = y_b*P/(Y_b*P_b);

    err = abs((x_a + x_b) - 1);
    T = T + 0.01;

end


printf(" Equilibrium Temperature of the system at
   pressure 1 atm  is %0.4f atm\n",T);
printf(" Mole fraction of ethanaol in liquid phase
   is     %0.4f\n",x_a);
printf(" Mole fraction of water in the liquid phase
   is    %0.4f",x_b);
```

Scilab code Exa 8.13 Isothermal flashes

```
clear;
//clc();

// Example 8.13
// Page: 194
printf("Example-8.13  Page no.-194\n\n");

//***Data***//

// Here again, we will denote ethanol as species 'a'
    and water as the species 'b'
x_aF = 0.126;
x_bF = (1-x_aF);
P = 1.00;//[atm] Given total pressure
T = 91.8;//[C]

// We will go with graphical approach for solving
   this problem
```

```
// This problem requires T - x_a diagram at the given pressure i.e. 1 atm
// This diagram is provided on page 196( figure 8.19) in the book
// We will draw horizontal and vertical lines corresponding to the specified T and x_a.
// Drawing a horizontal line from temperature 91.8 degC and vertical line corresponding to the x_aF = 0.126, we see that these two intersect in the two phase region, which tells that our feed contains both liquid and vapour phase
// Now liquid phase composition in equilibrium is found by reading the x-axis where the bubble-point vs x_a curve and horizontal line corresponding to     T = 91.8 degC intersect and viz.
x_a = 0.0401;
x_b = (1 - x_a);

// Similarily vapour phase composition in equilibrium is found by reading the x-axis where the dew-point vs y_a curve and horizontal line corresponding to T = 91.8 degC intersect and viz.
y_a = 0.2859;
y_b = ( 1 - y_a);

// Now vapour fraction is given by
V_by_F = ( x_aF - x_a )/(y_a - x_a);

printf(" Mole fraction of the ethanol in the liquid phase in equilibrium at the given condition is %f\n",x_a);
printf(" Mole fraction of the water in the liquid phase in equilibrium at the given condition is %f\n",x_b);
printf(" Mole fraction of the ethanol in the vapour phase in equilibrium at the given condition is %f\n",y_a);
printf(" Mole fraction of the water in the vapour
   phase in equilibrium at the given condition is %f
   \n",y_b);
printf(" Vapor fraction of the given water-ethanol
   mixture after the flash in equilibrium is %f",
   V_by_F);
```

Scilab code Exa 8.14 Use of de prester chart

```
clear;
//clc();

// Example 8.14
// Page: 198
printf("Example-8.14  Page no.-198\n\n");

//***Data***//

P = 100;//[psia]
// Composition in liquid phase is
x_a = 0.05;// Mole fraction of methane
x_b = 0.40;// Mole fraction of butane
x_c = 0.55;// mole fraction of pentane

//  We have to take the help of the following
   equations
// ( x_a + x_b + x_c ) = 1        and    ( y_a + y_b +
    y_c ) = 1
// ( y_a/x_a ) = K_a;     ( y_b/x_b ) = K_b;
   and    ( y_c/x_c ) = K_c;

// We draw a straight line across figure 8.20 from
   100psia to different temperatures like
   0,5,10,15,20,25,30 degF and read the three K
   factors
```

```
T = [15.8 0.087 0.024;16 0.105 0.026;16.2 0.115
    0.03;16.8 0.13 0.035;17.2 0.15 0.04;17.8 0.17
    0.045;18.2 0.175 0.0472727];
printf(" Calculations for the various assumed
    temperatures are given in the table below\n\n");
printf(" Temperature \t\t    y_a \t\t   y_b \t\t\t
     y_c \t\t\t   y \n\n");

T_b = 0;//[F] Bubble point
j=1;
for i = 1:7
        y_a = x_a*T(i,j);
        y_b = x_b*T(i,j+1);
        y_c = x_c*T(i,j+2);
        y = y_a + y_b + y_c;
        T_b = T_b + 5;
        printf(" %f \t\t %f \t\t %f \t\t %f \t\t %f\
           n ",T_b,y_a,y_b,y_c,y);
    end
printf(" \n For the temperature 30 deg F the
    summation of the mole fractions in the vapor
    phase is close enough to unity, so, bubble point
    is 30 degF\n");
printf(" And compositions in the vapor phase are the
     values given in the above table corresonding to
    the temperature 30 deg F, i.e.\n\n");
printf(" y_methane = %f \n y_butane = %f \n
    y_pentane = %f",y_a,y_b,y_c);
```

Scilab code Exa 8.15 Non volatile solutes boiling point elevation

```
clear;
//clc();

// Example 8.15
// Page: 199
printf("Example-8.15   Page no.-199\n\n");

//***Data***//

n_sugar = 1;//[mol]
n_water = 1000/18;//[mol]
x_sugar = n_sugar/(n_sugar+n_water);
x_water = n_water/(n_sugar+n_water);
// At 100C we have
p_water = 1;//[atm]
p_sugar = 0;//[atm]
// and the relation
P = x_water*p_water+x_sugar*p_sugar;//[atm]
// The situation is sketched in the figure 8.21 in
   the book[page 199].
// Now for the second part of the question
// To find the temperature at which the solution
   will boil, we see on the figure that we must
   raise the temperature to increase p_i to a value
   high enough  that the total pressure P_1 = 1atm,
   with x_water calculated above.
P_1 = 1;//[atm]
p_water = P_1/x_water;//[atm]
// Interpolating in the steam table[12] reported in
   the book, we find
T = 100.51;//[C]
// We may restate this that the boiling-point
   elevation caused by this dissolved, nonvolatile
   solute is
T_eb = T-100;//[C]
printf("Vapour pressure of this solution at the 100C
    is                %f atm\n",P);
printf(" The temperature at which this solution will
    boil at 1 atm is %f C",T);
```

Scilab code Exa 8.16 Freezing point depression

```
clear;
//clc();

// Example 8.16
// Page: 201
printf("Example-8.16   Page no.-201\n\n");

//***Data***//
n_sugar = 1;//[mol]
n_water = 1000/18;//[mol]
x_sugar = n_sugar/(n_sugar+n_water);
x_water = n_water/(n_sugar+n_water);

// Here we can assert that for liquid solution and
   solid ice to be in equilibrium, the fugacity of
   water in the liquid must be same as that of water
    in the solid ice.
// Crystalline solid formed from such a solution is
   nearly pure H2O, with no dissolved sugar.
// At the low pressures involved here, these
   fugacities are practically equal to partial
   pressures, so that
//P = x_water*p_water+x_sugar*p_sugar = p_ice;

// but
p_sugar = 0;
// so
p_ice_by_p_water = x_water;

// Figure 5.8 reported in the book  (page 100) shows
    the vapour pressure of subcooled water and of
   ice.
```

```
//The values in the given table from which that
   figure were made can be represented by the
   following totally empirical data-fitting equation
   .
// p_ice/p_water = 1+0.0096686*T+4.0176*10^(-5)*T
   ^(2)
// We eliminate p_ice/p_water by x_water

deff('[y]=f(T)', 'y = 1+0.0096686*T+4.0176*10^(-5)*T
   ^(2)-p_ice_by_p_water');
T = fsolve(0,f);//[C]

printf("Freezing-point temperature of the given
   solution is %f C",T);
```

Chapter 9

Correlating And Predicting Nonideal VLE

Scilab code Exa 9.1 Van Laar equation

```
clear;
//clc();

// Example 9.1
// Page: 219
printf("Example-9.1   Page no.-219\n\n");

//***Data***//
x_isopropanol = 0.4720;
x_water = 0.5280;
// From the table A.7 (page 427) reported in the
   book the Van Laar coefficients for isopropanol-
   water system at 1atm are given by
A = 1.0728;
B = 0.4750;
// Van Laar equations are given
// log10(Y_a) = A*x_b^(2)/(A/B*x_a+x_b)^(2)
// log10(Y_b) = B*x_a^(2)/(B/A*x_b+x_a)^(2)
// We calculate Y_isopropanol and Y_water as
```

```
Y_isopropanol = 10^(A*x_water^(2)/(A/B*x_isopropanol
   +x_water)^(2));
Y_water = 10^(B*x_isopropanol^(2)/(B/A*x_water+
   x_isopropanol)^(2));
printf(" Value of the liquid-phase activity
   coefficient for isopropanol is %f\n\n",
   Y_isopropanol);
printf(" And value of the liquid-phase activity
   coefficient for water is   %f",Y_water);
```

Scilab code Exa 9.2 Excess Gibbs free energy and activity coefficient equations

```
clear;
//clc();

// Example 9.2
// Page: 221
printf("Example-9.2  Page no.-221\n\n");

//***Data***//
// Recieving the VLE data from the example 8.2, we
   have
x_acetone = 0.05;
x_water = 0.95;
// And the activity coefficient is given by
y_acetone = 7.04;
y_water = 1.01;
// we hve the relation g_E/RT = summation(x_i*log(
   y_i))
// let C = g_E/RT , so
C = (x_acetone*log(y_acetone)+x_water*log(y_water));
// Now let M = (g_E/RT )/(x_acetone*x_water)
// So
M = C/(x_acetone*x_water);
printf("The value of g_E/RT for acetone-water
   solution at 1 atm pressure is              %f\n\n"
   ,C);
printf(" The value of (g_E/RT)/(x_a*x_b) for acetone
   -water solution at 1 atm pressure is %f",M);
```

Scilab code Exa 9.3 Excess Gibbs free energy and activity coefficient equations

```
clear;
//clc();

// Example 9.3
// Page: 221
printf("Example-9.3  Page no.-221\n\n");

//***Data***//

printf("This is a theoratical question and there are
    no any numerical components. For the derivation,
    refer to page no 221 of the book.");
```

Scilab code Exa 9.4 Activity coefficient at infinite dilution

```
clear;
//clc();

// Example 9.4
// Page: 224
printf("Example-9.4  Page no.-224\n\n");

//***Data***//

```

```
printf("This is a theoratical question and there are
    no any numerical components. For the derivation ,
    refer to page no 220 of the book.");
```

Scilab code Exa 9.5 Constants in the morgules equation

```
clear;
//clc();

// Example 9.5
// Page: 224
printf("Example-9.5  Page no.-224\n\n");

//***Data***//
y_acetone_infinity = 10;
y_water_infinty = 5
Pressure = 1;//[atm]
// From equation 9.L and 9.M (page 224) as reported
   in the book, we have
// Constants in morgules equation b and c as
b = log(y_acetone_infinity);
c = log(y_water_infinty);
printf("Values of the constants in Morgules equation
    for acetone-water at 1 atm are b = %f\n",b);
printf("

   and  c = %f",c);
```

Scilab code Exa 9.6 Effect of pressure changes on liquid phase activity coefficien

```
clear;
//clc();

```

```
// Example 9.6
// Page: 225
printf("Example-9.6   Page no.-225\n\n");

//***Data***//
P_1 = 10;//[atm]
x_a_1  = 0.1238;// mole fraction of ethanol at 10
   atm pressure
Temp = 273.15+85.3;//[K]
R = 0.08206;//[(L*atm)/(mol*K)]
P_0 = 1;//[atm]
// so
delta_P = (P_1-P_0);//[atm]
// Molecular weight of ethanol and water are
   respectively
M_ethanol = 46;//[g/mol]
M_water = 18;//[g/mol]
// Now changing the mol fraction of ethanol in the
   wt fraction
m_a_1 = x_a_1*M_ethanol/(x_a_1*M_ethanol+(1-x_a_1)*
   M_water);
// From example 8.9(page 188) we know that at this T
    and 1 atm and x_a_0 , activity coefficient for
   ethanol
y_ethanol_0 = 2.9235;
// Now from figure 6.15(page 129), we read that at
   20C and m_a_1 mass fraction ethanol ,
v_ethanol_1 = 1.16;//[cm^(3)/g]
// Similarily for mass fraction corresponding to
   mole fraction x_a_1
v_ethanol_0 = 1.27;//[cm^(3)/]
// Difference of thes etwo values is
v = v_ethanol_1-v_ethanol_0;//[cm^(3)/g]
v = v*46;//[L/g]
// If we assume that this value is more or less
   independent of temperature, we  can use it as the
    corresponding value at 85.3C, and compute
// From equation 7.31(page 225)
// d(log(y_i))/dP = (v_1−v_0)/(R*T);  at constant
   temperature and mole fraction
// Let d(log(y_i))/dP = C, then
C = (v_ethanol_1-v_ethanol_0)/(R*Temp);
// Also we can have
// delta_log(y_i) = (d(log(y_i))/dP)*delta_P
// or
// delta_log(y_i) = C*delta_P
// and delta_log(y_i) = log(y_ehtanol_1)−log(
   y_ethanol_0)
// So
y_ethanol_1 = exp(log(y_ethanol_0)+C*delta_P);
printf("The activity coefficient of ethanol in the
   solution at 10 atm pressure is %f",y_ethanol_1);
```

Scilab code Exa 9.7 Effect of temperature changes on liquid phase activity coeffic

```
clear;
//clc();

// Example 9.7
// Page: 226
printf("Example−9.7  Page no.−226\n\n");

//***Data***//
x_ethanol = 0.1238;
Temp_1 = 273.15+85.3;//[K]
P = 1;//[atm]
Temp_2 = 273.15+70;//[K]
R = 8.314;//[j/(mol*K)]
// From example 8.9, at temperature 85.3C the
   activity coefficient is
y_ethanol_1 = 2.9235;
// From figure 9.5[4] (page 227) as reported in the
   book, we read the value of (h_i_average−h_i_0) at
```

```
    temperatures 90C and 70C for ethanol.
// which are respectively
delta_h_2 = 0.2;//[kJ/mol]
delta_h_1 = 1.0;//[kJ/mol]
// Taking the average of these two values we have
delta_h_average = (delta_h_1+delta_h_1)/2*1000;//[J/
   mol]
// From the equation 7.32 (page 225) reported in the
    book
// d(log(y_i))/dT = (h_i_average-h_i_0)/(R*T^(2));
   at constant pressure and mole fraction
// So
// integrate(d(log(y_i)) = integrate((h_i_average-
   h_i_0)/(R*T^(2)))*dT
// it can be taken approximately as
// integrate(d(log(y_i)) = ((h_i_average-h_i_0)
   _average/R)*integrate(1/T^(2))*dT
// we have integrate(d(log(y_i)) = log(y_ethanol_2/
   y_ethanol_1)
// So
y_ethanol_2 = y_ethanol_1*exp((delta_h_average/R)*
   integrate('1/T^(2)','T',Temp_1,Temp_2));
printf("The activity coefficient for ethanol in the
   solution at 70 deg C and 1 atm is %f",y_ethanol_2
   );
```

Scilab code Exa 9.8 Liquid phase activity coefficients for ternary mixtures

```
clear;
//clc();

// Example 9.8
// Page: 229
printf("Example-9.8  Page no.-229\n\n");

```

```
//***Data***//
// In this solution we will give the identity to the
    three species as
// a- Acetone
// b- Methanol
// c- Water
// Given
x_a = 0.1200;
x_b = 0.1280;
x_c = 0.7520;
Temp = 66.70;//[C]
P = 1;//[atm]  pressure
// As reported in the book that from [5] we get the
   following values
// acetone-methanol(a-b)
A_ab = 0.2634;
A_ba = 0.2798;
// acetone-water(a-c)
A_ac = 0.9709;
A_ca = 0.5579;
// methanol-water(b-c)
A_bc = 0.3794;
A_cb = 0.2211;
// Now consider the equation 9.10 (page 228)
// The first term on the right of the equation is
T_1 = x_b^(2)*(A_ab+2*x_a*(A_ba-A_ab));
// similarily the second and third terms are given
   respectively as
T_2 = x_c^(2)*(A_ac+2*x_a*(A_ca-A_ac));
T_3 = x_b*x_c*(0.5*(A_ba+A_ab+A_ac-A_bc-A_cb)+x_a*(
   A_bc-A_ab+A_ca-A_ac)+(x_b-x_c)*(A_bc-A_cb)-(1-2*
   x_a)*0.00);
// thus whole term on the right hand side is
T = T_1+T_2+T_3;
// So
y_a = 10^(T);
// for this temperature vapour pressure of the
   acetone is calculated as
```

```
p_acetone = 1.417;//[atm]
// So that we estimate
y_acetone = x_a*y_a*p_acetone;
printf("The activity coefficient of acetone in the
    given mixture is %f",y_a);
// The experimental value is y_acetone  = 0.698
```

Scilab code Exa 9.9 `Application of mixing rule`

```
clear;
//clc();

// Example 9.9
// Page: 234
printf("Example-9.9  Page no.-234\n\n");

