

Scilab Textbook Companion for
Chemical Engineering Thermodynamics
by P. Ahuja¹

Created by
Shashwat
B.Tech
Chemical Engineering
IIT BHU Varanasi
College Teacher
Prakash Kotecha
Cross-Checked by

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: Chemical Engineering Thermodynamics

Author: P. Ahuja

Publisher: PHI Learning Private Limited, New Delhi

Edition: 1

Year: 2009

ISBN: 9788120336377

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	5
2 Equations of state	21
3 The First Law and Its Applications	51
4 The Second Law and Its Applications	89
5 Exergy	127
6 Chemical reactions	148
7 Thermodynamic property relations of pure substance	170
8 Thermodynamic Cycles	214
10 Residual Properties by Equations of State	233
11 Properties of a Component in a Mixture	302
12 Partial Molar Volume and Enthalpy from Experimental Data	314
13 Fugacity of a Component in a Mixture by Equations of State	328
14 Activity Coefficients Models for Liquid Mixtures	340
15 Vapour Liquid Equilibria	365

16 Other Phase Equilibria	415
17 Chemical Reactions Equilibria	434
18 Adiabatic Reaction Temperature	488

List of Scilab Codes

Exa 1.1	Calculation of pressure and heat transfer in piston cylinder assembly	5
Exa 1.2	Calculation of mass of air contained in a room	6
Exa 1.3	Determination of work done	6
Exa 1.4	Determination of wind energy per unit mass and diameter of the wind turbine	8
Exa 1.5	Determination of temperature	9
Exa 1.6	Calculation of dryness fraction of steam	10
Exa 1.7	Determination of pressure mass and volume	11
Exa 1.8	Determination of heat supplied	12
Exa 1.9	Calculation of saturation temperature	14
Exa 1.10	Calculation of pressure and temperature at triple point	15
Exa 1.11	Determination of value of R Cp0 and Cv0	15
Exa 1.12	Calculation of molar heat capacity	16
Exa 1.13	Determination of mean heat capacity	17
Exa 1.14	Calculation of enthalpy of water	18
Exa 2.1	Relations in virial coefficients	21
Exa 2.2	Determination of acentric factor	21
Exa 2.3	Calculation of acentric factor	22
Exa 2.4	Calculation of virial coefficients	23
Exa 2.5	Calculation of mass using virial equation of state	24
Exa 2.6	Calculation of molar volume	25
Exa 2.7	Calculation of molar volume and virial coefficients	26
Exa 2.8	Determination of second and third virial coefficients	29
Exa 2.9	Estimation of second virial coefficient	30
Exa 2.10	Estimation of molar volume	31
Exa 2.11	Calculation of maximum temperature	33
Exa 2.12	Calculation of pressure	37

Exa 2.13	Calculation of pressure	38
Exa 2.14	Determination of compressibility factor	40
Exa 2.15	Determination of molar volume	42
Exa 2.16	Calculation of volume	44
Exa 2.17	Estimation of compressibility factor	47
Exa 3.1	Calculation of temperature	51
Exa 3.2	Calculation of heat required	54
Exa 3.3	Calculation of temperature internal energy and enthalpy	55
Exa 3.4	Calculation of work done	57
Exa 3.5	Calculation of work done	59
Exa 3.6	Calculation of work done	61
Exa 3.7	Proving a mathematical relation	66
Exa 3.8	Calculation of work done	66
Exa 3.9	Calculation of final temperature	68
Exa 3.10	Finding expressions for temperature and pressure . . .	69
Exa 3.11	Calculation of final pressure	70
Exa 3.12	Calculation of slope and work done	71
Exa 3.13	Calculation of work done and final temperature	73
Exa 3.14	Calculation of power and discharge head	75
Exa 3.15	Calculation of discharge velocity	76
Exa 3.16	Calculation of change in enthalpy	78
Exa 3.17	Calculation of work done and change in enthalpy . . .	78
Exa 3.18	Calculation of work done per unit mass	80
Exa 3.19	Calculation of inlet and outlet velocity and power . . .	81
Exa 3.20	Proving a mathematical relation	83
Exa 3.21	Determination of equilibrium temperature	84
Exa 3.22	Determination of mass	86
Exa 4.1	Calculation of entropy change	89
Exa 4.2	Determination of whether the process is reversible or not	91
Exa 4.3	Calculation of final pressure temperature and increase in entropy	93
Exa 4.4	Calculation of final temperature heat transfer and change of entropy	95
Exa 4.5	Calculation of final temperature work and heat transfer	98
Exa 4.6	Calculation of final temperature and work done	101
Exa 4.7	Determination of index of isentropic expansion	102
Exa 4.8	Determination of entropy production	104
Exa 4.9	Determination of work required and exit temperature .	105

Exa 4.10	Determination of work required and exit temperature .	107
Exa 4.11	Determination of work required and exit temperature .	109
Exa 4.12	Determination of work required and discharge temperature	111
Exa 4.13	Determination of power output entropy and exit temperature	113
Exa 4.14	Calculation of work output per unit mass	116
Exa 4.15	Estimation of final velocity	118
Exa 4.16	Calculation of final velocity and increase in entropy . .	120
Exa 4.17	Calculation of work done and heat transfer	122
Exa 4.18	Calculation of air velocity and change in entropy . . .	124
Exa 5.1	Determination of fraction of the availability loss	127
Exa 5.2	Determination of availability change and irreversibility	128
Exa 5.3	Determination of availability change and irreversibility	130
Exa 5.4	Determination of useful work and irreversibility	134
Exa 5.5	Determination of reversible work and irreversibility . .	137
Exa 5.6	Determination of maximum obtainable work and efficiency	139
Exa 5.7	Determination of entropy generation rate and irreversibility	141
Exa 5.8	Calculation of exit temperature entropy and irreversibility rate	143
Exa 5.9	Determination of exit temperature availability change and irreversibility	145
Exa 6.1	Determination of enthalpy entropy and Gibbs free energy change of reaction	148
Exa 6.2	Determination of standard enthalpy and Gibbs free energy change of reaction	150
Exa 6.3	Determination of standard enthalpy and Gibbs free energy change of reaction	153
Exa 6.4	Determination of standard enthalpy and Gibbs free energy change of reaction	155
Exa 6.5	Determination of standard enthalpy and Gibbs free energy change of reaction	157
Exa 6.6	Calculation of heat exchange	159
Exa 6.7	Calculation of change in entropy	161
Exa 6.8	Calculation of standard enthalpy change and Gibbs free energy change	163

