

Scilab Textbook Companion for
Physical And Chemical Equilibrium For
Chemical Engineers
by N. de Nevers¹

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

May 23, 2016

¹Funded 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	4
1 Introduction to Equilibrium	5
2 Basic Thermodynamics	8
3 The Simplest Phase Equilibrium Examples and Some Simple Estimating Rules	13
4 Minimization of Gibbs Free energy	24
5 Vapor Pressure The Clapeyron Equation And Single Pure Chemical Species Phase Equilibrium	28
6 Partial Molal Properties	36
7 Fugacity Ideal Solutions Activity Activity Coefficient	47
8 Vapor Liquid Equilibrium VLE at Low Pressures	57
9 Correlating And Predicting Nonideal VLE	85
10 Vapor Liquid Equilibrium VLE at High Pressures	100
11 LIiquid Liquid Liquid Solid And Gas Solid Equilibrium	110
12 Chemical Equilibrium	128
13 Equilibrium In Complex Chemical Reactions	149

14 Equilibrium With Gravity Or Centrifugal Force Osmotic Equilibrium Equilibrium With Surface Tension	168
15 The Phase Rule	183

List of Scilab Codes

Exa 1.1	Units Conversion Factors and Notations	5
Exa 1.2	Density dependency of concentration	6
Exa 2.1	Calculation of the thermodynamics properties of steam	8
Exa 2.2	Work done in reversible adiabatic steady flow steam turbine	10
Exa 2.3	Estimation of Compressibility factor	11
Exa 3.1	Mole fraction of water vapour in air	13
Exa 3.2	Roult's and Henry's Law	14
Exa 3.3	Composition of Air and Water	15
Exa 3.4	Some Simple Applications of Raoult's and Henry's laws	17
Exa 3.5	Some Simple Applications of Raoult's and Henry's laws	19
Exa 3.6	Some Simple Applications of Raoult's and Henry's laws	19
Exa 3.7	Some Simple Applications of Raoult's and Henry's laws	21
Exa 3.8	Some Simple Applications of Raoult's and Henry's laws	22
Exa 4.1	Gibbs free energy calculation	24
Exa 4.2	Gibbs free energy diagram for the graphite diamond system	25
Exa 4.3	Gibbs free energy for chemical reactions	26
Exa 5.1	Application of Clapeyron Equation	28
Exa 5.2	Application of Clausius Clapeyron Equation	29
Exa 5.3	Application of Clausius Clapeyron Equation	30
Exa 5.4	The Acentric Factor	31
Exa 5.5	Estimation of NBP using Antoine equation	32
Exa 5.6	Applying the Clapeyron equation to other kind of equilibrium	33
Exa 6.1	Tangent Slopes	36
Exa 6.2	Volume Change on Mixing	37
Exa 6.3	Volume change on mixing	38

Exa 6.4	Tangent Intercept Concept	39
Exa 6.5	Tangent Intercept concept	40
Exa 6.6	Idea of Tangent Intercept	41
Exa 6.7	Partial Mass Properties	41
Exa 6.8	Differential Heat of Mixing	42
Exa 6.9	Integral Heat of Mixing	43
Exa 6.10	Integral Heat of Mixing	44
Exa 6.11	Application of Gibbs Duhem equation	45
Exa 6.12	Application of the gibbs Duhem equation	45
Exa 7.1	The fugacity of pure gases	47
Exa 7.2	Compressibility factor and volume residual	48
Exa 7.3	Fugacity of pure liquid	49
Exa 7.4	Activity and activity coefficient	50
Exa 7.5	Fugacities from gas PvT data	53
Exa 7.6	Fugacities from gas PvT data	55
Exa 7.7	Fugacities from an EOS for gas mixtures	56
Exa 8.1	Calculation of K factors	57
Exa 8.2	Liquid phase activity coefficient	58
Exa 8.3	Non ideal solution behaviour	59
Exa 8.4	Two liquid phase	62
Exa 8.5	Two liquid phase	63
Exa 8.6	Activity coefficient of water and n butanol	64
Exa 8.7	Zero Solubility and Steam distillation	66
Exa 8.8	The little EOS	68
Exa 8.9	Dew Point Calculations	69
Exa 8.10	Pressure specified dew point	71
Exa 8.11	Temperature specified bubble point	73
Exa 8.12	Pressure specified bubble point	75
Exa 8.13	Isothermal flashes	78
Exa 8.14	Use of de prester chart	80
Exa 8.15	Non volatile solutes boiling point elevation	81
Exa 8.16	Freezing point depression	83
Exa 9.1	Van Laar equation	85
Exa 9.2	Excess Gibbs free energy and activity coefficient equations	86
Exa 9.3	Excess Gibbs free energy and activity coefficient equations	87
Exa 9.4	Activity coefficient at infinite dilution	87

Exa 9.5	Constants in the morgules equation	88
Exa 9.6	Effect of pressure changes on liquid phase activity coefficient	88
Exa 9.7	Effect of temperature changes on liquid phase activity coefficient	90
Exa 9.8	Liquid phase activity coefficients for ternary mixtures	91
Exa 9.9	Application of mixing rule	93
Exa 9.10	Solubility parameter	97
Exa 9.11	Henrys Law Constant estimation	98
Exa 9.12	Calculation of the activity coefficient using Henrys law	99
Exa 10.1	Bubble point temperature estimation	100
Exa 10.2	Bubble point temperature estimation	103
Exa 10.3	Bubble point temperature estimation using SRK EOS	105
Exa 11.1	Reporting and presenting LLE data	110
Exa 11.2	Reporting and presenting LLE data	111
Exa 11.3	ternary LLE	112
Exa 11.4	The elementary theory of LLE	113
Exa 11.5	Plot of the Gibbs free energy vs mole fraction	115
Exa 11.6	Two liquid phase	116
Exa 11.7	Effect of temperature on LLE	117
Exa 11.8	Effect of temperature on LLE	118
Exa 11.9	Distribution coefficients	119
Exa 11.10	The experimental determination of LSE	121
Exa 11.11	The experimental determination of LSE	122
Exa 11.12	Estimation of the activity coefficients	123
Exa 11.13	Solubility of NaCl in water	125
Exa 11.14	Gas solid equilibrium at low pressures	126
Exa 11.15	GSE at high pressures	127
Exa 12.1	Gibbs free energy	128
Exa 12.2	Calculation of the Equilibrium constants	129
Exa 12.3	Change of reactant concentration	131
Exa 12.4	Mass action law	132
Exa 12.5	Mass action law	134
Exa 12.6	Reversible reaction	135
Exa 12.7	Standard state Gibbs free energy	137
Exa 12.8	Effect of temperature on chemical reaction equilibrium	139
Exa 12.9	Equilibrium conversion of a mixture	141
Exa 12.10	Ideal solution of ideal gases	143

Exa 12.11	Non ideal solution non ideal gases	144
Exa 12.12	Liquids and solids	145
Exa 12.13	Equilibrium constant K_p	148
Exa 13.1	Reactions Involving Ions	149
Exa 13.2	Sequential reactions	150
Exa 13.3	Simultaneous reactions	153
Exa 13.4	Solubility product	156
Exa 13.5	Gas liquid reactions	157
Exa 13.6	Gas liquid reactions	160
Exa 13.7	Electrochemical reactions	161
Exa 13.8	Electrochemical reactions	162
Exa 13.9	Electrochemical reactions	163
Exa 13.10	Dimerization	164
Exa 13.11	Dimerization	165
Exa 13.12	Dimerization	166
Exa 14.1	Equilibrium in the presence of gravity	168
Exa 14.2	Equilibrium in the presence of gravity	170
Exa 14.3	Equilibrium in the presence of gravity	171
Exa 14.4	Centrifuges	172
Exa 14.5	Osmotic Pressure	174
Exa 14.6	Pressure difference across a droplet	175
Exa 14.7	Equilibrium with surface tension	177
Exa 14.8	Equilibrium with surface tension	179
Exa 14.9	Equilibrium with surface tension	181
Exa 15.1	Phase rule	183
Exa 15.2	Two component system	183
Exa 15.3	Degree of freedom and number of components	185
Exa 15.4	Number of components in a reactions	186
Exa 15.5	Degree of freedom and Phases	187
Exa 15.6	Number of components in ionic reaction	188
Exa 15.7	Dependency of the number of components	189
Exa 15.8	A formal way to find the number of the independent equations	190
Exa 15.9	Isothermal behaviour of the given reaction set	191

Chapter 1

Introduction to Equilibrium

Scilab code Exa 1.1 Units Conversion Factors and Notations

```
1
2 clear;
3 //clc();
4
5 // Example 1.1
6 // Page: 9
7 printf("Example -1.1 Page no.-9\n\n");
8
9 /***Data***/
10 m_i = 10;//[g]
11 m_w = 990;//[g]
12 M_i = 342.3;//[g]
13 M_w = 18;//[g]
14 // The mass fraction is
15 // ( mass fraction of sucrose ) = x_i (by mass) =
    m_i/(sum of all substances)
16 x_i = m_i/(m_i+m_w);
17 x_i = x_i*100; // [in percentage]
18 // This is also the weight fraction.
19 // The mole fraction is
20 // ( mole fraction of sucrose ) = x_j (by mole) =
```

```

    n_i/(sum of number moles of all the substances)
21 n_i = m_i/M_i; // number of moles of sucrose
22 n_w = m_w/M_w; // number of moles of water
23 x_j = n_i/(n_i+n_w);
24 x_j = x_j*100; // [in percentage]
25 // The molality, a concentration unit is widely used
    in equilibrium calculations, is defined as
26 // m (molality) = (moles of solute)/(kg of solvent)
27 m = n_i/m_w*1000; // [molal]
28 // For solutions of solids and liquids (but not
    gases) ppm almost always means ppm by mass, so
29 x_ppm = x_i*10^(6)/100; // [ppm]
30 printf(" sucrose concentration in terms of the mass
    fraction is %f%%\n",x_i);
31 printf(" sucrose concentration in terms of the mole
    fraction is %f%%\n",x_j);
32 printf(" sucrose concentration in terms of the
    molality is %f molal\n",m);
33 printf(" sucrose concentration in terms of the ppm
    is %f ppm",x_ppm);

```

Scilab code Exa 1.2 Density dependency of concentration

```

1 clear;
2 //clc();
3
4 // Example 1.2
5 // Page: 9
6 printf("Example-1.2 Page no.-10\n\n");
7
8 //***Data***/
9 T = 20; // [C]
10 d = 1.038143/1000*10^(6); // [kg/m^(3)]
11 m_i = 10; // [g] mass of sucrose
12 M_i = 342.3; // [g/mol] molecular weight of sucrose

```

```

13 // In the previous example i.e. example 1.1 the mass
    was chosen to be 1.00 kg, so that
14 m = 1.00;//[kg]
15 V = m/d*1000;//[L]
16 // The mass concentration is
17 // m_1 ( mass concentration of sucrose ) = (mass of
    sucrose)/(volume of solution)
18 m_1 = m_i/V;//[g/L]
19 // The mole concentration is
20 // m_2 ( mole concentration of sucrose ) = (moles of
    sucrose)/(volume of solution)
21 m_2 = (m_i/M_i)/V;//[mol/L]
22 printf(" Mass concentration of the solution is %f g/
    L\n",m_1);
23 printf(" Mole concentration of the solution is %f
    mol/L\n",m_2);
24 // By the definition of the molarity , molarity is
    mole concentration of the solute
25 // so molarity
26 m = m_2;
27 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

```
1 clear;
2 // clc ();
3
4 // Example 2.1
5 // Page: 27
6 printf("Example -2.1 Page no. -27\n\n");
7
8 /***Data***/
9 m = 1; //[lbm] Mass of the steam
10 T_1 = 300; //[F] Initial temperature
11 P_1 = 14.7; //[psia] Initial pressure
12 P_sorrounding = 14.7; //[psia]
13 Q = 50; //[Btu] Amount of the energy added to the
    system as heat
14
15 // This is a closed system and we can apply the
    following equations
16 //  $\Delta U_{\text{system}} = \sum(dQ_{\text{in}} - \text{minus\_out}) + \sum(dW_{\text{in}} - \text{minus\_out})$  (
    A)
17 //  $dS_{\text{system}} = (m \cdot ds)_{\text{system}} = \sum((dQ)/T)$ 
```

```

        _in_minus_out + dS_reversible
                                (B)
18
19 // From the steam tables , we look up the properties
    of steam at temperature 300F and pressure 14.7
    psia and find
20 u_initial = 1109.6; //[Btu/lbm] Internal energy of
    the steam
21 h_initial = 1192.6; //[Btu/lbm] Enthalpy of the steam
22 s_initial = 1.8157; //[Btu/(lbm*R)] Entropy of the
    steam
23
24 // The work here is done by the system , equal to
25 // -delta_w = P*A_piston*delta_x = P*m*delta_v
26
27 // Substituting this in the equation (A) and
    rearranging , we have
28 // m*delta_(u + P*v) = m*delta_h = delta_Q
29 // From which we can solve for the final specific
    enthalpy
30 h_final = h_initial + Q; //[Btu/lbm]
31
32 // Now, by the linear interpolation we find that at
    h = 1242.6 Btu/lbm and P = 1 atm, temperature of
    the steam is given
33 T_2 = 405.7; //[F] Final temperature
34
35 // At this final temperature and pressure we have
    the steam properties
36 u_final = 1147.7; //[Btu/lbm]
37 s_final = 1.8772; //[Btu/(lbm*R)]
38
39 // Thus, increase in the internal energy , enthalpy
    and entropy are
40 delta_u = u_final - u_initial; //[Btu/lbm]
41 delta_s = s_final - s_initial; //[Btu/(lbm*R)]
42 delta_h = Q; //[Btu/lbm]
43

```

```

44 // The work done on the atmosphere is given by
45 w = delta_h - delta_u;//[Btulbm]
46
47 printf("The increase in internal energy of the steam
    by adding the heat is %0.2f Btu/lbm\n",delta_u);
48 printf("The increase in enthalpy of the steam by
    adding the heat is          %0.2f Btu/lbm\n",
    delta_h);
49 printf("The increase in entropy of the steam by
    adding the heat is          %0.4f Btu/lbm\n",
    delta_s);
50 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

```

1 clear;
2 //clc();
3
4 // Example 2.2
5 // Page: 28
6 printf("Example-2.2   Page no.-28\n\n");
7
8 //***Data***/
9 T_in = 600;//[F] Input steam temperature
10 P_in = 200;//[psia] Input steam pressure
11 P_exit = 50;//[psia]
12
13 // Because this is a steady-state , steady-flow
    process , we use
14 // (work per pound) = W/m = -( h_in - h_out )
15
16 // From the steam table we can read the the inlet
    enthalpy and entropy as

```

```

17 h_in = 1322.1; //[Btu/lbm]
18 s_in = 1.6767; //[Btu/(lb*R)]
19
20 // Now, we need the value of h_out
21
22 // For a reversible adiabatic steady-state, steady-
    flow process, we have
23 // sum(s*m_in_minus_out) = ( s_in - s_out ) = 0
24
25 // Which indicates that inlet and outlet entropies
    are same
26 // We can find the outlet temperature by finding the
    value of the temperature in the steam table
27 // For which the inlet entropy at 50 psia is the
    same as the inlet entropy, 1.6767 Btu/(lb*R).
28 // By linear interpolation in the table we find
29 T_in = 307.1; //[R]
30
31 // and by the linear interpolation in the same table
    we find that
32 h_out = 1188.1; //[Btu/lb]
33
34 // Thus, we find
35 W_per_pound = -(h_in - h_out); //[Btu/lb]
36
37 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

```

1 clear;
2 //clc();
3
4 // Example 2.3
5 // Page: 38

```



```

6 printf("Example -2.3 Page no.-38\n\n");
7
8 /***Data***/
9 T = 500; //[F]
10 P = 680; //[psi]
11 // It is reported in the book in the table A.1(page
    417) that for water
12 // We know that  $T_r = T/T_c$  and  $P_r = P/P_c$ , so
13 T_c = 647.1*1.8/[R]
14 P_c = 220.55*14.51; //[psia]
15 w = 0.345;
16 T_r = (T+459.67)/T_c;
17 P_r = P/P_c;
18 z_0 = 1+P_r/T_r*(0.083-0.422/T_r^(1.6));
19 z_1 = P_r/T_r*(0.139-0.172/T_r^(4.2));
20 z = z_0+w*z_1;
21 printf("The compressibility factor of steam at the
    given state is %0.3f",z);
22 // 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

```
1 clear;
2 //clc();
3
4 // Example 3.1
5 // Page: 52
6 printf("Example-3.1 Page no.-52\n\n");
7
8
9 /**Data**/
10 T = 20; // [C]
11 P = 1; // [atm]
12 // From Raoult's law  $y_i * P = x_i * p_i$ 
13 // Rearranging
14 //  $y_i = x_i * p_i / P$ ;
15 // Here we have ternary mixture of nitrogen, oxygen,
    and water. If we let the subscript i stand for
    water, we can say that
```

```

16 // x_water = 1-x_N2-x_O2;
17 // 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
18 x_N2 = 0;
19 x_O2 = 0;
20 x_water = 1-x_N2-x_O2;
21 // From any steam table we may look up the value of
    the vapour pressure of water at 20C, finding
22 p_water = 0.023; //[atm]
23 // So
24 y_water = x_water*p_water/P;
25 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

```

1 clear;
2 //clc();
3
4 // Example 3.2
5 // Page: 53
6 printf("Example-3.2 Page no.-53\n\n");
7
8 /***Data***/
9 T = 20; //[C]
10 P = 1; //[atm]
11 // From previous example i.e. example 3.1
12 y_water = 0.023;
13 //so that
14 //y = y_N2+y_O2
15 y = 1-y_water;
16 // The oxygen is 0.21 mole fraction of this mix, so
    that
17 y_O2 = y*0.21;

```

```

18 // It is reported in the book in table A.3 (page
    419) that Henry's law constant for oxygen in
    water at 20C is
19 H_O2 = 40100; //[atm]
20 // From Henry's law, we have
21 //  $y_i = x_i * H_i / P$ 
22 // rearranging
23 //  $x_i = y_i * P / H_i$ 
24 // so
25 x_O2 = y_O2 * P / H_O2;
26 // By the same logic we find that
27 y_N2 = y * 0.79;
28 //and Henry's law constant for nitrogen in water at
    20C is
29 H_N2 = 80400; //[atm]
30 // hence
31 x_N2 = y_N2 * P / H_N2;
32 // Now expressing the dissolved oxygen concentration
    in terms of the volume of the oxygen at STP viz.
    taken as 1 atm and 20C
33 // c = (concentration of dissolved oxygen in
    equilibrium with air at 1 atm and 20C)
34 c = x_O2 * 998.2 / 18; // [(mole O2) / (L solution)]
35 // V = (volume of O2, STP) / (L solution)
36 V = c * 24.06; // [(L O2, STP) / (L solution)]
37 V = V * 1000; // [(ml O2, STP) / (L solution)]
38 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

```

1 clear;
2 //clc();
3

```

```

4 // Example 3.3
5 // Page: 52
6 printf("Example -3.3 Page no.-56\n\n");
7
8
9 /***Data***/
10
11 P = 1.0; //[atm]
12 p_w = 0.023; //[atm] Vapor pressure of pure water
13 H_o = 40100; //[atm] Vapor pressure of pure oxygen
14 H_n = 80400; //[atm] Vapor pressure of pure nitrogen
15 // From Raoult's law, we have
16 // ( y_i*P ) = ( x_i*p_i )
17 // So
18 //For water
19 // ( y_w*P ) = ( x_w*p_w )
20 // For oxygen
21 //( y_o*P ) = ( x_o*p_o )
22 // And for nitrogen
23 //( y_n*P ) = ( x_n*p_n )
24
25 // Also
26 // ( y_w + y_o + y_n ) = 1
27 // ( x_w + x_o + x_n ) = 1
28
29 // In air, the mole fraction of nitrogen and oxygen
    are 0.79 and 0.21 respectively. So,
30 // y_o/y_n = 0.21/0.79
31
32 // We will take the help of matrix method to solve
    these six equations for six unknowns
33 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];
34 B = [0;0;0;1;1;0];
35 X = A^(-1)*B;
36
37 printf(" The composition in liquid and vapor phase
    are summarized in the following table:\n\n");

```

```

38 printf("      y_water      \t %f\n",X(4));
39 printf("      y_oxygen     \t %f\n",X(5));
40 printf("      y_nitrogen    \t %f\n",X(6));
41 printf("      x_water      \t %f\n",X(1));
42 printf("      x_oxygen     \t %e\n",X(2));
43 printf("      x_nitrogen    \t %e",X(3));

```

Scilab code Exa 3.4 Some Simple Applications of Raoult's and Henry's laws

```

1 clear;
2 //clc();
3
4 // Example 3.4
5 // Page: 57
6 printf("Example-3.4   Page no.-57\n\n");
7
8
9 //***Data***/
10 T = 20; //[C]
11 x_b = 0.80;
12 x_t = 0.20;
13 // Here we calculate the vapour pressures of benzene
    and toluene at 20C using the Antoine equation
14 // log10(p) = A-B/(T+C)
15 // here pressure p is in torr and temperature T is
    in C
16 // From the reported table A.2 (page 418) in the
    book, the constant A,B,C in the above equation
    for benzene have the values as
17 A_b = 6.90565;
18 B_b = 1211.033;
19 C_b = 220.79;
20 // So, for benzene
21 p_b = 10^(A_b-B_b/(T+C_b));
22 // 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

```

23 A_t = 6.95334;
24 B_t = 1343.943;
25 C_t = 219.337;
26 // So, for toluene
27 p_t = 10^(A_t-B_t/(T+C_t));
28 // Now we can compute that for benzene
29 // y_b*P = x_b*p_b
30 // let y_b*P = p_1 , so
31 p_1 = x_b*p_b;
32 // and correspondingly for toluene
33 // y_t*P = x_t*p_t
34 // let y_t*P = p_2 , so
35 p_2 = x_t*p_t;
36 // Now adding these two values of benzene and
    toluene , we have
37 // y_b*P+y_t*P = (y_b+y_t)*P
38 // i.e.
39 // P = (p_1+p_2)/(y_b+y_t)
40 // But we know that (y_b+y_t) must be equal to one i
    .e.
41 y = 1.00; // y =(y_b+y_t) sum of the mole fractions
    of the benzene and toluene in the gaseous phase
42 // Hence total pressure is
43 P = (p_1+p_2)/y;
44 // 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
45 // so
46 y_b = x_b*p_b/P;
47 y_t = x_t*p_t/P;
48 printf(" Vapour pressure of the mixture in the
    gaseous phase is %f torr\n",P);
49 printf(" Mole fraction of the benzene in the vapour
    phase is %f\n",y_b);
50 printf(" Mole fraction of the toluene in the vapour
    phase is %f",y_t);

```

Scilab code Exa 3.5 Some Simple Applications of Raoult's and Henry's laws

```
1 clear;
2 // clc ();
3
4 // Example 3.5
5 // Page: 57
6 printf("Example -3.5   Page no. -57\n\n");
7
8
9 //***Data***/
10 T = 20; //[C]
11 x_benzene = 1.00;
12 p_i = 75.2; //[torr] vapour pressure of the benzene
13 P = 760; //[torr] Pressure of the atmosphere
14
15 // So
16 y_benzene = (x_benzene*p_i)/P;
17
18 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 Raoult's and Henry's laws

```
1 clear;
2 // clc ();
3
4 // Example 3.6
5 // Page: 58
6 printf("Example -3.6   Page no. -58\n\n");
7
```



```

8
9  //***Data***//
10
11 P = 760; // [mm Hg]
12 x_b = 0.8; // Mole fraction of benzene in liquid
    phase
13 x_t = 0.2; // Mole fraction of toluene in liquid
    phase
14
15 // We will take the help of trial and error method
    to solve this problem
16 // From the table A.2 ( page 418 ), Antoine equation
    constants for benzene are
17 A_b = 6.90565;
18 B_b = 1211.003;
19 C_b = 220.79;
20
21 // and that for the toluene are
22 A_t = 6.95334;
23 B_t = 1343.943;
24 C_t = 219.337;
25
26 T = 82; // [C]
27 err = 1
28
29 while err > 10(-3)
30     p_b = 10(6.90565 - 1211.003/(T + 220.79));
31     p_t = 10(6.95334 - 1343.943/(T + 219.337));
32     y_b = x_b*p_b/P;
33     y_t = x_t*p_t/P;
34     err = abs((y_b + y_t) - 1);
35     T = T + 0.01;
36 end
37
38 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 Raoult's and Henry's laws

```
1 clear;
2 //clc();
3
4 // Example 3.7
5 // Page: 60
6 printf("Example -3.7 Page no. -60\n\n");
7
8
9 /***Data***/
10
11 V = 0.25; //[L] Volume of water
12 T_1 = 0; //[C] Initial temperature of water
13 T_2 = 20; //[C] Final temperature of water
14
15 // From the example 3.3 the mol fractions of oxygen
    and nitrogen in water at temperature 20 deg C are
16 x_o = 5.12*10^(-6); // mole fraction of oxygen
17 x_n = 9.598*10^(-6); // mole fraction of nitrogen
18
19
20 // 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
21 // From the table A.3( page 419), Henry's constant
    of oxygen and nitrogen are
22 H_o = 2.55*10^4; //[atm]
23 H_n = 5.29*10^4; //[atm]
24
25 // And vapor pressure of water at 0 deg C is
26 p_w = 0.006; //[atm]
27
28 // 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

```
29
30 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];
31 B = [0;0;0;1;1;0];
32 X = A^(-1)*B;
33
34 // Here the mole fraction of oxygen and nitrogen in
      water will be X(2) and X(3) respectively
35 // So oxygen rejected is
36 M_o_rej = V*( X(2) - x_o )/0.018;//[mole] oxygen
37 // Now At STP volume of the rejected oxygen is given
      as
38 V_o = M_o_rej*24200;//[ml] oxygen
39
40 // And rejected nitrogen is
41 M_n_rej = V*( X(3) - x_n )/0.018;//[mole] nitrogen
42 // In terms of volume
43 V_n = M_n_rej*24200;//[ml]
44
45 printf(" At equilibrium at 20 deg C the rejected
      amount of oxygen will be %0.2f ml\n",V_o);
46 printf(" At equilibrium at 20 deg C the rejected
      amount of nitrogen will be %0.2f ml\n",V_n);
47 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 Raoult's and Henry's laws