//***Data***//

T = 85.3+273.15;//[K] Temperature
P = 1;//[atm] Pressure of the mixture
R = 8.314;//[(Pa*m(3)/(K*mol))]
R_1 = 0.08206;//[(L*atm)/(mol*K)]
y_i = 0.1238;// mole fraction of the ethanol in the
    vapor phase
y_j = (1-y_i);// mole fraction of the water vapor in
     the vapor phase

// From the table A.1( table 417), the properties of
     water and ethanol are given as
// Critical temperatures are
T_c_ii = 513.9;//[K] Critical temperature of the
    ethanol
T_c_jj = 647.1;//[K] Criatical temperature of water

// Critical pressure are
```

```
P_c_ii = 61.48;//[bar] Critical pressure of ethanol
P_c_jj = 220.55;//[bar] Critical pressure of water

// Acccentric factor
w_ii = 0.645;// accentric factor of the ethanol
w_jj = 0.345;// accentric factor of the water

// Compressibility factor are
z_c_ii = 0.24;// compressibility factor of ethanol
z_c_jj = 0.229;// compressibility factor of the
   water

// Critical volume are given by
V_c_ii = z_c_ii*R*T_c_ii/(P_c_ii*100000)*10^(6);//
   critical volume the ethanol
V_c_jj = z_c_jj*R*T_c_jj/(P_c_jj*100000)*10^(6);//
   critical volume the ethanol

// Now
// for k_ij = 0.0
T_c_ij_0 = (T_c_ii*T_c_jj)^(1/2);//[K]
w_ij = (w_ii + w_jj)/2;
z_c_ij = (z_c_ii + z_c_jj)/2;
V_c_ij = ( (V_c_ii^(1/3) + V_c_jj^(1/3))/2)^(3);
P_c_ij_0 = (z_c_ij*R*T_c_ij_0)/(V_c_ij/10^(6))
   /10^(5);//[bar]

// again for k_ij = 0.01
T_c_ij_1 = (T_c_ii*T_c_jj)^(1/2)*(1-0.01);//[K]
P_c_ij_1 = (z_c_ij*R*T_c_ij_1)/(V_c_ij/10^(6))
   /10^(5);//[bar]

// Now
T_r_ii = T/T_c_ii;
T_r_jj = T/T_c_jj;
T_r_ij_0 = T/T_c_ij_0;
T_r_ij_1 = T/T_c_ij_1;

```

```
// and
P_r_ii = P/P_c_ii;
P_r_jj = P/P_c_jj;
P_r_ij_0 = P/P_c_ij_0;
P_r_ij_1 = P/P_c_ij_1;

// Now we  will calculate f(T_r) for each component
   and mixture
f_Tr_ii = ( 0.083 - 0.422/T_r_ii^(1.6) ) + w_ii*(
   0.139 - 0.172/T_r_ii^(4.2));
f_Tr_jj = ( 0.083 - 0.422/T_r_jj^(1.6) ) + w_jj*(
   0.139 - 0.172/T_r_jj^(4.2));
f_Tr_ij0 = ( 0.083 - 0.422/T_r_ij_0^(1.6) ) + w_ij*(
    0.139 - 0.172/T_r_ij_0^(4.2));
f_Tr_ij1 = ( 0.083 - 0.422/T_r_ij_1^(1.6) ) + w_ij*(
    0.139 - 0.172/T_r_ij_1^(4.2));

// Let us define A = (P_r*f(T_r)/T_r) , so
A_ii = P_r_ii*f_Tr_ii/T_r_ii;
A_jj = P_r_jj*f_Tr_jj/T_r_jj;

// We are given
v_ii = 0.975;
v_jj = 0.986;

// Now,
B_ii = ( f_Tr_ii*R*T_c_ii/P_c_ii)*(10^(3)/10^(5));//
   [L/mol]
B_jj = ( f_Tr_jj*R*T_c_jj/P_c_jj)*(10^(3)/10^(5));//
   [L/mol]
B_ij0 = ( f_Tr_ij0*R*T_c_ij_0/P_c_ij_0)*(10^(3)
   /10^(5));//[L/mol]
B_ij1 = ( f_Tr_ij1*R*T_c_ij_1/P_c_ij_1)*(10^(3)
   /10^(5));//[L/mol]

// now we will calculate 'delta'
delta_ij0 = 2*B_ij0 - B_ii - B_jj;//[L/mol]
delta_ij1 = 2*B_ij1 - B_ii - B_jj;//[L/mol]
```

```

// We have
// b_a = B_aa + y_b^(2)*delta      and     b_b = B_bb +
   y_a^(2)*delta
// so,
b_ethanol0 = B_ii + y_j^(2)*delta_ij0;//[L/mol]
b_water0 = B_jj + y_i^(2)*delta_ij0;//[L/mol]
b_ethanol1 = B_ii + y_j^(2)*delta_ij1;//[L/mol]
b_water1 = B_jj + y_i^(2)*delta_ij1;//[L/mol]

// Now
// phi_i = exp(b_i*P/(R*T))
// So,
phi_ethanol0 = exp((b_ethanol0*P)/(R_1*T));
phi_water0 = exp((b_water0*P)/(R_1*T));
phi_ethanol1 = exp((b_ethanol1*P)/(R_1*T));
phi_water1 = exp((b_water1*P)/(R_1*T));

// and
// Y_i = phi_i/v_i
// So,
Y_ethanol0 = phi_ethanol0/v_ii;
Y_water0 = phi_water0/v_jj;
Y_ethanol1 = phi_ethanol1/v_ii;
Y_water1 = phi_water1/v_jj;

printf(" The results are summarize in the following
   table\n\n");
printf("   Property \t\t\t Mix, ij, Assuming k_ij =
   0.0 \t\t\t  Mix, ij, Assuming k_ij = 0.01\n");
printf("  phi_ethanol \t\t\t\t %f \t\t\t\t\t %f\n ",
   phi_ethanol0,phi_ethanol1);
printf(" phi_water   \t\t\t\t %f \t\t\t\t\t %f \n",
   phi_water0,phi_water1);
printf("  Y_ethanol   \t\t\t\t %f \t\t\t\t\t %f \n",
   Y_ethanol0,Y_ethanol1);
printf("  Y_water   \t\t\t\t %f \t\t\t\t\t %f \n\n",
   Y_water0,Y_water1);
```

```
printf(" Value of ''v'' for ethanol is %f\n",v_ii);
printf(" Value of ''v'' water is %f",v_jj);
```

Scilab code Exa 9.10 `Solubility parameter`

```
clear;
//clc();

// Example 9.10
// Page: 239
printf("Example-9.10   Page no.-239\n\n");

//***Data***//

T = 65+273.15;//[K] Temperature
R = 8.314;//[(m^(3)*Pa)/(mol*K)] Universal gas
    constant
// From the table 9.C ( page 239 ) given in the book
     the molar volumes and solubility of n-hexane and
     diethylketone at 25 deg C are given as
v_hex = 131.6;//[ml/mol] Molar volume of n-Hexane
v_dketone = 106.4;//[ml/mol] Molar volume of
    diethylketone
s_hex = 14.9;//[MPa^(0.5)] Solubility of n-Hexane
s_dketone = 18.1;//[MPa^(0.5)] Solubility of
    diethylketone

// Here we will use these values with the assumption
     that   Y_i,65C = Y_i,25C
// At infinite dilution, the volume fraction of the
    other species is 1.00, so,
// logY_a = v_a*phi_b^(2)*(delta_a - delta_b)^(2)/(R
    *T)
// so, for n-Hexane
Y_hex = exp(v_hex*1^(2)*(s_hex - s_dketone)^(2)/(R*T
   ));

// And that for diethylketone
Y_dketone = exp(v_dketone*1^(2)*( s_dketone - s_hex
   )^(2)/(R*T));
printf(" The infinite dilution activity coefficient
   of n-Hexane is %f\n",Y_hex);
printf(" The infinite dilution activity coefficient
   of diethlyketone is %f",Y_dketone);
```

Scilab code Exa 9.11 `Henrys Law Constant estimation`

```
clear
clear;
//clc();

// Example 9.11
// Page: 243
printf("Example-9.11  Page no.-243\n\n");

//***Data***//

P = 1;//[atm]
T = 25;//[C]
y_i = 1.00;// amount of the oxygen in the vapour
// Using the constants for O2 in table A.2
A = 6.69147;
B = 319.0117;
C = 266.7;
// By Antoine equation
// log10(P_i) = A-B/(T+C)
P_i = 10^(A-B/(T+C));//[mmHg]
P_i = P_i/760;//[atm]
// This is extrapolated vapour pressure of O2 at 25C
// We will take this value as equal to the Henry's
```

```
   law constant
H_i = P_i;
x_i = y_i*P/H_i;
printf(" Henry''s law constant for O2 is %f atm\n",
   P_i);
printf(" solubility of O2 is                %e",x_i);
```

Scilab code Exa 9.12 Calculation of the activity coefficient using Henrys law

```
clear;
//clc();

// Example 9.12
// Page: 244
printf("Example-9.12  Page no.-244\n\n");

//***Data***//
y_a = 1.00;
P = 1.00;//[atm]
x_a = 0.231*10^(-4);
// Using the constants for O2 in table A.2 in the
   Antoine equation , we find the vapour pressure of
    the oxygen at 25C viz.
p_a = 521.15;//[atm]
// Thus activity coefficient is calculated by
   rewriting the equation 8.6 and using the above
   values
Y_O2 = (y_a*P)/(x_a*p_a);
printf("The activity coefficient of the oxygen in
   the water is %f",Y_O2);
```

Chapter 10

Vapor Liquid Equilibrium VLE at High Pressures

Scilab code Exa 10.1 Bubble point temperature estimation

```
clear;
//clc();

// Example 10.1
// Page: 260
printf("Example-10.1   Page no.-260\n\n");

//***Data***//

P = 100;//[psia] Bubble point pressure
x_ethane = 0.10;// Mole fraction of ethane in liquid
    phase
x_heptane = (1-x_ethane);

// a) From figure 10.7( page 260 ) given in the book
// We read the chart to get the bubble-point
   temperature
// The dew point curve for 100 psia crosses the 10
   mol% ethane line at about temperature
```

```
T1 = 165;//[C]
// Now, we horizontally from that intersection point
    to the dew-point curve, finding the vapor phase
   composition of ethane
y1_e = 0.92;
y1_h = (1- y1_e);

// b) By Raoult's law, we use a trial and error
   procedureon the temperature
// Antoine equation constants for ethanol are given
A_e = 6.80267;
B_e = 656.4028;
C_e = 255.99;

// and that for n-heptane are
A_h = 6.9024;
B_h = 1268.115;
C_h = 216.9;

// Antoine equation is given by
// (log10p) = (A - B/(T+C))
 T = 50;//[C]
 err = 1;

while err > 10^(-4)
    p1_e = (10^(A_e - B_e/(C_e + T)))*(14.7/760);
    p1_h = (10^(A_h - B_h/(C_h + T)))*(14.7/760);
    y2_e = p1_e*x_ethane/P;
    y2_h = p1_h*x_heptane/P;
    err = abs((y2_e + y2_h) - 1);
    T = T + 0.0001;
  end

// Changing the temperature in deg F
T2 = T*9/5 + 32;//[F] Bubble-point temperature

// c) In this method, we use L-R rule, instead of
   simple Raoult's law
```

```
// So,
// y_i = (x_i*p_i)/(v_i*P)
// Where calculated values of v_i from EOS are given
    in the table 10.A and are
v_e = 0.950;// For ethane
v_h = 0.459;// For n-heptane

// We again use trial and error on the temperature
// Let us assume the initial temperature
 Ti = 50;//[C]
 err = 1;

while err > 10^(-4)
    p2_e = (10^(A_e - B_e/(C_e + Ti)))*(14.7/760);
    p2_h = (10^(A_h - B_h/(C_h + Ti)))*(14.7/760);
    y3_e = p2_e*x_ethane/(P*v_e);
    y3_h = p2_h*x_heptane/(P*v_h);
    err = abs((y3_e + y3_h) - 1);
    Ti = Ti + 0.0001;
  end

// Changing the temperature in deg F
T3 = Ti*9/5 + 32;//[F] Bubble-point temperature

printf(" The results are summarized in the following
    table:\n\n");
printf(" \t  Variable \t\t\t Values calculated from
   \t\t\t Values calculated from \t\t\t Values
   calculated \n \t\t\t\t\t from figure 10.7 \t\t\t
   Raoult''s law \t\t\t\t\t from L-R rule\n\n");
printf(" \t  T(deg F) \t\t\t\t %f \t\t\t\t %f \t\t\t
   \t    %f\n",T1,T2,T3);
printf(" \t  y_ethane \t\t\t\t %f \t\t\t\t %f \t\t\t
   \t    %f\n",y1_e,y2_e,y3_e);
printf(" \t  y_heptane \t\t\t\t %f \t\t\t\t %f \t\t\
   t\t    %f\n\n",y1_h,y2_h,y3_h);
printf(" Where T is boiling point temperature");
```

Scilab code Exa 10.2 Bubble point temperature estimation

```
clear;
//clc();

// Example 10.2
// Page: 262
printf("Example-10.2  Page no.-262\n\n");
printf(" Please wait for the result\n\n");

//***Data***//

P = 800;//[psia] Bubble point pressure
x_ethane = 0.60;// Mole fraction of ethane in liquid
    phase
x_heptane = (1-x_ethane);

// a) From figure 10.7( page 260 ) given in the book
// We read the chart to get the bubble-point
   temperature
// The dew point curve for 800 psia crosses the 60
   mol% ethane line at about temperature
// T1 = 165
// Now, we horizontally from that intersection point
    to the dew-point curve, finding the vapor phase
   composition of ethane
// y1_e = 0.95
// But, by linear interpolation in the experimental
   data on which Figure 10.7 is based we make a
   slightly more reliable estimate and get
T1 = 209;//[F]
y1_e = 0.945;
y1_h = (1- y1_e);

```

```
// b) By Raoult's law, we use a trial and error
   procedureon the temperature
// Antoine equation constants for ethanol are given
A_e = 6.80267;
B_e = 656.4028;
C_e = 255.99;

// and that for n-heptane are
A_h = 6.9024;
B_h = 1268.115;
C_h = 216.9;

// Antoine equation is given by
// (log10p) = (A - B/(T+C))
 T = 50;//[C]
 err = 1;

while err > 10^(-4)
    p1_e = (10^(A_e - B_e/(C_e + T)))*(14.7/760);
    p1_h = (10^(A_h - B_h/(C_h + T)))*(14.7/760);
    y2_e = p1_e*x_ethane/P;
    y2_h = p1_h*x_heptane/P;
    err = abs((y2_e + y2_h) - 1);
    T = T + 0.0001;
  end

// Changing the temperature in deg F
T2 = T*9/5 + 32;//[F] Bubble-point temperature

// c) In this method, we use L-R rule, instead of
   simple Raoult's law
// So,
// y_i = (x_i*p_i)/(v_i*P)
// Where calculated values of v_i from EOS are given
v_e = 0.6290642;// For ethane
v_h = 0.0010113;// For n-heptane

// We again use trial and error on the temperature
// Let us assume the initial temperature
 Ti = 10;//[C]
 err = 1;

while err > 10^(-4)
    p2_e = (10^(A_e - B_e/(C_e + Ti)))*(14.7/760);
    p2_h = (10^(A_h - B_h/(C_h + Ti)))*(14.7/760);
    y3_e = p2_e*x_ethane/(P*v_e);
    y3_h = p2_h*x_heptane/(P*v_h);
    err = abs((y3_e + y3_h) - 1);
    Ti = Ti + 0.0001;
  end

// Changing the temperature in deg F
T3 = Ti*9/5 + 32;//[F] Bubble-point temperature

printf(" The results are summarized in the following
    table:\n\n");
printf(" \t   Variable \t\t\t Values calculated from
   \t\t\t Values calculated from \t\t\t Values
   calculated \n \t\t\t\t\t from figure 10.7 \t\t\t
   Raoult''s law \t\t\t\t\t from L-R rule\n\n");
printf(" \t  T(deg F) \t\t\t\t %f \t\t\t\t %f \t\t\t
   \t    %f\n",T1,T2,T3);
printf(" \t  y_ethane \t\t\t\t %f \t\t\t\t %f \t\t\t
   \t    %f\n",y1_e,y2_e,y3_e);
printf(" \t  y_heptane \t\t\t\t %f \t\t\t\t %f \t\t\
   t\t    %f\n\n",y1_h,y2_h,y3_h);
printf(" Where T is boiling point temperature");
```

Scilab code Exa 10.3 Bubble point temperature estimation using SRK EOS

```
clear;
//clc();

```

```
// Example 10.2
// Page: 262
printf("Example-8.2   Page no.-262\n\n");

//***Data***//

// The initial data for this example is same as that
    of example 10.2, i.e.
P = 800;//[psia] Bubble point pressure
x_e = 0.60;// Mole fraction of ethane in liquid
   phase
x_h = (1-x_e);// Mole fraction of n-heptane in the
   liquid phase
R = 0.08314;//( L*bar/(mol*K)) Universal gas
   constant