Exa 6.9	Calculation of standard enthalpy change and Gibbs free energy change	165
Exa 6.10	Determination of standard enthalpy change and Gibbs free energy change	167
Exa 7.1	Proving a mathematical relation	170
Exa 7.2	Proving a mathematical relation	171
Exa 7.3	Proving a mathematical relation	171
Exa 7.4	Proving a mathematical relation	172
Exa 7.5	Proving a mathematical relation	172
Exa 7.6	Estimation of entropy change	173
Exa 7.7	Determination of work done	174
Exa 7.8	Proving a mathematical relation	175
Exa 7.9	Proving a mathematical relation	176
Exa 7.10	Proving a mathematical relation	177
Exa 7.11	Proving a mathematical relation	177
Exa 7.12	Evaluation of beta and K for nitrogen gas	178
Exa 7.13	Calculation of temperature change and entropy change of water	179
Exa 7.14	Estimation of change in entropy and enthalpy	181
Exa 7.15	Calculation of percentage change in volume	182
Exa 7.16	Determination of enthalpy and entropy change	183
Exa 7.17	Proving a mathematical relation	185
Exa 7.18	Proving a mathematical relation	185
Exa 7.19	Proving a mathematical relation	186
Exa 7.20	Proving a mathematical relation	186
Exa 7.21	Proving a mathematical relation	187
Exa 7.22	Calculation of volume expansivity and isothermal compressibility	188
Exa 7.23	Estimation of specific heat capacity	189
Exa 7.24	Estimation of specific heat capacity	190
Exa 7.25	Calculation of volume expansivity and isothermal compressibility	191
Exa 7.26	Calculation of mean Joule Thomson coefficient	192
Exa 7.27	Estimation of Joule Thomson coefficient	194
Exa 7.28	Proving a mathematical relation	195
Exa 7.29	Proving a mathematical relation	196
Exa 7.30	Calculation of pressure	197
Exa 7.31	Calculation of enthalpy change and entropy change	198

Exa 7.32	Estimation of ratio of temperature change and pressure change	199
Exa 7.33	Determination of boiling point of water	200
Exa 7.34	Calculation of enthalpy and entropy of vaporization of water	201
Exa 7.35	Estimation of heat of vaporization of water	202
Exa 7.36	Calculation of latent heat of vaporization	203
Exa 7.37	Determination of temperature dependence	204
Exa 7.38	Calculation of fugacity of water	207
Exa 7.39	Estimation of fugacity of saturated steam	208
Exa 7.40	Estimation of fugacity of steam	209
Exa 7.41	Determination of fugacities at two states	212
Exa 8.1	Calculation of work done	214
Exa 8.2	Calculation of efficiency of Rankine cycle	217
Exa 8.3	Calculation of COP of carnot refrigerator and heat rejected	219
Exa 8.4	Calculation of minimum power required	220
Exa 8.5	Determination of COP and power required	221
Exa 8.6	Determination of maximum refrigeration effect	221
Exa 8.7	Determination of refrigeration effect power consumed and COP of refrigerator	222
Exa 8.8	Calculation of amount of air	225
Exa 8.9	Calculation of amount of air and temperature	227
Exa 8.10	Determination of temperature of air	230
Exa 10.1	Determination of expression for residual enthalpy internal energy and Gibbs free energy	233
Exa 10.2	Preparation of fugacity and fugacity coefficient	234
Exa 10.3	Calculation of enthalpy entropy and internal energy change	235
Exa 10.4	Calculation of molar heat capacity	238
Exa 10.5	Calculation of final temperature after expansion	240
Exa 10.6	Calculation of fugacity of liquid benzene	242
Exa 10.7	Calculation of molar enthalpy	244
Exa 10.8	Determination of second and third virial coefficients and fugacity	245
Exa 10.9	Determination of second and third virial coefficients	248
Exa 10.10	Determination of work done and the exit temperature	250
Exa 10.11	Calculation of temperature and pressure	256

Exa 10.12	Calculation of change of internal energy enthalpy entropy and exergy	260
Exa 10.13	Calculation of change in enthalpy	263
Exa 10.14	Calculation of final temperature	265
Exa 10.15	Proving a mathematical relation	266
Exa 10.16	Proving a mathematical relation	267
Exa 10.17	Determination of work done and the exit temperature	267
Exa 10.18	Proving a mathematical relation	271
Exa 10.19	Calculation of molar volume and fugacity	271
Exa 10.20	Calculation of enthalpy and entropy change	273
Exa 10.21	Calculation of fugacity	276
Exa 10.22	Calculation of enthalpy change	278
Exa 10.23	Calculation of fugacity of water vapour	282
Exa 10.24	Determination of change in internal energy	284
Exa 10.25	Calculation of enthalpy and entropy change	287
Exa 10.26	Calculation of final temperature and pressure	290
Exa 10.27	Calculation of vapour pressure	295
Exa 10.28	Determination of vapour pressure	298
Exa 11.1	Determination of volumes of ethanol and water	302
Exa 11.2	Developing an expression	303
Exa 11.3	Determination of partial molar volume	304
Exa 11.4	Calculation of enthalpies	305
Exa 11.5	Developing an expression and calculation for enthalpy change of mixture	307
Exa 11.6	Proving a mathematical relation	308
Exa 11.7	Proving a mathematical relation	308
Exa 11.8	Proving a mathematical relation	309
Exa 11.9	Calculation of minimum work required	309
Exa 11.10	Calculation of fugacity of the mixture	311
Exa 11.11	Proving a mathematical relation	312
Exa 11.12	Proving a mathematical relation	312
Exa 11.13	Proving a mathematical relation	313
Exa 12.1	Calculation of partial molar volume	314
Exa 12.2	Determination of volume of the mixture	316
Exa 12.3	Determination of volumes	318
Exa 12.4	Determination of partial molar volumes	322
Exa 12.5	Determination of enthalpy	325
Exa 13.1	Proving a mathematical relation	328