```
1 clear;
2 //clc();
3
4 // Example 3.8
5 // Page: 61
```

```

6 printf("Example -3.8 Page no.-61\n\n");
7
8
9 /***Data***/
10
11 P_1 = 5; //[atm]
12 y_n = 0.79; // Mole fraction of nitrogen in
    atmosphere
13 P_2 = 1.0; //[atm]
14 M = 55; //[kg] Mass of the diver
15 x_w = 0.75; // Fraction of water in human body
16 T = 37; //[C] Body temperature of the diver
17
18 // At 37 deg temperature , the Henry's constant for
    N2 from the table A.3 ( page 419 ) by the linear
    interpolation is
19 H_n = 10.05*10^(4); //[atm]
20
21 // Now, moles of nitrogen rejected will be
22 // M_rej = (moles of body fluid)*( x_N2,5 atm - x_N2
    ,1 atm)
23 // So
24 M_rej = (M*1000*x_w/18)*( P_1*y_n/H_n - P_2*y_n/H_n)
    ;//[mol]
25 // At STP the volume of the rejected nitrogen will
    be
26 V_n = M_rej*24.2; //[L]
27
28 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

```
1 clear;
2 //clc();
3
4 // Example 4.1
5 // Page: 67
6 printf("Example-4.1 Page no.-67\n\n");
7
8 /***Data***/
9 T = 671.7;//[R] Equilibrium temperature
10 m_steam = 1;//[lbm] Condensing amount of the steam
11 // Using values from the steam table [1], we find
    that
12 delta_h_condensation = -970.3//[Btu/lbm] Enthalpy
    change of the steam
13 delta_s_condensation = -1.4446;//[Btu/(lbm*R)]
    Entropy change of the steam
14
15 // Gibb's free energy change of the steam is
16 delta_g_condensation = delta_h_condensation - T*
```

```

    delta_s_condensation; //[Btu/lbm]
17
18 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

```

1 clear;
2 //clc();
3
4 // Example 4.2
5 // Page: 77
6 printf("Example -4.2 Page no. -77\n\n");
7
8 //***Data***/
9
10 // let we denote graphite by 'g' and diamond by 'd'
11 // Gibb's free energies of graphite and diamond are
    given by
12 g_g = 0.00; //[kJ/mol]
13 g_d = 2.90; //[kJ/mol]
14
15 // Specific volumes of graphite and diamond are
    given by
16 v_g = 5.31*10^(-1); //[kJ/(mol*kbar)]
17 v_d = 3.42*10^(-1); //[kJ/(mol*kbar)]
18
19 // Now from the equation 4.32 ( page 74) given in
    the book, we have
20 // (dg/dP) = v , at constant temperature
21 // where 'v' is specific volume
22 // let us denote (dg/dP) by 'D' ,so
23 D_g = v_g; //[J/(mol*Pa)] For graphite
24 D_d = v_d; //[J/(mol*Pa)] For diamond

```

```

25
26 // Now we can take our plot from P = 0( =1 ),
    however, total pressure is 1 atm.
27 // 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
28 // So the equation of the line for graphite is
29 //  $g = D_g * P + g_g$ 
30 // and that for diamond
31 //  $g = D_d * P + g_d$ 
32
33 P = [0:1:30]';
34
35 plot2d(P,[ D_d*P+g_d D_g*P+g_g ],style=[color("
    darkgreen"),color("red")]);
36
37 xlabel("Pressure , P, kbar");
38 ylabel("Gibb ' 's free energy per mol, g, kJ/mol");
39
40 printf(" Gibb ' 's free energy-pressure diagram for
    graphite-diamond system at 25 degC is as shown in
    the graphic window. ");
41 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

```

1 clear;
2 //clc();
3
4 // Example 4.3
5 // Page: 80
6 printf("Example -4.3 Page no. -80\n\n");
7
8 //***Data***/

```

```

9 // We have the system which consists of isobutane
    and normal butane and isomerisation is taking
    place between them
10 // The equilibrium constant for this reaction is
    given by
11 //  $K = (\text{mole fraction of isobutane}) / (\text{mole fraction}$ 
    of n-butane) =  $x_{\text{iso}} / x_{\text{normal}}$ 
12
13 // For this reaction , at 25C,
14 K = 4.52;
15
16 // and
17 //  $x_{\text{iso}} + x_{\text{normal}} = 1$ 
18 // so
19 //  $K = x_{\text{iso}} / (1 - x_{\text{iso}})$ 
20
21 // solving for  $x_{\text{iso}}$ 
22 deff('[y]=f(x_iso)', 'y = x_iso / (1 - x_iso) - K');
23 x_iso = fsolve(0,f);
24
25 printf(" Mole fraction of isobutane isomer in
    equilibrium is %0.2 f", 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

```
1 clear;
2 //clc();
3
4 // Example 5.1
5 // Page: 89
6 printf("Example -5.1 Page no.-89\n\n");
7
8 //***Data***//
9 T=212; // [F]
10
11 //*****//
12 //From the steam table, we have
13 delta_h=970.3; // [Btu/lbm]
14 delta_v=26.78; // [ft^(3)/lbm] and
15
```

```

16 // changing the units
17 delta_h1=delta_h*778; //[ ft*lb_f/lbm ]
18 delta_v1=delta_v*144; //[ ft*in^(2)/lbm ]
19 T=671.7; //[R]
20
21 // We have dP/dT = delta_h/(T*delta_v)
22 //Thus
23 dP_by_dT=delta_h1/(T*delta_v1); //[ psi/R]
24
25 printf("The value of dP/dT is %f psi/R",dP_by_dT);
26 //Using the nearest adjacent steam table entries for
    vapour pressure , wee have
27 //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

```

1 clear;
2 //clc();
3
4 // Example 5.2
5 // Page: 90
6 printf("Example-5.2 Page no.-90\n\n");
7
8 //***Data**//
9 p_2=0.005; // [psia]
10 R=1.987/18; //[1/R]
11
12 //*****//
13
14 //From the steam tables at the tripple point , we
    find
15 T_1=460+32.018; //[R]
16 p_1=0.0887; //[psia]
17

```

```

18 //delta_h(solid to gas) = delta_h(sublimation) =
    1218.7;//[Btu/lbm]
19 delta_H=1218.7;//[Btu/lbm]
20
21 //Assuming that the enthalpy change of vaporization
    is independent of temperature (a fairly good
    approximation in this case)
22 //we start with Eq. 5.10 and rearrange:
23 //1/T_2 = 1/T_1-(log(p_2/p_1))*R/delta_H
24 //So
25
26 T_2=1/(1/T_1-(log(p_2/p_1))*R/delta_H);//[R]
27 //Changing the temperature in fahrenheit
28 T_2F=T_2-460;//[F]
29
30 printf("The temperature is %f R",T_2F);
31 //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

```

1 clear;
2 //clc();
3
4 // Example 5.3
5 // Page: 91
6 printf("Example -5.3 Page no.-91\n\n");
7
8 //***Data***/
9
10 T_3=1155.2;//[R]
11 T_2=652.9;//[R]
12 T_1=787.5;//[R]

```

```

13 p_2=10;//[psia]
14 p_1=100;//[psia]
15
16 //*****//
17 //Here we can write Eq. 5.9 as reported in the book
    in the form most often seen.
18 // log(p)=A-B/T
19 //Where A and B are constants to be determined from
    the pair of T and p values above.
20
21 //we simply write
22 //log(10)=A-B/652.9;
23 //log(100)=A-B/787.5;
24 // We have to solve the above two simultaneous
    equations having two variables A and B.
25
26 M=[1 -1/652.9;1 -1/787.5];
27 C=[log(10);log(100)];
28 X=inv(M)*C;
29 A=X(1);
30 B=X(2);
31
32 // By straightforward algebra we find the values of
    A and B. Thus, for 1155.2 R we have
33 p_3=exp(A-B/T_3);
34
35 printf("Vapour pressure of water at given
    temperature is %f psia\n\n",p_3);
36 // p_3=3499 psia.
37 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

```

1 clear;
2 //clc();
3
4 // Example 5.4
5 // Page: 94
6 printf("Example-5.4 Page no.-94\n\n");
7
8 /***Data***/
9 // At Tr = 0.7, we read
10 Pr=0.023;
11 // and thus accentric factor is given by
12
13 w=-log10(0.023)-1;
14
15 printf("The accentric factor based on the given data
        is %f",w);
16 //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

```

1 clear;
2 //clc();
3
4 // Example 5.5
5 // Page: 94
6 printf("Example-5.5 Page no.-94\n\n");
7
8 /***Data***/
9 //From Antoine equation we have
10 // log(p) = A-B/(T+C)
11 //Solving above equation for T, we have
12 // T = B/(A-log(p))-C
13 //Inserting the values of the constants for the

```

```

    water which are reported in the given book in the
    table A.2 (page 419),
14 // and the value of 1.00 atm expressed in torr, we
    find that
15
16 A=7.96681;
17 B=1668.21;
18 C=228.0;
19 p=760;//[ torr]
20
21 //Thus
22 T=B/(A-log10(p))-C;
23
24 printf("NBP of water using antoine equation and
    table A.2 is %f C",T);
25
26 //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

```

1 clear;
2 //clc();
3
4 // Example 5.6
5 // Page: 96
6 printf("Example -5.6 Page no.-96\n\n");
7
8 //***Data***/
9 T_2=-22;//[C]
10 // converting temperature in fahrenheit

```

```

11 T_2F=T_2*9/5+32; //[F]
12
13 //Expressing T_2 in Rankine
14 T_2R=460+T_2F; //[R]
15
16 //*****//
17
18 //delta_h = delta_h(fusion)
19 delta_h=143.35*778; //[ft*lbf/lbm]
20
21 //delta_v = v_water-v_ice
22 delta_v=0.01602-0.01747; //[ft^(3)/lbm]
23
24 // changing the unit
25 delta_v1=delta_v*144; //[ft*in/lbm]
26
27 //and
28 T_1=460+32; //[R]
29 dP_by_dT=delta_h/(T_1*delta_v1); //[psi/R] at 32F
30 delta_T=T_2R-T_1;
31
32 //This gives the rigorously correct slope of the
    liquid-solid curve at 32F on a P-T diagram.
33 //Here we use P instead of p because neither phase
    is a gas, so this is not a vapour pressure.
34 //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
35 // delta_P = integrate(dP_by_dT)*dT = (dP_by_dT)*
    delta_T
36 //So
37
38 delta_P=(dP_by_dT)*delta_T; //[psi]
39
40 // From this we can estimate the final pressure as
41 delta_P=delta_P+0.09; //[psi]

```

```
42
43 printf("Freezing preeure of water at given
    temperature is %f psi",delta_P);
44 // In this case, the experimental pressure is well
    known, because this temperature corresponds to
    the tripple point between liquid and water,
45 // 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).
46 // The measured value is 30000 psia, which shows
    that our assumption of a straight line on a P-T
    plot ( $\Delta h / (T \Delta v) = \text{constant}$ ) is only
    approximately correct.
```

Chapter 6

Partial Molal Properties

Scilab code Exa 6.1 Tagent Slopes

```
1 clear;
2 // clc ();
3
4 // Example 6.1
5 // Page: 108
6 printf("Example -6.1 Page no. -108\n\n");
7
8 //***Data***/
9 T = 20; //[C]
10 m_1 = 0; //[molal]
11 m_2 = 1; //[molal]
12 // The data given in the figure 6.2 , as reported in
    // book, can be repersented with excellent accuracy
    // by a simple data fitting equation
13 //V = 1.0019+0.054668*m-0.000418*m^(2);
14 // Where 'V' is( solution volume, liters per 1000g
    // of water ) and 'm' is the molality of ethanol in
    // water
15 //The partial molal volume is obtained by
    // differentiating the expression of the 'V' with
    // respect to 'm'
```

```

16 // v_ethanol = dV/dm = 0.054668 - 2*0.000418*m
17 // So that at zero molality
18 m = 0; // [molal]
19 // the partial molal volume is
20 v_1 = 0.054668 - 2*0.000418*m; // [L/mol]
21 // and at
22 m = 1; // [molal]
23 v_2 = 0.054668 - 2*0.000418*m; // [L/mol]
24 v_1 = v_1*1000; // [cm^(3)/mol]
25 v_2 = v_2*1000; // [cm^(3)/mol]
26 printf("Partial molal volume of ethanol in water at
    zero molality is %f cm^(3)/mol\n", v_1);
27 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

```

1 clear;
2 // clc ();
3
4 // Example 6.2
5 // Page: 109
6 printf("Example -6.2 Page no. -109\n\n");
7
8 // ***Data***//
9 n_eth = 1; // [mol]
10 W_water = 1; // [kg]
11 Temp = 20; // [C]
12 // For pure ethanol at 20C
13 v_ethanol = 58.4; // [cm^(3)/mol]
14 v_ethanol = v_ethanol/1000; // [L/mol]
15 v_water = 1.0019; // [L/1000g]
16 // Molality of ethanol in water is
17 m = n_eth/W_water; // [molal]
18 // We have the equation used in the previous example

```

```

as
19 V_final_mix = 1.0019+0.054668*m-0.000418*m^(2);
20 // Where 'V' is( solution volume, liters per 1000g
    of water ) and 'm' is the molality of ethanol in
    water
21 // V is the final volume of the solution
22 // The volume expansion on mixing is
23 V_exp = V_final_mix-v_ethanol-v_water;//[L]
24 V_exp = V_exp*1000;//[cm^(3)]
25 printf("Volume change on mixing ethanol and water is
    %0.3f cubic cm",V_exp);
26 // 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

```

1 clear;
2 //clc();
3
4 // Example 6.3
5 // Page: 109
6 printf("Example-6.3 Page no.-109\n\n");
7
8 //***Data***//
9 // All the data are same as in the previous example
10 // The equation 6.5 reported in the book is
11 //  $\Delta V_{\text{mixing}} = V_{\text{solution final}} - V_{\text{(solution and material to be mixed)}} = \int_{n_0}^{n_1} (v_i - v_{i_0}) dn$ 
12 // Here the integrated average value of  $v_i$  over the
    molality range from 0 to 1 is
13 v_i_average = 0.05425;//[L/mol]
14 // and
15 v_i_0 = 0.0584;//[L/mol]
16 delta_n = 1.00;//[mol]

```

```

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

```

Scilab code Exa 6.4 Tangent Intercept Concept

```

1 clear;
2 //clc();
3
4 // Example 6.4
5 // Page: 113
6 printf("Example-6.4 Page no.-113\n\n");
7
8 //***Data***/
9 m = 1;//[molal] Molality of the solution with
    respect to ethanol
10 M_water = 18;//[g/mol] molecular weight of water
11
12 // First we convert molality to mole fraction
13 x_ethanol = m/(m + 1000/M_water);
14
15 // For the low range of data point on figure 6.5(
    page 112), we can fit an equation
16 // (Specific volume ) = 0.018032 + 0.037002*
    x_ethanol - 0.039593*x_ethanol^(2) + 0.21787*
    x_ethanol^(3)
17 // This is applicable for (0 < x_ethanol < 0.04 ),
    which is the case we have
18
19 // So
20 v_tan = 0.018032 + 0.037002*x_ethanol - 0.039593*
    x_ethanol^(2) + 0.21787*x_ethanol^(3);//[L/mol]
21

```

```

22 // Now we will find the derivative of the specific
    volume with respect to x_ethanol at the known
    point x_ethanol
23 // (dv/dx_ethanol) = 0.037002 - 2*0.039593*
    x_ethanol + 3*0.21787*x_ethanol^(2)
24 // Hence
25 v_derv_tan = 0.037002 - 2*0.039593*x_ethanol +
    3*0.21787*x_ethanol^(2); // [L/mol]
26
27 // By simple geometry from the figure 6.6(page 113)
    of the book we find
28 // a = v_tan + (1-x_tan)*(dv/dx_1)_tan
29 // b = v_tan - x_tan*(dv/dx_1)_tan
30
31 // We have a = v_ethanol and b = v_water
32 x_tan = x_ethanol;
33 // So
34 v_ethanol = v_tan + (1-x_tan)*(v_derv_tan); // [L/mol]
35 v_water = v_tan - x_tan*(v_derv_tan); // [L/mol]
36
37 printf(" Partial molar volume of the ethanol in the
    given solution is %f L/mol\n",v_ethanol);
38 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

```

1 clear;
2 //clc();
3
4 // Example 6.5
5 // Page: 115
6 printf("Example-6.5 Page no.-115\n\n");
7
8 //***Data***/

```

```
9 printf("This is a theoretical 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

```
1 clear;
2 //clc();
3
4 // Example 6.6
5 // Page: 115
6 printf("Example -6.6   Page no.-115\n\n");
7
8 /***Data***/
9 printf("This is a theoretical question and there are
    no any numerical components. Refer to page no
    115 of the book.");
```

Scilab code Exa 6.7 Partial Mass Properties

```
1 clear;
2 //clc();
3
4 // Example 6.7
5 // Page: 117
6 printf("Example -6.7   Page no.-117\n\n");
7
8 /***Data***/
9 x_sulph = 0.6;
10 x_water = 0.4;
11 Temp = 200;//[F]
12 // 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.
13 h_water_per_pound = 25;//[Btu/lbm]
14 h_sulph_per_pound = -100;//[Btu/lbm]
15 // also molecular weight of water and sulphuric acid
    are
16 M_water = 18;//[lbm/lbmol]
17 M_sulph = 98;//[lbm/lbmol]
18 // Using equation 6.20 given in the book we have
19 h_water = h_water_per_pound*M_water;//[Btu/lbmol]
20 h_sulph = h_sulph_per_pound*M_sulph;//[Btu/lbmol]
21 printf("Partial molar enthalpy of water in the
    mixture is %f Btu/lbmol\n",h_water);
22 printf(" Partial molar enthalpy of H2SO4 in the
    mixture is %f Btu/lbmol",h_sulph);

```

Scilab code Exa 6.8 Differential Heat of Mixing

```

1 clear;
2 //clc();
3
4 // Example 6.8
5 // Page: 119
6 printf("Example-6.8 Page no.-119\n\n");
7
8 //***Data***/
9 x_sulph = 0.6;
10 x_water = 0.4;
11 M_i = 18;//[lbm/lbmol]
12 Temp = 200;//[F]
13 // From Equation 6.11 as given in the book, we have
14 //  $dQ/dm_{in} = h_i - h_{in}$ 
15 // 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.
16 // So at 200F we have
17 h_i = 25;//[Btu/lbm]
18 h_in = 168;//[Btu/lbm]
19 // hence
20 dQ_by_dm_in = h_i-h_in;;//[Btu/lbm]
21 // Now
22 dQ_by_dn_in = M_i*dQ_by_dm_in;//[Btu/lbmol]
23 printf("The amount of heat removed to keep the
    temperature constant is %f Btu/lbm of water added
    ",dQ_by_dm_in);
24 // 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

```

1 clear;
2 //clc();
3
4 // Example 6.9
5 // Page: 119
6 printf("Example-6.9 Page no.-119\n\n");
7
8 //***Data***/
9 m_sulph = 0.6;
10 m_water = 0.4;
11 m = m_sulph+m_water;
12 Temp = 200;//[F]
13 // Here at 200F we can read the solution enthalpy
    h_solution and pure H2SO4 enthalpy h_sulph such
    that
14 h_solution = -50;//[Btu/lbm]
15 h_sulph = 53;//[Btu/lbm]

```



```

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

```

Scilab code Exa 6.10 Integral Heat of Mixing

```

1 clear;
2 //clc();
3
4 // Example 6.10
5 // Page: 120
6 printf("Example-6.10 Page no.-120\n\n");
7
8 /***Data***/
9 x_sulph = 0.6;
10 x_water = 0.4;
11 Temp = 200;//[F]
12 // At the 200F we have
13 h_water = 25;//[Btu/lbm]
14 h_sulph = -100;//[Btu/lbm]
15 // From equation 6.16 (as reperated in the book),
    rewritten for masses instead of moles we have
16 h_solution = h_water*x_water+h_sulph*x_sulph;//[Btu
    /lbm]
17 printf("Enthalpy of the solution is %f Btu/lbm",
    h_solution);

```

Scilab code Exa 6.11 Application of Gibbs Duhem equation

```
1 clear;
2 // clc ();
3
4 // Example 6.11
5 // Page: 121
6 printf("Example -6.11 Page no. -121\n\n");
7
8 /***Data***/
9 x_b = 0;
10 x_a = 1;
11 // We have
12 //  $dv_a/dx_a = 3x_b^2 + 2x_b$ 
13 // We have the equation
14 //  $dv_b/dx_a = -(dv_a/dx_a)/(x_b/x_a)$ 
15 // So
16 //  $dv_b/dx_a = -(x_a/x_b) * (3x_b^2 + 2x_b)$ 
17 dv_b_by_dx_a = x_a * (-3*x_b - 2);
18 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

```
1 clear;
2 // clc ();
3
4 // Example 6.12
5 // Page: 122
6 printf("Example -6.12 Page no. -122\n\n");
7
```

```
8  //***Data***//
9  x_b = 0;
10 x_a = 1;
11 // We have
12 //dv_a/dx_a = 3*x_b^(2)+2*x_b+1
13 // We have the equation
14 // dv_b/dx_a = -(dv_a/dx_a)/(x_b/x_a)
15 // So
16 // dv_b/dx_a = -(x_a/x_b)*(3*x_b^(2)+2*x_b+1)
17 //dv_b_by_dx_a = -x_a*(3*x_b+2+1)/x_b;
18 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

```
1 clear;
2 // clc ();
3
4 // Example 7.1
5 // Page: 134
6 printf("Example -7.1 Page no. -134\n\n");
7
8 //***Data***//
9 T = 220+459.67; // [R] Temperature in Rankine
10 P = 500; // [psia] Pressure
11 R = 10.73; // [(psi*ft^(3))/(lbmol*R)] Gas constant
12
13 // We will follow the method 'a' as the book has
14 // given the multiple methods to solve this problem
15 // From the equation 7.10 given in the book (page
16 // 132), we have
17 //  $(f/P) = \exp((-1/(R*T))*\text{intgrate}(a*dp))$ , with
18 // integration limits from zero to 'P'
19 // Where 'a' is known as volume residual
```

```

17 // Let us say , I = integrate(a*dp)
18
19 // From the table 7.A(page 134) given in the book,
    the average value of alpha(a) is
20 a = 4.256; //[ft^(3)/lbmol]
21 // so
22 I = integrate('a*p^(0)', 'p', 0, P);
23
24 // Now
25 f = P*exp((-1/(R*T))*I); //[psia]
26 printf("Fugacity of propane gas at the given
    condition is %f psia", f);

```

Scilab code Exa 7.2 Compressibility factor and volume residual

```

1 clear;
2 //clc();
3
4 // Example 7.2
5 // Page: 138
6 printf("Example-7.2 Page no.-138\n\n");
7
8 //***Data***/
9 T = 100 + 460; //[R] Temperature of the system in
    Rankine
10 P = 1; // [psia]
11 R = 10.73; //[(psi*ft^(3)/(lbmol*R))] Gas constant
12
13 // 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
14 v = 0.016136*18; //[ft^(3)/lbmol]
15 z = (P*v)/(R*T);
16
17 // and volume residual is given by

```

```

18 a = ((R*T)/P)*(1-z);//[ft^(3)/lbmol]
19
20 printf(" Compressibility factor the liquid water at
    the given condition is %f\n ",z);
21 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

```

1 clear;
2 //clc();
3
4 // Example 7.3
5 // Page: 138
6 printf("Example-7.3 Page no.-138\n\n");
7
8 //***Data***/
9
10 T = 100+460;//[R] Temperature
11 P = 1000;//[psia] Pressure
12 R = 10.73;//[ (psi*ft^(3))/(lbmol*R)] Gas constant
13
14 // 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
15 // fugacity at point b is calculated by the equation
16 // (f/P)_b = exp((-1/(R*T))*integrate(a*dp)), with
    integration limit of p, 0 and P = 0.9503
17 // We have
18 f_b = 0.95;//[psia]
19
20 // We also can write
21 f_c = f_b;//[psia]
22
23 // To find the value of f_d, we use the equation

```

```

24 // integrate(d(logf))_T = integrate((v/(R*T))*dp)_T
    .....(1)

25 // here 'v' is practically constant(for a liquid),
    and
26 v = 0.016136*18; //[ft^(3)/lbmol]
27
28 // and from the figure 7.3, we have
29 P_d = 1000; //[psia]
30 P_c = 1; //[psia]
31
32 // integrating the left hand side of the equation
    with the integration limits f_c and f_d and
    solving, we have
33 f_d = f_c*exp((v/(R*T))*integrate('p^(0)', 'p', P_c,
    P_d));
34
35 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

```

1 clear;
2 //clc();
3
4 // Example 7.4
5 // Page: 145
6 printf("Example-7.4 Page no.-145\n\n");
7
8 //***Data***/
9
10 T = 78.15; //[C]
11 P = 1.0; //[atm]
12 // 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.
13 // Thus vapor pressures of the pure species at the
    given temperature are
14 p_a_0 = 0.993; //[atm] Pure ethanol vapor pressure at
    78.15C
15 p_b_0 = 0.434; //[atm] Pure water vapor pressure at
    78.15C
16
17 // Also composition of the azeotrope is
18 x_a = 0.8943; // Amount of ethanol in the liquid
    phase
19 x_b = 0.1057; // Amount of water in liquid phase
20
21 // Also, for an azeotrope mixture
22 y_a = x_a; // Amount of ethanol in vapor phase
23 y_b = x_b; // Amount of water in the vapor phase
24
25 // For ideal gas , fugacity is equal to the total
    pressure of the system , i.e.
26 // f_i_0 = P , (where P is the system pressure)
27 // 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.
28 // f_i_0 = p_i
29
30 // Now, fugacity of each species in gaseous phase
    and liquid phase will be equal
31 // so, writing the expression for both liquid and
    gas phase fugacity and equatinh them, we have
32 // f_a_2 = f_a_1 = (y*Y*P)_a_1 = (x*Y*p)_a_2
    ..... (1)
33 // f_b_2 = f_b_1 = (y*Y*P)_b_1 = (x*Y*p)_b_2
    ..... (2)
34
35 // We observe that this system has four values of 'Y
    ', one for each of the two species in each of two
    phases.

```