// Changing the pressure in bar
Pb = (800/14.7)*(1.01325);//[bar]

// In this problem we will denote ethane by 'e' and
   that to n-heptane by 'h'
// From table A.1 ( page 417 ) given in the book,
   critical temperatures of ethane and heptane are
T_c_e = 305.3;//[K]
T_c_h = 540.2;//[K]

// and critical pressures are
P_c_e = 48.72;//[bar]
P_c_h = 27.40;//[bar]

// also the accentric facors are
w_e = 0.1;
w_h = 0.35;

// Thus we have
P_r_e = Pb/P_c_e;
P_r_h = Pb/P_c_h;

```

```
// Now from equations (F.13) and (F.14) ( page 459 )
    given in the book we have
// A_e = 0.42747 + ( 1 + (0.480 + 1.574*w_e - 0.17*
   w_e^(2))*( 1 - T_r_e^(0.5)))^(2)*(P_r_e/T_r_e^(2)
   );
// A_h = 0.42747 + ( 1 + (0.480 + 1.574*w_h - 0.17*
   w_h^(2))*( 1 - T_r_h^(0.5)))^(2)*(P_r_h/T_r_h^(2)
   );
// and
// B_e = 0.08664*(P_r_e/T_r_e);
// B_h = 0.08664*(P_r_h/T_r_h);

// We will take the help trial and error method both
    on Temperature and the vapor phase composition
   of ethane
// Let us assume the starting temperature 200 deg F.
    Changing this temperature in K
T = (200-32)*5/9 + 273.15;//[K]
err = 1;

while err > 10^(-4)
    T_r_e = T/T_c_e;
    T_r_h = T/T_c_h;
    A_e = 0.42747*( 1 + (0.480 + 1.574*w_e - 0.17*
       w_e^(2))*( 1 - T_r_e^(0.5)))^(2)*(P_r_e/T_r_e
       ^(2));
    A_h = 0.42747*( 1 + (0.480 + 1.574*w_h - 0.17*
       w_h^(2))*( 1 - T_r_h^(0.5)))^(2)*(P_r_h/T_r_h
       ^(2));

    B_e = 0.08664*(P_r_e/T_r_e);
    B_h = 0.08664*(P_r_h/T_r_h);

    // Now we will take the starting value of vapor
       phase composition of ethane as
    y_e = 0.9;
    err1 = 1;

    while err1 > 10^(-6)
        // Now value of A_mix and B_mix for both liquid and vapor phase are calculated as

        A_mix_l = (x_e*sqrt(A_e) + x_h*sqrt(A_h))^(2);// For liquid phase
        A_mix_v = (y_e*sqrt(A_e) + (1 - y_e)*sqrt(A_h))^(2);// For vapor phase
        B_mix_l = (x_e*B_e + x_h*B_h);// For liquid
        B_mix_v = (y_e*B_e + (1 - y_e)*B_h);// For liquid
        deff('[y]=f(z1)','y = z1^(3) - z1^(2) + z1*(A_mix_l - B_mix_l - B_mix_l^(2)) - A_mix_l*B_mix_l');
        z_l = fsolve(0.2,f);
        // and
        deff('[y]=g(z2)','y = z2^(3) - z2^(2) + z2*(A_mix_v - B_mix_v - B_mix_v^(2)) - A_mix_v*B_mix_v');
        z_v = fsolve(0.3,g);
        // Now
        phi_el = B_e/B_mix_l*( z_l - 1) - log(z_l - B_mix_l) - (A_mix_l/B_mix_l)*(2*sqrt(A_e/A_mix_l)-B_e/B_mix_l)*log(1-B_mix_l/z_l);
        phi_hl = B_h/B_mix_l*( z_l - 1) - log(z_l - B_mix_l) - (A_mix_l/B_mix_l)*(2*sqrt(A_h/A_mix_l)-B_h/B_mix_l)*log(1-B_mix_l/z_l);
        phi_ev = B_e/B_mix_v*( z_v - 1) - log(z_v - B_mix_v) - (A_mix_v/B_mix_v)*(2*sqrt(A_e/A_mix_v)-B_e/B_mix_v)*log(1-B_mix_v/z_v);
        phi_hv = B_h/B_mix_v*( z_v - 1) - log(z_v - B_mix_v) - (A_mix_v/B_mix_v)*(2*sqrt(A_h/A_mix_v)-B_h/B_mix_v)*log(1-B_mix_v/z_v);
        K_e = phi_el/phi_ev;
        K_h = phi_hl/phi_hv;
        y_e1 = K_e*x_e;
        y_h1 = K_h*x_h;
        err1 =abs((y_e1 - y_e));
        y_e = y_e1;
    end

    err = abs((y_e1 + y_h1) -1);
     T = T + 0.1;

end

// Changing the temperature in deg F, we have
Tf = ( T - 273.15)*9/5 + 32;//[F]

printf(" Bubble point of the given ethanol and n-
   heptane mixture at 800 psia is %f deg F\n",Tf);
printf(" Amount of ethanol in the vapour phase of
   the mixture at the given condition is %f \n",y_e1
   );
printf(" Amount of n-heptane in the vapour phase of
   the mixture at the given condition is %f ",y_h1);
```

Chapter 11

LIquid Liquid Liquid Solid And Gas Solid Equilibrium

Scilab code Exa 11.1 Reporting and presenting LLE data

```
clear;
//clc();

// Example 11.1
// Page: 272
printf("Example-11.1  Page no.-272\n\n");

//***Data***//
V_water = 1;//[L] volume of the water
Temp = 25;//[C]
d_benzene = 0.88;//[g/ml] density of the benzene
M_benzene = 78;//[g/mol] molecular weight of the
    benzene
M_water = 18;//[g/mol]
// Typically a buret will deliver about 20 drops of
    lliquid per millimeter, so moles of benzene in
    one drop is
n_1drop = 1/20*(d_benzene/M_benzene);//[mol/drop]
// No of moles in 1 litre of the water is
```

```
n_water = 1000/M_water;//[mol]
// Because 1 litre = 1000 g
// Now from the table 11.1 (page 273), at the
   saturated condition at the temperature 25C,
   solubility of benzene in the water is
s_benzene = 0.000405;
n_benzene_saturate = s_benzene*n_water;//[mol]
// Thus no of the drops of the benzene is
N_benzene = n_benzene_saturate/n_1drop;//[drops]
printf("The total number of the drops of the
   benznene required to saturate the water is %0.0f
   drops",N_benzene);
```

Scilab code Exa 11.2 Reporting and presenting LLE data

```
clear;
//clc();

// Example 11.2
// Page: 273
printf("Example-11.2   Page no.-273\n\n");

//***Data***//
m_benzene = 1000;//[lbm]
M_benzene = 78;//[lbm/lbmol]
// The total moles benzene are
 n_benzene = m_benzene/M_benzene;//[lbmol]
 // To saturate the water with benzene
 // the mole fraction of the benzene in the water
    will be
 x_benzene = 0.000405;
 // and
// n_benzene = x_benzene*n_T;
 // in this case n_benzene << n_water, so n_T =
    n_water
 // Thus
 //n_benzene = x_benzene*n_water;
n_water = n_benzene/x_benzene;//[lbmol]
m_water = n_water*18;//[lbm]
 printf("The amount of the ground water that will
    make a saturated solution will be %e lbm",
    m_water);
```

Scilab code Exa 11.3 ternary LLE

```
clear;
//clc();

// Example 11.3
// Page: 277
printf("Example-11.3  Page no.-277\n\n");

//***Data***//

Temp = 25;//[C]
n_water = 3.75;//[mol]
n_ethanol = 2.5;//[mol]
n_benzene = 3.75;//[mol]

// By the simple stoichiomtetry the overall mole
   fractions are

x_water = 0.375;
x_ethanol = 0.250;
x_benzene = 0.375;

// We locate the point corresponding to this
   concentration on the diagram 11.1 (page 277), by
   drawing any two of the three straight lines
   corresponding to those mole fractions.
```

```
// We find that the point falls almost exactly on
   the fifth tie line from the top.
//For this the end-point values are read from the
   table 11.4 ( page 276 ), which is fifth row from
   the bottom
// Thus in water reach phase we have the composition
    as
x_water_w = 64.9;//[%]
x_ethanol_w = 31.75;//[%]
x_benzene_w = 3.37;//[%]

// and in the benzene reach phase composition is
x_water_b = 6.43;//[%]
x_ethanol_b = 18.94;//[%]
x_benzene_b = 74.62;//[%]
printf("The composition of the two equilibrium
   phases i.e. water-reach phase and benzene reach
   phase is as \n\n");
printf("\t\t\t\tWater-reach phase\t\t\t\t\t\t\
   tbenzene-reach phase\n");
printf("  Mol%% water\t\t\t%f\t\t\t\t\t\t\t\t%f\n\n"
   ,x_water_w,x_water_b);
printf("  Mol%% ethanol\t\t\t%f\t\t\t\t\t\t\t\t%f\n\
   n",x_ethanol_w,x_ethanol_b);
printf("  Mol%% benzene\t\t\t%f\t\t\t\t\t\t\t\t%f\n\
   n",x_benzene_w,x_benzene_b);
```

Scilab code Exa 11.4 `The elementary theory of LLE`

```
clear;
//clc();

// Example 11.4
// Page: 282
printf("Example-11.4  Page no.-282\n\n");
```

```

//***Data***//

Temp = 25;//[C]
// Here we assume benzene to be component 1 and
   water to be componenet 2
// From table 11.1 given in the book(page 273)
// The mole fraction of benzene in water is
x_1in2 = 405;//[ppm]
// and the mole fraction of water in benzene is
x_2in1 = 3000;//[ppm]

// Thus mole fraction of water in water rich phase
   is
x_water_w = (10^(6)-405)/(10^(6));
x_benzene_w = 1-x_water_w;

// and mole fraction of the benzene in benzene rich
   phase is
x_benzene_b =(10^(6)-3000)/(10^(6));
x_water_b = 1-x_benzene_b;

// Here both x_water and x_benzene are nearly equal
   to 1
// Thus assumption used for derivation of the
   equation 11.4(page 282) are suitable here and the
    equation is
// x_i_1 = y_i_1 , where y_i_1 is activity
   coefficient

// So activity coefficient of benzene in water is
y_benzene = 1/(x_benzene_w);
// and activty coefficient of the water in benzene
   is
y_water = 1/(x_water_b);
printf(" Activity coefficient of benzene in water is
    %f\n\n",y_benzene);
printf(" Activity coefficient of water in benzene is
    %f",y_water);
```

Scilab code Exa 11.5 `Plot of the Gibbs free energy vs mole fraction`

```
clear;

//clc();

// Example 11.5

// Page: 283

printf("Example-11.5   Page no.-283\n\n");

//***Data***//


R = 8.314;//[J/(mol*K)] Universal gas constant
T = 298.15;//[K] Temperature
g_a_0 = 2;//[kj/mol] Gibb's free energy of the pure
    species 'a'
g_b_0 = 1;//[kj/mol] Gibb's free energy of the pure
    species 'b'

for a = 0:3
    deff("[y]=f(x)","y= x*g_a_0 + (1-x)*g_b_0 + (R*T
       )/1000*(x*log(x) + (1-x)*log(1-x) + x*a*(1-x)
       ^(2) + (1-x)*a*(x)^(2)) ")

x=[0.000001:0.01:0.99999];

fplot2d(x,f)
xlabel(" mole fraction of species a,x_a");
ylabel(" gibb''s free energy per mole of mixture,
   g_mixture kJ/mol");
```

```
end

printf(" The plot is shown in the graphic window.");
```

Scilab code Exa 11.6 Two liquid phase

```
clear;
//clc();

// Example 11.6
// Page: 283
printf("Example-11.6  Page no.-284\n\n");

//***Data***//

T = 92 + 273.15;//[K] Temperature of the system
R = 8.314;//[m^(3)*Pa/(mol*K)] universal gas
    constant
// Van Laar equation coefficients are given
A = 1.2739;
B = 3.9771;


// From van laar equation, for water, we have
// lnY_a = B^(2)*A*(1-x_a)^(2)/(A*x_a + B*(1-x_a))
    ^(2);
// Similarily for n-butanol
// lnY_b = A^(2)*B*x_a^(2)/(A*x_a + B*(1-x_a))^(2);

// Let us say,   g = g_mix - sum(x_i*g_i_0)
// So, plotting g vs x_i we have

deff("[y]=f(x_a)","y = (R*T/1000)*( x_a*log(x_a) +
    (1-x_a)*log(1-x_a) + x_a*(B^(2)*A*(1-x_a)^(2)/(A*
    x_a + B*(1-x_a))^(2)) + (1-x_a)*(A^(2)*B*x_a^(2)
    /(A*x_a + B*(1-x_a))^(2)) )")

x_a = [0.000001:0.01:0.99999];
fplot2d(x_a,f)
xlabel(" Mol fraction of species a, x_a");
ylabel(" g_mix - sum(x_i*g_i0)");

// Now drawing tangent
x = [0.000001:0.01:0.99999];
plot2d(x,[1.2090312*x - 1.251495764])

// Figure shows the results of the calculation of
    the whole range of x_a
// Drawing the tangent to the curve, the line
    touches the curve at two points x_a = 0.47 and
    0.97
printf(" Thus based on the several assumptions that
    the Van Laar equation is an accurate
    representation of LLE,\n");
printf(" we would conclude that at 92 deg C water-n-
    butanol does form two liquid phases.");
```

Scilab code Exa 11.7 Effect of temperature on LLE

```
clear;
//clc();

// Example 11.7
// Page: 286
printf("Example-11.7  Page no.-286\n\n");

//***Data***//
R = 8.314;//[J/(ml*K)]
// We find that the water in benzene least squares
    fit line has the equation
```

```
// log(x_water(benzene-reach phase)) = 4.175-2967.7/T

// equation 11.7 in the book (page 286) is
// log(x_i_1) = (h_i_0-h_i_average)/(RT) + constant of integration

// Comparing the two equations term by term, we have

// (h_i_0-h_i_average)/(RT) = -2967.7/T
// let us say (h_i_0-h_i_average) = h_mix
// So that
h_mix = -2967.7*R/1000;//[kJ/mol] Heat of mixing of water-in-benzene

// Now, for benzene-in-water the constant in the above equation is -522.9K, so
h_mix_1 = 522.9*R/1000;//[kJ/mol] Heat of mixing of benzene-in-water

printf(" Heat of mixing of water-in-benzene is given as %f kJ/mol\n",h_mix);
printf(" Heat of mixing of benzene-in-water is given as    %f kJ/mol",h_mix_1);
```

Scilab code Exa 11.8 `Effect of temperature on LLE`

```
clear;
//clc();

// Example 11.8
// Page: 287
printf("Example-11.8  Page no.-287\n\n");

//***Data***//
```

```

T_i = 50;//[F] Initial temperature of the system
T_f = 20;//[F] Final temperature of the system
M_gas = 115;//[g/mol] Molecular weight of gasoline
   at room temperature
M_water = 18;//[g/mol] Molecular weight of water at
   the room temperaature
d = 720;//[g/L] density of gasoline at the room
   temperature

// From Figure 11.10 ( page 288 ), solubility of the
    water in gasoline ( similar to solubility of
   water in cyclohexane ) at 50 deg F is given as
s_50 = 0.00026;//[mol fraction]

// And linearly extraploting the cyclohexane curve
   in figure 11.10 to 20 deg F, we get the
   solubility of water at 20deg F as
s_20 = 0.0001;//[mol fraction]

// So, rejected water is
s_rej = s_50 - s_20;// mol of water per mole of
   gasoline

// In terms of weight, rejected water will be
w = (s_rej*d*M_water)/M_gas;//[g water/L gasoline]

printf(" The amount of water that will come out of
   the solution in the gasoline will be %f g water/L
    gasoline\n",w);
printf(" At 20 deg F we would expect this water to
   become solid ice, forming a piece large enough to
    plug the fuel line of a parked auto.");
```

Scilab code Exa 11.9 Distribution coefficients

```
clear;
//clc();

// Example 11.9
// Page: 290
printf("Example-11.9   Page no.-290\n\n");

//***Data***//
Temp = 25;//[C]
x_water = 5;//[mo]
x_benzene = 0.1;//[mol]

// The fugacity of the ethanol must be same in both
   phases so that we have distribution coefficient

// K = ( distribution coefficient ) = x_ethanol(
   water_rich phase)/x_ethanol(benzene-rich phase) =
    y_ethanol(benzene-rich phase)/y_ethanol(water-
   rich phase)
// Here distribution coefficient is defined as rhe
   ratio of mole fractions of the distributed solute
    between the two phases.