Exa 13.2	Calculation of fugacity coefficients	329
Exa 13.3	Proving a mathematical relation	331
Exa 13.4	Proving a mathematical relation	332
Exa 13.5	Proving a mathematical relation	332
Exa 13.6	Proving a mathematical relation	333
Exa 13.7	Calculation of fugacity coefficients	333
Exa 14.1	Determination of expression for activity coefficients . .	340
Exa 14.2	Proving a mathematical relation	341
Exa 14.3	Proving a mathematical relation	341
Exa 14.4	Determination of value of Gibbs free energy change and enthalpy change	342
Exa 14.5	Proving a mathematical relation	343
Exa 14.6	Proving a mathematical relation	344
Exa 14.7	Proving a mathematical relation	344
Exa 14.8	Proving a mathematical relation	345
Exa 14.9	Proving a mathematical relation	346
Exa 14.10	Comparison of Margules and van Laar equations	346
Exa 14.11	Calculation of activity coefficients	348
Exa 14.12	Calculation of the value of activity coefficients	350
Exa 14.13	Calculation of the value of activity coefficients	353
Exa 14.14	Calculation of the value of activity coefficients	354
Exa 14.15	Calculation of the value of activity coefficients	356
Exa 14.16	Proving a mathematical relation	357
Exa 14.17	Calculation of pressure	357
Exa 14.18	Proving a mathematical relation	358
Exa 14.19	Determination of enthalpy	359
Exa 14.20	Determination of an expression	360
Exa 14.21	Calculation of enthalpy entropy and Gibbs free energy	361
Exa 14.22	Determination of Gibbs free energy and enthalpy change	363
Exa 15.1	Calculation of number of moles in liquid and vapour phase	365
Exa 15.2	Calculation of pressure temperature and composition .	366
Exa 15.3	Calculation of pressure temperature and composition .	370
Exa 15.4	Calculation of pressure and composition	374
Exa 15.5	Calculation of pressure temperature and composition .	376
Exa 15.6	Determination of DPT and BPT	378
Exa 15.7	Determination of range of temperature for which two phase exists	380

Exa 15.8	Calculation of DPT and BPT	382
Exa 15.9	Calculation of range of pressure for which two phase exists	385
Exa 15.10	Determination of vapour and liquid phase composition	388
Exa 15.11	Calculation of temperature	389
Exa 15.12	Determination of number of moles in liquid and vapour phase	391
Exa 15.13	Determination of vapour and liquid phase composition	392
Exa 15.14	Preparation of table having composition and pressure data	394
Exa 15.15	Calculation of DPT and BPT	396
Exa 15.16	Calculation of pressure	397
Exa 15.17	Calculation of van Laar activity coefficient parameters	399
Exa 15.18	Prediction of azeotrope formation	400
Exa 15.19	Tabulation of activity coefficients relative volatility and compositions	402
Exa 15.20	Tabulation of partial pressure and total pressure data of components	405
Exa 15.21	Determination of azeotrope formation	408
Exa 15.22	Tabulation of pressure and composition data	410
Exa 15.23	Determination of van Laar parameters	412
Exa 16.1	Determination of solubility	415
Exa 16.2	Determination of solubility	417
Exa 16.3	Proving a mathematical relation	419
Exa 16.4	Determination of composition	420
Exa 16.5	Determination of equilibrium composition	422
Exa 16.6	Proving a mathematical relation	423
Exa 16.7	Determination of freezing point depression	423
Exa 16.8	Determination of freezing point	425
Exa 16.9	Proving a mathematical relation	426
Exa 16.10	Determination of boiling point elevation	426
Exa 16.11	Determination of osmotic pressure	427
Exa 16.12	Determination of pressure	428
Exa 16.13	Determination of amount of precipitate	430
Exa 16.14	Calculation of pressure	431
Exa 17.1	Proving a mathematical relation	434
Exa 17.2	Determination of number of moles	434
Exa 17.3	Determination of equilibrium composition	438

Exa 17.4	Determination of the value of equilibrium constant . . .	439
Exa 17.5	Determination of mole fraction	442
Exa 17.6	Determination of number of moles	442
Exa 17.7	Calculation of mole fraction	445
Exa 17.8	Calculation of heat exchange	448
Exa 17.9	Determination of heat of reaction	450
Exa 17.10	Tabulation of equilibrium constant values	453
Exa 17.11	Determination of mean standard enthalpy of reaction .	456
Exa 17.12	Derivation of expression for enthalpy of reaction . . .	457
Exa 17.13	Determination of equilibrium composition	458
Exa 17.14	Determination of equilibrium composition	462
Exa 17.15	Determination of the	463
Exa 17.16	Calculation of the value of Gibbs free energy	465
Exa 17.17	Calculation of nu	467
Exa 17.18	Determination of value of the equilibrium constant . .	469
Exa 17.19	Determination of the value of equilibrium constant . .	471
Exa 17.20	Calculation of standard equilibrium cell voltage	473
Exa 17.21	Calculation of number of chemical reactions	475
Exa 17.22	Calculation of number of chemical reactions	477
Exa 17.23	Calculation of equilibrium composition	482
Exa 17.24	Determination of number of moles	484
Exa 18.1	Calculation of heat transfer	488
Exa 18.2	Calculation of adiabatic flame temperature	490
Exa 18.3	Calculation of mole fraction and average heat capacity	492
Exa 18.4	Determination of adiabatic flame temperature	493
Exa 18.5	Calculation of conversion	495
Exa 18.6	Calculation of maximum pressure	496
Exa 18.7	Calculation of number of moles	498

Chapter 1

Introduction

Scilab code Exa 1.1 Calculation of pressure and heat transfer in piston cylinder assembly

```
1 clear;
2 clc;
3
4 //Example - 1.1
5 //Page number - 6
6 printf("Example - 1.1 and Page number - 6\n\n");
7
8 //(a)
9 // The pressure in the cylinder is due to the weight
   of the piston and due to surroundings pressure
10 m = 50; //[kg] - Mass of piston
11 A = 0.05; //[m^(2)] - Area of piston
12 g = 9.81; //[m/s^(2)] - Acceleration due to gravity
13 Po = 101325; //[N/m^(2)] - Atmospheric pressure
14 P = (m*g/A)+Po; //[N/m^(2)]
15 P = P/100000; //[bar]
16 printf(" (a). Pressure = %f bar\n",P);
17
18 //(b)
19 printf(" (b). Since the piston weight and
```

surroundings pressure are the same, the gas pressure in the piston-cylinder assembly remains %f bar",P);

Scilab code Exa 1.2 Calculation of mass of air contained in a room

```
1 clear;
2 clc;
3
4 //Example - 1.2
5 //Page number - 8
6 printf("Example - 1.2 and Page number - 8\n\n");
7
8 // Given
9 P = 1; // [atm] - Atmospheric pressure
10 P = 101325; // [N/m^(2)]
11 R = 8.314; // [J/mol*K] - Universal gas constant
12 T = 30; // [C] - Temperature of air
13 T = 30+273.15; // [K]
14 V = 5*5*5; // [m^(3)] - Volume of the room
15
16 //The number of moles of air is given by
17 n = (P*V)/(R*T); // [mol]
18
19 //Molecular weight of air (21 vol% O2 and 79 vol% N2)
    =(0.21*32)+(0.79*28)= 28.84 g/mol
20 m = n*28.84; // [g]
21 m = m/1000; // [kg]
22 printf("The mass of air is , m = %f kg",m);
```