```

36 // 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
37 Y_a_1 = 1.0;
38 Y_b_1 = 1.0;
39
40 // 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
41 Y_a_2 = ((y_a*P)/(x_a*p_a_0));
42 Y_b_2 = ((y_b*P)/(x_b*p_b_0));
43
44 // From equations 1 and 2, the fugacity of each
    species in each phase is given by
45 f_a_1 = (y_a*Y_a_1*P);//[atm]
46 f_b_1 = (y_b*Y_b_1*P);//[atm]
47 // and from the definition we have
48 f_a_2 = f_a_1;//[atm]
49 f_b_2 = f_b_1;//[atm]
50
51 // As we have defined above about the pure species
    fugacity, so
52 // For vapor phase
53 f_a_1_0 = P;//[atm]
54 f_b_1_0 = P;//[atm]
55
56 // For liquid phase
57 f_a_2_0 = p_a_0;//[atm]
58 f_b_2_0 = p_b_0;//[atm]
59
60 printf(" The results are summarized in the following
    table:\n\n \tPhase\t\t\t\t\t Etahnol, i=a\t\t\t\t\t
    Water, i=b\n\n");
61 printf(" \tVAPOR, PHASE 1\n");
62 printf(" \t f_i-1 , atm \t\t\t\t %f \t\t\t\t\t %f\n",
    f_a_1, f_b_1);
63 printf(" \t f_i-1_0 , atm \t\t\t\t %f \t\t\t\t\t %f\n",

```

```

        f_a_1_0 , f_b_1_0 );
64 printf(" \t Y_i_1 (assumed) \t\t %f \t\t\t\t %f\n\
n", Y_a_1 , Y_b_1 );
65 printf(" \t LIQUID, PHASE 2\n");
66 printf(" \t f_i_2 , atm \t\t\t %f \t\t\t\t %f\n",
        f_a_2 , f_b_2 );
67 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 );
68 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

```

1 clear;
2 //clc();
3
4 // Example 7.5
5 // Page: 149
6 printf("Example -7.5 Page no. -149\n\n");
7
8 //***Data***//
9
10 T = 220+460; // [R] Temperature in rankine
11 P = 1000; // [psia] Pressure
12 y_methane = 0.784; // Mol fraction of methane in the
        given mixture
13 y_butane = (1-y_methane); // Mol fraction of n-butane
        in the given mixture
14 R = 10.73; // [(psia*ft^(3))/(lbmol*R)] gas constant
15
16 // In this problem, we need the partial molar volume
        residual.
17 // We find its value at 100 psia by plotting the
        volume residuals at 100 psia as a function of mole
        fraction, as shown in figure 7.9( page 150 )

```

```

18 // 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
19 // similarly volume residual is determined for all
    other pressures and plot them vs pressure, as
    shown in Figure 7.10 (page 151).
20 // From this plot we find the integral we need by
    numerical integration ( trapezoid rule ) as 290
    ft^(3)/lbmol.
21
22 // Thus, for methane
23 //  $f_i/(P*y_i) = \exp((-1/(R*T))*\int_0^P a_i dp)$ 
    with integral limits 0 to  $P = 1000$  psia
24 // Let  $I = \int_0^P a_i dp$  and  $J = f_i/(P*y_i)$ 
    , so
25  $I_m = 290$ ; //[ft^(3)/lbmol]
26
27 // and
28  $J_m = \exp((-1/(R*T))*I_m)$ ;
29
30 // hence
31  $f_{\text{methane}} = J_m * P * y_{\text{methane}}$ ; //[psia] fugacity of
    methane
32
33 // doing the same process for butane, we find
34  $I_b = 5859$ ; //[ft^(3)/lbmol]
35 // so, for butane we find
36  $J_b = \exp((-1/(R*T))*I_b)$ ;
37 // hence
38  $f_{\text{butane}} = J_b * P * y_{\text{butane}}$ ; //[psia] fugacity of butane
39
40 printf(" Fugacity of the methane in the gaseous
    mixture is %0.0f psia\n",f_methane);
41 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

```
1 clear;
2 //clc();
3
4 // Example 7.6
5 // Page: 153
6 printf("Example -7.6 Page no. -153\n\n");
7
8 //***Data***//
9
10 T = 220+460; //[R] Temperature in rankine
11 P = 1000; //[psia] Pressure
12 x_methane = 0.784; // Mol fraction of methane in the
    given mixture
13 x_butane = (1-x_methane); // Mol fraction of n-butane
    in the given mixture
14
15 // From the example 7.5, we found directly from the
    PvT data that for methane
16 //  $(f_i/(P*x_i)) = 0.961 = (v_i*Y_i) = \phi_{cap_i}$ 
17 // So, we can write that
18 v_i_into_Y_i = 0.961;
19 phi_cap_i = 0.961;
20
21 // From Starling's tables of hydrocarbon properties
    we read that for pure methane at this T and P,
22 //  $(F_i/P) = v_i = \phi_i$ , from which it follows
23 v_i = 0.954;
24 phi_i = v_i;
25 Y_i = phi_cap_i/v_i;
26
27 printf(" The value of v_i is %f\n",v_i);
28 printf(" The value of Y_i is %f\n",Y_i);
```

```
29 printf(" The value of phi_cap_i is %f",phi_cap_i);
```

Scilab code Exa 7.7 Fugacities from an EOS for gas mixtures

```
1 clear;
2 //clc();
3
4 // Example 7.7
5 // Page: 154
6 printf("Example-7.7 Page no.-154\n\n");
7
8 /***Data***/
9
10 T_r = 0.889;
11 P_r = 1.815;
12
13 // Using the properties of n-butane from appendix A
14 // .1 and the equation 7.W, we find that
15 //  $(f/P) = v = \phi = \exp((P_r/T_r)*f(T_r,w))$ 
16 // Say,  $f(T_r,w) = f_f$ 
17 f_f = -0.48553;
18 // so
19 v = exp((P_r/T_r)*f_f);
20 phi = v;
21 printf(" The value of v=phi for n-butane at given
22 condition is %f",v);
```

Chapter 8

Vapor Liquid Equilibrium VLE at Low Pressures

Scilab code Exa 8.1 Calculation of K factors

```
1 clear;
2 // clc ();
3
4 // Example 8.1
5 // Page: 163
6 printf("Example-8.1 Page no.-163\n\n");
7
8 /***Data***/
9 x_acetone = 0.05; // Mole fraction of Acetone in
   liquid
10 x_water = (1-x_acetone);
11 // Using the values from table 8.1(page 162) as
   reported in the book we have
12 y_acetone = 0.6381; // Mole fraction of Acetone in
   vapour
13 y_water = (1-y_acetone);
14 // We know that
15 //  $K_i = y_i/x_i$ 
16 // So 'K' factors are
```

```

17 K_acetone = y_acetone/x_acetone;
18 K_water = y_water/x_water;
19 // and relative volatility is
20 a = K_acetone/K_water;
21 printf("The K factor of acetone is %f\n",K_acetone);
22 printf(" The K factor of water is      %f\n",K_water);
23 printf(" The relative volatility is    %f",a)

```

Scilab code Exa 8.2 Liquid phase activity coefficient

```

1 clear;
2 //clc();
3
4 // Example 8.2
5 // Page: 165
6 printf("Example-8.2 Page no.-165\n\n");
7
8 //***Data**//
9 P = 1;//[atm]
10 Temp = 74.8;//[C]
11 // Here we need to know the vapour pressure p_i
    corresponding to the temperatures of each of the
    values in the table.
12 // We can estimate them using Antoine equation by
    the help of the values given in table A.2(page
    418) in the book
13 // log10(p_i) = A-B/(T+C)
14 // for acetone the constants are given as
15 A_a = 7.02447;
16 B_a = 1161;
17 C_a = 224;
18 // So p_acetone is given by
19 p_acetone = 10^(A_a-B_a/(Temp+C_a));//[mmHg]
20 // similarly for water the constants are given as
21 A_w = 7.94917;

```

```

22 B_w = 1657.462;
23 C_w = 227.02;
24 // So p_water is given by
25 p_water = 10^(A_w-B_w/(Temp+C_w)); // [mmHg]
26 // expressing the pressures in atm
27 p_acetone = p_acetone/760; // [atm]
28 p_water = p_water/760; // [atm]
29 // Now from table 8.1 given the book
30 y_acetone = 0.6381;
31 x_acetone = 0.05;
32 y_water = (1-y_acetone);
33 x_water = (1-x_acetone);
34 // Hence the liquid-phase activity coefficients for
    acetone and water are given as
35 Y_acetone = y_acetone*P/(x_acetone*p_acetone);
36 //and
37 Y_water = y_water*P/(x_water*p_water);
38 printf("Liquid-phase activity coefficient for
    acetone is %f\n",Y_acetone);
39 printf(" Liquid-phase activity coefficient for water
    is %f\n",Y_water);

```

Scilab code Exa 8.3 Non ideal solution behaviour

```

1 clear;
2 //clc();
3
4 // Example 8.3
5 // Page: 167
6 printf("Example-8.3 Page no.-167\n\n");
7
8 //***Data***/
9
10 x_a = 0.05; // mole fraction of acetone in liquid
    phase

```



```

11 x_w = (1-x_a); // mole fraction of the water in the
    liquid phase
12 P = 1.00; // [atm] Total pressure in vapor phase
13
14 // Let us assume that the solution is ideal
15 // We will take the help of trial and error method
    and find a temperature at which sum of the
    computed ideal solution vapor phase mole fraction
    is 1.00
16 // For our first try let the temperature is
17 T_1 = 80; // [C]
18 // Now from Table A.2( page 418), the Antoine
    equation constant for acetone are
19 A_a = 7.02447;
20 B_a = 1161;
21 C_a = 224;
22 // and that for water
23 A_w = 7.94917;
24 B_w = 1657.462;
25 C_w = 227.02;
26
27 // Now from Antoine equation
28 //  $\log_{10}(p) = A - B/(T+C)$ 
29 // So, vapor pressure for pure acetone at 80 C (in
    atm) is
30 p_a_1 = (1/760)*10^(A_a - B_a/(T_1+C_a)); // [atm]
31 // and that of water is
32 p_w_1 = (1/760)*10^(A_w - B_w/(T_1+C_w)); // [atm]
33
34 // Now from Raoult's law
35 //  $y_i * P = x_i * p_i$ 
36 // so, vapor phase composition at this temperature
    is
37 y_a_1 = (x_a * p_a_1) / P;
38 y_w_1 = (x_w * p_w_1) / P;
39
40 // Sum of these two composition is
41 y_1 = (y_a_1 + y_w_1);

```

```

42 // Since, y_1 is not equal to 1.00, so assumed
    temperature is wrong
43
44 // Now we will assume our temperature as
45 T_2 = 96.4060; //[C]
46
47 // Again, from Antoine equation
48 // log10(p) = A - B/(T+C)
49 // So, vapor pressure for pure acetone at 80 C (in
    atm) is
50 p_a_2 = (1/760)*10^(A_a - B_a/(T_2+C_a)); //[atm]
51 // and that of water is
52 p_w_2 = (1/760)*10^(A_w - B_w/(T_2+C_w)); //[atm]
53
54 // Now from Raoult's law
55 // y_i*P = x_i*p_i
56 // so, vapor phase composition at this temperature
    is
57 y_a_2 = (x_a*p_a_2)/P;
58 y_w_2 = (x_w*p_w_2)/P;
59
60 // Sum of these two composition is
61 y_2 = (y_a_2 + y_w_2);
62 // Value of y_2 is equal to 1.00, so our assumption
    is right
63 // These are the values when the solution would
    behave as ideal, but this is not the actual scene
64 // 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
65 T_e = 74.8; //[C] Boiling temperature
66 y_a_e = 0.6381; // vapor phase composition of acetone
67
68 printf(" Comparison of experimental values to those
    computed by the ideal solution assumption ,
    x_acetone = 0.05 and P = 1.00 atm\n\n");
69 printf(" \t\t\t Experimental Values from Table 8.1 \
\t\t\t\t Values calculated assuming idea solution\n

```



```

24 // n_1/n_total = (x_feed-x_a_2)/(x_a_1-x_a_2)
25 // and hence
26 n_1 = (x_feed-x_a_2)/(x_a_1-x_a_2)*n_total;//[mol]
27 n_2 = (n_total-n_1);//[mol]
28 // so
29 n_a_1 = 0.65*n_1;//[mol]
30 // and
31 n_a_2 = 0.98*n_2;//[mol]
32 printf(" Total moles of water present in the first
    phase is %f mol\n",n_a_1);
33 printf(" Total moles of water present in the second
    phase is %f mol",n_a_2);

```

Scilab code Exa 8.5 Two liquid phase

```

1 clear;
2 //clc();
3
4 // Example 8.5
5 // Page: 178
6 printf("Example-8.5 Page no.-178\n\n");
7
8 /***Data***/
9 // At equilibrium on dew-point the conditions are
    given as
10 P = 1;//[atm]
11 y_water = 0.60;
12 // From the figure 8.12d, if we start at 130C and 60
    mol% water and cool.
13 // We meet the dew-point line at 99C, and at the
    same temperature the bubble-point curve shows
14 x_water_1 = 0.22;
15 // Doing the same procedure with y_water = 0.90, we
    get the dew-point at the rightmost side at 98C
16 // In this case, the bubble-point line is the

```

```

    steeply sloping one at the right, from which we
    read
17 x_water_2 = 0.99;
18 // Similarly with y_water = 0.73, we get that two
    dew-point meet at 92C.
19 // Vapour of this composition is in equilibrium with
    both liquid phases, as sketched in the figure
    8.12d.
20 // Vapour with any other composition is in
    equilibrium with only one liquid i.e.
21 // if y_water < 0.73, then
22 // x_water < 0.65
23 // and if y_water > 0.73, then
24 // x_water > 0.98
25 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);
26 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

```

1 clear;
2 //clc();
3
4 // Example 8.6
5 // Page: 178
6 printf("Example-8.6 Page no.-178\n\n");
7
8 //***Data***/
9
10 P = 1.00; //[atm] assumed total vapor pressure
11 // In psia unit
12 P1 = 14.7; //[psia]

```

```

13 // 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
14 x_1_water = 0.65;
15 x_1_butanol = (1-x_1_water);
16 x_2_water = 0.98;
17 x_2_butanol = (1-x_2_water);
18 y_water = 0.73;
19 y_butanol = (1-y_water);
20 T = 92; //[C]
21
22 // At this temperature we have to estimate the vapor
    pressure of pure water and n-butanol with the
    help of Antoine equation
23 //  $\log_{10}(p) = A - B/(T+C)$ 
24 // From Table A.2( page 418), the Antoine equation
    constants for water are
25 A_w = 7.94917;
26 B_w = 1657.462;
27 C_w = 227.02;
28
29 // and that for n-butanol are
30 A_b = 7.838;
31 B_b = 1558.190;
32 C_b = 196.881;
33
34 // Thus vapor pressure of water and n-butanol are
    respectively
35 p_water = (14.7/760)*10^(A_w - B_w/(T+C_w));
36 p_butanol = (14.7/760)*10^(A_b - B_b/(T+C_b));
37
38 // fugacity of the water and n-butanol are given as
39 //  $f_i = (y*Y*P)_i$ 
40 // Where Y is the gas phase activity coefficient and
    its value is 1.00 in ideal gas mixture, so
41 f_water = (y_water*P);
42 f_butanol = (y_butanol*P);
43 // The fugacity will be same in both the phase 1 and

```

2

```
44
45 // Now, liquid-phase activity coefficients are given
    by
46 //  $Y_i = (y_i * P) / (x_i * p_i)$ 
47 // so,
48 Y_water_1 = (y_water * P1) / (x_1_water * p_water);
49 Y_butanol_1 = (y_butanol * P1) / (x_1_butanol * p_butanol)
    ;
50
51 // For phase 2
52 Y_water_2 = (y_water * P1) / (x_2_water * p_water);
53 Y_butanol_2 = (y_butanol * P1) / (x_2_butanol * p_butanol)
    ;
54
55 printf(" Four activity coefficients and fufacities
    are shown in the following table:\n\n");
56 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");
57 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);
58 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 distillation

```
1 clear;
2 //clc();
3
4 // Example 8.7
5 // Page: 179
6 printf("Example-8.7 Page no.-179\n\n");
```

```

7
8 //***Data***//
9
10 P = 1; //[atm] Total pressure in the vapor phase
11
12 // Since the two liquids are not soluble in each
    other so Raoult's law will apply separately for
    these two phases.
13 // From Raoult's law we have
14 //  $(y_i * P) = (x_i * p_i)$ 
15 // Here two phases are in pure stages so  $x_i=1$  for
    both phases
16 // So
17 //  $y_i = (p_i / P)$ 
18 // Writing this equation for each species, adding
    the equations, and solving for P, we find
19 //  $P = \text{summation}(y_i * P) = \text{summation}(p_i / P * P) =$ 
    summation( $p_i$ )
20
21 // The total pressure is the sum of the individual
    pure species vapor pressure
22 // To find the boiling point temperature we perform
    a trial and error
23 // Let us assume the boiling point temperature
24 T = 89; //[C]
25 // Antoine equation constants for water is given by
26 A_w = 7.94917;
27 B_w = 1657.462;
28 C_w = 227.02;
29
30 // and that for n-butanol are
31 A_b = 7.838;
32 B_b = 1558.190;
33 C_b = 196.881;
34
35 // Antoine equation is given by
36 //  $\log_{10}(p) = A - B / (T + C)$ 
37 // Thus vapor pressure of water and n-butanol are

```



```

    respectively
38 p_water = (1/760)*10^(A_w - B_w/(T+C_w));
39 p_butanol = (1/760)*10^(A_b - B_b/(T+C_b));
40
41 // Now, vapor phase composition are
42 y_water = p_water/P;
43 y_butanol = p_butanol/P;
44 // summing these, we get
45 y = y_water + y_butanol;
46
47 // Value of y is nearly equal to one so our
    assumption of the temperature is correct
48 // So the boiling point of the mixture is 'T'
49
50 printf(" Boiling point of the two phase system is %0
    .0f deg C\n",T);
51 printf(" In vapor phase, mole fraction of the water
    is %0.2f",y_water);

```

Scilab code Exa 8.8 The little EOS

```

1 clear;
2 //clc();
3
4 // Example 8.8
5 // Page: 184
6 printf("Example -8.8 Page no. -184\n\n");
7
8 //***Data***//
9 Temp = 68;//[F]
10 P = 1;//[atm]
11 // Changing the temperature in 'K' and pressure in '
    bar' we have
12 Temp = 273.15+(Temp-32)*5/9;//[K]
13 P = P*1.01325;//[bar]

```

```

14 // For water from the table A.1(page 417)
15 T_c = 647.1;//[K]
16 P_c = 220.55;//[bar]
17 // Now
18 T_r = Temp/T_c;
19 P_r = P/P_c;
20 w = 0.345;
21 //Now applying the result for the little EOS from
    the example 7.1(page 135 ), we have
22 // f/P = exp(P_r/T_r*f(T_r))
23 // From the chapter 2 of this book, we have
24 f_T_r = (0.083-0.422/T_r^(1.6))+w*(0.139-0.172/T_r
    ^(4.2));
25 // So
26 f_by_P = exp(P_r/T_r*f_T_r);
27 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

```

1 clear;
2 //clc();
3
4 // Example 8.9
5 // Page: 189
6 printf("Example-8.9 Page no.-189\n\n");
7
8 //***Data***//
9
10 // Here we will denote ethanol as species 'a' and
    water as the species 'b'
11 x_a = 0.1238;
12 x_b = (1-x_a);
13 T = 85.3;//[C] Given boiling temperature
14

```

```

15 // We have
16 //  $x_a + x_b = 1$  and  $y_a + y_b = 1$ 
17
18 // The Antoine equation constants for ethanol from
    the table A.2(page 418) given in the book, are
19 A_a = 8.04494;
20 B_a = 1554.3;
21 C_a = 222.65;
22
23 // and that for water
24 A_b = 7.96681;
25 B_b = 1668.21;
26 C_b = 228.0;
27
28 // Thus vapor pressure of ethanol and water are
    respectively
29 p_a = (1/760)*10^(A_a - B_a/(T+C_a));
30 p_b = (1/760)*10^(A_b - B_b/(T+C_b));
31
32 // Also the activity coefficients are given by
33 //  $Y_a = 10^{((B^2)*A*x_b^2)/(A*x_a+B*x_b)^2}$ 
    and
34 //  $Y_b = 10^{((A^2)*B*x_a^2)/(A*x_a+B*x_b)^2}$ 
35 // 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
36 A = 0.7292;
37 B = 0.4104;
38
39 // hence
40 Y_a = 10^{((B^2)*A*x_b^2)/(A*x_a+B*x_b)^2};
41 Y_b = 10^{((A^2)*B*x_a^2)/(A*x_a+B*x_b)^2};
42
43 // Now taking into account of nonideality of the
    gaseous phase, the modified Raoult's law gives
44 //  $(y_a/x_a) = (Y_a*p_a)/P$  and  $(y_b/x_b) = ($ 
    Y_b*p_b)/P
45

```

```

46 // we will take a simple method
47 // solving the above two equation for y_a and y_b
    and adding them, we get
48 P = (Y_a*p_a*x_a)+(Y_b*p_b*x_b);//[atm]
49
50 // So,
51 y_a = (Y_a*p_a*x_a)/P;
52 // and
53 y_b = (Y_b*p_b*x_b)/P;
54
55 printf(" Boiling pressure of the liquid at 85.3 deg
    C is %0.4f atm\n",P);
56 printf(" Mole fraction of ethanaol in vapor phase is
    %0.4f\n",y_a);
57 printf(" Mole fraction of water in the vapor phase
    is %0.4f",y_b);

```

Scilab code Exa 8.10 Pressure specified dew point

```

1 clear;
2 //clc();
3
4 // Example 8.10
5 // Page: 191
6 printf("Example-8.10 Page no.-191\n\n");
7
8 //***Data***/
9
10 // This problem is similar to the example 8.9 except
    that, we are provided pressure instead of
    temperature and different liquid composition
11 // Here again, we will denote ethanol as species 'a'
    and water as the species 'b'
12 x_a = 0.2608;
13 x_b = (1-x_a);

```

```

14 P = 1.00;//[atm] Given boiling pressure
15
16 // We have
17 // x_a + x_b = 1    and    y_a + y_b = 1
18
19 // The Antoine equation constants for ethanol from
    the table A.2(page 418) given in the book, are
20 A_a = 8.04494;
21 B_a = 1554.3;
22 C_a = 222.65;
23
24 // and that for water
25 A_b = 7.96681;
26 B_b = 1668.21;
27 C_b = 228.0;
28
29 // Thus vapor pressure of ethanol and water are
    respectively
30 // p_a = (1/760)*10^(A_a - B_a/(T+C_a))
31 // p_b = (1/760)*10^(A_b - B_b/(T+C_b))
32 // Adding these two equation, we get
33 // ( 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)
34
35 // Also the activity coefficients are given by
36 // Y_a = 10^((B^(2)*A*x_b^(2))/(A*x_a+B*x_b)^(2))
    and
37 // Y_b = 10^((A^(2)*B*x_b^(2))/(A*x_a+B*x_b)^(2))
38 // 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
39 A = 0.7292;
40 B = 0.4104;
41
42 // hence
43 Y_a = 10^((B^(2)*A*x_b^(2))/(A*x_a+B*x_b)^(2));
44 Y_b = 10^((A^(2)*B*x_a^(2))/(A*x_a+B*x_b)^(2));

```

```

45
46 // Now we will solve for T running the loop
47 // Let us assume the startup temperature
48 T = 80;
49 err = 1;
50
51 while err > 10(-3)
52     P_a = (10(8.04494 - 1554.3/(222.65 + T)))/760;
53     P_b = (10(7.96681 - 1668.21/(228 + T)))/760;
54     y_a = Y_a*P_a*x_a/P;
55     y_b = Y_b*P_b*x_b/P;
56     err = abs((y_a + y_b) - 1);
57     T = T + 0.01;
58 end
59
60 printf(" Boiling temperature of the liquid at 1 atm
        pressure is %0.4f atm\n",T);
61 printf(" Mole fraction of ethanaol in vapor phase is
        \t%0.4f\n",y_a);
62 printf(" Mole fraction of water in the vapor phase
        is \t%0.4f",y_b);

```

Scilab code Exa 8.11 Temperature specified bubble point

```

1 clear;
2 //clc();
3
4 // Example 8.11
5 // Page: 192
6 printf("Example-8.11 Page no.-192\n\n");
7
8 //***Data***/
9
10 // This problem is identical to that of the example
    8.9 except difference in the boiling temperature

```

```

    and liquid composition
11 // Here we will again denote ethanol as species 'a'
    and water as the species 'b'
12 y_a = 0.6122;
13 y_b = (1-y_a);
14 T = 80.7; //[C] Given boiling temperature
15
16 // We have
17 // x_a + x_b = 1    and    y_a + y_b = 1
18
19 // The Antoine equation constants for ethanol from
    the table A.2(page 418) given in the book, are
20 A_a = 8.04494;
21 B_a = 1554.3;
22 C_a = 222.65;
23
24 // and that for water
25 A_b = 7.96681;
26 B_b = 1668.21;
27 C_b = 228.0;
28
29 // Thus vapor pressure of ethanol and water are
    respectively
30 p_a = (1/760)*10^(A_a - B_a/(T+C_a));
31 p_b = (1/760)*10^(A_b - B_b/(T+C_b));
32
33 // Also the activity coefficients are given by
34 //  $Y_a = 10^{((B^2)*A*x_b^2)/(A*x_a+B*x_b)^2}$ 
    and
35 //  $Y_b = 10^{((A^2)*B*x_b^2)/(A*x_a+B*x_b)^2}$ 
36 // 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
37 A = 0.7292;
38 B = 0.4104;
39
40 // Now taking into account of nonideality of the
    gaseous phase, the modified Raoult's law gives

```

```

41 // (y_a/x_a) = (Y_a*p_a)/P    and    (y_b/x_b) = (
    Y_b*p_b)/P
42
43 // Now can take the help of trial and error method
    to solve the above equations
44 // 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
45 x_a = 0.6122; // Initial assumption of liquid phase
    composition of ethanol
46 x_b = 0.3; // Initial assumption of liquid phase
    composition water
47 P = 0.80; // [atm]
48 err = 1;
49
50 while err > 2* 10(-2)
51     P = P + 0.01;
52     Y_a = 10((B(2)*A*x_b(2)))/(A*x_a+B*x_b)(2));
53     Y_b = 10((A(2)*B*x_a(2)))/(A*x_a+B*x_b)(2));
54
55     err = abs((x_a + x_b) - 1);
56     x_a = y_a*P/(Y_a*p_a);
57     x_b = y_b*P/(Y_b*p_b);
58 end
59
60
61
62 printf(" Boiling pressure of the liquid at 80.7 deg
    C is %0.4f atm\n",P);
63 printf(" Mole fraction of ethanaol in liquid phase
    is %0.4f\n",x_a);
64 printf(" Mole fraction of water in the liquid phase
    is %0.4f",x_b);