// We observe that for the first experimental data
   set in table 11.4 in the book(page 276)
x_ethanol_water_rich = 3.817;//[%]
x_ethanol_benzene_rich = 1.010;//[%]

// So
K = x_ethanol_water_rich/x_ethanol_benzene_rich;

// Now for all the data set in the table 11.4 in the
    book(page 276), we wiil draw a plot

X =
   [3.817,7.968,12.977,18.134,23.540,24.069,27.892,31.725,35.510,39.

Y =
   [1.010,3.323,5.860,9.121,12.939,13.340,16.090,18.943,22.444,26.21
Z = X./Y;

// Plotting the graph between 'Z' and 'Y'
plot(Y,Z);
xgrid();
xlabel("Mol% ethanol in benzene-rich phase ");
ylabel("Distribution coefficient of ethanol ,
   K_ethanol");

// We see from the plot that at the low mole percent
    of ethanol , the distribution coefficient is
   approximately
K_1 = 4;

// Thus ratio of the amount of the ethanol
   distributed in the two phases will be 4
//the amount in mol % is
// for water rich phase
m_water_rich = 100*K_1/(K_1+1);
m_benzene_rich = 100/(K_1+1);

printf(" Ethanol''s 0.1 mol distributed in the water
    rich phase will be  %f mol%% of the total mol\n
   ",m_water_rich);
printf(" Ethanol''s 0.1 mol distributed in the
   benzene rich phase will be %f mol%% of the total
   mol",m_benzene_rich);
```

Scilab code Exa 11.10 The experimental determination of LSE

```
clear;
//clc();

```

```
// Example 11.10
// Page: 293
printf("Example-11.10  Page no.-293\n\n");

//***Data***//
Temp = 20;//[C]
// At this temperature solubility of NaCl is
s = 36.0;//[g per 100 g of water]
M_NaCl = 58.5;//[g/mol] molecular weight of NaCl
M_water = 18;//[g/mol] molecular weight of water

// weight fraction of NaCl
w = s/(s+100);
// In weight percentage
w_percent = w*100;//[wt %]

// Mol fraction of the NaCl
x = (s/M_NaCl)/((s/M_NaCl)+(100/M_water));
// In mol percentage
x_percent = x*100;//[mol %]

printf(" Weight fraction of the NaCl in the
    saturated solution is %0.1f wt %%\n",w_percent);
printf(" Mol fraction of the NaCl in the saturated
    solution is    %0.0f mol %%\n",x_percent);
```

Scilab code Exa 11.11 `The experimental determination of LSE`

```
clear;
//clc();

// Example 11.11
// Page: 293
printf("Example-11.11  Page no.-293\n\n");

```

```
//***Data***//
T_inlet = 68;//[F]
T_outlet = 110;//[F]

// from the figure 11.13 we read that at 68F the
   solubility of CaCO3 and CaSO4.2H2O are
s_inlet_carbonate = 60;//[ppm]
s_inlet_sulphate = 2020;//[ppm]

// At 110F the solubility of the CaCO3 is
s_outlet_carbonate = 40;//[ppm]
// at 110F the least soluble form of the CaSO4 is
   anhydride with the solubility
s_outlet_sulphate = 2000;//[ppm]
 // This is close enough to the solubility of the
    gypsum at 68F
 // so we conclude that we would not expect either
    form of CaSO4 to prdcipitate

 // Thus total amount of the calcium carbonate which
      will cime out of the solution and will remain
    in the heater will be
 w = s_inlet_carbonate - s_outlet_carbonate;//[ppm]
 printf (" Total amount of the solid left behind in
    the heater will be        %0.1f ppm\n\n",w);

 // Now if a typical houshold water heater heats 100
      gallons/per day , we would expect to deposite
 w_per_day = w*10^(-6)*100*8.33;//[lb/day]
printf (" Total amount of the solid left behind in
   the heater per day will be %f lb/day",w_per_day);
```

Scilab code Exa 11.12 Estimation of the activity coefficients

```
clear;
```

```
//clc();

// Example 11.12
// Page: 298
printf("Example-11.12  Page no.-298\n\n");

//***Data***//
x_2 = 0.1;

//y_i_1 = (x_i__ideal /x_i_1) , at constant
   temperature
// From figures 11.15 and 11.16 given in the book (
   page 298-299) ( or the equations of the lines on
   those figures, presented in [14] ) we can
   compute the values in Table 11.6

// We see that at x_solute = 10%
// T_m/T for the solution in benzene at which log(
   x_experimental) = -1 is equal to 1.332
// and that for the solution in CCl4 is equal to
   1.288

//Now at the that value of the T_m/T
x_ideal_benzene = 0.114;
x_ideal_CCl4 = 0.152;


// In benzene the average these compounds is
y_i_1 = x_ideal_benzene/x_2;// corresponding to
   practically ideal solution

// and in benzene the average of these compounds is
y_i_2 = x_ideal_CCl4/x_2;// corresponding to mild
   type II behavior

printf(" Activity coefficient in benzene
   corresponding to practically ideal solution is %0
   .2f\n",y_i_1);
```

```
printf(" Activity coefficient in CCl4 corresponding
   to mild type II behavior          is %0.2f",y_i_2)
   ;
```

Scilab code Exa 11.13 Solubility of NaCl in water

```
clear;
//clc();

// Example 11.13
// Page: 299
printf("Example-11.13   Page no.-299\n\n");

//***Data***//

T = 273.15+20;//[K]

// The equation 11.15 (page 297) is given by
// log(1/(x_i_1*y_i_1)) = log(p_i_solid_phase/
   p_i_subcooled_liquid) = delta_h_solid_to_liquid/(
   R*T_melting_point)*(T_melting_point/T-1)

// Ignoring the moment the wild extraplation
   involved, we simply insert the appropriate values
T_m = 273.15+800;//[K]
delta_h_fusion = 30219;//[J/g]
R = 8.314;//[J/(mol*K)]

// Let log(1/(x_i_1*y_i_1)) = a
a = delta_h_fusion/(R*T)*(T_m/T-1);

// Now
x_NaCl_into_y_i_1 = 1/exp(a);

// If we make the plausible assumption that y_i_1 =
    1.00, then
x_NaCl = 1/exp(a)*1;
printf(" The solubility of the NaCl in water at 20
    deg C is %e \n", x_NaCl);
printf(" But the experimental value is 0.1, so,
    Similar to the results in book, our results are
    very far wrong");
```

Scilab code Exa 11.14 Gas solid equilibrium at low pressures

```
clear;
//clc();

// Example 11.14
// Page: 301
printf("Example-11.14  Page no.-301\n\n");

//***Data***//
P = 1*14.7;//[psia]
T = 30;//[F]
//******//
 //The vapour pressure of ice at 30F is 0.0808 psia
     i.e.
 p_ice = 0.0808;//[psia]
 // We may assume that the solubility of nitrogen
     and oxygen in solid ice is negligible
 //Thus
 x_water_in_ice = 1.00;
 //and thus use Raoult's law,finding
 y_water_vapour = x_water_in_ice*p_ice/P;
 printf(" Equilibrium concentration of water vapour
     in the air is %0.4f",y_water_vapour);
```

Scilab code Exa 11.15 GSE at high pressures

```
clear;
//clc();

// Example 11.15
// Page: 302
printf("Example-11.15  Page no.-302\n\n");

//***Data***//
T = 273.15+35;//[K]
p_d = 100;//[atm]
R = 82.06;//[(cm^(3)*atm)/(mol*K)]
 //******//

//The calculated vapour pressure of naphthalene at
   35C is
p_naphthalene = 0.00065;//[atm]
//The solid is practically pure naphthalene
x_naphthalene = 1.00;
//Total pressure
P = p_d;
//By Raoult's law
y_naphthalene = x_naphthalene*p_naphthalene/P;
//At this high a pressure the volume of solid
   naphthalene is
v = 132;//[cm^(3)/mol]
// We have equation log(f_d/f_c) = v/(R*T)*(p_d-p_c)
p_c = 1;//[atm]
f_d_by_f_c = exp(v/(R*T)*(p_d-p_c));
//and the estimated
y_naphthalene = f_d_by_f_c*y_naphthalene;
printf("Estimated solubility of naphthalene in CO2
   is %e",y_naphthalene);
```

Chapter 12

Chemical Equilibrium

Scilab code Exa 12.1 `Gibbs free energy`

```
clear;
//clc();

// Example 12.1
// Page: 311
printf("Example-12.1   Page no.-311\n\n");

//***Data***//
T = 298.15;//[K] temperature
P = 1;//[atm] pressure
R = 8.314*10^(-3);//[kJ/(mol*K)]

// For an ideal binary solution the Gibbbs free
    energy is given by
// g_mix = summation(x_i*g_i_0) + R*T*summation(x_i*
    log(x_i))
// Differentiating the above equation with respect
    to x_a , remembering that for a binary mixture
    dx_b = dx_a , finding

// dg_mix/dx_a = g_a_0-g_b_0+R*T*[log(x_a)+1-(log(
   x_b)+1)]
// and x_a+x_b = 1
// so
// dg_mix/dx_a = g_a_0-g_b_0+R*T*[log(x_a/(1-x_a))]

// setting up this equal to zero ( to find the
   minimum on the g-x curve ) and solving gives
// x_a/(1-x_a) = exp((g_b_0-g_a_0)/(R*T))

// From the table A.8 (page 427) reported in the
   book, pure component Gibbs free energies for
   isobutane,a,and n-butane,b, we find
g_a_0 = -20.9;//[kJ/mol]
g_b_0 = -17.2;//[kJ/mol]

// Now solving the above equation for x_a, we have
x_a = exp((g_b_0-g_a_0)/(R*T))/(1+exp((g_b_0-g_a_0)
   /(R*T)));
x_b = 1-x_a;
printf(" The chemical equilibrium composition of the
    gaseous mixture contains %f mol fraction
   isobutane\n \t\t\t\t\t\t\tand %f mol fraction n
   -butane",x_a,x_b);
```

Scilab code Exa 12.2 Calculation of the Equilibrium constants

```
clear;
//clc();

// Example 12.2
// Page: 319
printf("Example-12.2  Page no.-319\n\n");

//***Data***//
T = 298.15;//[K] temperature
```

```
P = 0.987;//[atm] pressure
g_0_NO = 86.6;//[kJ/mol] Free energy of formation
    the NO from elements
R = 8.314;//[J/(mol*K)]

// And the corresponding values for the elements N2
    and O2 are
g_0_O2 = 0.00;
g_0_N2 = 0.00;

// The reaction of the nitrogen and oxygen to form
    nitric oxide at 298.15 K is
// N2 + O2 = NO

//  Here
delta_g_0 = 2*g_0_NO - g_0_O2 - g_0_N2;//[kJ/mol]
// Changing in the J/mol
delta_g_01 = delta_g_0*1000;//[J/mol]

// hence
K_298 = exp((-delta_g_01)/(R*T));

// The activities are all
// a_i = f_i/f_i_0
// f_i_0 correspond to the standard state, which for
    gas at idael gas state are
f_0_N2 = 1;//[bar]
f_0_O2 = 1;//[bar]
f_0_NO = 1;//[bar]

// If we make the most general statement of the
    activities (for gases ) we would have
// a_i = y_i*v_i*Y_i*P/f_i_0 = y_i*phi*P/f_i_0

// At this low pressure we may safely asssume that
    the NO,O2 and N2 behave as ideal gases for which
    v_i*Y_i = phi = 1.00 and substituting these we
    find
```

```
// K_298 = [a_NO]^(2)/([a_N2]*[a_O2]) = [y_NO]^(2)
   /([y_N2]*[y_O2])

// Now using this equilibrium constant we can
   calculare he equilibrium concentratin of NO in
   the air sample in which
//oxygen = 21%, nitrogen = 78% and argon = 1% ,so
y_N2 = 0.78;
y_O2 = 0.21;

// Hence From above expression , we have
y_NO_298 = sqrt(K_298*y_N2*y_O2);

// Making the similar calculations for the
   temperature 2000 K, we  have
T_1 = 2000;//[K]
K_2000 = exp((-delta_g_01)/(R*T_1));

// So,
y_NO_2000 = sqrt(K_2000*y_N2*y_O2)*10^(6);//[ppm]

printf(" The equilibrium constant for the reaction
   at 298.15 K is \t\t\t %e\n",K_298);
printf(" The concentration of NO at equilibrium at
   temperature 298.15 K is \t\t%e\n",y_NO_298);
printf(" The equilibrium constant for the reaction
   at 2000 K is \t\t\t %e\n",K_2000);
printf(" The concentration of NO at equilibrium at
   temperature 2000 K is \t\t%f ppm",y_NO_2000);
```

Scilab code Exa 12.3 Change of reactant concentration

```
clear;
//clc();

```

```
// Example 12.3
// Page: 321
printf("Example-12.3  Page no.-321\n\n");

//***Data***//
Temp = 2000;//[K]
n_air = 1;//[mol] no of moles of the air

// Let the moless of the NO formed be 2*x
// Then at equilibrium the unreacted moles of the N2
    and O2 will be (0.78-x) and (0.21-x)
   respectively

// from the previous example, we have
// [y_NO]^(2) = K_298*[y_N2]*[y_O2]
// here
K_2000 = 4*10^(-4);
// Substituting all the values, we have
// (2*x)^(2) = K_2000*(0.78-x)*(0.21-x)

//Now
deff('[y]=f(x)','y = (2*x)^(2) - K_2000*(0.78-x)
   *(0.21-x)');
//deff('[y]=f(x)','y = (K_2000-2)*x^(2)-K_2000
   *(0.78+0.21)*x+K_2000*0.78*0.21');
x = fsolve(0,f);
// Here negative root is meaningless,so
// concentration of NO
c_NO = 2*x*10^(6);//[ppm]
// now
p = c_NO/8100*100;
printf(" The calculated NO cocentration is %f ppm,
   which %f%% of the value computed in example 12.1"
   ,c_NO,p);
```

Scilab code Exa 12.4 `Mass action law`

```
clear;
//clc();

// Example 12.4
// Page: 323
printf("Example-12.4   Page no.-323\n\n");

//***Data***//
n_water_0 = 0.833;//[mol]
n_ethylene_0 = 1;//[mol]
n_ethanol_0 = 0;//[mol]
n_T_0 = (n_water_0+n_ethylene_0+n_ethanol_0);//[mol]

// In general, we must have
// K = [a_C2H5OH]/([a_C2H4]*[a_H2O])

// Here we will assume that we have an ideal
   solution of the ideal gases for which in equation
    12.18 (page 320), we have
// v_i*Y_i = phi = 1.00 , and that for each reactant
    or product f_i_0 = 1 bar so that
// [a_C2H5OH]/([a_C2H4]*[a_H2O]) = K = [x_C2H5OH*P
   /(1 bar)]/([x_C2H4*P/(1 bar)]*[x_H2O*P/(1 bar)])
// So

// K = [x_C2H5OH]/([x_C2H4]*[x_H2O])*(1 bar)/P
// Here the stoichiometric coefficients are -1,-1
   and +1, so that summation(v_i) = -1 and

//  [(0+e)/(1.833-e)]/([(1-e)/(1.833-e)]*[(0.833-e)
   /(1.833-e)]) = K*P/(1 bar)
printf("''The mass action law '' statement for the
   given reaction is\n\n [(0+e)/(1.833-e)]/([(1-e)
   /(1.833-e)]*[(0.833-e)/(1.833-e)]) = K*P/(1 bar)"
   )
```

Scilab code Exa 12.5 Mass action law

```
clear;
//clc();

// Example 12.5
// Page: 324
printf("Example-12.5  Page no.-324\n\n");

//***Data***//
Temp = 298;//[K]
K = 29.6;// equilibrium constant at 298 K
P = 1;//[bar]
n_water_0 = 0.833;//[mol]
n_ethylene_0 = 1;//[mol]
n_ethanol_0 = 0;//[mol]
n_T_0 = (n_water_0+n_ethylene_0+n_ethanol_0);//[mol]

// From the previous example, we have
// [(0+e)/(1.833-e)]/([(1-e)/(1.833-e)]*[(0.833-e)
    /(1.833-e)]) = K*P/(1 bar)
// let y = [(0+e)/(1.833-e)]/([(1-e)/(1.833-e)
    ]*[(0.833-e)/(1.833-e)])- K*P/(1 bar)
deff('[y]=f(e)','y = [(0+e)/(1.833-e)]/([(1-e)
    /(1.833-e)]*[(0.833-e)/(1.833-e)])-K*P/(1)');
e_1 = fsolve(0,f);
e_2 = fsolve(0.5,f);

// Here the root 'e_2' is meaningless, Then
y_ethanol = [(0+e_2)/(1.833-e_2)];
y_ethylene = [(1-e_2)/(1.833-e_2)];
y_water = [(0.833-e_2)/(1.833-e_2)];

printf("Concentration of the ethylene at the
    equilibrium is %f\n",y_ethylene);
printf(" Concentration of the water at the
    equilibrium is   %f\n",y_water);
printf(" Concentration of the ethanol at the
    equilibrium is  %f",y_ethanol);
```

Scilab code Exa 12.6 Reversible reaction