Scilab code Exa 1.3 Determination of work done

```
1 clear;
```



```

2  clc;
3
4  //Example - 1.3
5  //Page number - 13
6  printf("Example - 1.3 and Page number - 13\n\n");
7
8  // Given
9  P1 = 3; // [bar] - initial pressure
10 V1 = 0.5; // [m^(3)] - initial volume
11 V2 = 1.0; // [m^(3)] - final volume
12
13 // (a)
14 n = 1.5;
15
16 //Let  $P*V^n=C$  //Given relation
17 //W (work done per mole)= (integrate('P', 'V', V1, V2))
18 //W = (integrate('C/V^(n)', 'V', V1, V2)) = (C*((V2
19     ^*(1-n))-(V1^(1-n)))/(1-n)
20 //Where  $C=P*V^n=P1*V1^n=P2*V2^n$ 
21 //Thus  $w=((P2*V2^n)*V2^(1-n))-(P1*V1^n)*V1^(1-n))$ 
22 //w = ((P2*V2^n)-(P1*V1^n))/(1-n)
23 //and thus  $W=((P2*V2)-(P1*V1))/(1-n)$ 
24 //The above expression is valid for all values of n,
25     except n=1.0
26 P2 = (P1*((V1/V2)^(n))); // [bar] //pressure at state
27     2
28
29 //we have,  $(V1/V2)=(V1t/(V2t))$ , since the number of
30     moles are constant. Thus
31 W = ((P2*V2)-(P1*V1))/(1-n)*10^(5); // [J]
32 W = W/1000; // [kJ]
33 printf(" (a).The work done (for n=1.5) is %f kJ\n", W
34     );
35
36 // (b)
37 //For n=1.0, we have,  $PV=C$ .
38 // w(wok done per mol)= (integrate('P', 'V', V1, V2)) =

```

```

        (integrate('C/V', 'V', V1, V2)) = C*ln(V2/V1)=P1*V1
        *ln(V2/V1)
34 W1 = P1*V1*log(V2/V1)*10^(5); //[J]
35 W1 = W1/1000; //[kJ]
36 printf(" (b).The work done (for n=1.0) is %f kJ\n",
        W1);
37
38 //(c)
39 //For n=0,we get P=Constant and thus
40 P = P1; //[bar]
41 // w =(integrate('P', 'V', V1, V2)) = P*(V2-V1)
42 W2 = P*(V2-V1)*10^(5); //[J]
43 W2 = W2/1000; //[kJ]
44 printf(" (c).The work done (for n=0) is %f kJ\n\n",
        W2);

```

Scilab code Exa 1.4 Determination of wind energy per unit mass and diameter of the wind turbine

```

1 clear;
2 clc;
3
4 //Example - 1.4
5 //Page number - 17
6 printf("Example - 1.4 and Page number - 17\n\n");
7
8 //(a)
9 //Given
10 V = 9; // [m/s] - velocity
11 d = 1; // [m] - diameter
12 A = 3.14*(d/2)^(2); // [m^(2)] - area
13 P = 1; // [atm] - pressure
14 P = 101325; // [N/m^(2)]
15 T = 300; // [K] - Temperature
16 R = 8.314; // [J/mol*K] - Universal gas constant

```

```

17
18 E = (V^(2))/2; //[J/kg]
19 printf(" (a).The wind energy per unit mass of air is
        %f J/kg\n",E);
20
21 //(b)
22 // Molecular weight of air(21 vol% O2 and 79 vol% N2
        )=(0.21*32)+(0.79*28)= 28.84 g/mol
23 M = 28.84*10^(-3); //[kg/mol]
24 r = (P*M)/(R*T); //[kg/m^(3)] - density
25 m = r*V*A; // [kg/s] - mass flow rate of air
26 pi = m*E; //[Watt] - power input
27 printf(" (b).The wind power input to the turbine is
        %f Watt",pi);

```

Scilab code Exa 1.5 Determination of temperature

```

1 clear;
2 clc;
3
4 //Example - 1.5
5 //Page number - 23
6 printf("Example - 1.5 and Page number - 23\n\n");
7
8 // Given
9 P = 1; // [bar] - atmospheric pressure
10 P1guz = 0.75; // [bar] - gauze pressure in 1st
    evaporator
11 P2Vguz = 0.25; // [bar] - vaccum gauze pressure in 2
    nd evaporator
12 P1abs = P + P1guz; // [bar] - absolute pressure in 1
    st evaporator
13 P2abs = P - P2Vguz; // [bar] -absolute pressure in 2
    nd evaporator
14

```

```

15 //From saturated steam table as reported in the book
16 printf(" For P1abs (absolute pressure) = %f bar\n",
    P1abs);
17 printf(" The saturation temperature in first
    evaporator is 116.04 C\n\n");
18 printf(" For P2abs (absolute pressure) = %f bar\n",
    P2abs);
19 printf(" The saturation temperature in second
    evaporator is 91.76 C\n");

```

Scilab code Exa 1.6 Calculation of dryness fraction of steam

```

1 clear;
2 clc;
3
4 //Example - 1.6
5 //Page number - 23
6 printf("Example - 1.6 and Page number - 23\n\n");
7
8 // Given
9 V = 1; // [kg] - volume of tank
10 P = 10; // [bar] - pressure
11
12 //Here degree of freedom =1(C=1,P=2, therefore F=1)
13 //From steam table at 10 bar as reported in the book
14 V_liq = 0.001127; // [m3/kg] - volume in liquid
    phase
15 V_vap = 0.19444; // [m3/kg] - volume in vapour
    phase
16
17 //x*Vv=(1-x)*Vl // since two volumes are equal
18 x = (V_liq/(V_liq+V_vap)); // [kg]
19 y = (1-x); // [kg]
20
21 printf(" Mass of saturated vapour is %f kg\n",x);