```

Scilab code Exa 8.12 Pressure specified bubble point

```
1 clear;
2 //clc();
3
4 // Example 8.12
5 // Page: 193
6 printf("Example-8.12 Page no.-193\n\n");
7
8 /***Data***/
9
10 // This problem is similar to the example 8.10
    // except that, we are provided vapour phase
    // composition
11 // Here again, we will denote ethanol as species 'a'
    // and water as the species 'b'
12 y_a = 0.1700;
13 y_b = (1-y_a);
14 P = 1.00; //[atm] Given boiling pressure
15
16 // We have
17 // x_a + x_b = 1 and y_a + y_b = 1
18
19 // The Antoine equation constants for ethanol from
    // the table A.2(page 418) given in the book, are
20 A_a = 8.04494;
21 B_a = 1554.3;
22 C_a = 222.65;
23
24 // and that for water
25 A_b = 7.96681;
26 B_b = 1668.21;
27 C_b = 228.0;
28
29 // Thus vapor pressure of ethanol and water are
    // respectively
30 // p_a = (1/760)*10^(A_a - B_a/(T+C_a))
31 // p_b = (1/760)*10^(A_b - B_b/(T+C_b))
```

```

32
33 // Also the activity coefficients are given by
34 //  $Y_a = 10^{((B^2)*A*x_b^2)/(A*x_a+B*x_b)^2}$ 
      and
35 //  $Y_b = 10^{((A^2)*B*x_a^2)/(A*x_a+B*x_b)^2}$ 
36 // 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
37 A = 0.7292;
38 B = 0.4104;
39
40 // Now taking into account of nonideality of the
      gaseous phase, the modified Raoult's law gives
41 //  $(y_a/x_a) = (Y_a*p_a)/P$  and  $(y_b/x_b) = ($ 
       $Y_b*p_b)/P$ 
42
43 // Now we can take the help of trial and error
      method to solve the above equations
44 // 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
45
46 x_a = 0.0100; // Initial assumption of liquid phase
      composition of ethanol
47 x_b = 0.9; // Initial assumption of liquid phase
      composition water
48 T = 80; // [C] Initial guess of the temperature
49 err = 1;
50
51 while err > 1/16*10^(-2)
52     P_a = (10^(8.04494 - 1554.3/(222.65 + T)))/760;
53     P_b = (10^(7.96681 - 1668.21/(228 + T)))/760;
54
55     Y_a = 10^((B^2)*A*x_b^2)/(A*x_a+B*x_b)^2);
56     Y_b = 10^((A^2)*B*x_a^2)/(A*x_a+B*x_b)^2);
57
58     x_a = y_a*P/(Y_a*P_a);

```

```

59     x_b = y_b*P/(Y_b*P_b);
60
61     err = abs((x_a + x_b) - 1);
62     T = T + 0.01;
63
64 end
65
66
67 printf(" Equilibrium Temperature of the system at
        pressure 1 atm  is %0.4f atm\n",T);
68 printf(" Mole fraction of ethanaol in liquid phase
        is      %0.4f\n",x_a);
69 printf(" Mole fraction of water in the liquid phase
        is      %0.4f",x_b);

```

Scilab code Exa 8.13 Isothermal flashes

```

1  clear;
2  //clc();
3
4  // Example 8.13
5  // Page: 194
6  printf("Example-8.13  Page no.-194\n\n");
7
8  //***Data***/
9
10 // Here again, we will denote ethanol as species 'a'
    and water as the species 'b'
11 x_aF = 0.126;
12 x_bF = (1-x_aF);
13 P = 1.00;//[atm] Given total pressure
14 T = 91.8;//[C]
15
16 // We will go with graphical approach for solving
    this problem

```

```

17 // This problem requires T - x_a diagram at the
    // given pressure i.e. 1 atm
18 // This diagram is provided on page 196( figure
    // 8.19) in the book
19 // We will draw horizontal and vertical lines
    // corresponding to the specified T and x_a.
20 // 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
21 // 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.
22 x_a = 0.0401;
23 x_b = (1 - x_a);
24
25 // Similarly 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.
26 y_a = 0.2859;
27 y_b = (1 - y_a);
28
29 // Now vapour fraction is given by
30 V_by_F = ( x_aF - x_a )/(y_a - x_a);
31
32 printf(" Mole fraction of the ethanol in the liquid
    // phase in equilibrium at the given condition is %f
    // \n",x_a);
33 printf(" Mole fraction of the water in the liquid
    // phase in equilibrium at the given condition is %f
    // \n",x_b);
34 printf(" Mole fraction of the ethanol in the vapour
    // phase in equilibrium at the given condition is %f
    // \n",y_a);

```

```

35 printf(" Mole fraction of the water in the vapour
    phase in equilibrium at the given condition is %f
    \n",y_b);
36 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

```

1 clear;
2 //clc();
3
4 // Example 8.14
5 // Page: 198
6 printf("Example-8.14 Page no.-198\n\n");
7
8 //***Data***/
9
10 P = 100;//[psia]
11 // Composition in liquid phase is
12 x_a = 0.05; // Mole fraction of methane
13 x_b = 0.40; // Mole fraction of butane
14 x_c = 0.55; // mole fraction of pentane
15
16 // We have to take the help of the following
    equations
17 // ( x_a + x_b + x_c ) = 1          and    ( y_a + y_b +
    y_c ) = 1
18 // ( y_a/x_a ) = K_a;    ( y_b/x_b ) = K_b;
    and    ( y_c/x_c ) = K_c;
19
20 // 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

```

```

21 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];
22 printf(" Calculations for the various assumed
      temperatures are given in the table below\n\n");
23 printf(" Temperature \t\t y_a \t\t y_b \t\t\t
      y_c \t\t\t y \n\n");
24
25 T_b = 0;//[F] Bubble point
26 j=1;
27 for i = 1:7
28     y_a = x_a*T(i,j);
29     y_b = x_b*T(i,j+1);
30     y_c = x_c*T(i,j+2);
31     y = y_a + y_b + y_c;
32     T_b = T_b + 5;
33     printf(" %f \t\t %f \t\t %f \t\t %f \t\t %f\
      n ",T_b,y_a,y_b,y_c,y);
34     end
35 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");
36 printf(" And compositions in the vapor phase are the
      values given in the above table corresponding to
      the temperature 30 deg F, i.e.\n\n");
37 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

```

1 clear;
2 //clc();
3
4 // Example 8.15

```

```

5 // Page: 199
6 printf("Example-8.15 Page no.-199\n\n");
7
8 /***Data***/
9
10 n_sugar = 1;//[mol]
11 n_water = 1000/18;//[mol]
12 x_sugar = n_sugar/(n_sugar+n_water);
13 x_water = n_water/(n_sugar+n_water);
14 // At 100C we have
15 p_water = 1;//[atm]
16 p_sugar = 0;//[atm]
17 // and the relation
18 P = x_water*p_water+x_sugar*p_sugar;//[atm]
19 // The situation is sketched in the figure 8.21 in
    the book[page 199].
20 // Now for the second part of the question
21 // 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.
22 P_1 = 1;//[atm]
23 p_water = P_1/x_water;//[atm]
24 // Interpolating in the steam table[12] reported in
    the book, we find
25 T = 100.51;//[C]
26 // We may restate this that the boiling-point
    elevation caused by this dissolved , nonvolatile
    solute is
27 T_eb = T-100;//[C]
28 printf("Vapour pressure of this solution at the 100C
    is %f atm\n",P);
29 printf(" The temperature at which this solution will
    boil at 1 atm is %f C",T);

```

Scilab code Exa 8.16 Freezing point depression

```
1 clear;
2 //clc();
3
4 // Example 8.16
5 // Page: 201
6 printf("Example - 8.16 Page no. - 201\n\n");
7
8 /***Data***/
9 n_sugar = 1; // [mol]
10 n_water = 1000/18; // [mol]
11 x_sugar = n_sugar/(n_sugar+n_water);
12 x_water = n_water/(n_sugar+n_water);
13
14 // 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.
15 // Crystalline solid formed from such a solution is
    // nearly pure H2O, with no dissolved sugar.
16 // At the low pressures involved here, these
    // fugacities are practically equal to partial
    // pressures, so that
17 //  $P = x_{\text{water}} \cdot p_{\text{water}} + x_{\text{sugar}} \cdot p_{\text{sugar}} = p_{\text{ice}}$ ;
18
19 // but
20 p_sugar = 0;
21 // so
22 p_ice_by_p_water = x_water;
23
24 // Figure 5.8 reported in the book (page 100) shows
    // the vapour pressure of subcooled water and of
    // ice.
```



```

25 //The values in the given table from which that
    figure were made can be represented by the
    following totally empirical data-fitting equation
    .
26 // p_ice/p_water = 1+0.0096686*T+4.0176*10^(-5)*T
    ^ (2)
27 // We eliminate p_ice/p_water by x_water
28
29 deff(' [y]=f(T) ', 'y = 1+0.0096686*T+4.0176*10^(-5)*T
    ^ (2)-p_ice_by_p_water ');
30 T = fsolve(0,f); // [C]
31
32 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

```
1 clear;
2 //clc();
3
4 // Example 9.1
5 // Page: 219
6 printf("Example-9.1 Page no.-219\n\n");
7
8 /***Data***/
9 x_isopropanol = 0.4720;
10 x_water = 0.5280;
11 // From the table A.7 (page 427) reported in the
    book the Van Laar coefficients for isopropanol-
    water system at 1atm are given by
12 A = 1.0728;
13 B = 0.4750;
14 // Van Laar equations are given
15 //  $\log_{10}(Y_a) = A \cdot x_b^2 / (A/B \cdot x_a + x_b)^2$ 
16 //  $\log_{10}(Y_b) = B \cdot x_a^2 / (B/A \cdot x_b + x_a)^2$ 
17 // We calculate Y_isopropanol and Y_water as
```

```

18 Y_isopropanol = 10^(A*x_water^(2)/(A/B*x_isopropanol
    +x_water)^(2));
19 Y_water = 10^(B*x_isopropanol^(2)/(B/A*x_water+
    x_isopropanol)^(2));
20 printf(" Value of the liquid-phase activity
    coefficient for isopropanol is %f\n\n",
    Y_isopropanol);
21 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

```

1 clear;
2 //clc();
3
4 // Example 9.2
5 // Page: 221
6 printf("Example-9.2 Page no.-221\n\n");
7
8 //***Data***/
9 // Recieving the VLE data from the example 8.2, we
    have
10 x_acetone = 0.05;
11 x_water = 0.95;
12 // And the activity coefficient is given by
13 y_acetone = 7.04;
14 y_water = 1.01;
15 // we hve the relation  $g_E/RT = \text{summation}(x_i \cdot \log(y_i))$ 
16 // let  $C = g_E/RT$  , so
17 C = (x_acetone*log(y_acetone)+x_water*log(y_water));
18 // Now let  $M = (g_E/RT)/(x_acetone \cdot x_water)$ 
19 // So
20 M = C/(x_acetone*x_water);

```

```

21 printf("The value of g_E/RT for acetone-water
        solution at 1 atm pressure is           %f\n\n"
        ,C);
22 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

```

1 clear;
2 //clc();
3
4 // Example 9.3
5 // Page: 221
6 printf("Example-9.3 Page no.-221\n\n");
7
8 //***Data***/
9
10 printf("This is a theoretical 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

```

1 clear;
2 //clc();
3
4 // Example 9.4
5 // Page: 224
6 printf("Example-9.4 Page no.-224\n\n");
7
8 //***Data***/
9

```

```
10 printf("This is a theoretical 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

```
1 clear;
2 //clc();
3
4 // Example 9.5
5 // Page: 224
6 printf("Example -9.5   Page no. -224\n\n");
7
8 /***Data***/
9 y_acetone_infinity = 10;
10 y_water_infinty = 5
11 Pressure = 1;//[atm]
12 // From equation 9.L and 9.M (page 224) as reported
    in the book, we have
13 // Constants in morgules equation b and c as
14 b = log(y_acetone_infinity);
15 c = log(y_water_infinty);
16 printf("Values of the constants in Morgules equation
    for acetone-water at 1 atm are b = %f\n",b);
17 printf("
    and c = %f",c);
```

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

```
1 clear;
2 //clc();
```

```

3
4 // Example 9.6
5 // Page: 225
6 printf("Example-9.6 Page no.-225\n\n");
7
8 /***Data***/
9 P_1 = 10; //[atm]
10 x_a_1 = 0.1238; // mole fraction of ethanol at 10
    atm pressure
11 Temp = 273.15+85.3; //[K]
12 R = 0.08206; //[(L*atm)/(mol*K)]
13 P_0 = 1; //[atm]
14 // so
15 delta_P = (P_1-P_0); //[atm]
16 // Molecular weight of ethanol and water are
    respectively
17 M_ethanol = 46; //[g/mol]
18 M_water = 18; //[g/mol]
19 // Now changing the mol fraction of ethanol in the
    wt fraction
20 m_a_1 = x_a_1*M_ethanol/(x_a_1*M_ethanol+(1-x_a_1)*
    M_water);
21 // From example 8.9(page 188) we know that at this T
    and 1 atm and x_a_0, activity coefficient for
    ethanol
22 y_ethanol_0 = 2.9235;
23 // Now from figure 6.15(page 129), we read that at
    20C and m_a_1 mass fraction ethanol ,
24 v_ethanol_1 = 1.16; //[cm^(3)/g]
25 // Similarly for mass fraction corresponding to
    mole fraction x_a_1
26 v_ethanol_0 = 1.27; //[cm^(3)/g]
27 // Difference of these two values is
28 v = v_ethanol_1-v_ethanol_0; //[cm^(3)/g]
29 v = v*46; //[L/g]
30 // 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

```

```

31 // From equation 7.31(page 225)
32 //  $d(\log(y_i))/dP = (v_1-v_0)/(R*T)$ ; at constant
    temperature and mole fraction
33 // Let  $d(\log(y_i))/dP = C$ , then
34 C = (v_ethanol_1-v_ethanol_0)/(R*Temp);
35 // Also we can have
36 //  $\Delta \log(y_i) = (d(\log(y_i))/dP)*\Delta P$ 
37 // or
38 //  $\Delta \log(y_i) = C*\Delta P$ 
39 // and  $\Delta \log(y_i) = \log(y_{ethanol_1})-\log($ 
    y_ethanol_0)
40 // So
41 y_ethanol_1 = exp(log(y_ethanol_0)+C*delta_P);
42 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 coefficient

```

1 clear;
2 //clc();
3
4 // Example 9.7
5 // Page: 226
6 printf("Example-9.7 Page no.-226\n\n");
7
8 //***Data***/
9 x_ethanol = 0.1238;
10 Temp_1 = 273.15+85.3; // [K]
11 P = 1; // [atm]
12 Temp_2 = 273.15+70; // [K]
13 R = 8.314; // [j/(mol*K)]
14 // From example 8.9, at temperature 85.3C the
    activity coefficient is
15 y_ethanol_1 = 2.9235;

```

```

16 // 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.
17 // which are respectively
18 delta_h_2 = 0.2; //[kJ/mol]
19 delta_h_1 = 1.0; //[kJ/mol]
20 // Taking the average of these two values we have
21 delta_h_average = (delta_h_1+delta_h_2)/2*1000; //[J/
    // mol]
22 // From the equation 7.32 (page 225) reported in the
    // book
23 //  $d(\log(y_i))/dT = (h_i\_average - h_i\_0)/(R*T^2)$ ;
    // at constant pressure and mole fraction
24 // So
25 //  $\int d(\log(y_i)) = \int ((h_i\_average - h_i\_0)/(R*T^2)) * dT$ 
26 // it can be taken approximately as
27 //  $\int d(\log(y_i)) = ((h_i\_average - h_i\_0) / R) * \int 1/T^2 * dT$ 
28 // we have  $\int d(\log(y_i)) = \log(y\_ethanol\_2 / y\_ethanol\_1)$ 
29 // So
30 y_ethanol_2 = y_ethanol_1 * exp((delta_h_average/R) *
    //  $\int 1/T^2$ , 'T', Temp_1, Temp_2));
31 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

```

1 clear;
2 //clc();
3
4 // Example 9.8
5 // Page: 229

```



```

6 printf("Example-9.8 Page no.-229\n\n");
7
8 /***Data***/
9 // In this solution we will give the identity to the
   three species as
10 // a- Acetone
11 // b- Methanol
12 // c- Water
13 // Given
14 x_a = 0.1200;
15 x_b = 0.1280;
16 x_c = 0.7520;
17 Temp = 66.70; // [C]
18 P = 1; // [atm] pressure
19 // As reported in the book that from [5] we get the
   following values
20 // acetone-methanol(a-b)
21 A_ab = 0.2634;
22 A_ba = 0.2798;
23 // acetone-water(a-c)
24 A_ac = 0.9709;
25 A_ca = 0.5579;
26 // methanol-water(b-c)
27 A_bc = 0.3794;
28 A_cb = 0.2211;
29 // Now consider the equation 9.10 (page 228)
30 // The first term on the right of the equation is
31 T_1 = x_b^(2)*(A_ab+2*x_a*(A_ba-A_ab));
32 // similarly the second and third terms are given
   respectively as
33 T_2 = x_c^(2)*(A_ac+2*x_a*(A_ca-A_ac));
34 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);
35 // thus whole term on the right hand side is
36 T = T_1+T_2+T_3;
37 // So
38 y_a = 10^(T);

```

```

39 // for this temperature vapour pressure of the
    acetone is calculated as
40 p_acetone = 1.417; //[atm]
41 // So that we estimate
42 y_acetone = x_a*y_a*p_acetone;
43 printf("The activity coefficient of acetone in the
    given mixture is %f",y_a);
44 // The experimental value is y_acetone = 0.698

```

Scilab code Exa 9.9 Application of mixing rule

```

1 clear;
2 //clc();
3
4 // Example 9.9
5 // Page: 234
6 printf("Example -9.9 Page no. -234\n\n");
7
8 //***Data***//
9
10 T = 85.3+273.15; //[K] Temperature
11 P = 1; //[atm] Pressure of the mixture
12 R = 8.314; // [(Pa*m(3))/(K*mol)]
13 R_1 = 0.08206; // [(L*atm)/(mol*K)]
14 y_i = 0.1238; // mole fraction of the ethanol in the
    vapor phase
15 y_j = (1-y_i); // mole fraction of the water vapor in
    the vapor phase
16
17 // From the table A.1( table 417), the properties of
    water and ethanol are given as
18 // Critical temperatures are
19 T_c_ii = 513.9; //[K] Critical temperature of the
    ethanol
20 T_c_jj = 647.1; //[K] Criatical temperature of water

```

```

21
22 // Critical pressure are
23 P_c_ii = 61.48; //[bar] Critical pressure of ethanol
24 P_c_jj = 220.55; //[bar] Critical pressure of water
25
26 // Accentric factor
27 w_ii = 0.645; // accentric factor of the ethanol
28 w_jj = 0.345; // accentric factor of the water
29
30 // Compressibility factor are
31 z_c_ii = 0.24; // compressibility factor of ethanol
32 z_c_jj = 0.229; // compressibility factor of the
    water
33
34 // Critical volume are given by
35 V_c_ii = z_c_ii*R*T_c_ii/(P_c_ii*100000)*10^(6); //
    critical volume the ethanol
36 V_c_jj = z_c_jj*R*T_c_jj/(P_c_jj*100000)*10^(6); //
    critical volume the ethanol
37
38 // Now
39 // for k_ij = 0.0
40 T_c_ij_0 = (T_c_ii*T_c_jj)^(1/2); // [K]
41 w_ij = (w_ii + w_jj)/2;
42 z_c_ij = (z_c_ii + z_c_jj)/2;
43 V_c_ij = ( (V_c_ii^(1/3) + V_c_jj^(1/3))/2 )^(3);
44 P_c_ij_0 = (z_c_ij*R*T_c_ij_0)/(V_c_ij/10^(6))
    /10^(5); //[bar]
45
46 // again for k_ij = 0.01
47 T_c_ij_1 = (T_c_ii*T_c_jj)^(1/2)*(1-0.01); // [K]
48 P_c_ij_1 = (z_c_ij*R*T_c_ij_1)/(V_c_ij/10^(6))
    /10^(5); //[bar]
49
50 // Now
51 T_r_ii = T/T_c_ii;
52 T_r_jj = T/T_c_jj;
53 T_r_ij_0 = T/T_c_ij_0;

```

```

54 T_r_ij_1 = T/T_c_ij_1;
55
56 // and
57 P_r_ii = P/P_c_ii;
58 P_r_jj = P/P_c_jj;
59 P_r_ij_0 = P/P_c_ij_0;
60 P_r_ij_1 = P/P_c_ij_1;
61
62 // Now we will calculate f(T_r) for each component
    and mixture
63 f_Tr_ii = ( 0.083 - 0.422/T_r_ii^(1.6) ) + w_ii*(
        0.139 - 0.172/T_r_ii^(4.2));
64 f_Tr_jj = ( 0.083 - 0.422/T_r_jj^(1.6) ) + w_jj*(
        0.139 - 0.172/T_r_jj^(4.2));
65 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));
66 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));
67
68 // Let us define A = (P_r*f(T_r)/T_r) , so
69 A_ii = P_r_ii*f_Tr_ii/T_r_ii;
70 A_jj = P_r_jj*f_Tr_jj/T_r_jj;
71
72 // We are given
73 v_ii = 0.975;
74 v_jj = 0.986;
75
76 // Now,
77 B_ii = ( f_Tr_ii*R*T_c_ii/P_c_ii)*(10^(3)/10^(5)); //
        [L/mol]
78 B_jj = ( f_Tr_jj*R*T_c_jj/P_c_jj)*(10^(3)/10^(5)); //
        [L/mol]
79 B_ij0 = ( f_Tr_ij0*R*T_c_ij_0/P_c_ij_0)*(10^(3)
        /10^(5)); // [L/mol]
80 B_ij1 = ( f_Tr_ij1*R*T_c_ij_1/P_c_ij_1)*(10^(3)
        /10^(5)); // [L/mol]
81
82 // now we will calculate 'delta '

```

```

83 delta_ij0 = 2*B_ij0 - B_ii - B_jj; // [L/mol]
84 delta_ij1 = 2*B_ij1 - B_ii - B_jj; // [L/mol]
85
86 // We have
87 //  $b_a = B_{aa} + y_b^2 \cdot \delta$  and  $b_b = B_{bb} + y_a^2 \cdot \delta$ 
88 // so,
89 b_ethanol0 = B_ii + y_j^2 * delta_ij0; // [L/mol]
90 b_water0 = B_jj + y_i^2 * delta_ij0; // [L/mol]
91 b_ethanol1 = B_ii + y_j^2 * delta_ij1; // [L/mol]
92 b_water1 = B_jj + y_i^2 * delta_ij1; // [L/mol]
93
94 // Now
95 //  $\phi_i = \exp(b_i \cdot P / (R \cdot T))$ 
96 // So,
97 phi_ethanol0 = exp((b_ethanol0 * P) / (R_1 * T));
98 phi_water0 = exp((b_water0 * P) / (R_1 * T));
99 phi_ethanol1 = exp((b_ethanol1 * P) / (R_1 * T));
100 phi_water1 = exp((b_water1 * P) / (R_1 * T));
101
102 // and
103 //  $Y_i = \phi_i / v_i$ 
104 // So,
105 Y_ethanol0 = phi_ethanol0 / v_ii;
106 Y_water0 = phi_water0 / v_jj;
107 Y_ethanol1 = phi_ethanol1 / v_ii;
108 Y_water1 = phi_water1 / v_jj;
109
110 printf(" The results are summarize in the following
table\n\n");
111 printf(" Property \t\t\t Mix, ij , Assuming k_ij =
0.0 \t\t\t\t Mix, ij , Assuming k_ij = 0.01\n");
112 printf(" phi_ethanol \t\t\t\t %f \t\t\t\t\t %f\n ",
phi_ethanol0, phi_ethanol1);
113 printf(" phi_water \t\t\t\t\t %f \t\t\t\t\t %f \n",
phi_water0, phi_water1);
114 printf(" Y_ethanol \t\t\t\t\t %f \t\t\t\t\t %f \n",
Y_ethanol0, Y_ethanol1);

```

```

115 printf("  Y_water    \t\t\t\t %f \t\t\t\t\t %f \n\n",
        Y_water0,Y_water1);
116 printf(" Value of ''v'' for ethanol is %f\n",v_ii);
117 printf(" Value of ''v'' water is %f",v_jj);

```

Scilab code Exa 9.10 Solubility parameter

```

1 clear;
2 //clc();
3
4 // Example 9.10
5 // Page: 239
6 printf("Example-9.10 Page no.-239\n\n");
7
8 /***Data***/
9
10 T = 65+273.15; //[K] Temperature
11 R = 8.314; //[(m^(3)*Pa)/(mol*K)] Universal gas
    constant
12 // 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
13 v_hex = 131.6; //[ml/mol] Molar volume of n-Hexane
14 v_dketone = 106.4; //[ml/mol] Molar volume of
    diethylketone
15 s_hex = 14.9; //[MPa^(0.5)] Solubility of n-Hexane
16 s_dketone = 18.1; //[MPa^(0.5)] Solubility of
    diethylketone
17
18 // Here we will use these values with the assumption
    that  $Y_{i,65C} = Y_{i,25C}$ 
19 // At infinite dilution, the volume fraction of the
    other species is 1.00, so,
20 //  $\log Y_a = v_a \phi_b^2 (\delta_a - \delta_b)^2 / (R$ 
    *T)

```

```

21 // so, for n-Hexane
22 Y_hex = exp(v_hex*1^(2)*(s_hex - s_dketone)^(2)/(R*T
    ));
23
24 // And that for diethylketone
25 Y_dketone = exp(v_dketone*1^(2)*( s_dketone - s_hex
    )^(2)/(R*T));
26 printf(" The infinite dilution activity coefficient
    of n-Hexane is %f\n",Y_hex);
27 printf(" The infinite dilution activity coefficient
    of diethylketone is %f",Y_dketone);

```

Scilab code Exa 9.11 Henrys Law Constant estimation