```
clear;
//clc();

// Example 12.6
// Page: 324
printf("Example-12.6  Page no.-324\n\n");

//***Data***//
Temp = 273.15+25;//[C]
P = 1;//[bar]
R = 8.314;//[J/(mol*K)]

// We have the reaction as
//  H2 + 0.5O2 = H2O
// Using values of the Gibbs free energies of
   formation in the Table A.8(page 427) we have
g_H2O_0 = -237.1;//[kJ/mol]
g_O2_0 = 0;//[kJ/mol]
g_H2_0 = 0;//[kJ/mol]
// now
delta_g_0 = g_H2O_0 - 0.5*g_O2_0-g_H2_0;//[kJ/mol]
// expressing delta_g_0 in [J/mol]
delta_g_1 = delta_g_0*1000;//[J/mol]

// and
K = exp((-delta_g_1)/(R*Temp));

```

```
// And we have
// K = [a_H2O]/([a_H2]*[a_O2]^(0.5))
// Here we will again assume as in the previous
   example that we have an ideal solution of the
   ideal gases for which in equation 12.18 (page
   320),we have
// v_i*Y_i = phi = 1.00 , and that for each reactant
    or product f_i_0 = 1 bar, putting all the values
    and simplifying

// K = [y_H2O]/([y_H2]*[y_O2]^(0.5))*((1 bar)/P)
   ^(0.5)
// Choosing oxygen as the selected reactant, and
   assuming that we begin with 0.5 mol of oxygen and
    1 mol of hydrogen,
// we have the stoichiometric coefficients of -1,
   -0.5 and +1
// and
n_T_0 = 1.5;//[mol]
// Also summation(v_i) = -0.5

// Thus
// K = [e/(1.5-0.5*e)]/([(1-e)/(1.5-0.5*e)
   ]*[(0.5-0.5*e)/(1.5-0.5*e)]^(0.5))

// Now
//   deff('[y]=f(e)','y =[e/(1.5-0.5*e)]/([(1-e)
   /(1.5-0.5*e)]*[(0.5-0.5*e)/(1.5-0.5*e)]^(0.5))');
//   e = fsolve(.99999,f);
// e = (1-2.4e-28);

// So the equilibrium concentration of the hydrogen
   and oxygen are as
// y_H2 = [(1-e)/(1.5-0.5*e)];
// y_O2 = [(0.5-0.5*e)/(1.5-0.5*e)];
// These values are so less that scilab consol is
   displaying them zero, however we get
y_H2 = 2.4e-28;
```

```
y_O2 = 0.5*y_H2;

printf(" The equilibrium mol fraction of the
    hydrogen is   %0.3e\n",y_H2);
printf(" And the equilibrium mol fraction of the
    oxygen is %e",y_O2);
```

Scilab code Exa 12.7 Standard state Gibbs free energy

```
clear;
//clc();

// Example 12.7
// Page: 327
printf("Example-12.7  Page no.-327\n\n");

//***Data***//
Temp = 298.15;//[K]
Press = 1*10^(5);//[Pa]
R = 8.314;//[J/(mol*K)]

// We will calculate the free energy change from
    liquid to hypothetical gas in three steps
// 1) The liquid is reduced in pressure from the
    standard pressure of 1 bar to its vapour pressure
     at 298.15K and for this cange in the state we
    have
v_liquid = 1.805*10^(-5);//[m^(3)/mol] this liquid
    specific volume and we will treat is as a
    constant

// The vapour preesure of the water 25C is given as
P_vapour_25 = 0.0317*10^(5);//[Pa]

// thus change in the Gibbs free energy is
```

```
delta_g_0_1 = integrate('v_liquid*P^(0)','P',Press,
   P_vapour_25);//[J/mol]

// 2) In the second step the liquid is vaporized at
   that pressure, for which
delta_g_0_2 = 0;//[J/mol]
// because this is an equilibrium vaporization.

// 3) And in this last step the vapour is replaced
   by an ideal gas, which  will not condence, and
   compressed from the vapour pressure at 298.15K to
    1 bar
// In this case the specific volume v_ideal of the
   ideal gas is replaced by the ideal gas law viz. (
   R*T)/P
delta_g_0_3 = (R*Temp)*integrate('1/P','P',
   P_vapour_25,Press);//[J/mol]

// Thus total change in free energy is
delta_g_0 = delta_g_0_1+delta_g_0_2+delta_g_0_3;//[J
   /mol]
//expressing the result in kJ/mol
delta_g_1 = delta_g_0/1000;//[kJ/mol]

printf(" Total change in the free energy of water,
   going under given conditions, is %0.2f kJ/mol\n\n
   ",delta_g_1);

// From Table A.8 we find
delta_g_0_ideal_gas = -228.6;//[kJ/mol]
delta_g_0_liquid = -237.1;//[kJ/mol]
// So
delta_g_o = delta_g_0_ideal_gas-delta_g_0_liquid;//[
   kJ/mol]

printf(" From the values of Table A.8 given in the
   book, the free energy change is %0.2f kJ/mol",
   delta_g_o);
```

Scilab code Exa 12.8 Effect of temperature on chemical reaction equilibrium

```
clear;
//clc();

// Example 12.8
// Page: 330
printf("Example-12.8  Page no.-330\n\n");

//***Data***//

T1 = 273.15+25;//[K]
T2 = 273.15+400;//[K]
R = 8.314;//[J/(mol*K)]

// Using the table A.8, we have
// Gibb's free energy of the various species at
    298.15 K are
g0_NH3 = -16.5;//[kJ/mol]
g0_N2 = 0;//[kJ/mol]
g0_H2 = 0;//[kJ/mol]

// We have the reaction as
// 0.5N2 + 1.5H2 = NH3

// So, Gibb's free energy change in the reaction is
    given as
delta_g_0 = g0_NH3 - 0.5*g0_N2 - 1.5*g0_H2;//[kJ/mol
    ]

// and
K_1 = exp(-delta_g_0*1000/(R*T1));// Equilibrium
    constant of the reaction at temperature 298.15 K

```

```
// Similarly enthalpy of the various species are
h0_NH3 = -46.1;//[kJ/mol]
h0_N2 = 0;//[kJ/mol]
h0_H2 = 0;//[kJ/mol]

// So, enthalpy change of the reaction is given as
del_h_1 = h0_NH3 - 0.5*h0_N2 - 1.5*h0_H2;//[kJ/mol]

// Now, from Table 12.3( page 332 )
a_NH3 = 3.578;
a_H2 = 3.249;
a_N2 = 3.280;
b_NH3 = 3.020*10^(-3);//[1/K]
b_H2 = 0.422*10^(-3);
b_N2 = 0.593*10^(-3);
c_NH3 = 0;//[1/K^(2)]
c_H2 = 0;//[1/K^(2)]
c_N2 = 0;//[1/K^(2)]
d_NH3 = -0.186*10^(5);//[K^(2)]
d_H2 = 0.083*10^(5);//[K^(2)]
d_N2 = 0.040*10^(5);//[K^(2)]

// So,
del_a = a_NH3 - 0.5*a_N2 - 1.5*a_H2;
del_b = b_NH3 - 0.5*b_N2 - 1.5*b_H2;
del_c = c_NH3 - 0.5*c_N2 - 1.5*c_H2;
del_d = d_NH3 - 0.5*d_N2 - 1.5*d_H2;

// Now, enthalpy change of the reaction at any other
   temparature is given by
//  del_h = del_h_1 + R*( integrate( del_a + del_b*T
   + del_c*T^(2) + del_d/T^(2) )*dT)  with lower
   limit 'T_1' and upper limit 'T'
// Integrating and putting the limits, we have
//  del_h = del_h_1 + R*( del_a*T + del_b*T^(2)/2 +
   del_c*T^(3)/3 - del_d/T) - R*( del_a*T_1 + del_b*
   T_1^(2)/2 + del_c*T_1^(3)/3 - del_d/T_1)
// let
```

```
I = R*( del_a*T1 + del_b*T1^(2)/2 + del_c*T1^(3)/3 -
    del_d/T1);//[J/mol]

// From equation 12.28 and above relations we have
// log(K_2/K_1) = 1/R*( integrate( del_h_1 - I + R*(
   del_a*T + del_b*T^(2)/2 + del_c*T^(3)/3 - del_d/T
   ))/T^(2)*dT)   with limits T1 and T2
// Let log(K_2/K_1) = X, So,
X = (1/R)*integrate('( del_h_1*1000 - I + R*(del_a*T
    + del_b*T^(2)/2 + del_c*T^(3)/3 - del_d/T))/T
   ^(2)','T',T1,T2);

// So,
K_2 = K_1*exp(X);

printf(" Equilibrium constants for the formation of
   ammonia from hydrogen and nitrogen are \n\n");
printf(" K = %0.0f at temperature 25 deg C\n\n",K_1)
   ;
printf(" K = %f at temperature 400 deg C\n",K_2);
```

Scilab code Exa 12.9 Equilibrium conversion of a mixture

```
clear;
//clc();

// Example 12.9
// Page: 335
printf("Example-12.9  Page no.-335\n\n");

//***Data***//
// Initial moles of the gases are
n_H2_0 = 1.5;//[mol]
n_N2_0 = 0.5;//[mol]
n_NH3_0 = 0;//[mol]
```

```
T_1 = 298.15;//[K]
T_2 = 673.15;//[K]
P = 1;//[bar]

// We start with the equation as
// [f_NH3/f_0_NH3]/([f_N2/f_0_N2]^(0.5)*[f_H2/f_0_H2
   ]^(1.5)) = K

// For a pressure of 1 bar with the assumption of
   ideal solution of ideal gases and standard state
   fugacities of 1 bar,
// a_i = [f_i/f_0_i] = [P*y_i/(1 bar)] = y_i
// The equilibrium relation is given by
// K = [y_NH3]/([y_N2]^(0.5)*[y_H2]^(1.5))

// We have the stoichiometric coefficient of N2, H2
   and NH3 as -0.5, -1.5 and +1 respectively, so
   summation(v_i) = -1
// Now using the equilibrium relations which are
   Equations 12.W, 12.X and 12.Y ( page 322 ), we
   have

// K = ((0+e)/(2-e))/(((0.5-0.5*e)/(2-e))^(0.5)
   *((1.5-1.5*e)/(2-e))^(1.5))
// Form the example 12.8 of this book we know that
K_298 = 778;// at temperature 298.15K
K_673 = 0.013;// at temperature 673.15K

// Solving for temperature 298.15
deff('[y]=g(e_1)','y = ((0+e_1)/(2-e_1))/(((0.5-0.5*
   e_1)/(2-e_1))^(0.5)*((1.5-1.5*e_1)/(2-e_1))^(1.5)
   )-K_298');
e_1 = fsolve(0.97,g);
y_NH3_298 = e_1/(2-e_1);

// Similarily solving for temperature 673.15K
deff('[y]=h(e_2)','y = ((0+e_2)/(2-e_2))/(((0.5-0.5*
   e_2)/(2-e_2))^(0.5)*((1.5-1.5*e_2)/(2-e_2))^(1.5)
```

```
    )-K_673');
e_2 = fsolve(0,h);
y_NH3_673 = e_2/(2-e_2);

printf(" The mole fraction of NH3 in the equilibrium
     at the temperature 298.15K and 1 bar is %f\n",
    y_NH3_298);
printf(" The mole fraction of NH3 in the equilibrium
     at the temperature 673.15K and 1 bar is %f",
    y_NH3_673);
```

Scilab code Exa 12.10 `Ideal solution of ideal gases`

```
clear;
//clc();

// Example 12.10
// Page: 337
printf("Example-12.10  Page no.-337\n\n");

//***Data***//
Temp = 273.15+400;//[K]
P = 150*1.01325;//[bar]

// Comparing this with the example 12.9, we see that
     we can use the same equation , but K_673 is
    replaced by K_673*(P/(1bar))^(1.5+0.5-1)
K_673 = 0.013;

// So
K = K_673*(P/1)^(1.5+0.5-1);

// We have
// K = ((0+e)/(2-e))/(((0.5-0.5*e)/(2-e))^(0.5)
    *((1.5-1.5*e)/(2-e))^(1.5))
```

```
deff('[y]=f(e)','y = ((0+e)/(2-e))/((((0.5-0.5*e)/(2-
    e))^(0.5)*((1.5-1.5*e)/(2-e))^(1.5))-K');
e=fsolve(0.5,f);

// Thus mole fraction of the ammonia in the gas is
    given by
y_NH3 = (0+e)/(2-e);

printf("The mole fraction of the ammonia in the
    equilibrium is %0.2f",y_NH3);
```

Scilab code Exa 12.11 Non ideal solution non ideal gases

```
clear;
//clc();

// Example 12.11
// Page: 338
printf("Example-12.11  Page no.-338\n\n");

//***Data***//
// The data used in this example will e same as in
    the example 12.10
T = 273.15+400;//[K] given temperature
P = 150*1.01325;//[bar] given pressure

// Here again the equation will be same as in the
    example 12.9 like we used in the example 12.10
    only K_673 is replaced by (K/K_v)*[P/(1 bar)
    ]^(1.5+0.5-1)
K_673 = 0.013;
// The value of 'K_v' is calculated by the equation
    12.BN, which is
// log10(1/K_v) = (0.1191849/T + 91.87212/T^(2) +
    25122730/T^(4))*P
```

```
// So
K_v = (10^((0.1191849/T + 91.87212/T^(2) + 25122730/
    T^(4))*P))^(-1);

// Thus
K = (K_673/K_v)*[P/1]^(1.5+0.5-1);

// Now from the previous example we have
// K = ((0+e)/(2-e))/(((0.5 -0.5*e)/(2-e))^(0.5)
    *((1.5 -1.5*e)/(2-e))^(1.5))

deff('[y]=f(e)','y = ((0+e)/(2-e))/(((0.5 -0.5*e)/(2-
    e))^(0.5)*((1.5 -1.5*e)/(2-e))^(1.5))-K');
e = fsolve(0.2,f);

// Mol fraction of the ammonia in the gas phase in
    the equilibrium is given by
y_NH3 = (0+e)/(2-e);

printf(" The mole fraction of the ammonia in the
    equilibrium is %0.2f",y_NH3);
```

Scilab code Exa 12.12 Liquids and solids

```
clear;
//clc();

// Example 12.12
// Page: 340
printf("Example-12.12   Page no.-340\n\n");

//***Data***//
p_i = 1;//[atm] initial pressure
P = 150;//[atm] final pressure
T = 273+25;//[K] Given temperature
```

```
R = 8.314;//[J/(mol*K)]

// Now ignoring the difference between 25C and 20C,
   we use the values given in the table A.8 (page
   427) to get
delta_g_0 = 10.54*1000;//[J/mol]
// And thus
K = exp((-delta_g_0)/(R*T));

// Now the chemical reaction is given by
// C2H5OH + CH3COOH = C2H5OOC2H5 + H2O

// Let we start with 1 mol each of ethanol and
   acetic acid, and at equilibrium 'e' moles each of
    the reactants reacted, then
// remaining amount of each of the two reactants
   will be (1-e) and that products formation will be
    'e' mol each

// We have
// K = (a_C2H5OOC2H5*a_H2O)/(a_C2H5OH*a_CH3COOH) = (
   x_C2H5OOC2H5*x_H2O)/(x_C2H5OH*x_CH3COOH) = (e*e)
   /((1-e)*(1-e))
// Now solving for 'e'
deff('[y]=f(e)','y = (e*e)/((1-e)*(1-e))-K');
e = fsolve(0,f);

// To see the effect of changing the pressure we
   first compute the volume increase of the reaction
// delta_v = v_C2H5OOC2H5 + v_H2O - v_C2H5OH -
   v_CH3COOH, where v_i is the molar volume of the
   ith component
// From the Table 12.4(page 340), we have
v_C2H5OOC2H5 = 97.67;//[ml/mol]
v_H2O = 18.03;//[ml/mol]
v_C2H5OH = 58.30;//[ml/mol]
v_CH3COOH = 57.20;//[ml/mol]

```

```
// Thus volume increase of the reaction is
delta_v = v_C2H5OOC2H5 + v_H2O - v_C2H5OH -
   v_CH3COOH;//[ml/mol]

// So, from Le Chatelier's principal, on increasing
   the pressure , the reaction is forced in the
   direction of the reactant or away from the
   product
// To calculate the extent of shifting we will take
   the help of the activity of each of the four
   component
// a_i = (f_i/f_i_0) = (x_i*Y_i*p_i)/p_i*exp(v/(R*T)
   *(P-p_i))
// we will assume that this is an ideal solution so
   that Y_i = 1.00, for every component