```

```
22 printf(" Mass of saturated liquid is %f kg",y);
```

Scilab code Exa 1.7 Determination of pressure mass and volume

```
1 clear;
2 clc;
3
4 //Example - 1.7
5 //Page number - 23
6 printf("Example - 1.7 and Page number - 23\n\n");
7
8 // Given
9 V = 1; // [m(3)] - volume of tank
10 M = 10; // [m(3)] - total mass
11 T = (90+273.15); // [K] - temperature
12
13 //From steam table at 90 C as reported in the book
14 //vapour pressure (pressure of rigid tank) = 70.14[
    kPa] = 0.7014[bar]
15 printf(" Pressure of tank = 0.7014 bar\n");
16
17 V_liq_sat=0.001036; // [m(3)/kg] - saturated liquid
    specific volume
18 V_vap_sat=2.36056; // [m(3)/kg] - saturated vapour
    specific volume
19
20 //1=(V_liq_sat*(10-x))+(V_vap_sat*x)
21 x = (1-(10*V_liq_sat))/(V_vap_sat-V_liq_sat); // [kg]
22 y = (10-x); // [kg]
23
24 printf(" The amount of saturated liquid is %f kg\n",
    y);
25 printf(" The amount of saturated vapour is %f kg \n"
    ,x);
26
```

```

27 z = y*V_liq_sat; // [m(3)] - Volume of saturated
    liquid
28 w = x*V_vap_sat; // [m(3)] - Volume of saturated
    vapour
29
30 printf(" Total volume of saturated liquid is %f m
    ^{(3)}\n",z);
31 printf(" Total volume of saturated vapour is %f m
    ^{(3)}",w);

```

Scilab code Exa 1.8 Determination of heat supplied

```

1 clear;
2 clc;
3
4 //Example - 1.8
5 //Page number - 24
6 printf("Example - 1.8 and Page number - 24\n\n");
7
8 // Given
9 V = 10; // [m(3)] - volume of vessel
10 P_1 = 1; // [bar] - initial pressure
11 V_liq_sat = 0.05; // [m(3)] - saturated liquid
    volume
12 V_gas_sat = 9.95; // [m(3)] - saturated vapour
    volume
13
14 //At 1 bar pressure
15 V_liq_1 = 0.001043; // [m(3)/kg] - specific
    saturated liquid volume
16 U_liq_1 = 417.33; // [kJ/kg] - specific internal
    energy
17 V_gas_1 = 1.69400; // [m(3)/kg] - specific saturated
    vapour volume
18 U_gas_1 = 2506.06; // [kJ/kg]

```

```

19
20 M_liq_1 = V_liq_sat/V_liq_1;// [kg] - mass of
    saturated liquid
21 M_gas_1 = V_gas_sat/V_gas_1;// [kg] - mass of
    saturated vapour
22 M = (M_liq_1+M_gas_1);// [kg] - total mass
23 U_1t = (M_liq_1*U_liq_1)+(M_gas_1*U_gas_1);// [kJ] -
    initial internal energy
24 V_gas_2 = (V/M);// [m^(3/kg)]
25
26 //from steam table at 10 bar pressure as reported in
    the book
27 V_vap_2 = 0.19444;// [m^(3/kg)]
28 U_vap_2 = 2583.64;// [kJ/kg]
29
30 //from steam table at 11 bar pressure as reported in
    the book
31 V_vap_3 = 0.17753;// [m^(3/kg)]
32 U_vap_3 = 2586.40;// [kJ/kg]
33
34 //Now computing pressure when molar volume of
    saturated vapour=Vg_2
35 //By interpolation (P2-10)/(11-10)=(Vg_2-Vv_2)/(Vv_3
    -Vv_2)
36 P_2 = (((V_gas_2 - V_vap_2)/(V_vap_3 - V_vap_2)*1)
    +10);// [bar] - final pressure
37
38 //By interpolation calculating internal energy at
    state 2
39 //(P2-10)/(11-10)=(U2-Uv_2)/(Uv_3-Uv_2)
40 U_2 = (((P_2-10)/(11-10))*(U_vap_3 - U_vap_2))+
    U_vap_2;// [kJ/kg]
41 U_2t = U_2*M;// [kJ]
42 H = U_2t - U_1t;// [kJ] - Heat supplied
43 H = H/1000;// [MJ]
44
45 printf(" Total heat supplied is %f MJ',H);
46 // since volume is constant ,no work is done by the

```

system and heat supplied is used in increasing the internal energy of the system.

Scilab code Exa 1.9 Calculation of saturation temperature

```
1 clear;
2 clc;
3
4 //Example - 1.9
5 //Page number - 26
6 printf("Example - 1.9 and Page number - 26\n\n");
7
8 //Given
9 //Antoine equation for water ln(Psat)
   =16.262-(3799.89/(T_sat + 226.35))
10 P = 2; //[atm] - Pressure
11 P = (2*101325)/1000; //[kPa]
12
13 P_sat = P; // Saturation pressure
14 T_sat = (3799.89/(16.262-log(P_sat)))-226.35; //[C] -
   Saturation temperature
15 //Thus boiling at 2 atm occurs at Tsat = 120.66 C.
16
17 //From steam tables ,at 2 bar ,Tsat = 120.23 C and at
   2.25 bar ,Tsat = 124.0 C
18 //From interpolation for T_sat = 120.66 C,P = 2.0265
   bar
19 //For P_= 2.0265 bar ,T_sat , from steam table by
   interpolation is given by
20 //((2.0265-2)/(2.25-2))=((Tsat-120.23)
   /(124.0-120.23))
21 T_sat_0 = (((2.0265-2)/(2.25-2))*(124.0-120.23))
   +120.23; //[C]
22
23 printf(" Saturation temperature (Tsat) = %f C which
```


is close to %f C as determined from Antoine equation",T_sat_0,T_sat);

Scilab code Exa 1.10 Calculation of pressure and temperature at triple point

```
1 clear;
2 clc;
3
4 //Example - 1.10
5 //Page number - 27
6 printf("Example - 1.10 and Page number - 27\n\n");
7
8 //Given
9 // log(P)=- (1640/T)+10.56 (solid)
10 // log(P)=- (1159/T)+7.769 (liquid), where T is in K
11 // F+P=C+2, at triple point F+3=1+2 or, F=0 i.e.,
    vapour pressure of liquid and solid at triple
    point are same, we get
12 // -(1640/T)+10.56 = -(1159/T)+7.769
13
14 T = (1640-1159)/(10.56-7.769); // [K]
15 P = 10^((-1640/T)+10.56); // [torr]
16
17 printf(" The temperature is %f K\n",T);
18 printf(" The pressure is %f torr (or mm Hg)",P);
```