```

1 clear
2 clear;
3 //clc ();
4
5 // Example 9.11
6 // Page: 243
7 printf(" Example -9.11 Page no.-243\n\n");
8
9 /***Data***/
10
11 P = 1; //[atm]
12 T = 25; //[C]
13 y_i = 1.00; // amount of the oxygen in the vapour
14 // Using the constants for O2 in table A.2
15 A = 6.69147;
16 B = 319.0117;
17 C = 266.7;
18 // By Antoine equation
19 // log10(P_i) = A-B/(T+C)
20 P_i = 10^(A-B/(T+C)); // [mmHg]
21 P_i = P_i/760; // [atm]

```

```

22 // This is extrapolated vapour pressure of O2 at 25C
23 // We will take this value as equal to the Henry's
    law constant
24 H_i = P_i;
25 x_i = y_i*P/H_i;
26 printf(" Henry ' 's law constant for O2 is %f atm\n",
    P_i);
27 printf(" solubility of O2 is                %e",x_i);

```

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

```

1  clear;
2  //clc();
3
4  // Example 9.12
5  // Page: 244
6  printf("Example-9.12  Page no.-244\n\n");
7
8  /***Data***/
9  y_a = 1.00;
10 P = 1.00;//[atm]
11 x_a = 0.231*10(-4);
12 // Using the constants for O2 in table A.2 in the
    Antoine equation , we find the vapour pressure of
    the oxygen at 25C viz.
13 p_a = 521.15;//[atm]
14 // Thus activity coefficient is calculated by
    rewriting the equation 8.6 and using the above
    values
15 Y_O2 = (y_a*P)/(x_a*p_a);
16 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

```
1 clear;
2 // clc ();
3
4 // Example 10.1
5 // Page: 260
6 printf("Example-10.1 Page no.-260\n\n");
7
8 //***Data***//
9
10 P = 100;//[psia] Bubble point pressure
11 x_ethane = 0.10; // Mole fraction of ethane in liquid
    phase
12 x_heptane = (1-x_ethane);
13
14 // a) From figure 10.7( page 260 ) given in the book
15 // We read the chart to get the bubble-point
    temperature
16 // The dew point curve for 100 psia crosses the 10
    mol% ethane line at about temperature
```

```

17 T1 = 165;//[C]
18 // Now, we horizontally from that intersection point
    to the dew-point curve, finding the vapor phase
    composition of ethane
19 y1_e = 0.92;
20 y1_h = (1- y1_e);
21
22 // b) By Raoult's law, we use a trial and error
    procedure on the temperature
23 // Antoine equation constants for ethanol are given
24 A_e = 6.80267;
25 B_e = 656.4028;
26 C_e = 255.99;
27
28 // and that for n-heptane are
29 A_h = 6.9024;
30 B_h = 1268.115;
31 C_h = 216.9;
32
33 // Antoine equation is given by
34 // (log10p) = (A - B/(T+C))
35 T = 50;//[C]
36 err = 1;
37
38 while err > 10(-4)
39     p1_e = (10(A_e - B_e/(C_e + T)))*(14.7/760);
40     p1_h = (10(A_h - B_h/(C_h + T)))*(14.7/760);
41     y2_e = p1_e*x_ethane/P;
42     y2_h = p1_h*x_heptane/P;
43     err = abs((y2_e + y2_h) - 1);
44     T = T + 0.0001;
45 end
46
47 // Changing the temperature in deg F
48 T2 = T*9/5 + 32;//[F] Bubble-point temperature
49
50 // c) In this method, we use L-R rule, instead of
    simple Raoult's law

```

```

51 // So,
52 //  $y_i = (x_i * p_i) / (v_i * P)$ 
53 // Where calculated values of  $v_i$  from EOS are given
    in the table 10.A and are
54  $v_e = 0.950$ ; // For ethane
55  $v_h = 0.459$ ; // For n-heptane
56
57 // We again use trial and error on the temperature
58 // Let us assume the initial temperature
59  $T_i = 50$ ; // [C]
60  $err = 1$ ;
61
62 while  $err > 10^{-4}$ 
63      $p2_e = (10^{(A_e - B_e / (C_e + T_i))}) * (14.7 / 760)$ ;
64      $p2_h = (10^{(A_h - B_h / (C_h + T_i))}) * (14.7 / 760)$ ;
65      $y3_e = p2_e * x_{ethane} / (P * v_e)$ ;
66      $y3_h = p2_h * x_{heptane} / (P * v_h)$ ;
67      $err = abs((y3_e + y3_h) - 1)$ ;
68      $T_i = T_i + 0.0001$ ;
69 end
70
71 // Changing the temperature in deg F
72  $T3 = T_i * 9 / 5 + 32$ ; // [F] Bubble-point temperature
73
74 printf(" The results are summarized in the following
    table:\n\n");
75 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");
76 printf(" \t T(deg F) \t\t\t\t %f \t\t\t\t %f \t\t\t\t
    \t %f\n", T1, T2, T3);
77 printf(" \t  $y_{ethane}$  \t\t\t\t %f \t\t\t\t %f \t\t\t\t
    \t %f\n", y1_e, y2_e, y3_e);
78 printf(" \t  $y_{heptane}$  \t\t\t\t %f \t\t\t\t %f \t\t\t\t
    \t\t\t %f\n\n", y1_h, y2_h, y3_h);
79 printf(" Where T is boiling point temperature");

```

Scilab code Exa 10.2 Bubble point temperature estimation

```
1 clear;
2 //clc();
3
4 // Example 10.2
5 // Page: 262
6 printf("Example-10.2 Page no.-262\n\n");
7 printf(" Please wait for the result\n\n");
8
9 /***Data***/
10
11 P = 800;//[psia] Bubble point pressure
12 x_ethane = 0.60; // Mole fraction of ethane in liquid
    phase
13 x_heptane = (1-x_ethane);
14
15 // a) From figure 10.7( page 260 ) given in the book
16 // We read the chart to get the bubble-point
    temperature
17 // The dew point curve for 800 psia crosses the 60
    mol% ethane line at about temperature
18 // T1 = 165
19 // Now, we horizontally from that intersection point
    to the dew-point curve, finding the vapor phase
    composition of ethane
20 // y1_e = 0.95
21 // But, by linear interpolation in the experimental
    data on which Figure 10.7 is based we make a
    slightly more reliable estimate and get
22 T1 = 209;//[F]
23 y1_e = 0.945;
24 y1_h = (1- y1_e);
25
```

```

26 // b) By Raoult's law, we use a trial and error
    procedure on the temperature
27 // Antoine equation constants for ethanol are given
28 A_e = 6.80267;
29 B_e = 656.4028;
30 C_e = 255.99;
31
32 // and that for n-heptane are
33 A_h = 6.9024;
34 B_h = 1268.115;
35 C_h = 216.9;
36
37 // Antoine equation is given by
38 // (log10p) = (A - B/(T+C))
39 T = 50; // [C]
40 err = 1;
41
42 while err > 10(-4)
43     p1_e = (10(A_e - B_e/(C_e + T)))*(14.7/760);
44     p1_h = (10(A_h - B_h/(C_h + T)))*(14.7/760);
45     y2_e = p1_e*x_ethane/P;
46     y2_h = p1_h*x_heptane/P;
47     err = abs((y2_e + y2_h) - 1);
48     T = T + 0.0001;
49 end
50
51 // Changing the temperature in deg F
52 T2 = T*9/5 + 32; // [F] Bubble-point temperature
53
54 // c) In this method, we use L-R rule, instead of
    simple Raoult's law
55 // So,
56 //  $y_i = (x_i * p_i) / (v_i * P)$ 
57 // Where calculated values of  $v_i$  from EOS are given
58 v_e = 0.6290642; // For ethane
59 v_h = 0.0010113; // For n-heptane
60
61 // We again use trial and error on the temperature

```

```

62 // Let us assume the initial temperature
63 Ti = 10; //[C]
64 err = 1;
65
66 while err > 10(-4)
67     p2_e = (10(A_e - B_e/(C_e + Ti)))*(14.7/760);
68     p2_h = (10(A_h - B_h/(C_h + Ti)))*(14.7/760);
69     y3_e = p2_e*x_ethane/(P*v_e);
70     y3_h = p2_h*x_heptane/(P*v_h);
71     err = abs((y3_e + y3_h) - 1);
72     Ti = Ti + 0.0001;
73 end
74
75 // Changing the temperature in deg F
76 T3 = Ti*9/5 + 32; //[F] Bubble-point temperature
77
78 printf(" The results are summarized in the following
       table:\n\n");
79 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");
80 printf(" \t T(deg F) \t\t\t\t %f \t\t\t\t %f \t\t\t\t
       \t %f\n", T1, T2, T3);
81 printf(" \t y_ethane \t\t\t\t %f \t\t\t\t %f \t\t\t\t
       \t %f\n", y1_e, y2_e, y3_e);
82 printf(" \t y_heptane \t\t\t\t %f \t\t\t\t %f \t\t\t\t
       \t\t\t\t %f\n\n", y1_h, y2_h, y3_h);
83 printf(" Where T is boiling point temperature");

```

Scilab code Exa 10.3 Bubble point temperature estimation using SRK EOS

```

1 clear;
2 //clc();

```

```

3
4 // Example 10.2
5 // Page: 262
6 printf("Example-8.2 Page no.-262\n\n");
7
8 //***Data***/
9
10 // The initial data for this example is same as that
    of example 10.2, i.e.
11 P = 800;//[psia] Bubble point pressure
12 x_e = 0.60; // Mole fraction of ethane in liquid
    phase
13 x_h = (1-x_e); // Mole fraction of n-heptane in the
    liquid phase
14 R = 0.08314; //( L*bar/(mol*K)) Universal gas
    constant
15
16 // Changing the pressure in bar
17 Pb = (800/14.7)*(1.01325);//[bar]
18
19 // In this problem we will denote ethane by 'e' and
    that to n-heptane by 'h'
20 // From table A.1 ( page 417 ) given in the book,
    critical temperatures of ethane and heptane are
21 T_c_e = 305.3;//[K]
22 T_c_h = 540.2;//[K]
23
24 // and critical pressures are
25 P_c_e = 48.72;//[bar]
26 P_c_h = 27.40;//[bar]
27
28 // also the accentric facors are
29 w_e = 0.1;
30 w_h = 0.35;
31
32 // Thus we have
33 P_r_e = Pb/P_c_e;
34 P_r_h = Pb/P_c_h;

```

```

35
36 // Now from equations (F.13) and (F.14) ( page 459 )
    given in the book we have
37 //  $A_e = 0.42747 + ( 1 + (0.480 + 1.574*w_e - 0.17*
    w_e^2) * ( 1 - T_{r_e}^{0.5} ) )^2 * (P_{r_e}/T_{r_e}^2)$ 
    );
38 //  $A_h = 0.42747 + ( 1 + (0.480 + 1.574*w_h - 0.17*
    w_h^2) * ( 1 - T_{r_h}^{0.5} ) )^2 * (P_{r_h}/T_{r_h}^2)$ 
    );
39 // and
40 //  $B_e = 0.08664 * (P_{r_e}/T_{r_e})$ ;
41 //  $B_h = 0.08664 * (P_{r_h}/T_{r_h})$ ;
42
43 // We will take the help trial and error method both
    on Temperature and the vapor phase composition
    of ethane
44 // Let us assume the starting temperature 200 deg F.
    Changing this temperature in K
45 T = (200-32)*5/9 + 273.15; // [K]
46 err = 1;
47
48 while err > 10(-4)
49     T_r_e = T/T_c_e;
50     T_r_h = T/T_c_h;
51     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);
52     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);
53
54     B_e = 0.08664*(P_r_e/T_r_e);
55     B_h = 0.08664*(P_r_h/T_r_h);
56
57 // Now we will take the starting value of vapor
    phase composition of ethane as
58 y_e = 0.9;
59 err1 = 1;

```



```

60
61 while err1 > 10(-6)
62     // Now value of A_mix and B_mix for both
        liquid and vapor phase are calculated as
63
64     A_mix_l = (x_e*sqrt(A_e) + x_h*sqrt(A_h))
        ^ (2); // For liquid phase
65     A_mix_v = (y_e*sqrt(A_e) + (1 - y_e)*sqrt(
        A_h))^ (2); // For vapor phase
66     B_mix_l = (x_e*B_e + x_h*B_h); // For liquid
67     B_mix_v = (y_e*B_e + (1 - y_e)*B_h); // For
        liquid
68     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');
69     z_l = fsolve(0.2,f);
70     // and
71     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');
72     z_v = fsolve(0.3,g);
73     // Now
74     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);
75     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);
76     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);
77     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);
78     K_e = phi_el/phi_ev;
79     K_h = phi_hl/phi_hv;
80     y_e1 = K_e*x_e;
81     y_h1 = K_h*x_h;

```

```

82         err1 =abs((y_e1 - y_e));
83         y_e = y_e1;
84     end
85
86     err = abs((y_e1 + y_h1) -1);
87     T = T + 0.1;
88
89 end
90
91 // Changing the temperature in deg F, we have
92 Tf = ( T - 273.15)*9/5 + 32; //[F]
93
94 printf(" Bubble point of the given ethanol and n-
      heptane mixture at 800 psia is %f deg F\n",Tf);
95 printf(" Amount of ethanol in the vapour phase of
      the mixture at the given condition is %f \n",y_e1
      );
96 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

```
1 clear;
2 //clc();
3
4 // Example 11.1
5 // Page: 272
6 printf("Example-11.1 Page no.-272\n\n");
7
8 /***Data***/
9 V_water = 1; //[L] volume of the water
10 Temp = 25; //[C]
11 d_benzene = 0.88; //[g/ml] density of the benzene
12 M_benzene = 78; //[g/mol] molecular weight of the
    benzene
13 M_water = 18; //[g/mol]
14 // Typically a buret will deliver about 20 drops of
    lliquid per millimeter , so moles of benzene in
    one drop is
15 n_1drop = 1/20*(d_benzene/M_benzene); //[mol/drop]
16 // No of moles in 1 litre of the water is
```

```

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

```

Scilab code Exa 11.2 Reporting and presenting LLE data

```

1 clear;
2 //clc();
3
4 // Example 11.2
5 // Page: 273
6 printf("Example-11.2 Page no.-273\n\n");
7
8 //***Data***/
9 m_benzene = 1000;//[lbm]
10 M_benzene = 78;//[lbm/lbmol]
11 // The total moles benzene are
12 n_benzene = m_benzene/M_benzene;//[lbmol]
13 // To saturate the water with benzene
14 // the mole fraction of the benzene in the water
    will be
15 x_benzene = 0.000405;
16 // and
17 // n_benzene = x_benzene*n_T;
18 // in this case n_benzene << n_water, so n_T =
    n_water

```

```

19 // Thus
20 //n_benzene = x_benzene*n_water;
21 n_water = n_benzene/x_benzene;//[lbmol]
22 m_water = n_water*18;//[lbm]
23 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

```

1 clear;
2 //clc();
3
4 // Example 11.3
5 // Page: 277
6 printf("Example-11.3 Page no.-277\n\n");
7
8 //***Data***/
9
10 Temp = 25;//[C]
11 n_water = 3.75;//[mol]
12 n_ethanol = 2.5;//[mol]
13 n_benzene = 3.75;//[mol]
14
15 // By the simple stoichiometry the overall mole
    fractions are
16
17 x_water = 0.375;
18 x_ethanol = 0.250;
19 x_benzene = 0.375;
20
21 // 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.

```



```

7
8  /**Data**/
9
10 Temp = 25; // [C]
11 // Here we assume benzene to be component 1 and
    water to be component 2
12 // From table 11.1 given in the book(page 273)
13 // The mole fraction of benzene in water is
14 x_1in2 = 405; // [ppm]
15 // and the mole fraction of water in benzene is
16 x_2in1 = 3000; // [ppm]
17
18 // Thus mole fraction of water in water rich phase
    is
19 x_water_w = (10^(6)-405)/(10^(6));
20 x_benzene_w = 1-x_water_w;
21
22 // and mole fraction of the benzene in benzene rich
    phase is
23 x_benzene_b =(10^(6)-3000)/(10^(6));
24 x_water_b = 1-x_benzene_b;
25
26 // Here both x_water and x_benzene are nearly equal
    to 1
27 // Thus assumption used for derivation of the
    equation 11.4(page 282) are suitable here and the
    equation is
28 //  $x_{i-1} = y_{i-1}$  , where  $y_{i-1}$  is activity
    coefficient
29
30 // So activity coefficient of benzene in water is
31 y_benzene = 1/(x_benzene_w);
32 // and activity coefficient of the water in benzene
    is
33 y_water = 1/(x_water_b);
34 printf(" Activity coefficient of benzene in water is
    %f\n\n", y_benzene);
35 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

```
1 clear;
2
3 //clc();
4
5 // Example 11.5
6
7 // Page: 283
8
9 printf("Example-11.5 Page no.-283\n\n");
10
11 /***Data***/
12
13
14 R = 8.314; // [J/(mol*K)] Universal gas constant
15 T = 298.15; // [K] Temperature
16 g_a_0 = 2; // [kJ/mol] Gibb's free energy of the pure
    species 'a'
17 g_b_0 = 1; // [kJ/mol] Gibb's free energy of the pure
    species 'b'
18
19 for a = 0:3
20     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)) ")
21
22 x=[0.000001:0.01:0.99999];
23
24 fplot2d(x,f)
25 xlabel("mole fraction of species a, x_a");
26 ylabel("gibb's free energy per mole of mixture,
    g_mixture kJ/mol");
```



```

27 end
28
29 printf(" The plot is shown in the graphic window.");

```

Scilab code Exa 11.6 Two liquid phase

```

1 clear;
2 //clc();
3
4 // Example 11.6
5 // Page: 283
6 printf("Example-11.6 Page no.-284\n\n");
7
8 //***Data***//
9
10 T = 92 + 273.15; // [K] Temperature of the system
11 R = 8.314; // [m^(3)*Pa/(mol*K)] universal gas
    constant
12 // Van Laar equation coefficients are given
13 A = 1.2739;
14 B = 3.9771;
15
16
17 // From van laar equation, for water, we have
18 //  $\ln Y_a = \frac{B^2 A (1-x_a)^2}{(A x_a + B(1-x_a))^2}$ ;
19 // Similarly for n-butanol
20 //  $\ln Y_b = \frac{A^2 B x_a^2}{(A x_a + B(1-x_a))^2}$ ;
21
22 // Let us say,  $g = g_{mix} - \sum(x_i g_{i_0})$ 
23 // So, plotting  $g$  vs  $x_i$  we have
24
25 def(" [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)) )")
26
27 x_a = [0.000001:0.01:0.99999];
28 fplot2d(x_a,f)
29 xlabel(" Mol fraction of species a, x_a");
30 ylabel(" g_mix - sum(x_i*g_i0)");
31
32 // Now drawing tangent
33 x = [0.000001:0.01:0.99999];
34 plot2d(x,[1.2090312*x - 1.251495764])
35
36 // Figure shows the results of the calculation of
    the whole range of x_a
37 // Drawing the tangent to the curve, the line
    touches the curve at two points x_a = 0.47 and
    0.97
38 printf(" Thus based on the several assumptions that
    the Van Laar equation is an accurate
    representation of LLE,\n");
39 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

```

1 clear;
2 //clc();
3
4 // Example 11.7
5 // Page: 286
6 printf(" Example-11.7 Page no.-286\n\n");
7
8 //***Data***/
9 R = 8.314; // [J/(ml*K)]
10 // We find that the water in benzene least squares
    fit line has the equation

```

```

11 // log(x_water(benzene-reach phase)) = 4.175 - 2967.7/
    T
12
13 // equation 11.7 in the book (page 286) is
14 // log(x_i_1) = (h_i_0 - h_i_average)/(RT) + constant
    of integration
15
16 // Comparing the two equations term by term, we have
17
18 // (h_i_0 - h_i_average)/(RT) = -2967.7/T
19 // let us say (h_i_0 - h_i_average) = h_mix
20 // So that
21 h_mix = -2967.7*R/1000; // [kJ/mol] Heat of mixing of
    water-in-benzene
22
23 // Now, for benzene-in-water the constant in the
    above equation is -522.9K, so
24 h_mix_1 = 522.9*R/1000; // [kJ/mol] Heat of mixing of
    benzene-in-water
25
26 printf(" Heat of mixing of water-in-benzene is given
    as %f kJ/mol\n", h_mix);
27 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

```

1 clear;
2 // clc ();
3
4 // Example 11.8
5 // Page: 287
6 printf("Example -11.8 Page no. -287\n\n");
7
8 //***Data***//

```

```

9
10 T_i = 50;//[F] Initial temperature of the system
11 T_f = 20;//[F] Final temperature of the system
12 M_gas = 115;//[g/mol] Molecular weight of gasoline
    at room temperature
13 M_water = 18;//[g/mol] Molecular weight of water at
    the room temperature
14 d = 720;//[g/L] density of gasoline at the room
    temperature
15
16 // 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
17 s_50 = 0.00026;//[mol fraction]
18
19 // And linearly extrapolating the cyclohexane curve
    in figure 11.10 to 20 deg F, we get the
    solubility of water at 20deg F as
20 s_20 = 0.0001;//[mol fraction]
21
22 // So, rejected water is
23 s_rej = s_50 - s_20;// mol of water per mole of
    gasoline
24
25 // In terms of weight, rejected water will be
26 w = (s_rej*d*M_water)/M_gas;//[g water/L gasoline]
27
28 printf(" The amount of water that will come out of
    the solution in the gasoline will be %f g water/L
    gasoline\n",w);
29 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

```

1 clear;
2 //clc();
3
4 // Example 11.9
5 // Page: 290
6 printf("Example-11.9 Page no.-290\n\n");
7
8 //***Data***/
9 Temp = 25;//[C]
10 x_water = 5;//[mo]
11 x_benzene = 0.1;//[mol]
12
13 // The fugacity of the ethanol must be same in both
14 // phases so that we have distribution coefficient
15 //  $K = (\text{distribution coefficient}) = \frac{x_{\text{ethanol}}(\text{water-rich phase})}{x_{\text{ethanol}}(\text{benzene-rich phase})} = \frac{y_{\text{ethanol}}(\text{benzene-rich phase})}{y_{\text{ethanol}}(\text{water-rich phase})}$ 
16 // Here distribution coefficient is defined as the
17 // ratio of mole fractions of the distributed solute
18 // between the two phases.
19 // We observe that for the first experimental data
20 // set in table 11.4 in the book(page 276)
21 x_ethanol_water_rich = 3.817;//[%]
22 x_ethanol_benzene_rich = 1.010;//[%]
23
24 // So
25 K = x_ethanol_water_rich/x_ethanol_benzene_rich;
26
27 // Now for all the data set in the table 11.4 in the
28 // book(page 276), we will draw a plot
29
30 X =
31     [3.817,7.968,12.977,18.134,23.540,24.069,27.892,31.725,35.510,39.
32
33 Y =

```

```

[1.010,3.323,5.860,9.121,12.939,13.340,16.090,18.943,22.444,26.21

29 Z = X./Y;
30
31 // Plotting the graph between 'Z' and 'Y'
32 plot(Y,Z);
33 xgrid();
34 xlabel("Mol% ethanol in benzene-rich phase ");
35 ylabel("Distribution coefficient of ethanol,
      K_ethanol");
36
37 // We see from the plot that at the low mole percent
      of ethanol , the distribution coefficient is
      approximately
38 K_1 = 4;
39
40 // Thus ratio of the amount of the ethanol
      distributed in the two phases will be 4
41 //the amount in mol % is
42 // for water rich phase
43 m_water_rich = 100*K_1/(K_1+1);
44 m_benzene_rich = 100/(K_1+1);
45
46 printf(" Ethanol ' 's 0.1 mol distributed in the water
      rich phase will be %f mol%% of the total mol\n
      ",m_water_rich);
47 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

```

1 clear;
2 //clc();
3

```

```

4 // Example 11.10
5 // Page: 293
6 printf("Example-11.10 Page no.-293\n\n");
7
8 /**Data**/
9 Temp = 20; //[C]
10 // At this temperature solubility of NaCl is
11 s = 36.0; //[g per 100 g of water]
12 M_NaCl = 58.5; //[g/mol] molecular weight of NaCl
13 M_water = 18; //[g/mol] molecular weight of water
14
15 // weight fraction of NaCl
16 w = s/(s+100);
17 // In weight percentage
18 w_percent = w*100; //[wt %]
19
20 // Mol fraction of the NaCl
21 x = (s/M_NaCl)/((s/M_NaCl)+(100/M_water));
22 // In mol percentage
23 x_percent = x*100; //[mol %]
24
25 printf(" Weight fraction of the NaCl in the
        saturated solution is %0.1f wt %%\n",w_percent);
26 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

```

1 clear;
2 //clc();
3
4 // Example 11.11
5 // Page: 293
6 printf("Example-11.11 Page no.-293\n\n");
7

```

```

8  //***Data***//
9  T_inlet = 68; // [F]
10 T_outlet = 110; // [F]
11
12 // from the figure 11.13 we read that at 68F the
   solubility of CaCO3 and CaSO4.2H2O are
13 s_inlet_carbonate = 60; // [ppm]
14 s_inlet_sulphate = 2020; // [ppm]
15
16 // At 110F the solubility of the CaCO3 is
17 s_outlet_carbonate = 40; // [ppm]
18 // at 110F the least soluble form of the CaSO4 is
   anhydride with the solubility
19 s_outlet_sulphate = 2000; // [ppm]
20 // This is close enough to the solubility of the
   gypsum at 68F
21 // so we conclude that we would not expect either
   form of CaSO4 to prdcipitate
22
23 // Thus total amount of the calcium carbonate which
   will cime out of the solution and will remain
   in the heater will be
24 w = s_inlet_carbonate - s_outlet_carbonate; // [ppm]
25 printf (" Total amount of the solid left behind in
   the heater will be          %0.1f ppm\n\n",w);
26
27 // Now if a typical houshold water heater heats 100
   gallons/per day , we would expect to deposite
28 w_per_day = w*10(-6)*100*8.33; // [lb/day]
29 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

```
1 clear;
```



```

2 //clc ();
3
4 // Example 11.12
5 // Page: 298
6 printf("Example-11.12 Page no.-298\n\n");
7
8 /***Data***/
9 x_2 = 0.1;
10
11 //y_i_1 = (x_i_ideal /x_i_1) , at constant
    temperature
12 // 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
13
14 // We see that at x_solute = 10%
15 // T_m/T for the solution in benzene at which log(
    x_experimental) = -1 is equal to 1.332
16 // and that for the solution in CCl4 is equal to
    1.288
17
18 //Now at the that value of the T_m/T
19 x_ideal_benzene = 0.114;
20 x_ideal_CCl4 = 0.152;
21
22
23 // In benzene the average these compounds is
24 y_i_1 = x_ideal_benzene/x_2;// corresponding to
    practically ideal solution
25
26 // and in benzene the average of these compounds is
27 y_i_2 = x_ideal_CCl4/x_2;// corresponding to mild
    type II behavior
28
29 printf(" Activity coefficient in benzene
    corresponding to practically ideal solution is %0
    .2f\n",y_i_1);

```