// Now substituting the activity of each component
   in the expression of the equilibrium constant
   given above, we have
// K = (x_C2H5OOC2H5*x_H2O)/(x_C2H5OH*x_CH3COOH)*exp
   [(delta_v)/(R*T)*(P-p_i)]
// or
// K = (e_1*e_1)/((1-e_1)*(1-e_1))*exp[(delta_v)/(R*
   T)*(P-p_i)]

// Solving for 'e_1'
deff('[y]=g(e_1)','y = (e_1*e_1)/((1-e_1)*(1-e_1))*
   exp((delta_v)/(R*T)*(P-p_i))-K');
e_1 = fsolve(0.2,g);

// Now if we carry out the calculation to enough
   significant figures then
a = e_1/e;

// It indicates that e_1 is 'a' times of that of the
    e
printf("On increasing the pressure from 1 atm to 150
    atm, the reacted amount of the equimolar
   reactants at equilibrium becomes %f times of
   initial",a);
```

Scilab code Exa 12.13 `Equilibrium constant Kp`

```
clear;
//clc();

// Example 12.13
// Page: 342
printf("Example-12.13   Page no.-342\n\n");

//***Data***//
P = 150;//[atm] given pressure
T = 400;//[C] temperature
// Using the values from the example 12.11, we know
    that
K = 0.013;
K_v = 0.84;
delta_v = 1.5+0.5-1;

// so
// K_p = (K/K_v)*[1/bar]^(-summation(v_i)) = (K/K_v)
    *[1/bar]^(delta_v)

K_p = (K/K_v)*[1/1]^(delta_v);//[1/bar]

printf(" Value of the K_p at the given condition is
    %f (1/bar)\n\n",K_p);

printf (" The basic K is dimensionless, but K_p has
    the dimensions of pressure to the power.")
```

Chapter 13

Equilibrium In Complex Chemical Reactions

Scilab code Exa 13.1 `Reactions Involving Ions`

```
clear;
//clc();

// Example 13.1
// Page: 349
printf("Example-13.1   Page no.-349\n\n");

//***Data***//
T =273.15+25;//[K] given temperature
R = 8.314;//[J/(mol*K)] universal gas constant

// We have the reaction as follows
// H2O = H+ + OH-

// Reading the free energy of species from the Table
    A.8 ( page 427), we have
g_0_H = 0;//[kJ/mol]
g_0_OH = -157.29;//[kJ/mol]
g_0_H2O = -237.1;//[kJ/mol]
```

```

// Thus free enaergy change of the reaction is
delta_g_0 = g_0_H + g_0_OH - g_0_H2O;//[kJ/mol]
// Changing in J/mol we have
delta_g_1 = delta_g_0*1000;//[J/mol]

// Now equilibrium constant of the reaction is given
   by
K = exp((-delta_g_1)/(R*T));

// Also, in terms of activity
// K = ([[H+]/(1 molal)]*[[OH-]/(1 molal)])/[a_water
   ]
// The activity of any pure liquid at its standard
   state is 1.00, and here water is practically pure
   , so
// K_w = [[H+]/(1 molal)]*[[OH-]/(1 molal)] = K
// or
K_w = K;

printf("At the equilibrium the product of the
   hydrogen ion and hydroxil ion is %0.1e",K_w);
```

Scilab code Exa 13.2 Sequential reactions

```
clear;
//clc();

// Example 13.2
// Page: 351
printf("Example-13.2  Page no.-351\n\n");

//***Data***//
n_H2SO4 = 1;//[mol] mole of the sulphuric acid
w_water = 1000;//[g] weight of the water
```

```
T =273.15+25;//[K] temperature
R = 8.314;//[J/(mol*K)]

// We the two sequential reaction , in which the
   first reaction is
// H2SO4 = HSO4- + H+

// From the Table A.8 (page 427) as given in the
   book , free energy of the above species are
g_0_H = 0;//[J/mol] free energy of the hydrogen ion
g_0_HSO4 = -756.01*1000;//[J/mol] free energy of the
    bisulphate ion
g_0_H2SO4 = -744.50*1000;//[J/mol] free enery of
   sulphuric acid

// So
delta_g_0 = g_0_H + g_0_HSO4 - g_0_H2SO4;//[J/mol]

// So equilibrium constant of the reaction is given
   by
K_1 = exp((-delta_g_0)/(R*T));

// Now the second reaction is which is going on
   sequentialy is
// HSO4- = SO4(-2) + H+

// Again from the Table A.8 reading the values of
   free energy of the species of the above reaction ,
    we have
g_0_H = 0;//[J/mol] free energy of the hydrogen ion
g_0_SO4 = -744.62*1000;;//[J/mol] free energy of
   sulphate ion
g_0_HSO4 = -756.01*1000;//[J/mol] free energy of the
    bisulphate ion

// So
delta_g_1 = g_0_H + g_0_SO4 - g_0_HSO4;//[J/mol]

```

```
// Equilibrium constant of thi reaction is
K_2 = exp((-delta_g_1)/(R*T));

// Now we have 1 mol of H2SO4 initially. Let e_1 mol
    of H2SO4 ionised at equilibrium
// Then amount of the each of two product i.e.
   bisulphate and hydrogen ion will be e_1 mol
// Now for the second reaction e_1 mol of the
   bisulphate ion will be treated as  initial
   concentration.
// If at equilibrium e_2 moles of bisulphate ion has
    ionised
// In this case the amount of each of two product of
    this reaction will be e_2 mol
// So final amount of each of the species (in moles)
    at equilibrium is given as
// n_H2SO4 = (1-e_1)
// n_HSO4 = (e_1-e_2)
// n_SO4 = e_2
// n_H = (e_1+e_2)

// now
// K_1 = ([HSO4]*[H])/[H2SO4] = ((e_1-e_2)*(e_1+e_2)
   )/(1-e_1)....................(1)
// and that for the second reaction
// K_2 = ([SO4]*[H])/[HSO4] = ((e_2)*(e_1+e_2))/(e_1
   -e_2)........................(2)

// e = [e_1 e_2]
// Solving the two given simultaneous equations,we
   have
function[f]=F(e)
    f(1) = ((e(1)-e(2))*(e(1)+e(2)))/(1-e(1)) - K_1;
    f(2) = ((e(2))*(e(1)+e(2)))/(e(1)-e(2)) - K_2;
    funcprot(0);
endfunction

// Initial guess:
```

```
e = [0.8 0.1];
y = fsolve(e,F);
e_1 = y(1);
e_2 = y(2);

// So, concentration of the various species in
   equilibrium is given as
m_H2SO4 = 1-e_1;// [molal]
m_HSO4 = e_1 - e_2;//[molal]
m_SO4 = e_2;//[molal]
m_H = e_1 + e_2;//[molal]

printf(" The equilibrium concentration of H2SO4 in
   terms of molality is %f molal\n",m_H2SO4);
printf(" The equilibrium concentration of HSO4- in
   terms of molality is %f molal\n",m_HSO4);
printf(" The equilibrium concentration of SO4-- in
   terms of molality is %f molal\n",m_SO4);
printf(" The equilibrium concentration of H+ in
   terms of molality is   %f molal",m_H);
```

Scilab code Exa 13.3 Simultaneous reactions

```
clear;
//clc();

// Example 13.3
// Page: 352
printf("Example-13.3  Page no.-352\n\n");

//***Data***//
P = 10;//[MPa] given pressure
T = 250;//[C] Temperature
// Let the total number of moles in the feed be one,
    then
```

```
n_T_0 = 1;//[mol]
n_CO = 0.15;//[mol]
n_CO2 = 0.08;//[mol]
n_H2 = 0.74;//[mol]
n_CH4 = 0.03;//[mol]

// The two simultaneous reactions taking place are
// CO + 2*H2 = CH3OH
// CO2 + H2  = CO + H2O

// Let us denote the first reaction by 1 and the
   second reaction by 2
// and K_i = (K/K_v)*[P/(1 atm)]^(-summation(v_i))
// and that    summation(v_i) = V_i

// Then from the table 13.C (page 353) as reported
   in the book, we have
V_1 = -2;
V_2 = 0;
K_1 = 49.9;// For the first reaction
K_2 = 0.032;// For the second reaction

// Now let v_i denotes the stoichiometric
   coefficient of species 'i', then
v_CO_1 = -1;
v_H2_1 = -2;
v_CH3OH_1 = +1;
v_CO2_2 = -1;
v_H2_2 = -1;
v_CO_2 = +1;
v_H2O_2 = +1;

// Let e_1 = the moles of CO reacted in reaction 1
   and e_2 = the moles of CO2 reacted in reaction 2.
// Now mol fractions of each of the species in the
   equilibrium is
// y_CO = (n_CO+v_CO_1*e_1+v_CO_2*e_2)/(n_T_0+e_1*
   V_1+e_2*V_2) = (0.15-1*e_1+1*e_2)/(1+e_1*(-2)+e_2
```

```
    *(0)) = (0.15 - e_1 + e_2)/(1 - 2*e_1)

// similarily
// y_H2 = (n_H2+v_H2_1*e_1+v_H2_2*e_2)/(n_T_0+e_1*
    V_1+e_2*V_2) = (0.74 - 2*e_1 - e_2)/(1 - 2*e_1)

// y_CH3OH = (n_CH3OH+v_CH3OH_1*e_1+v_CH3OH_2*e_2)/(
    n_T_0+e_1*V_1+e_2*V_2) = (0 + e_1)/(1 - 2*e_1)

// y_CO2 = (n_CO2+v_CO2_1*e_1+v_CO2_2*e_2)/(n_T_0+
    e_1*V_1+e_2*V_2) = (0.08 - e_2)/(1 - 2*e_1)

// y_H2O = (n_H2O+v_H2O_1*e_1+v_H2O_2*e_2)/(n_T_0+
    e_1*V_1+e_2*V_2) = (0 + e_2)/(1 - 2*e_1)

// Now putting the values in the expression of the
    equilibrium constant of the reactions , for the
    reaction 1 we have

// K_1 = ((0 + e_1)/(1 - 2*e_1))/(((0.15 - e_1 + e_2
    )/(1 - 2*e_1))*((0.74 - 2*e_1 - e_2)/(1 - 2*e_1))
    ^(2))

// K_2 = (((0.15 - e_1 + e_2)/(1 - 2*e_1))*((0 + e_2
    )/(1 - 2*e_1)))/(((0.08 - e_2)/(1 - 2*e_1))
    *((0.74 - 2*e_1 - e_2)/(1 - 2*e_1)))

// e = [e_1 e_2]
// Solving the two given simultaneous equations ,we
    have
function[f]=F(e)
    f(1) = ((0 + e(1))/(1 - 2*e(1)))/(((0.15 - e(1)
        + e(2))/(1 - 2*e(1)))*((0.74 - 2*e(1) - e(2))
        /(1 - 2*e(1)))^(2)) - K_1;
    f(2) = (((0.15 - e(1) + e(2))/(1 - 2*e(1)))*((0
        + e(2))/(1 - 2*e(1))))/(((0.08 - e(2))/(1 -
        2*e(1)))*((0.74 - 2*e(1) - e(2))/(1 - 2*e(1))
        )) - K_2;
    funcprot(0);
endfunction

// Initial guess:
e = [0.109 0];
y = fsolve(e,F);
e_1 = y(1);
e_2 = y(2);

// So, percent conversion of CO2 is given as
// (moles of CO2 reacted)/(moles of CO2 fed) i.e.
c_CO2 = e_2/(n_CO2)*100;
// Number of moles of CO Formed by the second
   reaction is 0.032
// So, percent conversion of CO is given as
c_CO = e_1/(n_CO + 0.032)*100;

printf(" Percent conversion of CO is %f%%\n",c_CO);
printf(" Percent conversion of CO2 is %f%%",c_CO2);
```

Scilab code Exa 13.4 Solubility product

```
clear;
//clc();

// Example 13.4
// Page: 354
printf("Example-13.4   Page no.-354\n\n");

//***Data***//
T = 273.15+25;//[K] Temperature
R = 8.314;//[J/(mol*K)] universal gas constant

// Solubility of AgCl in water follows
// AgCl = Ag+  +  Cl-
// From the Table A.8, free energy of above species
    are
g_0_Ag = 77.12*1000;//[J/mol]
g_0_Cl = -131.26*1000;//[J/mol]
g_0_AgCl = -109.8*1000;//[J/mol]

// Free energy change of the reacton is given by
delta_g_0 = g_0_Ag + g_0_Cl - g_0_AgCl;//[J/mol]

// Now equilbrium constant of the reaction is given
    by
K = exp((-delta_g_0)/(R*T));

// In terms of activity of the components,
    equilibrium constant is
// K = [[Ag+]/(1 molal)*[Cl-]/(1 molal)]/[a_AgCl]

// For solids f_i_0 is normaly taken as the fugacity
     of the pure crystalline solid ,and the activity
    of the pure crystalline solid is = 1.00, so
a_AgCl = 1.00;
// hence
// [[Ag+]/(1 molal)*[Cl-]/(1 molal)]= K = K_sp ,
    solubility product
printf("The amount of solid dissolved in terms of
    solubility product is %0.2e",K);
```

Scilab code Exa 13.5 Gas liquid reactions

```
clear;
//clc();

// Example 13.5
// Page: 357
printf("Example-13.5  Page no.-357\n\n");
```

```

//***Data***//
T = 273.15+25;//[K] Given temperature of air
P = 1;//[atm] Pressure of the air
y_CO2 = 350*10^(-6);// Amount of CO2 present in air
    at the given condition
R = 8.314;//[J/(mol*K)]

// At equilibrium there are two ionisition reactions
     takin place sequentily
// First ionisation reaction is
// H2CO3 = H+ + HCO3-
// Free energy of the species of the above reation
    is
g_0_H2CO3 = -623.1*1000;//[J/mol]
g_0_H = 0;//[J/mol]
g_0_HCO3 = -586.85*1000;//[J/mol]

// So free energy change of the reaction is given by
delta_g_0 = g_0_H + g_0_HCO3 - g_0_H2CO3;//[J/mol]
// Equilibrium constant of the reaction is given by
K_1 = exp((-delta_g_0)/(R*T));

// And the second one is
// HCO3- = H+ + CO3(-2)
// Free energy of the species of the second reacion
    are
g_0_CO3 = -527.89*1000;//[J/mol]

// Free energy change of the second reacion is
delta_g_1 = g_0_H + g_0_CO3 - g_0_HCO3;//[J/mol]
// So equilibrium constant of the reaction is given
    by
K_2 = exp((-delta_g_1)/(R*T));

// Now, writing the expression of the equilibrium
    constant of the first reaction , we have
// K_1 = ([HCO3-]*[H+])/[H2CO3]
// and that for the second reaction
// K_2 = ([CO3]*[H+])/[CO3-]

// From the Table A.3 (page 419) as reported in the
   book, Henry's law constant is
H = 1480;//[atm]

// From Henry's law
// P*y_CO2 = x_O2*H   , so
x_CO2 = P*y_CO2/H;

// This gives the mol fracion. The dissociation
   constant are based on molaities a standard states
   , so
// Molality of the CO2 in the solution is
// m_CO2 = x_CO2*n_water   , where 'n_water' is
   number of moles of water in 1000g of water, so
n_water = 1000/18;//[mol]
m_CO2 = x_CO2*n_water;//[molal]

// Then we assume that almost all the H+ comes from
   the dissociation of dissolved CO2, so
// m_HCO3 = m_H, i.e. molality of bicarbonate is
   approximately equal to molality of hydrogen ion
   in the solution and hence
m_HCO3 = sqrt(K_1*m_CO2);//[molal]
m_H = m_HCO3;//[molal]

// Then we compute
m_CO3 = K_2*(m_HCO3/m_H);//[molal]

printf(" Amount of the CO2 dissolved in water in
   equilibrium with air is \t\t\t%0.2e molal\n",
   m_CO2);
printf(" Conentration of HCO3 ion and hydrogen ion H
   - in solution in equilibrium with air is    %0.2e
    molal\n",m_HCO3);
printf(" And concentration of CO3 ion in the
   solution in equilibrium with air is\t\t%0.2e
   molal",m_CO3);
```

Scilab code Exa 13.6 Gas liquid reactions

```
clear;
//clc();

// Example 13.6
// Page: 358
printf("Example-13.6  Page no.-358\n\n");

//***Data***//
// All the data are taken from the previous example
   13.5
m_H = 10^(-10);//[molal] molality of hydrogen ion
K_1 = 4.5*10^(-7);
K_2 = 4.7*10^(-11);

// Our Henry's law calculations are independent of
   the subsequent fate of the dissolved CO2.
// The concentration of dissolved CO2 in equilibrium
    with atmosphere is
m_CO2 = 1.32*10^(-5);//[molal] from previous example
// It is independent of that acidity or basicity of
   the water, and hence
m_HCO3 = K_1*(m_CO2/m_H);//[molal]