Scilab code Exa 1.11 Determination of value of R Cp0 and Cv0

```
1 clear;
2 clc;
3
4 //Example - 1.11
```

```

5 //Page number - 29
6 printf("Example - 1.11 and Page number - 29\n\n");
7
8 //Given
9 M_O2 = 31.999; //molecular weight of oxygen
10 M_N2 = 28.014; //molecular weight of nitrogen
11 Y = 1.4; //molar heat capacities ratio for air
12
13 //Molecular weight of air(21 vol% O2 and 79 vol% N2)
    is given by
14 M_air = (0.21*M_O2)+(0.79*M_N2); //(vol% = mol%)
15
16 R = 8.314; // [J/mol*K] - Universal gas constant
17 R = (R*1/M_air); // [kJ/kg*K]
18
19 printf(" The value of universal gas constant (R) =
    %f kJ/kg-K \n",R);
20
21 //Y=Cp0/Cv0 and Cp0-Cv0=R
22 Cv_0 = R/(Y-1); // [kJ/kg*K]
23 Cp_0 = Y*Cv_0; // [kJ/kg*K]
24 printf(" The value of Cp_0 for air is %f kJ/kg-K\n',
    Cp_0);
25 printf(" The value of Cv_0 for air is %f kJ/kg-K',
    Cv_0);

```

Scilab code Exa 1.12 Calculation of molar heat capacity

```

1 clear;
2 clc;
3
4 //Example - 1.12
5 //Page number - 30
6 printf("Example - 1.12 and Page number - 30\n\n");
7

```

```

8 //Given
9 Y = 1.4; //molar heat capacities ratio for air
10 R = 8.314; // [J/mol*K] - Universal gas constant
11 Cv_0 = R/(Y-1); // [J/mol*K]
12 Cp_0 = Y*Cv_0; // [J/mol*K]
13
14 printf(" The molar heat capacity at constant volume
        (Cv_0) is %f J/mol-K\n", Cv_0);
15 printf(" The molar heat capacity at constant
        pressure (Cp_0) is %f J/mol-K', Cp_0);

```

Scilab code Exa 1.13 Determination of mean heat capacity

```

1 clear;
2 clc;
3
4 //Example - 1.13
5 //Page number - 30
6 printf("Example - 1.13 and Page number - 30\n\n");
7
8 //Given
9 // Cp0=7.7+(0.04594*10^(-2)*T)+(0.2521*10^(-5)*T^(2)
    )-(0.8587*10^(-9)*T^(3))
10 T_1 = 400; // [K]
11 T_2 = 500; // [K]
12
13 //(C)avg = q/(T_2 - T_1) = 1/(T_2 - T_1)*{(integrate
    ('C', 'T', T_1, T_2))}
14 //(Cp0)avg = 1/(T_2 - T_1)*{(integrate('Cp0', 'T', T_1
    , T_2))}
15 Cp0_avg = (1/(T_2 - T_1))*integrate('
    7.7+(0.04594*10^(-2)*T)+(0.2521*10^(-5)*T^(2))
    -(0.8587*10^(-9)*T^(3))', 'T', T_1, T_2);
16
17 printf(" The mean heat capacity (Cp0_avg) for

```

temperature range of 400 to 500 K is %f cal/mol-K"
,Cp0_avg);

Scilab code Exa 1.14 Calculation of enthalpy of water

```
1 clear;
2 clc;
3
4 //Example - 1.14
5 //Page number - 31
6 printf("Example - 1.14 and Page number - 31\n\n");
7
8 //Given
9 //(a)
10 P_1 = 0.2; // [MPa] - pressure
11 x_1 = 0.59; // mole fraction
12
13 //From saturated steam tables at 0.2 MPa
14 H_liq_1 = 504.7; // [kJ/kg] - Enthalpy of saturated
    liquid
15 H_vap_1 = 2706.7; // [kJ/kg]- Enthalpy of saturated
    vapour
16 H_1 = (H_liq_1*(1-x_1))+(x_1*H_vap_1); // [kJ/kg]
17 printf(" (a). Enthalpy of 1 kg of water in tank is %f
    kJ/kg\n", H_1);
18
19 //(b)
20 T_2 = 120.23; // [C] - temperature
21 V_2 = 0.6; // [m^(3)/kg] - specific volume
22
23 //From saturated steam tables at 120.23 C, as
    reported in the book
24 V_liq_2=0.001061; // [m^(3)/kg]
25 V_vap_2=0.8857; // [m^(3)/kg]
26 //since V_2 < Vv_2, dryness factor will be given by,
```

```

    V = ((1-x)*V_liq)+(x*V_vap)
27 x_2 = (V_2- V_liq_2)/(V_vap_2 - V_liq_2);
28
29 //From steam table ,at 120.2 C,the vapour pressure of
    water is 0.2 MPa.So,enthalpy is given by
30 H_2 = (H_liq_1*(1-x_2))+(H_vap_1*x_2); //kJ/kg]
31 printf(" (b).Enthalpy of saturated steam is %f kJ/kg
    \n",H_2);
32
33 //(c)
34 P_3 = 2.5; // [MPa]
35 T_3 = 350; // [C]
36 //From steam tables at 2.5 MPa, T_sat = 223.99 C, as
    reported in the book
37 //since ,T_3 > Tsat, steam is superheated
38 printf(" (c).As steam is superheated ,from steam
    table ,enthalpy (H) is 3126.3 kJ/kg\n");
39
40 //(d)
41 T_4 = 350; // [C]
42 V_4 = 0.13857; // [m^(3)/kg]
43 //From steam table ,at 350 C, V_liq = 0.001740 m^(3)/
    kg and V_vap = 0.008813 m^(3)/kg. Since ,V > V_vap ,
    therefore it is superheated.
44 //From steam table at 350 C and 1.6 MPa, V = 0.17456
    m^(3)/kg
45 //At 350 C and 2.0 MPa, V = 0.13857 m^(3)/kg. So ,
46 printf(" (d).The enthalpy of superheated steam (H)
    is 3137.0 kJ/kg\n");
47
48 //(e)
49 P_4 = 2.0; // [MPa]
50 U_4 = 2900; // [kJ/kg] - internal energy
51 //From saturated table at 2.0 MPa, U_liq = 906.44kJ
    and U_vap = 2600.3 kJ/kg
52 //scince ,U_4 > Uv, it is saturated.
53 //From superheated steam table at 2.0 MPa and 350 C,
    as reported in the book