```

30 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

```

1  clear;
2  //clc();
3
4  // Example 11.13
5  // Page: 299
6  printf("Example-11.13   Page no.-299\n\n");
7
8  //***Data***//
9
10 T = 273.15+20; // [K]
11
12 // The equation 11.15 (page 297) is given by
13 //  $\log(1/(x_{i-1}y_{i-1})) = \log(p_{i-solid\_phase}/$ 
     $p_{i-subcooled\_liquid}) = \Delta h_{solid\_to\_liquid}/($ 
     $R*T_{melting\_point})*(T_{melting\_point}/T-1)$ 
14
15 // Ignoring the moment the wild extrapolation
    involved, we simply insert the appropriate values
16 T_m = 273.15+800; // [K]
17 delta_h_fusion = 30219; // [J/g]
18 R = 8.314; // [J/(mol*K)]
19
20 // Let  $\log(1/(x_{i-1}y_{i-1})) = a$ 
21 a = delta_h_fusion/(R*T)*(T_m/T-1);
22
23 // Now
24 x_NaCl_into_y_i_1 = 1/exp(a);
25
26 // If we make the plausible assumption that  $y_{i-1} =$ 

```

```

    1.00, then
27 x_NaCl = 1/exp(a)*1;
28 printf(" The solubility of the NaCl in water at 20
    deg C is %e \n", x_NaCl);
29 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

```

1 clear;
2 //clc();
3
4 // Example 11.14
5 // Page: 301
6 printf("Example-11.14 Page no.-301\n\n");
7
8 //***Data***/
9 P = 1*14.7;//[psia]
10 T = 30;//[F]
11 //*****/
12 //The vapour pressure of ice at 30F is 0.0808 psia
    i.e.
13 p_ice = 0.0808;//[psia]
14 // We may assume that the solubility of nitrogen
    and oxygen in solid ice is negligible
15 //Thus
16 x_water_in_ice = 1.00;
17 //and thus use Raoult's law, finding
18 y_water_vapour = x_water_in_ice*p_ice/P;
19 printf(" Equilibrium concentration of water vapour
    in the air is %0.4f",y_water_vapour);

```

Scilab code Exa 11.15 GSE at high pressures

```
1 clear;
2 //clc();
3
4 // Example 11.15
5 // Page: 302
6 printf("Example-11.15 Page no.-302\n\n");
7
8 /***Data***/
9 T = 273.15+35; // [K]
10 p_d = 100; // [atm]
11 R = 82.06; // [(cm3*atm)/(mol*K)]
12 /*****/
13
14 //The calculated vapour pressure of naphthalene at
15 35C is
16 p_naphthalene = 0.00065; // [atm]
17 //The solid is practically pure naphthalene
18 x_naphthalene = 1.00;
19 //Total pressure
20 P = p_d;
21 //By Raoult's law
22 y_naphthalene = x_naphthalene*p_naphthalene/P;
23 //At this high a pressure the volume of solid
24 naphthalene is
25 v = 132; // [cm3/mol]
26 // We have equation  $\log(f_d/f_c) = v/(R*T)*(p_d-p_c)$ 
27 p_c = 1; // [atm]
28 f_d_by_f_c = exp(v/(R*T)*(p_d-p_c));
29 //and the estimated
30 y_naphthalene = f_d_by_f_c*y_naphthalene;
31 printf("Estimated solubility of naphthalene in CO2
32 is %e", y_naphthalene);
```

Chapter 12

Chemical Equilibrium

Scilab code Exa 12.1 Gibbs free energy

```
1 clear;
2 // clc ();
3
4 // Example 12.1
5 // Page: 311
6 printf("Example -12.1 Page no.-311\n\n");
7
8 //***Data***/
9 T = 298.15; // [K] temperature
10 P = 1; // [atm] pressure
11 R = 8.314*10^(-3); // [kJ/(mol*K)]
12
13 // For an ideal binary solution the Gibbs free
14 // energy is given by
15 //  $g_{mix} = \sum(x_i * g_{i,0}) + R*T*\sum(x_i * \log(x_i))$ 
16 // Differentiating the above equation with respect
17 // to  $x_a$ , remembering that for a binary mixture
18 //  $dx_b = dx_a$ , finding
19 //  $dg_{mix}/dx_a = g_{a,0} - g_{b,0} + R*T*[\log(x_a) + 1 - (\log(x_a) + \log(x_b))]$ 
```



```

10 P = 0.987; //[atm] pressure
11 g_0_NO = 86.6; //[kJ/mol] Free energy of formation
    the NO from elements
12 R = 8.314; //[J/(mol*K)]
13
14 // And the corresponding values for the elements N2
    and O2 are
15 g_0_O2 = 0.00;
16 g_0_N2 = 0.00;
17
18 // The reaction of the nitrogen and oxygen to form
    nitric oxide at 298.15 K is
19 // N2 + O2 = NO
20
21 // Here
22 delta_g_0 = 2*g_0_NO - g_0_O2 - g_0_N2; //[kJ/mol]
23 // Changing in the J/mol
24 delta_g_01 = delta_g_0*1000; //[J/mol]
25
26 // hence
27 K_298 = exp((-delta_g_01)/(R*T));
28
29 // The activities are all
30 // a_i = f_i/f_i_0
31 // f_i_0 correspond to the standard state , which for
    gas at idael gas state are
32 f_0_N2 = 1; //[bar]
33 f_0_O2 = 1; //[bar]
34 f_0_NO = 1; //[bar]
35
36 // If we make the most general statement of the
    activities (for gases ) we would have
37 // a_i = y_i*v_i*Y_i*P/f_i_0 = y_i*phi*P/f_i_0
38
39 // 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

```

```

40 //  $K_{298} = \frac{[a_{NO}]^2}{([a_{N2}] * [a_{O2}])} = \frac{[y_{NO}]^2}{([y_{N2}] * [y_{O2}])}$ 
41
42 // Now using this equilibrium constant we can
    calculare he equilibrium concentratin of NO in
    the air sample in which
43 //oxygen = 21%, nitrogen = 78% and argon = 1% ,so
44 y_N2 = 0.78;
45 y_O2 = 0.21;
46
47 // Hence From above expression , we have
48 y_NO_298 = sqrt(K_298*y_N2*y_O2);
49
50 // Making the similar calculations for the
    temperature 2000 K, we have
51 T_1 = 2000; // [K]
52 K_2000 = exp((-delta_g_01)/(R*T_1));
53
54 // So,
55 y_NO_2000 = sqrt(K_2000*y_N2*y_O2)*10^(6); // [ppm]
56
57 printf(" The equilibrium constant for the reaction
    at 298.15 K is \t\t\t %e\n",K_298);
58 printf(" The concentration of NO at equilibrium at
    temperature 298.15 K is \t\t %e\n",y_NO_298);
59 printf(" The equilibrium constant for the reaction
    at 2000 K is \t\t\t %e\n",K_2000);
60 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

```

1 clear;
2 //clc ();
3

```



```

4 // Example 12.3
5 // Page: 321
6 printf("Example-12.3 Page no.-321\n\n");
7
8 /**Data**/
9 Temp = 2000; // [K]
10 n_air = 1; // [mol] no of moles of the air
11
12 // Let the moless of the NO formed be 2*x
13 // Then at equilibrium the unreacted moles of the N2
    and O2 will be (0.78-x) and (0.21-x)
    respectively
14
15 // from the previous example, we have
16 // [y_NO]^(2) = K_298*[y_N2]*[y_O2]
17 // here
18 K_2000 = 4*10^(-4);
19 // Substituting all the values, we have
20 // (2*x)^(2) = K_2000*(0.78-x)*(0.21-x)
21
22 //Now
23 deff(' [y]=f(x)', 'y = (2*x)^(2) - K_2000*(0.78-x)
    *(0.21-x) ');
24 // deff(' [y]=f(x)', 'y = (K_2000-2)*x^(2)-K_2000
    *(0.78+0.21)*x+K_2000*0.78*0.21 ');
25 x = fsolve(0,f);
26 // Here negative root is meaningless, so
27 // concentration of NO
28 c_NO = 2*x*10^(6); // [ppm]
29 // now
30 p = c_NO/8100*100;
31 printf(" The calculated NO cocentration is %f ppm,
    which %d%% of the value computed in example 12.1"
    ,c_NO,p);

```

Scilab code Exa 12.4 Mass action law

```
1 clear;
2 //clc();
3
4 // Example 12.4
5 // Page: 323
6 printf("Example-12.4 Page no.-323\n\n");
7
8 //***Data***/
9 n_water_0 = 0.833;//[mol]
10 n_ethylene_0 = 1;//[mol]
11 n_ethanol_0 = 0;//[mol]
12 n_T_0 = (n_water_0+n_ethylene_0+n_ethanol_0);//[mol]
13
14 // In general, we must have
15 //  $K = \frac{a_{\text{C}_2\text{H}_5\text{OH}}}{[a_{\text{C}_2\text{H}_4}][a_{\text{H}_2\text{O}}]}$ 
16
17 // Here we will assume that we have an ideal
18 // solution of the ideal gases for which in equation
19 // 12.18 (page 320), we have
20 //  $v_i Y_i = \phi_i = 1.00$ , and that for each reactant
21 // or product  $f_{i,0} = 1$  bar so that
22 //  $\frac{a_{\text{C}_2\text{H}_5\text{OH}}}{[a_{\text{C}_2\text{H}_4}][a_{\text{H}_2\text{O}}]} = K = \frac{[x_{\text{C}_2\text{H}_5\text{OH}}P]}{/(1 \text{ bar})} / \left( \frac{[x_{\text{C}_2\text{H}_4}P]}{/(1 \text{ bar})} * \frac{[x_{\text{H}_2\text{O}}P]}{/(1 \text{ bar})} \right)$ 
23 // So
24
25 //  $K = \frac{[x_{\text{C}_2\text{H}_5\text{OH}}]}{[x_{\text{C}_2\text{H}_4}][x_{\text{H}_2\text{O}}]} * (1 \text{ bar}) / P$ 
26 // Here the stoichiometric coefficients are -1,-1
27 // and +1, so that summation( $v_i$ ) = -1 and
28
29 //  $\frac{[(0+e)/(1.833-e)]}{/(1.833-e)} / \left( \frac{[(1-e)/(1.833-e)] * [(0.833-e)]}{/(1.833-e)} \right) = K * P / (1 \text{ bar})$ 
30
31 printf("''The mass action law '' statement for the
32 given reaction is\n\n  $\frac{[(0+e)/(1.833-e)]}{/(1.833-e)} * \frac{[(0.833-e)/(1.833-e)]}{/(1.833-e)} = K * P / (1 \text{ bar})$ 
33 ")
```

Scilab code Exa 12.5 Mass action law

```
1 clear;
2 //clc();
3
4 // Example 12.5
5 // Page: 324
6 printf("Example-12.5 Page no.-324\n\n");
7
8 //***Data***//
9 Temp = 298;//[K]
10 K = 29.6; // equilibrium constant at 298 K
11 P = 1;//[bar]
12 n_water_0 = 0.833;//[mol]
13 n_ethylene_0 = 1;//[mol]
14 n_ethanol_0 = 0;//[mol]
15 n_T_0 = (n_water_0+n_ethylene_0+n_ethanol_0);//[mol]
16
17 // From the previous example, we have
18 //  $[(0+e)/(1.833-e)]/([(1-e)/(1.833-e)]*[(0.833-e)/(1.833-e)]) = K*P/(1 \text{ bar})$ 
19 // let  $y = [(0+e)/(1.833-e)]/([(1-e)/(1.833-e)]*[(0.833-e)/(1.833-e)])-K*P/(1 \text{ bar})$ 
20 def(' [y]=f(e)', 'y = [(0+e)/(1.833-e)]/([(1-e)/(1.833-e)]*[(0.833-e)/(1.833-e)])-K*P/(1) ');
21 e_1 = fsolve(0,f);
22 e_2 = fsolve(0.5,f);
23
24 // Here the root 'e_2' is meaningless, Then
25 y_ethanol = [(0+e_2)/(1.833-e_2)];
26 y_ethylene = [(1-e_2)/(1.833-e_2)];
27 y_water = [(0.833-e_2)/(1.833-e_2)];
28
29 printf("Concentration of the ethylene at the
```

```

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

```

Scilab code Exa 12.6 Reversible reaction

```

1 clear;
2 //clc();
3
4 // Example 12.6
5 // Page: 324
6 printf("Example-12.6 Page no.-324\n\n");
7
8 /***Data***/
9 Temp = 273.15+25;//[C]
10 P = 1;//[bar]
11 R = 8.314;//[J/(mol*K)]
12
13 // We have the reaction as
14 // H2 + 0.5O2 = H2O
15 // Using values of the Gibbs free energies of
    formation in the Table A.8(page 427) we have
16 g_H2O_0 = -237.1;//[kJ/mol]
17 g_O2_0 = 0;//[kJ/mol]
18 g_H2_0 = 0;//[kJ/mol]
19 // now
20 delta_g_0 = g_H2O_0 - 0.5*g_O2_0-g_H2_0;//[kJ/mol]
21 // expressing delta_g_0 in [J/mol]
22 delta_g_1 = delta_g_0*1000;//[J/mol]
23
24 // and
25 K = exp((-delta_g_1)/(R*Temp));
26

```

```

27 // And we have
28 //  $K = [a_{\text{H}_2\text{O}}]/([a_{\text{H}_2}][a_{\text{O}_2}]^{0.5})$ 
29 // 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
30 //  $v_i Y_i = \phi_i = 1.00$ , and that for each reactant
    or product  $f_{i,0} = 1$  bar, putting all the values
    and simplifying
31
32 //  $K = [y_{\text{H}_2\text{O}}]/([y_{\text{H}_2}][y_{\text{O}_2}]^{0.5}) * ((1 \text{ bar})/P)^{0.5}$ 
33 // Choosing oxygen as the selected reactant, and
    assuming that we begin with 0.5 mol of oxygen and
    1 mol of hydrogen,
34 // we have the stoichiometric coefficients of -1,
    -0.5 and +1
35 // and
36 n_T_0 = 1.5; // [mol]
37 // Also  $\sum v_i = -0.5$ 
38
39 // Thus
40 //  $K = [e/(1.5 - 0.5 * e)] / ([ (1 - e)/(1.5 - 0.5 * e) ] * [ (0.5 - 0.5 * e)/(1.5 - 0.5 * e) ]^{0.5})$ 
41
42 // Now
43 //  $\text{deff}(' [y]=f(e)', 'y = [e/(1.5 - 0.5 * e)] / ([ (1 - e)/(1.5 - 0.5 * e) ] * [ (0.5 - 0.5 * e)/(1.5 - 0.5 * e) ]^{0.5})');$ 
44 //  $e = \text{fsolve}(.99999, f);$ 
45 //  $e = (1 - 2.4e-28);$ 
46
47 // So the equilibrium concentration of the hydrogen
    and oxygen are as
48 //  $y_{\text{H}_2} = [(1 - e)/(1.5 - 0.5 * e)];$ 
49 //  $y_{\text{O}_2} = [(0.5 - 0.5 * e)/(1.5 - 0.5 * e)];$ 
50 // These values are so less that scilab consol is
    displaying them zero, however we get
51 y_H2 = 2.4e-28;
```

```

52 y_02 = 0.5*y_H2;
53
54 printf(" The equilibrium mol fraction of the
    hydrogen is %0.3e\n",y_H2);
55 printf(" And the equilibrium mol fraction of the
    oxygen is %e",y_02);

```

Scilab code Exa 12.7 Standard state Gibbs free energy

```

1 clear;
2 //clc();
3
4 // Example 12.7
5 // Page: 327
6 printf("Example-12.7 Page no.-327\n\n");
7
8 //***Data***/
9 Temp = 298.15;//[K]
10 Press = 1*10^(5);//[Pa]
11 R = 8.314;//[J/(mol*K)]
12
13 // We will calculate the free energy change from
    liquid to hypothetical gas in three steps
14 // 1) The liquid is reduced in pressure from the
    standard pressure of 1 bar to its vapour pressure
    at 298.15K and for this change in the state we
    have
15 v_liquid = 1.805*10^(-5);//[m^(3)/mol] this liquid
    specific volume and we will treat it as a
    constant
16
17 // The vapour pressure of the water 25C is given as
18 P_vapour_25 = 0.0317*10^(5);//[Pa]
19
20 // thus change in the Gibbs free energy is

```

```

21 delta_g_0_1 = integrate('v_liquid*P^(0)', 'P', Press,
    P_vapour_25); // [J/mol]
22
23 // 2) In the second step the liquid is vaporized at
    that pressure, for which
24 delta_g_0_2 = 0; // [J/mol]
25 // because this is an equilibrium vaporization.
26
27 // 3) And in this last step the vapour is replaced
    by an ideal gas, which will not condense, and
    compressed from the vapour pressure at 298.15K to
    1 bar
28 // In this case the specific volume v_ideal of the
    ideal gas is replaced by the ideal gas law viz. (
    R*T)/P
29 delta_g_0_3 = (R*Temp)*integrate('1/P', 'P',
    P_vapour_25, Press); // [J/mol]
30
31 // Thus total change in free energy is
32 delta_g_0 = delta_g_0_1+delta_g_0_2+delta_g_0_3; // [J
    /mol]
33 //expressing the result in kJ/mol
34 delta_g_1 = delta_g_0/1000; // [kJ/mol]
35
36 printf(" Total change in the free energy of water,
    going under given conditions, is %0.2f kJ/mol\n\n
    ", delta_g_1);
37
38 // From Table A.8 we find
39 delta_g_0_ideal_gas = -228.6; // [kJ/mol]
40 delta_g_0_liquid = -237.1; // [kJ/mol]
41 // So
42 delta_g_o = delta_g_0_ideal_gas - delta_g_0_liquid; // [
    kJ/mol]
43
44 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

```
1 clear;
2 //clc();
3
4 // Example 12.8
5 // Page: 330
6 printf("Example -12.8 Page no.-330\n\n");
7
8 /***Data***/
9
10 T1 = 273.15+25; // [K]
11 T2 = 273.15+400; // [K]
12 R = 8.314; // [J/(mol*K)]
13
14 // Using the table A.8, we have
15 // Gibb's free energy of the various species at
    298.15 K are
16 g0_NH3 = -16.5; // [kJ/mol]
17 g0_N2 = 0; // [kJ/mol]
18 g0_H2 = 0; // [kJ/mol]
19
20 // We have the reaction as
21 // 0.5N2 + 1.5H2 = NH3
22
23 // So, Gibb's free energy change in the reaction is
    given as
24 delta_g_0 = g0_NH3 - 0.5*g0_N2 - 1.5*g0_H2; // [kJ/mol
    ]
25
26 // and
27 K_1 = exp(-delta_g_0*1000/(R*T1)); // Equilibrium
    constant of the reaction at temperature 298.15 K
```



```

28
29 // Similarly enthalpy of the various species are
30 h0_NH3 = -46.1; // [kJ/mol]
31 h0_N2 = 0; // [kJ/mol]
32 h0_H2 = 0; // [kJ/mol]
33
34 // So, enthalpy change of the reaction is given as
35 del_h_1 = h0_NH3 - 0.5*h0_N2 - 1.5*h0_H2; // [kJ/mol]
36
37 // Now, from Table 12.3( page 332 )
38 a_NH3 = 3.578;
39 a_H2 = 3.249;
40 a_N2 = 3.280;
41 b_NH3 = 3.020*10^(-3); // [1/K]
42 b_H2 = 0.422*10^(-3);
43 b_N2 = 0.593*10^(-3);
44 c_NH3 = 0; // [1/K^(2)]
45 c_H2 = 0; // [1/K^(2)]
46 c_N2 = 0; // [1/K^(2)]
47 d_NH3 = -0.186*10^(5); // [K^(2)]
48 d_H2 = 0.083*10^(5); // [K^(2)]
49 d_N2 = 0.040*10^(5); // [K^(2)]
50
51 // So,
52 del_a = a_NH3 - 0.5*a_N2 - 1.5*a_H2;
53 del_b = b_NH3 - 0.5*b_N2 - 1.5*b_H2;
54 del_c = c_NH3 - 0.5*c_N2 - 1.5*c_H2;
55 del_d = d_NH3 - 0.5*d_N2 - 1.5*d_H2;
56
57 // Now, enthalpy change of the reaction at any other
    temperature is given by
58 // 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'
59 // Integrating and putting the limits , we have
60 // 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)

```

```

61 // let
62 I = R*( del_a*T1 + del_b*T1^(2)/2 + del_c*T1^(3)/3 -
        del_d/T1);//[J/mol]
63
64 // From equation 12.28 and above relations we have
65 // 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
66 // Let log(K_2/K_1) = X, So,
67 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);
68
69 // So,
70 K_2 = K_1*exp(X);
71
72 printf(" Equilibrium constants for the formation of
        ammonia from hydrogen and nitrogen are \n\n");
73 printf(" K = %0.0f at temperature 25 deg C\n\n",K_1)
        ;
74 printf(" K = %f at temperature 400 deg C\n",K_2);

```

Scilab code Exa 12.9 Equilibrium conversion of a mixture

```

1 clear;
2 //clc();
3
4 // Example 12.9
5 // Page: 335
6 printf(" Example -12.9 Page no.-335\n\n");
7
8 //***Data***/
9 // Initial moles of the gases are
10 n_H2_0 = 1.5;//[mol]
11 n_N2_0 = 0.5;//[mol]

```

```

12 n_NH3_0 = 0;//[mol]
13 T_1 = 298.15;//[K]
14 T_2 = 673.15;//[K]
15 P = 1;//[bar]
16
17 // We start with the equation as
18 // 
$$\frac{f_{\text{NH}_3}/f_{0,\text{NH}_3}}{([f_{\text{N}_2}/f_{0,\text{N}_2}]^{0.5} * [f_{\text{H}_2}/f_{0,\text{H}_2}]^{1.5})} = K$$

19
20 // For a pressure of 1 bar with the assumption of
    ideal solution of ideal gases and standard state
    fugacities of 1 bar,
21 //  $a_i = [f_i/f_{0,i}] = [P*y_i/(1 \text{ bar})] = y_i$ 
22 // The equilibrium relation is given by
23 //  $K = [y_{\text{NH}_3}]/([y_{\text{N}_2}]^{0.5} * [y_{\text{H}_2}]^{1.5})$ 
24
25 // We have the stoichiometric coefficient of N2, H2
    and NH3 as -0.5, -1.5 and +1 respectively , so
    summation(v_i) = -1
26 // Now using the equilibrium relations which are
    Equations 12.W, 12.X and 12.Y ( page 322 ), we
    have
27
28 //  $K = ((0+e)/(2-e))/(((0.5-0.5*e)/(2-e))^{0.5} * ((1.5-1.5*e)/(2-e))^{1.5})$ 
29 // Form the example 12.8 of this book we know that
30 K_298 = 778; // at temperature 298.15K
31 K_673 = 0.013; // at temperature 673.15K
32
33 // Solving for temperature 298.15
34 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');
35 e_1 = fsolve(0.97,g);
36 y_NH3_298 = e_1/(2-e_1);
37
38 // Similarly solving for temperature 673.15K
39 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');
40 e_2 = fsolve(0,h);
41 y_NH3_673 = e_2/(2-e_2);
42
43 printf(" The mole fraction of NH3 in the equilibrium
    at the temperature 298.15K and 1 bar is %f\n",
    y_NH3_298);
44 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

```

1 clear;
2 //clc();
3
4 // Example 12.10
5 // Page: 337
6 printf("Example-12.10 Page no.-337\n\n");
7
8 //***Data***/
9 Temp = 273.15+400; // [K]
10 P = 150*1.01325; // [bar]
11
12 // 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)
13 K_673 = 0.013;
14
15 // So
16 K = K_673*(P/1)^(1.5+0.5-1);
17
18 // We have
19 // K = ((0+e)/(2-e))/(((0.5-0.5*e)/(2-e))^(0.5)

```

```

    *((1.5-1.5*e)/(2-e))^(1.5))
20 def f(' [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');
21 e=fsolve(0.5,f);
22
23 // Thus mole fraction of the ammonia in the gas is
    given by
24 y_NH3 = (0+e)/(2-e);
25
26 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

```

1 clear;
2 //clc();
3
4 // Example 12.11
5 // Page: 338
6 printf("Example-12.11 Page no.-338\n\n");
7
8 //***Data***/
9 // The data used in this example will e same as in
    the example 12.10
10 T = 273.15+400;//[K] given temperature
11 P = 150*1.01325;//[bar] given pressure
12
13 // 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)
14 K_673 = 0.013;
15 // The value of 'K_v' is calculated by the equation
    12.BN, which is
16 // log10(1/K_v) = (0.1191849/T + 91.87212/T^(2) +

```

```

    25122730/T^(4))*P
17 // So
18 K_v = (10^((0.1191849/T + 91.87212/T^(2) + 25122730/
    T^(4))*P))^(-1);
19
20 // Thus
21 K = (K_673/K_v)*[P/1]^(1.5+0.5-1);
22
23 // Now from the previous example we have
24 // K = ((0+e)/(2-e))/(((0.5-0.5*e)/(2-e))^(0.5)
    *((1.5-1.5*e)/(2-e))^(1.5))
25
26 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');
27 e = fsolve(0.2,f);
28
29 // Mol fraction of the ammonia in the gas phase in
    the equilibrium is given by
30 y_NH3 = (0+e)/(2-e);
31
32 printf(" The mole fraction of the ammonia in the
    equilibrium is %0.2 f", y_NH3);

```

Scilab code Exa 12.12 Liquids and solids

```

1 clear;
2 // clc ();
3
4 // Example 12.12
5 // Page: 340
6 printf(" Example -12.12 Page no. -340\n\n");
7
8 //***Data***/
9 p_i = 1; // [atm] initial pressure
10 P = 150; // [atm] final pressure

```