// and
m_CO3 = K_2*(m_HCO3/m_H);//[molal]
printf(" Amount of the CO2 dissolved in water in
   equilibrium with air is    \t%0.2e molal\n",m_CO2
   );
printf(" Conentration of HCO3 ion in solution in
   equilibrium with air is \t %0.2e molal\n",m_HCO3)
```

```
    ;
printf(" And concentration of CO3 ion in the
    solution in equilibrium with air is  %0.2e molal"
    ,m_CO3);
```

Scilab code Exa 13.7 Electrochemical reactions

```
clear;
//clc();

// Example 13.7
// Page: 362
printf("Example-13.7  Page no.-362\n\n");

//***Data***//
T = 298.15;//[K] Temperature
F = 96500;//[(coulomb)/(mole*electrons)] faraday
    constant

// The reaction is given as
// Al2O3 + 1.5C = 2Al + 1.5CO2

// No of the electron being exchanged are
n_e = 6;//[electron]
// All the reactants and products enter or leave the
     reactor as pure species in their standard states
    , so
// delta_g_0 = delta_g_1 and E = E_0
// Free energy of the species in the above equation
    as reported in the Table A.8 in the book is
g_0_CO2 = -394.4*1000;//[J/mol]
g_0_Al = 0;//[J/mol]
g_0_C = 0;//[J/mol]
g_0_Al2O3 = -1582.3*1000;//[J/mol]

```

```
// Free energy change of the reaction is
delta_g_0 = 1.5*g_0_CO2 + 2*g_0_Al - 1.5*g_0_C -
   g_0_Al2O3;//[J/mol]

// So, standard state cell voltage is
E_0 = (-delta_g_0)/(n_e*F);//[V]

printf("Standard state cell voltage for the
   production of aluminium is %f Volt",E_0);
```

Scilab code Exa 13.8 `Electrochemical reactions`

```
clear;
//clc();

// Example 13.8
// Page: 362
printf("Example-13.8  Page no.-362\n\n");

//***Data***//
T = 298.15;//[K] Temperature
F = 96500;//[(coulomb)/(mole*electrons)] faraday
   constant

// The reaction taking place between lithium and
   florine is
// Li + F = LiF

// From Table A.8 we find that
delta_g_0 = -587.7*1000;//[J/mol]
// We also know that
n_e = 1;//[electron] no of electron transferred
// That is because the valence Li and F change by 1,
    so one electron is transferred per molecule of
   LiF, thus
```

```
E_298_0 = (-delta_g_0)/(n_e*F);//[V]

printf("The reversible voltage for given
   electrochemical device is %f Volt",E_298_0);
```

Scilab code Exa 13.9 Electrochemical reactions

```
clear;
//clc();

// Example 13.9
// Page: 363
printf("Example-13.9  Page no.-363\n\n");

//***Data***//
T = 298.15;//[K] Temperature
P_0 = 1;//[atm]
P = 100;//[atm]
E_0 = -1.229;//[V]
F = 96500;//[(coulomb)/(mole*electrons)] faraday
   constant
R = 8.314;//[J/(mol*K)] universal gas constant

// The reaction is
// H2O(l) = H2(g) + 1/2O2(g)
// number of the valence electrons transferred in
   this reaction is
n_e = 2;//[(mole electrons)/mole]

// Gibb's free energy is given by
// g = g_0 + integrate(dg/dP)*dP, at constant
   temperature with integration limit P_0 and P
// or
// g = g_0 + integrate(v_T)*dP
// In the rightmost term we replace v_T by (R*T)/P,
    which is correct only for ideal gases, so
// g = g_0 + (R*T)*log(P/P_0)

// According to the assumption ,we can ignore the
    change in Gibb's free energy with pressure of the
     liquid water, so that
// delta_g = delta_g_0 + 1.5*(R*T)*log(P/P_0)

// and
// E = (-delta_g)/(n_e*F) = -(delta_g_0 + 1.5*(R*T)*
    log(P/P_0))/(n_e*F)
// So equilibrium cell voltage is given as
E = E_0 - 1.5*(R*T)*log(P/P_0)/(n_e*F);

printf("The equilibrium cell voltage of electrolytic
     cell if feed and product are at the pressure 100
     atm is %f Volt",E);
```

Scilab code Exa 13.10 Dimerization

```
clear;
//clc();

// Example 13.10
// Page: 365
printf("Example-13.10  Page no.-365\n\n");

//***Data***//
T = 273.15+25;//[K] Temperature
P = 11.38/760;//[atm] Pressure
R = 0.08206;//[(L*atm)/(mol*K)] Gas constant
v = 0.6525/0.04346;//[L/g] Specific volume
M = 60.05;//[g/mol] Molecular weight of HAc in the
    monomer form
// So the specific volume in [L/mol] is
```

```
V = v*M;//[L/mol]

// Compressibility factor is give by
z = (P*V)/(R*T);

printf("The value of the compressibility factor for
   HAc at given condition is %f",z);
```

Scilab code Exa 13.11 Dimerization

```
clear;
//clc();

// Example 13.11
// Page: 366
printf("Example-13.11  Page no.-366\n\n");

//***Data***//
T = 273.15+25;//[K] Temperature
P = 11.38;//[torr] Pressure

// Formation of the dimer from monomer in the gas
   phase follows the reaction
// 2*HAc = (HAc)2

// From the equation 13.BF(page 366) given in the
   book
// K = (P*y_HAc_2)/(P*y_HAc)^(2) , where 'y_HAc_2'
   is mol fraction of dimer and 'y_HAc' is mol
   fraction of monomer
// and
// log10(K) = -10.4184 + 3164/T , so
K = 10^(-10.4184 + 3164/T);//[1/torr]

//  Thus
// y_HAc_2 = K*(P*y_HAc)^(2)/P
// Since , (y_HAc + y_HAc_2) = 1
// y_HAc_2 = K*(P*(1-y_HAc))^(2)/P

// Solving for y_HAc_2
deff('[y]=f(y_HAc_2)','y = K*(P*(1-y_HAc_2))^(2)/P-
   y_HAc_2');
y_HAc_2 = fsolve(0,f);
// So
y_HAc = 1-y_HAc_2;

printf("Mole fraction of the monomer in the vapour
   phase is %f\n",y_HAc);
printf("Mole fraction of the dimer in the vapour
   phase is  %f",y_HAc_2);
```

Scilab code Exa 13.12 Dimerization

```
clear;
//clc();

// Example 13.12
// Page: 367
printf("Example-13.12  Page no.-367\n\n");

//***Data***//
// Getting the data from the example 13.10
T = 273.15+25;//[K] Temperature
P = 11.38/760;//[atm] Pressure
R = 0.08206;//[(L*atm)/(mol*K)] Gas constant
v = 0.6525/0.04346;//[L/g] Specific volume

// Now from the previous example ie example 13.11
   the mole fractions of the monomer and dimer in
   the gas phase is
```

```
y_HAc = 0.211;// monomer
y_HAc_2 = 0.789;// dimer

// Molecular weights of the monomer and dimer forms
   are
M_HAc = 60.05;//[g/mol] monomer
M_HAc_2 = 120.10;//[g/mol] dimer

// Now average molecular weight of the mixture is
M_avg = M_HAc*y_HAc + M_HAc_2*y_HAc_2;//[g/mol]

// So specific volume in [L/mol] is
V = v*M_avg;//[L/mol]

// Now compressibility factor is
z = (P*V)/(R*T);

printf("The compressibility factor z for the gaseous
    mixture is %f",z);
```

Chapter 14

Equilibrium With Gravity Or Centrifugal Force Osmotic Equilibrium Equilibrium With Surface Tension

Scilab code Exa 14.1 Equilibrium in the presence of gravity

```
clear;
//clc();

// Example 14.1
// Page: 379
printf("Example-14.1  Page no.-379\n\n");

//***Data***//
T = 300;//[K] Temperature of the natural gas well
R = 8.314;//[J/(mol*K)] universal gas constant
z_1 = 0;//[m]
// At the surface of the well mole fraction of the
   components are
y_methane_surf = 85/100;//[mol%]
y_ethane_surf = 10/100;//[mol%]
```

```
y_propane_surf = 5/100;//[mol%]
P = 2;//[MPa] Total equilibrium pressure
z_2 = 1000;//[m] Depth of the well

// Molecular weights of the components are
M_methane = 16/1000;//[kg/mol]
M_ethane = 30/1000;//[kg/mol]
M_propane = 44/1000;//[kg/mol]

// Now, we have the relation between the fugacities
    of a component at z_1 and z_2 as
// f_i_2/f_i_1 = exp((-M_i*g*(z_2-z_1))/(R*T))  ,
    where g is gravitational accelaration and its
    value is
g = 9.81;//[m/s^(2)]

// Fugacities of the various components at the
    surface i.e. at z = z_1 is
f_methane_1 = y_methane_surf*P;//[MPa]
f_ethane_1 = y_ethane_surf*P;//[MPa]
f_propane_1 = y_propane_surf*P;//[MPa]

// Now, fugacities at z = z_2 are
f_methane_2 = f_methane_1*exp((-M_methane*g*(z_1-z_2
    ))/(R*T));;;//[MPa]
f_ethane_2 = f_ethane_1*exp((-M_ethane*g*(z_1-z_2))
    /(R*T));//[MPa]
f_propane_2 = f_propane_1*exp((-M_propane*g*(z_1-z_2
    ))/(R*T));//[MPa]

// Let at z = z_1 total pressure of the gases are
    P_2
// Then, fugacities of the ith component is also
    given as
// f_i_2 = y_i_2*P_2
// Writing the expression for all the component ad
    adding them we get
// (f_methane_2 + f_ethane_2 + f_propane_2 ) =
    y_methane_2*P_2 + y_ethane_2*P_2 + y_propane_2*
    P_2
// or
// (f_methane_2 + f_ethane_2 + f_propane_2 ) = P_2*(
    y_methane_2 + y_ethane_2 + y_propane_2)
// and
// (y_methane_2 + y_ethane_2 + y_propane_2) = 1    ,
    so
P_2 = (f_methane_2 + f_ethane_2 + f_propane_2 );//[
    MPa]

// Now the mole fractions of the components are
//  y_i_2 = f_i_2/P_2   , so
y_methane_2 = f_methane_2/P_2;
y_ethane_2 = f_ethane_2/P_2;
y_propane_2 = f_propane_2/P_2;

printf("The mol fraction of the methane at the depth
     1000m is %f\n",y_methane_2);
printf("The mol fraction of the ethane at the depth
    1000m is %f\n",y_ethane_2);
printf("The mol fraction of the propane at the depth
     1000m is %f\n",y_propane_2);
```

Scilab code Exa 14.2 Equilibrium in the presence of gravity

```
clear;
//clc();

// Example 14.2
// Page: 380
printf("Example-14.2  Page no.-380\n\n");

//***Data***//
T = 288;//[K] Atmospheric temperature
```

```
R = 8.314;//[J/(mol*K)] universal gas constant
z_2 = 15000;//[m] Thickness of the atmosphere
z_1 = 0;//[m] Surface
// At the surface, the mole fraction of nitrogen and
    oxygen are
y_N2_1 = 0.79;
y_O2_1 = 0.21;
M_N2 = 28/1000;//[kg/mol]
M_O2 = 32/1000;//[kg/mol]

// For an ideal solution of ideal gases with only
   two species, we have
// y_i_2/y_i_1 = 1/(y_i_1 + y_j_1/a) , and
// a = exp(-(M_i-M_j)*g*(z_2-z_1)/(R*T))
// where 'g' is accelaration due to gravity and its
   value is
g = 9.81;//[m/s^(2)]

// So
a = exp(-(M_N2-M_O2)*g*(z_2-z_1)/(R*T));
// and
yi2_by_yi1 = 1/(y_N2_1 + y_O2_1/a);

printf(" Concentration of the nitrogen at the top of
    atmosphere with respect to the concentration of
   nitrogen at the surface of the earth is \n
          yi2_by_yi1 = %0.2f",yi2_by_yi1);
```

Scilab code Exa 14.3 Equilibrium in the presence of gravity

```
clear;
//clc();

// Example 14.3
// Page: 381
```

```
printf("Example-14.3   Page no.-381\n\n");

//***Data***//
// For this problem all the data are same as in
   previous Example 14.2 except z_1 and z_2
// So
T = 288;//[K] Atmospheric temperature
R = 8.314;//[J/(mol*K)] Universal gas constant
z_2 = 10;//[m] Height of the reactor
z_1 = 0;//[m] Surface
g = 9.81;//[m/s^(2)] Accelaration due to gravity
// At z = z_1, the mole fraction of nitrogen and
   oxygen are
y_N2_1 = 0.79;
y_O2_1 = 0.21;
M_N2 = 28/1000;//[kg/mol]
M_O2 = 32/1000;//[kg/mol]

// So
a = exp(-(M_N2-M_O2)*g*(z_2-z_1)/(R*T));
// and
yi2_by_yi1 = 1/(y_N2_1 + y_O2_1/a);

printf(" Concentration of the nitrogen at the top of
    reactor with respect to the concentration of
   nitrogen at the bottom of reactor is \n
   yi2_by_yi1 = %f",yi2_by_yi1);
```

Scilab code Exa 14.4 Centrifuges

```
clear;
//clc();

// Example 14.4
// Page: 382
```

```
printf("Example-14.4   Page no.-382\n\n");

//***Data***//
T = 300;//[K] Temperature of the centrifuge
R = 8.314;//[J/(mol*K)] Universal gas constant
// Mole fractions of the two components are
y_UF6_238_1 = 0.993; // Mole fraction of UF6 with
   238 isotope of uranium in feed
y_UF6_235_1 = 0.007;// Mole fraction of UF6 with 235
    isotope of uranium in feed
M_UF6_238 = 352/1000;//[kg/mol] Molecular weight of
   UF6 with 238 isotope of uranium
M_UF6_235 = 349/1000;//[kg/mol] Molecular weight of
   UF6 with 235 isotope of uranium
r_in = 2/100;//[m] Interanal raddi of the centrifuge
r_out = 10/100;//[m] outer raddi of the centrifuge
f = 800;//[revolution/second] Rotational frequency
   of centrifuge

// Here the accelaration will come due to
   centrifugal force and is
// g = w^(2)*r , where 'w' is angular speed and its
   value is w = 2*pie*f and 'r' is radius
// But in the present case 'r' is varies as we move
   away from the axis of centrifuge
// After making integration by taking small elements
    at the distance 'r' we find the expression
a = exp((M_UF6_235-M_UF6_238)*(2*3.141592*f)^(2)*(
   r_out^(2)-r_in^(2))/(2*R*T));

// Now Let the ratio y_i_2/y_i_1 = A
// Then we have
A = 1/(y_UF6_235_1 + y_UF6_238_1/a);

// Now say y_i_1/y_i_2 = 1/A = B , then
B = 1/A;

printf("The ratio of the mole fraction of UF6 (with
   uranium 235 isotope) at the 2 cm radius to that
   at the 10 cm radius is %0.3f",B);
```

Scilab code Exa 14.5 `Osmotic Pressure`

```
clear;
//clc();

// Example 14.5
// Page: 384
printf("Example-14.5  Page no.-384\n\n");

//***Data***//

// We have two phase system in this problem in which
     phase 1 is seawater and phase 2 is fresshwater
// Seawater contains mostly NaCl, Na2SO4, MgCl2, KCl
     and if they completly ionised then
x_water_1 = 0.98;// mole fraction of water in phase
    1 i.e. in seawater
x_water_2 = 1;// mole fraction of water in the phase
     2 i.e. in water
R = 10.73;//[(psi*ft^(3))/(lbmol*R)] Universal gas
    constant
T = 500;//[R] temperature
v_water_1 = 18/62.4//[ft^(3)/(lbmol)]

// The effect of the pressure on the fugacity of the
     liquid is given as
// f_i = (x_i*Y_i*p)*exp(integrate(v/(R*T)dP)) with
    integration limit from pure liquid pressure to
    solution liquid pressure

// Writing this equation twice, oncce for pure water
     and once for the water in the ocean water, and
```

```
   equating the fugacities , we get
// ((x_i*Y_i*p)*exp(integrate(v/(R*T)dP)))
   _pure_water = ((x_i*Y_i*p)*exp(integrate(v/(R*T)
   dP)))_seawater

// For pure water, x_i and Y_i are unity, and for
   the water in the solution, with mole fraction
   0.98, Raoult's law is certain to be practically
   obeyed
// So that Y_i is certain to be practically unity.