```

```

54 U_1 = 2859.8; //[kJ/kg]
55 H_1 = 3137.0; //[kJ/kg]
56 //At 2.0 MPa and 400 C,
57 U_2 = 2945.2; //[kJ/kg]
58 H_2 = 3247.6; //[kJ/kg]
59 T = (((U_4 - U_1)/(U_2 - U_1))*(400 - 350)) + 350; //
    [C] - By interpolation
60 H = (((T - 350)/(400 - 350))*(H_2 - H_1)) + H_1; //[
    kJ/kg]
61 printf(" (e).The enthalpy value (of superheated
    steam) obtained after interpolation is %f kJ/kg\n
    ",H);
62
63 //(f)
64 P_5 = 2.5; //[MPa]
65 T_5 = 100; //[C]
66 //At 100 C, P_sat=101350 N/m^(2). Since P_5 > P_sat ,
    it is compressed liquid
67 P_sat = 0.101350; //[MPa]
68 H_liq = 419.04; //[kJ/kg] - At 100 C and 0.10135 MPa
69 V_liq = 0.001044; //[m^(3)/kg] - At 100 C and 0.10135
    MPa
70 H_0 = H_liq + (V_liq*(P_5 - P_sat))*1000; //[kJ/kg]
71 printf(" (f).The enthalpy of compressed liquid is %f
    kJ/kg\n",H_0);

```

Chapter 2

Equations of state

Scilab code Exa 2.1 Relations in virial coefficients

```
1 clear;
2 clc;
3
4 //Example - 2.1
5 //Page number - 40
6 printf("Example - 2.1 and Page number - 40\n\n");
7
8 //This problem involves proving a relation in which
   no numerical components are involved.
9 //For prove refer to this example 2.1 on page number
   40 of the book.
10 printf(" This problem involves proving a relation in
   which no numerical components are involved.\n\n"
   );
11 printf(" For prove refer to this example 2.1 on page
   number 40 of the book.");
```

Scilab code Exa 2.2 Determination of acentric factor

```

1 clear;
2 clc;
3
4 //Example - 2.2
5 //Page number - 42
6 printf("Example - 2.2 and Page number - 42\n\n");
7
8 //Given
9 Tc = 647.1;//[K] - Critical temperature
10 Pc = 220.55;//[bar] - Critical pressure
11 Tr = 0.7;// Reduced temperature
12
13 T = Tr*Tc;//[K]
14 //From steam table ,vapour pressure of H2O at T is
15     10.02 [bar], as reported in the book
16 P = 10.02;//[bar]
17 w = -1-log10((P/Pc));
18 printf(" The acentric factor (w) of water at given
19     condition is %f ",w);

```

Scilab code Exa 2.3 Calculation of acentric factor

```

1 clear;
2 clc;
3
4 //Example - 2.3
5 //Page number - 42
6 printf("Example - 2.3 and Page number - 42\n\n");
7
8 //Given
9 //log10(Psat)=8.1122-(1592.864/(t+226.184))// 'Psat'
10     in [mm Hg] and 't' in [c]
11 Tc = 513.9;//[K] - Critical temperature
12 Pc = 61.48;//[bar] - Critical pressure
13 Pc = Pc*10^(5);//[N/m^(2)]

```



```

13 Tr = 0.7; // Reduced temperature
14
15 T = Tr*Tc; // [K] - Temperature
16 T = T - 273.15; // [C]
17 P_sat = 10^(8.1122 - (1592.864/(T + 226.184))); // [mm
    Hg]
18 P_sat = (P_sat/760)*101325; // [N/m^(2)]
19 Pr_sat = P_sat/Pc;
20 w = -1-log10(Pr_sat); // Acentric factor
21 printf(" The acentric factor (w) for ethanol at
    given condition is %f",w);

```

Scilab code Exa 2.4 Calculation of virial coefficients

```

1 clear;
2 clc;
3
4 //Example - 2.4
5 //Page number - 45
6 printf("Example - 2.4 and Page number - 45\n\n");
7
8 //Given
9 T = 380; // [K] - Temperature
10 Tc = 562.1; // [K] - Critical temperature
11 P = 7; // [atm] - Pressure
12 P = P*101325; // [N/m^(2)]
13 Pc = 48.3; // [atm] - Critical pressure
14 Pc = Pc*101325; // [N/m^(2)]
15 R = 8.314; // [J/mol*K] - Universal gas constant
16 w = 0.212; // acentric factor
17 Tr = T/Tc; // Reduced temperature
18
19 B_0 = 0.083-(0.422/(Tr)^(1.6));
20 B_1 = 0.139-(0.172/(Tr)^(4.2));
21

```

```

22 //We know ,(B*Pc)/(R*Tc) = B_0+(w*B_1)
23 B = ((B_0+(w*B_1))*(R*Tc))/Pc; // [m^(3)/mol]
24 printf(" The second virial coefficient for benzene
      is %e m^(3)/mol\n",B);
25
26 //Compressibility factor is given by
27 Z = 1 + ((B*P)/(R*T));
28 printf(" The compressibility factor at 380 K is %f\n
      ",Z);
29
30 //We know thar Z=(P*V)/(R/*T), therefore
31 V = (Z*R*T)/P; // [m^(3)/mol]
32 printf(" The molar volume is %e m^(3)/mol\n",V);

```

Scilab code Exa 2.5 Calculation of mass using virial equation of state

```

1 clear;
2 clc;
3
4 //Example - 2.5
5 //Page number - 46
6 printf("Example - 2.5 and Page number - 46\n\n");
7
8 //Given
9 V_1 = 0.3; // [m^(3)] // volume of cylinder
10 T = 60+273.15; // [K] - Temperature
11 P = 130*10^(5); // [N/m^(2)] - Pressure
12 Tc = 305.3; // [K] - Critical temperature
13 Pc = 48.72*10^(5); // [N/m^(2)] - Critical pressure
14 w = 0.100; //acentric factor
15 M = 30.07; //molecular weight of ethane
16 Tr = T/Tc; // Reduced temperature
17 R = 8.314; // [J/mol*K] - Universal gas constant
18
19 B_0 = 0.083-(0.422/(Tr)^(1.6));

```

```

20 B_1 = 0.139-(0.172/(Tr)^(4.2));
21
22 //We know ,(B*Pc)/(R*Tc) = B_0+(w*B_1)
23 B = ((B_0 + (w*B_1))*(R*Tc))/Pc; // [m^(3)/mol] -
    Second virial coefficient
24 Z = 1 + ((B*P)/(R*T)); // Compressibility factor
25 V = (Z*R*T)/P; // [m^(3)/mol] - Molar volume
26
27 //No. of moles in 0.3 m^(3) cylinder is given by
28 n1 = V_1/V; // [mol]
29
30 //Mass of gas in cylinder is given by
31 m1 = (n1*M)/1000; // [kg]
32 printf(" Under actual conditions ,the mass of ethane
    is , %f kg\n",m1);
33
34 //Under ideal condition , taking Z = 1,
35 V_ideal = (R*T)/P; // [m^(3)/mol]
36 n2 = V_1/V_ideal; // [mol]
37 m2 = (n2*M)/1000; // [kg]
38 printf(" Under ideal conditions ,the mass of ethane
    is , %f kg\n",m2);