```

11 T = 273+25;//[K] Given temperature
12 R = 8.314;//[J/(mol*K)]
13
14 // Now ignoring the difference between 25C and 20C,
    we use the values given in the table A.8 (page
    427) to get
15 delta_g_0 = 10.54*1000;//[J/mol]
16 // And thus
17 K = exp((-delta_g_0)/(R*T));
18
19 // Now the chemical reaction is given by
20 // C2H5OH + CH3COOH = C2H5OOC2H5 + H2O
21
22 // Let we start with 1 mol each of ethanol and
    acetic acid, and at equilibrium 'e' moles each of
    the reactants reacted, then
23 // remaining amount of each of the two reactants
    will be (1-e) and that products formation will be
    'e' mol each
24
25 // We have
26 //  $K = \frac{(a_{C2H5OOC2H5} \cdot a_{H2O})}{(a_{C2H5OH} \cdot a_{CH3COOH})} = \frac{(x_{C2H5OOC2H5} \cdot x_{H2O})}{(x_{C2H5OH} \cdot x_{CH3COOH})} = \frac{(e \cdot e)}{((1-e) \cdot (1-e))}$ 
27 // Now solving for 'e'
28 deff(' [y]=f(e) ', 'y = (e*e)/((1-e)*(1-e))-K');
29 e = fsolve(0,f);
30
31 // To see the effect of changing the pressure we
    first compute the volume increase of the reaction
32 //  $\Delta v = v_{C2H5OOC2H5} + v_{H2O} - v_{C2H5OH} - v_{CH3COOH}$ , where  $v_i$  is the molar volume of the
    ith component
33 // From the Table 12.4(page 340), we have
34 v_C2H5OOC2H5 = 97.67;//[ml/mol]
35 v_H2O = 18.03;//[ml/mol]
36 v_C2H5OH = 58.30;//[ml/mol]
37 v_CH3COOH = 57.20;//[ml/mol]

```

```

38
39 // Thus volume increase of the reaction is
40 delta_v = v_C2H5OOC2H5 + v_H2O - v_C2H5OH -
      v_CH3COOH; // [ml/mol]
41
42 // 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
43 // To calculate the extent of shifting we will take
      the help of the activity of each of the four
      component
44 //  $a_i = (f_i/f_{i_0}) = (x_i \cdot Y_i \cdot p_i) / p_i \cdot \exp(v/(R \cdot T) \cdot (P - p_i))$ 
45 // we will assume that this is an ideal solution so
      that  $Y_i = 1.00$ , for every component
46
47 // Now substituting the activity of each component
      in the expression of the equilibrium constant
      given above, we have
48 //  $K = (x_{C2H5OOC2H5} \cdot x_{H2O}) / (x_{C2H5OH} \cdot x_{CH3COOH}) \cdot \exp$ 
       $[(\text{delta}_v) / (R \cdot T) \cdot (P - p_i)]$ 
49 // or
50 //  $K = (e_1 \cdot e_1) / ((1 - e_1) \cdot (1 - e_1)) \cdot \exp[(\text{delta}_v) / (R \cdot T) \cdot (P - p_i)]$ 
51
52 // Solving for 'e_1'
53 def ('[y]=g(e_1)', 'y = (e_1 * e_1) / ((1 - e_1) * (1 - e_1)) *
      exp((delta_v) / (R * T) * (P - p_i)) - K');
54 e_1 = fsolve(0.2, g);
55
56 // Now if we carry out the calculation to enough
      significant figures then
57 a = e_1/e;
58
59 // It indicates that e_1 is 'a' times of that of the
      e
60 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

```
1 clear;
2 //clc();
3
4 // Example 12.13
5 // Page: 342
6 printf("Example-12.13 Page no.-342\n\n");
7
8 //***Data***//
9 P = 150;//[atm] given pressure
10 T = 400;//[C] temperature
11 // Using the values from the example 12.11, we know
    that
12 K = 0.013;
13 K_v = 0.84;
14 delta_v = 1.5+0.5-1;
15
16 // so
17 //  $K_p = (K/K_v) * [1/\text{bar}]^{(-\text{summation}(v_i))} = (K/K_v) * [1/\text{bar}]^{(\text{delta}_v)}$ 
18
19 K_p = (K/K_v)*[1/1]^(delta_v);//[1/bar]
20
21 printf(" Value of the K_p at the given condition is
    %f (1/bar)\n\n",K_p);
22
23 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

```
1 clear;
2 // clc ();
3
4 // Example 13.1
5 // Page: 349
6 printf("Example-13.1 Page no.-349\n\n");
7
8 //***Data***/
9 T =273.15+25; //[K] given temperature
10 R = 8.314; //[J/(mol*K)] universal gas constant
11
12 // We have the reaction as follows
13 // H2O = H+ + OH-
14
15 // Reading the free energy of species from the Table
    A.8 ( page 427), we have
16 g_0_H = 0; //[kJ/mol]
17 g_0_OH = -157.29; //[kJ/mol]
18 g_0_H2O = -237.1; //[kJ/mol]
```

```

19
20 // Thus free enaergy change of the reaction is
21 delta_g_0 = g_0_H + g_0_OH - g_0_H2O;//[kJ/mol]
22 // Changing in J/mol we have
23 delta_g_1 = delta_g_0*1000;//[J/mol]
24
25 // Now equilibrium constant of the reaction is given
    by
26 K = exp((-delta_g_1)/(R*T));
27
28 // Also, in terms of activity
29 // K = ([[H+]/(1 molal)]*[[OH-]/(1 molal)])/[a_water
    ]
30 // The activity of any pure liquid at its standard
    state is 1.00, and here water is practically pure
    , so
31 // K_w = [[H+]/(1 molal)]*[[OH-]/(1 molal)] = K
32 // or
33 K_w = K;
34
35 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

```

1 clear;
2 //clc();
3
4 // Example 13.2
5 // Page: 351
6 printf("Example -13.2 Page no.-351\n\n");
7
8 //***Data***/
9 n_H2SO4 = 1;//[mol] mole of the sulphuric acid
10 w_water = 1000;//[g] weight of the water

```

```

11 T =273.15+25;//[K] temperature
12 R = 8.314;//[J/(mol*K)]
13
14 // We the two sequential reaction , in which the
    first reaction is
15 //  $\text{H}_2\text{SO}_4 = \text{HSO}_4^- + \text{H}^+$ 
16
17 // From the Table A.8 (page 427) as given in the
    book, free energy of the above species are
18 g_0_H = 0;//[J/mol] free energy of the hydrogen ion
19 g_0_HSO4 = -756.01*1000;//[J/mol] free energy of the
    bisulphate ion
20 g_0_H2SO4 = -744.50*1000;//[J/mol] free energy of
    sulphuric acid
21
22 // So
23 delta_g_0 = g_0_H + g_0_HSO4 - g_0_H2SO4;//[J/mol]
24
25 // So equilibrium constant of the reaction is given
    by
26 K_1 = exp((-delta_g_0)/(R*T));
27
28 // Now the second reaction is which is going on
    sequentially is
29 //  $\text{HSO}_4^- = \text{SO}_4^{(-2)} + \text{H}^+$ 
30
31 // Again from the Table A.8 reading the values of
    free energy of the species of the above reaction ,
    we have
32 g_0_H = 0;//[J/mol] free energy of the hydrogen ion
33 g_0_SO4 = -744.62*1000; ;//[J/mol] free energy of
    sulphate ion
34 g_0_HSO4 = -756.01*1000;//[J/mol] free energy of the
    bisulphate ion
35
36 // So
37 delta_g_1 = g_0_H + g_0_SO4 - g_0_HSO4;//[J/mol]
38

```

```

39 // Equilibrium constant of thi reaction is
40 K_2 = exp((-delta_g_1)/(R*T));
41
42 // Now we have 1 mol of H2SO4 initially. Let e_1 mol
    of H2SO4 ionised at equilibrium
43 // Then amount of the each of two product i.e.
    bisulphate and hydrogen ion will be e_1 mol
44 // Now for the second reaction e_1 mol of the
    bisulphate ion will be treated as initial
    concentration.
45 // If at equilibrium e_2 moles of bisulphate ion has
    ionised
46 // In this case the amount of each of two product of
    this reaction will be e_2 mol
47 // So final amount of each of the species (in moles)
    at equilibrium is given as
48 // n_H2SO4 = (1-e_1)
49 // n_HSO4 = (e_1-e_2)
50 // n_SO4 = e_2
51 // n_H = (e_1+e_2)
52
53 // now
54 // K_1 = ([HSO4]*[H])/[H2SO4] = ((e_1-e_2)*(e_1+e_2)
    )/(1-e_1) ..... (1)
55 // and that for the second reaction
56 // K_2 = ([SO4]*[H])/[HSO4] = ((e_2)*(e_1+e_2))/(e_1
    -e_2) ..... (2)
57
58 // e = [e_1 e_2]
59 // Solving the two given simultaneous equations ,we
    have
60 function [f]=F(e)
61     f(1) = ((e(1)-e(2))*(e(1)+e(2)))/(1-e(1)) - K_1;
62     f(2) = ((e(2))*(e(1)+e(2)))/(e(1)-e(2)) - K_2;
63     funcprot(0);
64 endfunction
65
66 // Initial guess:

```

```

67 e = [0.8 0.1];
68 y = fsolve(e,F);
69 e_1 = y(1);
70 e_2 = y(2);
71
72 // So, concentration of the various species in
    equilibrium is given as
73 m_H2SO4 = 1-e_1;// [molal]
74 m_HSO4 = e_1 - e_2;//[molal]
75 m_SO4 = e_2;//[molal]
76 m_H = e_1 + e_2;//[molal]
77
78 printf(" The equilibrium concentration of H2SO4 in
    terms of molality is %f molal\n",m_H2SO4);
79 printf(" The equilibrium concentration of HSO4- in
    terms of molality is %f molal\n",m_HSO4);
80 printf(" The equilibrium concentration of SO4— in
    terms of molality is %f molal\n",m_SO4);
81 printf(" The equilibrium concentration of H+ in
    terms of molality is %f molal",m_H);

```

Scilab code Exa 13.3 Simultaneous reactions

```

1 clear;
2 //clc();
3
4 // Example 13.3
5 // Page: 352
6 printf("Example-13.3 Page no.-352\n\n");
7
8 //***Data***/
9 P = 10;//[MPa] given pressure
10 T = 250;//[C] Temperature
11 // Let the total number of moles in the feed be one,
    then

```

```

12 n_T_0 = 1; // [mol]
13 n_CO = 0.15; // [mol]
14 n_CO2 = 0.08; // [mol]
15 n_H2 = 0.74; // [mol]
16 n_CH4 = 0.03; // [mol]
17
18 // The two simultaneous reactions taking place are
19 // CO + 2*H2 = CH3OH
20 // CO2 + H2 = CO + H2O
21
22 // Let us denote the first reaction by 1 and the
    second reaction by 2
23 // and  $K_i = (K/K_v) * [P/(1 \text{ atm})]^{(-\text{summation}(v_i))}$ 
24 // and that  $\text{summation}(v_i) = V_i$ 
25
26 // Then from the table 13.C (page 353) as reported
    in the book, we have
27 V_1 = -2;
28 V_2 = 0;
29 K_1 = 49.9; // For the first reaction
30 K_2 = 0.032; // For the second reaction
31
32 // Now let  $v_i$  denotes the stoichiometric
    coefficient of species 'i', then
33 v_CO_1 = -1;
34 v_H2_1 = -2;
35 v_CH3OH_1 = +1;
36 v_CO2_2 = -1;
37 v_H2_2 = -1;
38 v_CO_2 = +1;
39 v_H2O_2 = +1;
40
41 // Let  $e_1$  = the moles of CO reacted in reaction 1
    and  $e_2$  = the moles of CO2 reacted in reaction 2.
42 // Now mol fractions of each of the species in the
    equilibrium is
43 //  $y_{\text{CO}} = (n_{\text{CO}} + v_{\text{CO}_1} * e_1 + v_{\text{CO}_2} * e_2) / (n_{\text{T}_0} + e_1 * V_1 + e_2 * V_2) = (0.15 - 1 * e_1 + 1 * e_2) / (1 + e_1 * (-2) + e_2$ 

```

```

44      *(0)) = (0.15 - e_1 + e_2)/(1 - 2*e_1)
45 // similarly
46 // 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)
47
48 // 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)
49
50 // 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)
51
52 // 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)
53
54 // Now putting the values in the expression of the
      equilibrium constant of the reactions , for the
      reaction 1 we have
55
56 // 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))
57
58 // 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))))
59
60 // e = [e_1 e_2]
61 // Solving the two given simultaneous equations ,we
      have
62 function [f]=F(e)
63     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;
64     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;

```



```

65     funcprot(0);
66 endfunction
67
68 // Initial guess:
69 e = [0.109 0];
70 y = fsolve(e,F);
71 e_1 = y(1);
72 e_2 = y(2);
73
74 // So, percent conversion of CO2 is given as
75 // (moles of CO2 reacted)/(moles of CO2 fed) i.e.
76 c_CO2 = e_2/(n_CO2)*100;
77 // Number of moles of CO Formed by the second
78 // reaction is 0.032
79 // So, percent conversion of CO is given as
80 c_CO = e_1/(n_CO + 0.032)*100;
81 printf(" Percent conversion of CO is %f%%\n",c_CO);
82 printf(" Percent conversion of CO2 is %f%%",c_CO2);

```

Scilab code Exa 13.4 Solubility product

```

1 clear;
2 //clc();
3
4 // Example 13.4
5 // Page: 354
6 printf("Example-13.4 Page no.-354\n\n");
7
8 //***Data**//
9 T = 273.15+25; // [K] Temperature
10 R = 8.314; // [J/(mol*K)] universal gas constant
11
12 // Solubility of AgCl in water follows
13 // AgCl = Ag+ + Cl-

```

```

14 // From the Table A.8, free energy of above species
    are
15 g_0_Ag = 77.12*1000; //[J/mol]
16 g_0_Cl = -131.26*1000; //[J/mol]
17 g_0_AgCl = -109.8*1000; //[J/mol]
18
19 // Free energy change of the reacton is given by
20 delta_g_0 = g_0_Ag + g_0_Cl - g_0_AgCl; //[J/mol]
21
22 // Now equilibrium constant of the reaction is given
    by
23 K = exp((-delta_g_0)/(R*T));
24
25 // In terms of activity of the components,
    equilibrium constant is
26 // K = [[Ag+]/(1 molal)*[Cl-]/(1 molal)]/[a_AgCl]
27
28 // 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
29 a_AgCl = 1.00;
30 // hence
31 // [[Ag+]/(1 molal)*[Cl-]/(1 molal)]= K = K_sp ,
    solubility product
32 printf("The amount of solid dissolved in terms of
    solubility product is %0.2e",K);

```

Scilab code Exa 13.5 Gas liquid reactions

```

1 clear;
2 //clc();
3
4 // Example 13.5
5 // Page: 357
6 printf("Example-13.5 Page no.-357\n\n");

```

```

7
8 //***Data***//
9 T = 273.15+25; //[K] Given temperature of air
10 P = 1; //[atm] Pressure of the air
11 y_CO2 = 350*10^(-6); // Amount of CO2 present in air
    at the given condition
12 R = 8.314; //[J/(mol*K)]
13
14 // At equilibrium there are two ionisation reactions
    takin place sequently
15 // First ionisation reaction is
16 //  $\text{H}_2\text{CO}_3 = \text{H}^+ + \text{HCO}_3^-$ 
17 // Free energy of the species of the above reaction
    is
18 g_0_H2CO3 = -623.1*1000; //[J/mol]
19 g_0_H = 0; //[J/mol]
20 g_0_HCO3 = -586.85*1000; //[J/mol]
21
22 // So free energy change of the reaction is given by
23 delta_g_0 = g_0_H + g_0_HCO3 - g_0_H2CO3; //[J/mol]
24 // Equilibrium constant of the reaction is given by
25 K_1 = exp((-delta_g_0)/(R*T));
26
27 // And the second one is
28 //  $\text{HCO}_3^- = \text{H}^+ + \text{CO}_3^{(-2)}$ 
29 // Free energy of the species of the second reacion
    are
30 g_0_CO3 = -527.89*1000; //[J/mol]
31
32 // Free energy change of the second reaction is
33 delta_g_1 = g_0_H + g_0_CO3 - g_0_HCO3; //[J/mol]
34 // So equilibrium constant of the reaction is given
    by
35 K_2 = exp((-delta_g_1)/(R*T));
36
37 // Now, writing the expression of the equilibrium
    constant of the first reaction , we have
38 //  $K_1 = ([\text{HCO}_3^-][\text{H}^+]) / [\text{H}_2\text{CO}_3]$ 

```

```

39 // and that for the second reaction
40 //  $K_2 = ([CO_3] * [H^+]) / [CO_3^-]$ 
41
42 // From the Table A.3 (page 419) as reported in the
    book, Henry's law constant is
43  $H = 1480$ ; // [atm]
44
45 // From Henry's law
46 //  $P * y_{CO_2} = x_{CO_2} * H$  , so
47  $x_{CO_2} = P * y_{CO_2} / H$ ;
48
49 // This gives the mol fracion. The dissociation
    constant are based on molaities a standard states
    , so
50 // Molality of the CO2 in the solution is
51 //  $m_{CO_2} = x_{CO_2} * n_{water}$  , where 'n_water' is
    number of moles of water in 1000g of water, so
52  $n_{water} = 1000 / 18$ ; // [mol]
53  $m_{CO_2} = x_{CO_2} * n_{water}$ ; // [molal]
54
55 // Then we assume that almost all the H+ comes from
    the dissociation of dissolved CO2, so
56 //  $m_{HCO_3} = m_H$ , i.e. molality of bicarbonate is
    approximately equal to molality of hydrogen ion
    in the solution and hence
57  $m_{HCO_3} = \text{sqrt}(K_1 * m_{CO_2})$ ; // [molal]
58  $m_H = m_{HCO_3}$ ; // [molal]
59
60 // Then we compute
61  $m_{CO_3} = K_2 * (m_{HCO_3} / m_H)$ ; // [molal]
62
63 printf(" Amount of the CO2 dissolved in water in
    equilibrium with air is \t\t\t%0.2e molal\n",
    m_CO2);
64 printf(" Conentration of HCO3 ion and hydrogen ion H
    - in solution in equilibrium with air is \t\t\t%0.2e
    molal\n", m_HCO3);
65 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

```
1 clear;
2 //clc();
3
4 // Example 13.6
5 // Page: 358
6 printf("Example-13.6 Page no.-358\n\n");
7
8 /***Data***/
9 // All the data are taken from the previous example
10 // 13.5
11 m_H = 10^(-10);//[molal] molality of hydrogen ion
12 K_1 = 4.5*10^(-7);
13 K_2 = 4.7*10^(-11);
14 // Our Henry's law calculations are independent of
15 // the subsequent fate of the dissolved CO2.
16 // The concentration of dissolved CO2 in equilibrium
17 // with atmosphere is
18 m_CO2 = 1.32*10^(-5);//[molal] from previous example
19 // It is independent of that acidity or basicity of
20 // the water, and hence
21 m_HCO3 = K_1*(m_CO2/m_H);//[molal]
22 // and
23 m_CO3 = K_2*(m_HCO3/m_H);//[molal]
24 printf(" Amount of the CO2 dissolved in water in
25 equilibrium with air is \t\t%0.2e molal\n",m_CO2
26 );
27 printf(" Concentration of HCO3 ion in solution in
28 equilibrium with air is \t %0.2e molal\n",m_HCO3)
```

```

;
24 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

```

1 clear;
2 //clc();
3
4 // Example 13.7
5 // Page: 362
6 printf("Example-13.7 Page no.-362\n\n");
7
8 /***Data***/
9 T = 298.15;//[K] Temperature
10 F = 96500;//[coulomb)/(mole*electrons)] faraday
    constant
11
12 // The reaction is given as
13 // Al2O3 + 1.5C = 2Al + 1.5CO2
14
15 // No of the electron being exchanged are
16 n_e = 6;//[electron]
17 // All the reactants and products enter or leave the
    reactor as pure species in their standard states
    , so
18 // delta_g_0 = delta_g_1 and E = E_0
19 // Free energy of the species in the above equation
    as reported in the Table A.8 in the book is
20 g_0_CO2 = -394.4*1000;//[J/mol]
21 g_0_Al = 0;//[J/mol]
22 g_0_C = 0;//[J/mol]
23 g_0_Al2O3 = -1582.3*1000;//[J/mol]
24

```

```

25 // Free energy change of the reaction is
26 delta_g_0 = 1.5*g_0_CO2 + 2*g_0_Al - 1.5*g_0_C -
    g_0_Al2O3; //[J/mol]
27
28 // So, standard state cell voltage is
29 E_0 = (-delta_g_0)/(n_e*F); //[V]
30
31 printf("Standard state cell voltage for the
    production of aluminium is %f Volt",E_0);

```

Scilab code Exa 13.8 Electrochemical reactions

```

1 clear;
2 //clc();
3
4 // Example 13.8
5 // Page: 362
6 printf("Example-13.8 Page no.-362\n\n");
7
8 //***Data***/
9 T = 298.15; //[K] Temperature
10 F = 96500; //[(coulomb)/(mole*electrons)] faraday
    constant
11
12 // The reaction taking place between lithium and
    fluorine is
13 // Li + F = LiF
14
15 // From Table A.8 we find that
16 delta_g_0 = -587.7*1000; //[J/mol]
17 // We also know that
18 n_e = 1; //[electron] no of electron transferred
19 // That is because the valence Li and F change by 1,
    so one electron is transferred per molecule of
    LiF, thus

```

```

20 E_298_0 = (-delta_g_0)/(n_e*F); // [V]
21
22 printf("The reversible voltage for given
        electrochemical device is %f Volt",E_298_0);

```

Scilab code Exa 13.9 Electrochemical reactions

```

1 clear;
2 // clc ();
3
4 // Example 13.9
5 // Page: 363
6 printf("Example-13.9 Page no.-363\n\n");
7
8 //***Data***/
9 T = 298.15; // [K] Temperature
10 P_0 = 1; // [atm]
11 P = 100; // [atm]
12 E_0 = -1.229; // [V]
13 F = 96500; // [(coulomb)/(mole*electrons)] faraday
    constant
14 R = 8.314; // [J/(mol*K)] universal gas constant
15
16 // The reaction is
17 // H2O(l) = H2(g) + 1/2O2(g)
18 // number of the valence electrons transferred in
    this reaction is
19 n_e = 2; // [(mole electrons)/mole]
20
21 // Gibb's free energy is given by
22 // g = g_0 + integrate(dg/dP)*dP, at constant
    temperature with integration limit P_0 and P
23 // or
24 // g = g_0 + integrate(v_T)*dP
25 // In the rightmost term we replace v_T by (R*T)/P,

```



```

    which is correct only for ideal gases , so
26 // g = g_0 + (R*T)*log(P/P_0)
27
28 // According to the assumption ,we can ignore the
    change in Gibb's free energy with pressure of the
    liquid water , so that
29 // delta_g = delta_g_0 + 1.5*(R*T)*log(P/P_0)
30
31 // and
32 // E = (-delta_g)/(n_e*F) = -(delta_g_0 + 1.5*(R*T)*
    log(P/P_0))/(n_e*F)
33 // So equilibrium cell voltage is given as
34 E = E_0 - 1.5*(R*T)*log(P/P_0)/(n_e*F);
35
36 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

```

1 clear;
2 //clc();
3
4 // Example 13.10
5 // Page: 365
6 printf("Example-13.10 Page no.-365\n\n");
7
8 //***Data***/
9 T = 273.15+25; // [K] Temperature
10 P = 11.38/760; // [atm] Pressure
11 R = 0.08206; // [(L*atm)/(mol*K)] Gas constant
12 v = 0.6525/0.04346; // [L/g] Specific volume
13 M = 60.05; // [g/mol] Molecular weight of HAc in the
    monomer form
14 // So the specific volume in [L/mol] is

```

```

15 V = v*M; //[L/mol]
16
17 // Compressibility factor is give by
18 z = (P*V)/(R*T);
19
20 printf("The value of the compressibility factor for
    HAc at given condition is %f",z);

```

Scilab code Exa 13.11 Dimerization

```

1 clear;
2 //clc();
3
4 // Example 13.11
5 // Page: 366
6 printf("Example-13.11 Page no.-366\n\n");
7
8 //***Data***/
9 T = 273.15+25; //[K] Temperature
10 P = 11.38; //[torr] Pressure
11
12 // Formation of the dimer from monomer in the gas
    phase follows the reaction
13 // 2*HAc = (HAc)2
14
15 // From the equation 13.BF(page 366) given in the
    book
16 //  $K = (P*y_{HAc_2}) / (P*y_{HAc})^2$  , where 'y_HAc_2'
    is mol fraction of dimer and 'y_HAc' is mol
    fraction of monomer
17 // and
18 //  $\log_{10}(K) = -10.4184 + 3164/T$  , so
19 K = 10^(-10.4184 + 3164/T); //[1/torr]
20
21 // Thus

```

```

22 //  $y_{\text{HAc}_2} = K \cdot (P \cdot y_{\text{HAc}})^2 / P$ 
23 // Since ,  $(y_{\text{HAc}} + y_{\text{HAc}_2}) = 1$ 
24 //  $y_{\text{HAc}_2} = K \cdot (P \cdot (1 - y_{\text{HAc}}))^2 / P$ 
25
26 // Solving for  $y_{\text{HAc}_2}$ 
27 deff(' [y]=f(y_HAc_2)', 'y = K*(P*(1-y_HAc_2))^2/P-
      y_HAc_2');
28 y_HAc_2 = fsolve(0,f);
29 // So
30 y_HAc = 1-y_HAc_2;
31
32 printf("Mole fraction of the monomer in the vapour
      phase is %f\n",y_HAc);
33 printf("Mole fraction of the dimer in the vapour
      phase is %f",y_HAc_2);

```

Scilab code Exa 13.12 Dimerization

```

1 clear;
2 // clc();
3
4 // Example 13.12
5 // Page: 367
6 printf("Example-13.12 Page no.-367\n\n");
7
8 //***Data***/
9 // Getting the data from the example 13.10
10 T = 273.15+25; // [K] Temperature
11 P = 11.38/760; // [atm] Pressure
12 R = 0.08206; // [(L*atm)/(mol*K)] Gas constant
13 v = 0.6525/0.04346; // [L/g] Specific volume
14
15 // Now from the previous example ie example 13.11
      the mole fractions of the monomer and dimer in
      the gas phase is

```