// The partial molal volume of water in pure water
   is practically the same as that in dilute
   solutions,
// Tkaing the logarithm of both sides and solving ,
   we get

// -log(x_water_1) = integrate(v_water_1/(R*T)dP)
// Integrating with the limit P_purewater and
   P_seawater we have
// -log(x_water_1) = (v_water_1/(R*T))*( P_seawater
   - P_purewater )
// ( P_seawater - P_purewater ) = delta_P
// So
delta_P = (-(R*T)*log(x_water_1))/v_water_1;//[psi]
printf("The pressure difference between the two
   phases is %0.1f psi",delta_P)
```

Scilab code Exa 14.6 Pressure difference across a droplet

```
clear;
//clc();

// Example 14.6
// Page: 386
```

```
printf("Example-14.6   Page no.-386\n\n");

//***Data***//
T = 100;//[C] Temperature of the outside
P_outside = 1;//[atm]
// At 100 C, the surface tension between steam and
   water is
T = 0.05892;//[N/m] From metric steam table (7, page
    267)

// Pressure difference between inside and outside of
    a drop is given by the expression
// (P_inside - P_outside) = (4*T)/d_i

// Let (P_inside - P_outside) = delta_P , so
//delta_P = (4*T)/d_i
// For the drop of diameter
d_1 = 0.001;//[m]
// So
delta_P_1 = (4*T)/d_1;//[Pa]

// Which is certainly negligible
// If we reduce the diameter to
d_2 = 10^(-6);//[m]

// So
delta_P_2 = (4*T)/d_2;//[Pa]

// If we reduce it to diameter that is smallest
   sized drop likely to exist
d_3 = 0.01*10^(-6)//[m]
// Then the calculated pressure difference is
delta_P_3 = (4*T)/d_3;//[Pa]

printf("Pressure difference with the change in
   radius of the drop of the water is given as in
   the following table\n\n");
printf("           Diameter of the droplet (d_i)(in
   meter)                        Pressure difference (
   P_inside - P_outside )(in atm)\n");
printf("                         %0.2e

   %0.2e\n",d_1,delta_P_1);
printf("                         %0.2e

   %0.2e\n",d_2,delta_P_2);
printf("                         %0.2e

   %0.2e\n",d_3,delta_P_3);
```

Scilab code Exa 14.7 Equilibrium with surface tension

```
clear;
//clc();

// Example 14.7
// Page: 387
printf("Example-14.7   Page no.-387\n\n");

//***Data***//
P_NBP = 1;//[atm]
Temp =273.15+100;//[C] Temperature
D = 0.01*10^(-6);//[m] Diameter of the condensation
    nuclei( due to impurity )
T = 0.05892;//[N/m] Surface tension between water
    drops and gas
R = 8.314;//[J/(mol*K)]

// At equilibrium the Gibb's free energy per pound
    will be the same inside and outside the drops.
// From the previous example 14.6, the pressure
    difference inside and outside of the drop is
// delta_P = ( P_inside-P_outside) = 4*T/D = 233 atm
```

```
    = 235.7 bar

// Taking the Gibb's free energy at the normal
   boiling point as g_NBP we have
// g_small_drop_equilibrium = g_NBP + integrate(
   v_water_gas)dP  , with integration limits P_NBP
   and P_gas
// also
// g_small_drop_equilibrium = g_NBP + integrate(
   v_water_liquid)dP  , with integration limits
   P_NBP and (P_gas + 4*T/D)
// and
v_water_liquid = 1/958.39*0.018;//[m^(3)/mol]

// If we assume that the specific volume of the
   liquid is a constant,and independent of pressure,
    and that the volume of the vapour is given by
   the gas law
// then we can perform the integrations and cancel
   the g_NBP terms, finding the Kelvin equation

// (R*Temp)*log(P_gas/P_NBP) = v_water_liquid*(P_gas
    + 4*T/D - P_NBP)
// For very small drops
// (P_gas - P_NBP) << 4*T/D
// So that we can write it approximately as

// P_gas/P_NBP = exp(v_water_liquid*(4*T/D)/(R*Temp)
   ) = I
// so
I = exp(v_water_liquid*(4*T/D)/(R*Temp));

// Substracting 1 from both sides in the above
   equation we have
// (P_gas-P_NBP)/P_NBP = I-1
// So
P_gas_minus_P_NBP = (I-1)*P_NBP;//[atm]
// Changing into the bar we have
```

```
delta_P = P_gas_minus_P_NBP*1.01325;//[bar]

// Now changing the unit to psi we have
delta_P_1 = delta_P*100*0.1450377;//[psi]

printf("The equilibrium pressure at which the steam
    begin to condence at this temperature on the
    nuclei is %f psi above the normal boiling point."
    ,delta_P_1);
```

Scilab code Exa 14.8 Equilibrium with surface tension

```
clear;
//clc();

// Example 14.8
// Page: 388
printf("Example-14.8   Page no.-388\n\n");

//***Data***//
Temp = 273.15+100;//[K] Temperature of the water
    drop
R = 8.314;//[J/(mol*K)] Universal gas constant
D = 0.01*10^(-6);//[m] Diameter of the water drop
P_g = 0.15;//[bar] guage pressure
T = 0.05892;//[N/m] Surface tension between water
    drop and gas

// The calculation of the pressure difference from
    inside to outside is the same as done in the
    example 14.7

// The specific Gibb's free energy of the liquid is
    thus given as
// (g_water_liquid - g_NBP) = integrate(
    v_water_liquid)dP   , with integration limits
    P_NBP and (P_gas + 4*T/D)
// Where
v_water_liquid = 0.018/958.39;//[m^(3)/mol]
P_NBP = 1.013;//[bar]
P_gas = 1.013+0.15;//[bar]

// Say
P_1 = P_gas + 4*T/D;//[bar]
// and (g_water_liquid - g_NBP) = delta_g_1
// So
delta_g_1 = integrate('v_water_liquid*P^(0)','P',
    P_NBP,P_1);//[J/mol]

// and for the gas, again using equation for Gibb's
    free energy, we have
// (g_water_liquid- g_NBP) = integrate(v_water_gas)
    dP   , with integration limits P_NBP and P_gas
// Here assuming that the vapour follows the ideal
    gas law we have
// v_water_gas = (R*Temp/P)
// and also let (g_water_liquid- g_NBP) = delta_g_2
// so
delta_g_2 = integrate('(R*Temp)/P','P',P_NBP,P_gas);

// Now
// (g_water_liquid - g_water_gas) = (g_water_liquid
    - g_NBP)-(g_water_gas - g_NBP) = delta_g
// So
delta_g = (delta_g_1 - delta_g_2);

// We have got the value of the delta_g positive, so

printf("The liquid can lower its free energy %0.2f J
    /mol by Changing to gas,\n",delta_g);
printf("So that even at 0.15 bar above the normal
    boiling point, a drop of this small size is
    unstable and will quickly evaporate.");
```

Scilab code Exa 14.9 Equilibrium with surface tension

```
clear;
//clc();

// Example 14.9
// Page: 390
printf("Example-14.9   Page no.-390\n\n");

//***Data***//
Temp = 904.7;//[R] Temperature of the pure liquid
    water
P_NBP = 400;//[psia] Saturation pressure of the pure
     liquid water at the given temperature
T = 1.76*10^(-4);//[lbf/inch] Surface tension of
    water
R = 10.73;//[(psi*ft^(3))/(lbmol*R)]

// In this problem the gas is inside the bubble, at
    a pressure much higher than that of the
    sorrounding liquid.
// The criterion of equilibrium is that the Gibb's
    free energy of the gas inside the bubble must be
    the same as that of the liquid outside the bubble
    .
// Thus we have
// g_small_drop_equilibrium = g_NBP + integrate(
    v_water_liquid)dP  , with integration limits
    P_NBP and P_liquid
// also
// g_small_drop_equilibrium = g_NBP + integrate(
    v_water_gas)dP  , with integration limits P_NBP
    and (P_liquid+4*T/D)
// where
```

```
v_water_liquid = 18*0.01934;//[ft^(3)/lbmol]
D = 10^(-5);//[inch]

// so
// g_NBP + integrate(v_water_liquid)dP = g_NBP +
   integrate(v_water_gas)dP

// Here we assume that the liquid has practically
   constant density and that the gas behaves as an
   ideal gas and find
// (R*Temp)*log((P_liquid+4*T/D)/P_NBP) =
   v_water_liquid*(P_liquid - P_NBP)
// let P_liquid = p

// We will solve the above equation for p
deff('[y]=f(p)','y = v_water_liquid*(p - P_NBP)-(R*
   Temp)*log((p+4*T/D)/P_NBP)');
P_liquid = fsolve(300,f);//[psia]

// At this external pressure the pressure inside the
    bubble is
P_inside = P_liquid + 4*T/D;//[psia]

printf("The liquid pressure at which these boiling
   nuclei will begin to grow and intiate boiling is
   %0.1f psia\n",P_liquid);
printf("At this external pressure the pressure
   inside the bubble is %0.1f psia",P_inside);
```

Chapter 15

The Phase Rule

Scilab code Exa 15.1 Phase rule

```
clear;
//clc();

// Example 15.1
// Page: 398
printf("Example-15.1   Page no.-398\n\n");

//***Data***//
// This is a theoratical question.
printf("This is a theoratical question and there are
    no any numerical components involve. Refer to
   page no 398 of the book.");
```

Scilab code Exa 15.2 Two component system

```
clear;
//clc();

```

```
// Example 15.2
// Page: 401
printf("Example-15.2   Page no.-401\n\n");

//***Data***//
// The system contains four species
printf(" In this system, there are four identifiable
    chemical species,which are C,O2,CO2 and CO.\n
   The balanced equations we can write among them
   are\n");

printf("     C + 0.5O2 = CO\n");
printf("     C + O2 = CO2\n");
printf("     CO + 0.5O2 = CO2\n");
printf("     CO2 + C = 2CO\n");

// Let we call these equations A, B, C and D
   respectively
// These relations are not independent.
// If we add A and C and cancel like terms, we
   obtain B.
// So, If we want independent chemical equilibria we
    must remove equation C

// Now, if we reverse the direction of B and add it
   to A, we see that D is also not independent.
// Thus, there are only two independent relations
   among these four species and
printf(" There are only two independent relations
   among these four species and\n");

// V = C + 2 - P
// and we have
V = 2;// No of the variable
P = 2;// No of the phases
// So
C = V + P - 2;
printf("     C = V + P - 2\n");
printf("     C = 4 - 2 = 2\n");
printf(" Thus, this is a two-component system");
```

Scilab code Exa 15.3 Degree of freedom and number of components

```
clear;
//clc();

// Example 15.3
// Page: 402
printf("Example-15.3  Page no.-402\n\n");

//***Data***//
// This contains three species.
printf(" The three species in this system are H2, N2
    and NH3\n");
N = 3;
printf(" There is only one balanced chemical
   reaction among these species\n");
Q = 1

// 2NH3 = N2 + 3H2
C = N - Q;
printf("     C = N - Q = %0.0f\n\n",C);
// Now let us we made the system by starting with
   pure ammonia.
// Assuming that all the species are in the gas
   phase, ammonia dissociates in H2 and N2 in the
   ratio of 3:1.
printf(" Let we start with pure ammonia in the
   system, then ammonia will dissociate in H2 and N2
    in the ratio of 3:1.\n");

// We can write an equation among their mole
   fractions, viz;
```

```
// y_H2 = 3*y_N2
printf(" And the relation between their mole
    fraction is\n     y_H2 = 3*y_N2\n\n");

// We might modify the phase rule to put in another
    symbol for stoichiometric restrictions , but the
    common usage is to write that
// Components = species - (independent reactions) -
    (stoichiometric restriction)
// and stoichiometric restriction SR is
SR = 1;
// so
c = N-Q-SR;
printf(" We have the modified phase rule as\n
    Components = species - (independent reactions) -
    (stoichiometric restriction)\n")
printf("     C = N - Q - SR = %0.0f",c);
```

Scilab code Exa 15.4 Number of components in a reactions

```
clear;
//clc();

// Example 15.4
// Page: 403
printf("Example-15.4   Page no.-403\n\n");

//***Data***//
// We have been given the reaction
// CaCO3(s) = CaO(s) + CO2(g)

// Here we have three species and one balanced
    chemical reaction between them
// So
N = 3;// No of species
```

```
Q = 1; // no of reaction

// Since CO2 will mostly be in the gas phase and
   CaCO3 and CaO will each form separate solid
   phases,
// there is no equation we can write among the mole
   fractions in any of the phases.
// Hence, there is no stoichiometric restriction i.e
   .
SR = 0
// and the number of the components is
C = N - Q - SR;

printf("Number of the components presents in the
   test tube are %0.0f",C);
```

Scilab code Exa 15.5 Degree of freedom and Phases

```
clear;
//clc();

// Example 15.5
// Page: 403
printf("Example-15.5  Page no.-403\n\n");

//***Data***//
// We have been given the reaction
// CaCO3(s) = CaO(s) + CO2(g)
// The CaCO3 and CaO form separate solid phases, so
   we have three phases, two solid and one gas.
// So
P = 3;
// This is a two component system, so
C = 2;

```

```
// From the phase rule
V = C + 2 - P;

// If there is only one degree of freedom, then the system should have a unique P-T curve.
// Reference [ 2, page 214 ] as reported in the book, shows the data to draw such a curve, which can be well represented by
// log(p/torr) = 23.6193 - 19827/T

printf(" The no. of phases present in the system are %0.0f \n",P);
printf(" Total no of degrees of freedom is %0.0f \n",V);
printf(" Since, there is only one degree of freedom, so the system has a unique P-T curve,\n");
printf(" which can be well represented by \n    log(p/torr) = 23.6193 - 19827/T");
```

Scilab code Exa 15.6 Number of components in ionic reaction

```
clear;
//clc();

// Example 15.6
// Page: 404
printf("Example-15.6  Page no.-404\n\n");

//***Data***//
// The system consists of five species.
printf(" The five species present in the system are H2O, HCl, H+, OH- and Cl-. \n");
// So
N = 5;// Number of the species
printf(" Here we have two chemical relations:\n");
printf("     H2O = H+  +  OH- \n");
printf("     HCl = H+  +  Cl- \n");

// so
Q = 2;// No of the reactions

// In addition we have electroneutrality , which says
   that at equilibrium the total no of positive
  ions in the solution must be the same as the
  total no of nagative ions ,or
// [H+] = [OH-] + [Cl-]
// To maintain electroneutrality number of positive
  and negative ion should be same.
// Here [H+] stands for the molality of hydrogen ion
  . This is convertible to a relation among the 'mu
  's'; hence ,
// it is an additional restriction , so
SR = 1;
// So
// The number of components is
C = N - Q - SR;

printf(" Number of the components present in the
  system are \n    C = N - Q - SR = %0.0f",C);
```

Scilab code Exa 15.7 Dependency of the number of components

```
clear;
//clc();

// Example 15.7
// Page: 405
printf("Example-15.7  Page no.-405\n\n");

//***Data***//
```

```
printf(" Our system consists of Au and H2O.\n");
// So
N = 2;// Number of the species
// If there is no chemical reaction , then
Q = 0;

//So
C = N - Q;// Number of the components
printf(" If no compound is formed , then number of
   the components in the system are \n    C = N − Q
   = 2 − 0 = %0.0 f\n\n",C);

// However , if there is also a chemical reaction
// Au + H2O = AuH2O
// so
n = 3;// Number of the species
q = 1;// Number of the reactions

// Thus , we have
c = n - q;// Number of the components

printf(" If there is also a chemical reaction , viz .\
   n     Au + H2O = AuH2O \n");
printf(" the number of the components in the system
   are\n     C = N − Q = %0.0 f\n\n",c);
printf(" The number of the components is independent
    of the existence or nonexistence of such
   compounds of questionable existence. ");
```

Scilab code Exa 15.8 A formal way to find the number of the indepedent equations

```
clear;
//clc();

// Example 15.8
// Page: 405
printf("Example-15.8   Page no.-405\n\n");

//***Data***//
// The species, we are given are CaCO3, CaO and CO2
// First we write the reaction for the formation of
   the above species from their elemental part.
// So, we have
// Ca + C + 1.5O2 = CaCO3
// Ca + 0.5O2 = CaO
// C + O2 = CO2

// We must eliminate Ca, C and O2 from these
   equations because they do not appear in the
   species list.

// Now, solving the 3rd equation for C and
   substituting in the first equation, we have
// Ca + CO2 - O2 + 1.5O2 = CaCO3
// Now, this equation to the equation second, we
   have
// CO2 = -CaO + CaCO3
// or
// CaCO3 = CO2 + CaO

printf(" There is only one balanced chemical
   reaction between the species on the species list,
    viz.\n");
printf("     CaCO3 = CO2 + CaO ");
```

Scilab code Exa 15.9 Isothermal behaviour of the given reaction set

```
clear;
//clc();

```

```
// Example 15.9
// Page: 408
printf("Example-15.9   Page no.-408\n\n");

//***Data***//
printf("This is a theoratical question and there are
    no any numerical components. Refer to page no
    408 of the book.");
```