```
16 y_HAc = 0.211; // monomer
17 y_HAc_2 = 0.789; // dimer
18
19 // Molecular weights of the monomer and dimer forms
   are
20 M_HAc = 60.05; // [g/mol] monomer
21 M_HAc_2 = 120.10; // [g/mol] dimer
22
23 // Now average molecular weight of the mixture is
24 M_avg = M_HAc*y_HAc + M_HAc_2*y_HAc_2; // [g/mol]
25
26 // So specific volume in [L/mol] is
27 V = v*M_avg; // [L/mol]
28
29 // Now compressibility factor is
30 z = (P*V)/(R*T);
31
32 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

```
1 clear;
2 //clc ();
3
4 // Example 14.1
5 // Page: 379
6 printf("Example -14.1 Page no.-379\n\n");
7
8 //***Data***//
9 T = 300; //[K] Temperature of the natural gas well
10 R = 8.314; //[J/(mol*K)] universal gas constant
11 z_1 = 0; //[m]
12 // At the surface of the well mole fraction of the
    components are
13 y_methane_surf = 85/100; //[mol%]
14 y_ethane_surf = 10/100; //[mol%]
```

```

15 y_propane_surf = 5/100; //[mol%]
16 P = 2; //[MPa] Total equilibrium pressure
17 z_2 = 1000; //[m] Depth of the well
18
19 // Molecular weights of the components are
20 M_methane = 16/1000; //[kg/mol]
21 M_ethane = 30/1000; //[kg/mol]
22 M_propane = 44/1000; //[kg/mol]
23
24 // Now, we have the relation between the fugacities
    of a component at z_1 and z_2 as
25 //  $f_{i_2}/f_{i_1} = \exp((-M_i * g * (z_2 - z_1)) / (R * T))$  ,
    where g is gravitational accelaration and its
    value is
26 g = 9.81; //[m/s^(2)]
27
28 // Fugacities of the various components at the
    surface i.e. at z = z_1 is
29 f_methane_1 = y_methane_surf * P; //[MPa]
30 f_ethane_1 = y_ethane_surf * P; //[MPa]
31 f_propane_1 = y_propane_surf * P; //[MPa]
32
33 // Now, fugacities at z = z_2 are
34 f_methane_2 = f_methane_1 * exp((-M_methane * g * (z_1 - z_2
    )) / (R * T));; //[MPa]
35 f_ethane_2 = f_ethane_1 * exp((-M_ethane * g * (z_1 - z_2))
    / (R * T)); //[MPa]
36 f_propane_2 = f_propane_1 * exp((-M_propane * g * (z_1 - z_2
    )) / (R * T)); //[MPa]
37
38 // Let at z = z_1 total pressure of the gases are
    P_2
39 // Then, fugacities of the ith component is also
    given as
40 //  $f_{i_2} = y_{i_2} * P_2$ 
41 // Writing the expression for all the component ad
    adding them we get
42 //  $(f_{methane_2} + f_{ethane_2} + f_{propane_2}) =$ 

```

```

        y_methane_2*P_2 + y_ethane_2*P_2 + y_propane_2*
        P_2
43 // or
44 // (f_methane_2 + f_ethane_2 + f_propane_2 ) = P_2*(
        y_methane_2 + y_ethane_2 + y_propane_2)
45 // and
46 // (y_methane_2 + y_ethane_2 + y_propane_2) = 1    ,
        so
47 P_2 = (f_methane_2 + f_ethane_2 + f_propane_2 );//[
        MPa]
48
49 // Now the mole fractions of the components are
50 //  $y_{i-2} = f_{i-2}/P_2$  , so
51 y_methane_2 = f_methane_2/P_2;
52 y_ethane_2 = f_ethane_2/P_2;
53 y_propane_2 = f_propane_2/P_2;
54
55 printf("The mol fraction of the methane at the depth
        1000m is %f\n",y_methane_2);
56 printf("The mol fraction of the ethane at the depth
        1000m is %f\n",y_ethane_2);
57 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

```

1 clear;
2 //clc ();
3
4 // Example 14.2
5 // Page: 380
6 printf("Example-14.2 Page no.-380\n\n");
7
8 //***Data***//
9 T = 288;//[K] Atmospheric temperature

```

```

10 R = 8.314; // [J/(mol*K)] universal gas constant
11 z_2 = 15000; // [m] Thickness of the atmosphere
12 z_1 = 0; // [m] Surface
13 // At the surface, the mole fraction of nitrogen and
    oxygen are
14 y_N2_1 = 0.79;
15 y_O2_1 = 0.21;
16 M_N2 = 28/1000; // [kg/mol]
17 M_O2 = 32/1000; // [kg/mol]
18
19 // For an ideal solution of ideal gases with only
    two species, we have
20 //  $y_{i-2}/y_{i-1} = 1/(y_{i-1} + y_{j-1}/a)$ , and
21 //  $a = \exp(-(M_i - M_j) * g * (z_2 - z_1) / (R * T))$ 
22 // where 'g' is acceleration due to gravity and its
    value is
23 g = 9.81; // [m/s^(2)]
24
25 // So
26 a = exp(-(M_N2 - M_O2) * g * (z_2 - z_1) / (R * T));
27 // and
28 yi2_by_yi1 = 1/(y_N2_1 + y_O2_1/a);
29
30 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

```

1 clear;
2 //clc();
3
4 // Example 14.3
5 // Page: 381

```



```

6 printf("Example-14.3 Page no.-381\n\n");
7
8 /***Data***/
9 // For this problem all the data are same as in
   previous Example 14.2 except z_1 and z_2
10 // So
11 T = 288; //[K] Atmospheric temperature
12 R = 8.314; //[J/(mol*K)] Universal gas constant
13 z_2 = 10; //[m] Height of the reactor
14 z_1 = 0; //[m] Surface
15 g = 9.81; //[m/s^(2)] Acceleration due to gravity
16 // At z = z_1, the mole fraction of nitrogen and
   oxygen are
17 y_N2_1 = 0.79;
18 y_O2_1 = 0.21;
19 M_N2 = 28/1000; //[kg/mol]
20 M_O2 = 32/1000; //[kg/mol]
21
22 // So
23 a = exp(-(M_N2-M_O2)*g*(z_2-z_1)/(R*T));
24 // and
25 yi2_by_yi1 = 1/(y_N2_1 + y_O2_1/a);
26
27 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

```

1 clear;
2 //clc();
3
4 // Example 14.4
5 // Page: 382

```

```

6 printf("Example-14.4 Page no.-382\n\n");
7
8 /***Data***/
9 T = 300; //[K] Temperature of the centrifuge
10 R = 8.314; //[J/(mol*K)] Universal gas constant
11 // Mole fractions of the two components are
12 y_UF6_238_1 = 0.993; // Mole fraction of UF6 with
    238 isotope of uranium in feed
13 y_UF6_235_1 = 0.007; // Mole fraction of UF6 with 235
    isotope of uranium in feed
14 M_UF6_238 = 352/1000; //[kg/mol] Molecular weight of
    UF6 with 238 isotope of uranium
15 M_UF6_235 = 349/1000; //[kg/mol] Molecular weight of
    UF6 with 235 isotope of uranium
16 r_in = 2/100; //[m] Interanal raddi of the centrifuge
17 r_out = 10/100; //[m] outer raddi of the centrifuge
18 f = 800; //[revolution/second] Rotational frequency
    of centrifuge
19
20 // Here the accelaration will come due to
    centrifugal force and is
21 //  $g = w^2 * r$  , where 'w' is angular speed and its
    value is  $w = 2 * \text{pie} * f$  and 'r' is radius
22 // But in the present case 'r' is varies as we move
    away from the axis of centrifuge
23 // After making integration by taking small elements
    at the distance 'r' we find the expression
24 a = exp((M_UF6_235 - M_UF6_238) * (2 * 3.141592 * f)^2 * (
    r_out^2 - r_in^2) / (2 * R * T));
25
26 // Now Let the ratio  $y_{i-2} / y_{i-1} = A$ 
27 // Then we have
28 A = 1 / (y_UF6_235_1 + y_UF6_238_1 / a);
29
30 // Now say  $y_{i-1} / y_{i-2} = 1 / A = B$  , then
31 B = 1 / A;
32
33 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

```
1 clear;
2 //clc();
3
4 // Example 14.5
5 // Page: 384
6 printf("Example-14.5 Page no.-384\n\n");
7
8 /***Data***/
9
10 // We have two phase system in this problem in which
11 // phase 1 is seawater and phase 2 is freshwater
12 // Seawater contains mostly NaCl, Na2SO4, MgCl2, KCl
13 // and if they completely ionised then
14 x_water_1 = 0.98; // mole fraction of water in phase
15 // 1 i.e. in seawater
16 x_water_2 = 1; // mole fraction of water in the phase
17 // 2 i.e. in water
18 R = 10.73; // [(psi*ft^(3))/(lbmol*R)] Universal gas
19 // constant
20 T = 500; // [R] temperature
21 v_water_1 = 18/62.4 // [ft^(3)/(lbmol)]
22
23 // The effect of the pressure on the fugacity of the
24 // liquid is given as
25 //  $f_i = (x_i * Y_i * p) * \exp(\int (v / (R * T)) dP)$  with
26 // integration limit from pure liquid pressure to
27 // solution liquid pressure
28
29 // Writing this equation twice, once for pure water
30 // and once for the water in the ocean water, and
```

```

equating the fugacities , we get
22 // ((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
23
24 // 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
25 // So that Y_i is certain to be practically unity .
26
27 // The partial molal volume of water in pure water
    is practically the same as that in dilute
    solutions ,
28 // Taking the logarithm of both sides and solving ,
    we get
29
30 // -log(x_water_1) = integrate(v_water_1/(R*T)dP)
31 // Integrating with the limit P_purewater and
    P_seawater we have
32 // -log(x_water_1) = (v_water_1/(R*T))*( P_seawater
    - P_purewater )
33 // ( P_seawater - P_purewater ) = delta_P
34 // So
35 delta_P = -(R*T)*log(x_water_1)/v_water_1;//[psi]
36 printf("The pressure difference between the two
    phases is %0.1f psi",delta_P)

```

Scilab code Exa 14.6 Pressure difference across a droplet

```

1 clear;
2 //clc();
3
4 // Example 14.6
5 // Page: 386

```

```

6 printf("Example-14.6 Page no.-386\n\n");
7
8 /***Data***/
9 T = 100; //[C] Temperature of the outside
10 P_outside = 1; //[atm]
11 // At 100 C, the surface tension between steam and
    water is
12 T = 0.05892; //[N/m] From metric steam table (7, page
    267)
13
14 // Pressure difference between inside and outside of
    a drop is given by the expression
15 // (P_inside - P_outside) = (4*T)/d_i
16
17 // Let (P_inside - P_outside) = delta_P , so
18 //delta_P = (4*T)/d_i
19 // For the drop of diameter
20 d_1 = 0.001; //[m]
21 // So
22 delta_P_1 = (4*T)/d_1; //[Pa]
23
24 // Which is certainly negligible
25 // If we reduce the diameter to
26 d_2 = 10^(-6); //[m]
27
28 // So
29 delta_P_2 = (4*T)/d_2; //[Pa]
30
31 // If we reduce it to diameter that is smallest
    sized drop likely to exist
32 d_3 = 0.01*10^(-6) //[m]
33 // Then the calculated pressure difference is
34 delta_P_3 = (4*T)/d_3; //[Pa]
35
36 printf("Pressure difference with the change in
    radius of the drop of the water is given as in
    the following table\n\n");
37 printf("          Diameter of the droplet (d_i)(in

```

```

        meter)           Pressure difference (
        P_inside - P_outside )(in atm)\n");
38 printf("                %0.2e
        %0.2e\n",d_1,delta_P_1);
39 printf("                %0.2e
        %0.2e\n",d_2,delta_P_2);
40 printf("                %0.2e
        %0.2e\n",d_3,delta_P_3);

```

Scilab code Exa 14.7 Equilibrium with surface tension

```

1 clear;
2 //clc();
3
4 // Example 14.7
5 // Page: 387
6 printf("Example-14.7 Page no.-387\n\n");
7
8 //***Data***/
9 P_NBP = 1; //[atm]
10 Temp =273.15+100; //[C] Temperature
11 D = 0.01*10^(-6); //[m] Diameter of the condensation
    nuclei( due to impurity )
12 T = 0.05892; //[N/m] Surface tension between water
    drops and gas
13 R = 8.314; //[J/(mol*K)]
14
15 // At equilibrium the Gibb's free energy per pound
    will be the same inside and outside the drops.
16 // From the previous example 14.6, the pressure
    difference inside and outside of the drop is
17 // delta_P = ( P_inside-P_outside) = 4*T/D = 233 atm

```

```

    = 235.7 bar
18
19 // Taking the Gibb's free energy at the normal
    boiling point as g_NBP we have
20 // g_small_drop_equilibrium = g_NBP + integrate(
    v_water_gas)dP , with integration limits P_NBP
    and P_gas
21 // also
22 // g_small_drop_equilibrium = g_NBP + integrate(
    v_water_liquid)dP , with integration limits
    P_NBP and (P_gas + 4*T/D)
23 // and
24 v_water_liquid = 1/958.39*0.018; // [m^(3)/mol]
25
26 // 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
27 // then we can perform the integrations and cancel
    the g_NBP terms, finding the Kelvin equation
28
29 // (R*Temp)*log(P_gas/P_NBP) = v_water_liquid*(P_gas
    + 4*T/D - P_NBP)
30 // For very small drops
31 // (P_gas - P_NBP) << 4*T/D
32 // So that we can write it approximately as
33
34 // P_gas/P_NBP = exp(v_water_liquid*(4*T/D)/(R*Temp)
    ) = I
35 // so
36 I = exp(v_water_liquid*(4*T/D)/(R*Temp));
37
38 // Subtracting 1 from both sides in the above
    equation we have
39 // (P_gas-P_NBP)/P_NBP = I-1
40 // So
41 P_gas_minus_P_NBP = (I-1)*P_NBP; // [atm]
42 // Changing into the bar we have

```

```

43 delta_P = P_gas_minus_P_NBP*1.01325; //[bar]
44
45 // Now changing the unit to psi we have
46 delta_P_1 = delta_P*100*0.1450377; //[psi]
47
48 printf("The equilibrium pressure at which the steam
        begin to condense 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

```

1  clear;
2  //clc();
3
4  // Example 14.8
5  // Page: 388
6  printf("Example-14.8 Page no.-388\n\n");
7
8  /***Data***/
9  Temp = 273.15+100; //[K] Temperature of the water
        drop
10 R = 8.314; //[J/(mol*K)] Universal gas constant
11 D = 0.01*10^(-6); //[m] Diameter of the water drop
12 P_g = 0.15; //[bar] guage pressure
13 T = 0.05892; //[N/m] Surface tension between water
        drop and gas
14
15 // The calculation of the pressure difference from
        inside to outside is the same as done in the
        example 14.7
16
17 // The specific Gibb's free energy of the liquid is
        thus given as
18 // (g_water_liquid - g_NBP) = integrate(

```



```

    v_water_liquid)dP , with integration limits
    P_NBP and (P_gas + 4*T/D)
19 // Where
20 v_water_liquid = 0.018/958.39; //[m^(3)/mol]
21 P_NBP = 1.013; //[bar]
22 P_gas = 1.013+0.15; //[bar]
23
24 // Say
25 P_1 = P_gas + 4*T/D; //[bar]
26 // and (g_water_liquid - g_NBP) = delta_g_1
27 // So
28 delta_g_1 = integrate('v_water_liquid*P^(0)', 'P',
    P_NBP, P_1); //[J/mol]
29
30 // and for the gas, again using equation for Gibb's
    free energy, we have
31 // (g_water_liquid - g_NBP) = integrate(v_water_gas)
    dP , with integration limits P_NBP and P_gas
32 // Here assuming that the vapour follows the ideal
    gas law we have
33 // v_water_gas = (R*Temp/P)
34 // and also let (g_water_liquid - g_NBP) = delta_g_2
35 // so
36 delta_g_2 = integrate('(R*Temp)/P', 'P', P_NBP, P_gas);
37
38 // Now
39 // (g_water_liquid - g_water_gas) = (g_water_liquid
    - g_NBP) - (g_water_gas - g_NBP) = delta_g
40 // So
41 delta_g = (delta_g_1 - delta_g_2);
42
43 // We have got the value of the delta_g positive, so
44
45 printf("The liquid can lower its free energy %0.2f J
    /mol by Changing to gas,\n", delta_g);
46 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

```
1 clear;
2 // clc ();
3
4 // Example 14.9
5 // Page: 390
6 printf("Example-14.9 Page no.-390\n\n");
7
8 //***Data***/
9 Temp = 904.7; //[R] Temperature of the pure liquid
   water
10 P_NBP = 400; //[psia] Saturation pressure of the pure
   liquid water at the given temperature
11 T = 1.76*10^(-4); //[lbf/inch] Surface tension of
   water
12 R = 10.73; //[(psi*ft^(3))/(lbmol*R)]
13
14 // In this problem the gas is inside the bubble, at
   a pressure much higher than that of the
   surrounding liquid.
15 // 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
   .
16 // Thus we have
17 // g_small_drop_equilibrium = g_NBP + integrate(
   v_water_liquid)dP , with integration limits
   P_NBP and P_liquid
18 // also
19 // g_small_drop_equilibrium = g_NBP + integrate(
   v_water_gas)dP , with integration limits P_NBP
   and (P_liquid+4*T/D)
20 // where
```

```

21 v_water_liquid = 18*0.01934; //[ft^(3)/lbmol]
22 D = 10^(-5); //[inch]
23
24 // so
25 // g_NBP + integrate(v_water_liquid)dP = g_NBP +
    integrate(v_water_gas)dP
26
27 // Here we assume that the liquid has practically
    constant density and that the gas behaves as an
    ideal gas and find
28 // (R*Temp)*log((P_liquid+4*T/D)/P_NBP) =
    v_water_liquid*(P_liquid - P_NBP)
29 // let P_liquid = p
30
31 // We will solve the above equation for p
32 deff(' [y]=f(p) ', 'y = v_water_liquid*(p - P_NBP) - (R*
    Temp)*log((p+4*T/D)/P_NBP) ');
33 P_liquid = fsolve(300,f); //[psia]
34
35 // At this external pressure the pressure inside the
    bubble is
36 P_inside = P_liquid + 4*T/D; //[psia]
37
38 printf("The liquid pressure at which these boiling
    nuclei will begin to grow and initiate boiling is
    %0.1f psia\n",P_liquid);
39 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

```
1 clear;
2 // clc ();
3
4 // Example 15.1
5 // Page: 398
6 printf("Example -15.1 Page no.-398\n\n");
7
8 /***Data***/
9 // This is a theoretical question.
10 printf("This is a theoretical 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

```
1 clear;
2 // clc ();
3
```

```

4 // Example 15.2
5 // Page: 401
6 printf("Example-15.2 Page no.-401\n\n");
7
8 /***Data***/
9 // The system contains four species
10 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");
11
12 printf("      C + 0.5O2 = CO\n");
13 printf("      C + O2 = CO2\n");
14 printf("      CO + 0.5O2 = CO2\n");
15 printf("      CO2 + C = 2CO\n");
16
17 // Let we call these equations A, B, C and D
        respectively
18 // These relations are not independent.
19 // If we add A and C and cancel like terms, we
        obtain B.
20 // So, If we want independent chemical equilibria we
        must remove equation C
21
22 // Now, if we reverse the direction of B and add it
        to A, we see that D is also not independent.
23 // Thus, there are only two independent relations
        among these four species and
24 printf(" There are only two independent relations
        among these four species and\n");
25
26 //  $V = C + 2 - P$ 
27 // and we have
28  $V = 2$ ; // No of the variable
29  $P = 2$ ; // No of the phases
30 // So
31  $C = V + P - 2$ ;
32 printf("      C = V + P - 2\n");

```

```
33 printf("    C = 4 - 2 = 2\n");
34 printf(" Thus, this is a two-component system");
```

Scilab code Exa 15.3 Degree of freedom and number of components

```
1 clear;
2 //clc();
3
4 // Example 15.3
5 // Page: 402
6 printf("Example-15.3 Page no.-402\n\n");
7
8 /***Data***/
9 // This contains three species.
10 printf(" The three species in this system are H2, N2
    and NH3\n");
11 N = 3;
12 printf(" There is only one balanced chemical
    reaction among these species\n");
13 Q = 1
14
15 // 2NH3 = N2 + 3H2
16 C = N - Q;
17 printf("    C = N - Q = %0.0f\n\n",C);
18 // Now let us we made the system by starting with
    pure ammonia.
19 // Assuming that all the species are in the gas
    phase, ammonia dissociates in H2 and N2 in the
    ratio of 3:1.
20 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");
21
22 // We can write an equation among their mole
    fractions, viz;
```

```

23 // y_H2 = 3*y_N2
24 printf(" And the relation between their mole
      fraction is\n      y_H2 = 3*y_N2\n\n");
25
26 // We might modify the phase rule to put in another
      symbol for stoichiometric restrictions , but the
      common usage is to write that
27 // Components = species - (independent reactions) -
      (stoichiometric restriction)
28 // and stoichiometric restriction SR is
29 SR = 1;
30 // so
31 c = N-Q-SR;
32 printf(" We have the modified phase rule as\n
      Components = species - (independent reactions) -
      (stoichiometric restriction)\n")
33 printf("      C = N - Q - SR = %0.0 f" ,c);

```

Scilab code Exa 15.4 Number of components in a reactions

```

1 clear;
2 //clc();
3
4 // Example 15.4
5 // Page: 403
6 printf(" Example -15.4   Page no.-403\n\n");
7
8 //***Data***/
9 // We have been given the reaction
10 // CaCO3(s) = CaO(s) + CO2(g)
11
12 // Here we have three species and one balanced
      chemical reaction between them
13 // So
14 N = 3; // No of species

```

```

15 Q = 1; // no of reaction
16
17 // Since CO2 will mostly be in the gas phase and
    CaCO3 and CaO will each form separate solid
    phases ,
18 // there is no equation we can write among the mole
    fractions in any of the phases.
19 // Hence, there is no stoichiometric restriction i.e
    .
20 SR = 0
21 // and the number of the components is
22 C = N - Q - SR;
23
24 printf("Number of the components presents in the
    test tube are %0.0f",C);

```

Scilab code Exa 15.5 Degree of freedom and Phases

```

1 clear;
2 //clc();
3
4 // Example 15.5
5 // Page: 403
6 printf("Example-15.5 Page no.-403\n\n");
7
8 /**Data**/
9 // We have been given the reaction
10 // CaCO3(s) = CaO(s) + CO2(g)
11 // The CaCO3 and CaO form separate solid phases, so
    we have three phases, two solid and one gas.
12 // So
13 P = 3;
14 // This is a two component system, so
15 C = 2;
16

```



```

17 // From the phase rule
18 V = C + 2 - P;
19
20 // If there is only one degree of freedom, then the
    system should have a unique P-T curve.
21 // Reference [ 2, page 214 ] as reported in the book
    , shows the data to draw such a curve, which can
    be well represented by
22 // log(p/torr) = 23.6193 - 19827/T
23
24 printf(" The no. of phases present in the system are
    %0.0f \n",P);
25 printf(" Total no of degrees of freedom is %0.0f \n"
    ,V);
26 printf(" Since, there is only one degree of freedom,
    so the system has a unique P-T curve,\n");
27 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

```

1 clear;
2 //clc();
3
4 // Example 15.6
5 // Page: 404
6 printf("Example-15.6 Page no.-404\n\n");
7
8 //***Data***/
9 // The system consists of five species.
10 printf(" The five species present in the system are
    H2O, HCl, H+, OH- and Cl-. \n");
11 // So
12 N = 5; // Number of the species
13 printf(" Here we have two chemical relations:\n");

```

```

14 printf("      H2O = H+  +  OH-  \n");
15 printf("      HCl = H+  +  Cl-  \n");
16
17 // so
18 Q = 2; // No of the reactions
19
20 // 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 negative ions ,or
21 // [H+] = [OH-] + [Cl-]
22 // To maintain electroneutrality number of positive
      and negative ion should be same.
23 // Here [H+] stands for the molality of hydrogen ion
      . This is convertible to a relation among the 'mu
      's'; hence ,
24 // it is an additional restriction , so
25 SR = 1;
26 // So
27 // The number of components is
28 C = N - Q - SR;
29
30 printf(" Number of the components present in the
      system are \n      C = N - Q - SR = %0.0 f" ,C);

```

Scilab code Exa 15.7 Dependency of the number of components

```

1 clear;
2 //clc ();
3
4 // Example 15.7
5 // Page: 405
6 printf("Example -15.7  Page no.-405\n\n");
7
8 //***Data***/

```

```

 9 printf(" Our system consists of Au and H2O.\n");
10 // So
11 N = 2; // Number of the species
12 // If there is no chemical reaction , then
13 Q = 0;
14
15 //So
16 C = N - Q; // Number of the components
17 printf(" If no compound is formed , then number of
    the components in the system are \n    C = N - Q
    = 2 - 0 = %0.0f\n\n", C);
18
19 // However , if there is also a chemical reaction
20 // Au + H2O = AuH2O
21 // so
22 n = 3; // Number of the species
23 q = 1; // Number of the reactions
24
25 // Thus , we have
26 c = n - q; // Number of the components
27
28 printf(" If there is also a chemical reaction , viz.\n
    n    Au + H2O = AuH2O \n");
29 printf(" the number of the components in the system
    are\n    C = N - Q = %0.0f\n\n", c);
30 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 independent equations

```

1 clear;
2 //clc ();
3

```

```

4 // Example 15.8
5 // Page: 405
6 printf("Example-15.8 Page no.-405\n\n");
7
8 /**Data**/
9 // The species, we are given are CaCO3, CaO and CO2
10 // First we write the reaction for the formation of
    the above species from their elemental part.
11 // So, we have
12 // Ca + C + 1.5O2 = CaCO3
13 // Ca + 0.5O2 = CaO
14 // C + O2 = CO2
15
16 // We must eliminate Ca, C and O2 from these
    equations because they do not appear in the
    species list.
17
18 // Now, solving the 3rd equation for C and
    substituting in the first equation, we have
19 // Ca + CO2 - O2 + 1.5O2 = CaCO3
20 // Now, this equation to the equation second, we
    have
21 // CO2 = -CaO + CaCO3
22 // or
23 // CaCO3 = CO2 + CaO
24
25 printf(" There is only one balanced chemical
    reaction between the species on the species list ,
    viz.\n");
26 printf(" CaCO3 = CO2 + CaO ");

```

Scilab code Exa 15.9 Isothermal behaviour of the given reaction set

```

1 clear;
2 //clc();

```

```
3
4 // Example 15.9
5 // Page: 408
6 printf("Example-15.9 Page no.-408\n\n");
7
8 /**Data**/
9 printf("This is a theoretical question and there are
    no any numerical components. Refer to page no
    408 of the book.");
```