```

Scilab code Exa 2.6 Calculation of molar volume

```

1 clear;
2 clc;
3 funcprot(0);
4
5 //Example - 2.6
6 //Page number - 47
7 printf("Example - 2.6 and Page number - 47\n\n");
8
9 //Given
10 T = 373.15; // [K] - Temperature

```

```

11 P = 101325; // [N/m^(2)] - Pressure
12 Tc = 647.1; // [K] - Critical temperature
13 Pc = 220.55*10^(5); // [N/m^(2)] - Critical pressure
14 w = 0.345; // acentric factor
15 Tr = T/Tc; // Reduced temperature
16 R = 8.314; // [J/mol*K] - UNiversal gas constant
17
18 B_0 = 0.083-(0.422/(Tr)^(1.6));
19 B_1 = 0.139-(0.172/(Tr)^(4.2));
20
21 //We know, (B*Pc)/(R*Tc) = B_0+(w*B_1)
22 B = ((B_0+(w*B_1))*(R*Tc))/Pc; // [m^(3)/mol] - Second
    virial coefficient
23
24 //We have, Z = 1+(B/V) and Z = (P*V)/(R*T).
    Substituting the value of Z, we get
25 // V^(2) - ((R*T)/P)*V - (B*R*T)/P = 0 . Solving the
    quadratic equation by shreedharcharya rule
26 V1 = (((R*T)/P) + (((R*T)/P)^(2) + 4*1*((B*R*T)/P))
    ^(1/2))/2*1;
27
28 printf(" The molar volume of water vapour is %f m
    ^(3)/mol", V1);
29
30 //The roots are, V1 = 0.0003670 [m^(3)/mol] and V2 =
    0.0302510 [m^(3)/mol].
31 //As 'V2' is near to ideal volume (0.030618 [m^(3)/
    mol]), it is taken as the molar volume
32 //The other root 'V1' has no physical significance

```

Scilab code Exa 2.7 Calculation of molar volume and virial coefficients

```

1 clear;
2 clc;
3 funcprot(0);

```

```

4
5 //Example - 2.7
6 // Page number - 47
7 printf("Example - 2.7 and Page number - 47\n\n");
8
9 // Given
10 T = 50+273.15; // [K] - Temperature
11 P = 15*10^(5); // [N/m^(2)] - Pressure
12 Tc = 305.3; // [K] - Critical temperature
13 Pc = 48.72*10^(5); // [N/m^(2)] - Critical pressure
14 w = 0.100; // Acentric factor
15 B = -157.31; // [cm^(3)/mol] - second virial
    coefficient
16 B = B*10^(-6); // [m^(3)/mol]
17 C = 9650; // [cm^(6)/mol^(2)] - third virial
    coefficient
18 C = C*10^(-12); // [cm^(6)/mol^(2)]
19 R = 8.314; // [J/mol*K] - Universal gas constant
20
21 // (1)
22 V_1 = (R*T)/P; // [m^(3)/mol] - Molar volume
23 printf(" (1).The molar volume for ideal equation of
    state is %e m^(3)/mol\n",V_1);
24
25 // (2)
26 Tr = T/Tc; // Reduced temperature
27 // At this temperature
28 B_0 = 0.083-(0.422/(Tr)^(1.6));
29 B_1 = 0.139-(0.172/(Tr)^(4.2));
30
31 // We know ,(B*Pc)/(R*Tc) = B_0+(w*B_1)
32 B_2 = ((B_0 + (w*B_1))*(R*Tc))/Pc; // [m^(3)/mol]//
    second virial coefficient
33 printf(" (2).The second virial coefficient using
    Pitzer correlation is found to be %e m^(3)/mol
    which is same as given value\n",B_2);
34
35 // (3)

```

```

36 // Given (virial equation),  $Z=1+(B/V)$ 
37  $V_3 = B + (R*T)/P$ ; // [m3/mol] – Molar volume
38 printf(" (3).The molar volume using virial equation
    of state is %e m3/mol\n",V_3);
39
40 // (4)
41 // Given (virial equation),  $Z = 1 + ((B*P)/(R*T)) +$ 
     $((C - B^2)/(R*T)^2)*P^2$ 
42  $V_4 = B + (R*T)/P + ((C - B^2)/(R*T))*P$ ; // [m3/
    mol]
43 printf(" (4).The molar volume using given virial
    equation of state is %e m3/mol\n",V_4);
44
45 // (5)
46 // Given,  $Z = 1 + (B/V)$ 
47 // Also,  $Z = (P*V)/(R*T)$ . Substituting the value of Z
    ,we get
48 //  $V^2 - ((R*T)/P)*V - ((B*R*T)/P) = 0$ . Solving the
    quadratic equation
49 deff(' [y]=f(V) ', 'y= $V^2 - ((R*T)/P)*V - ((B*R*T)/P)$  ');
50 V_5_1 = fsolve(0,f);
51 V_5_2 = fsolve(1,f);
52
53 printf(" (5).The molar volume using given virial
    equation of state is %e m3/mol\n",V_5_2);
54
55 // The roots are,  $V_5_1=0.0001743$  [m3/mol] and
     $V_5_2=0.0016168$  [m3/mol].
56 // As 'V_2' is near to ideal volume (0.0017911 [m
    ^3/mol]), it is taken as the molar volume
57
58 // (6)
59 // Given,  $Z = 1 + (B/V) + (C/V^2)$ 
60 // Also,  $Z = (P*V)/(R*T)$ . Substituting the value of Z
    ,we get
61 //  $V^3 - ((R*T)/P)*V^2 - ((B*R*T)/P)*V - ((C*R*T)/P)$ 
    =0. Solving the cubic equation
62 deff(' [y]=f1(V) ', 'y= $V^3 - ((R*T)/P)*V^2 - ((B*R*T)/P$ 

```

```

    )*V-((C*R*T)/P) ');
63 V_6_3=fsolve(-1,f1);
64 V_6_4=fsolve(0,f1);
65 V_6_5=fsolve(1,f1);
66 //The above equation has 1 real and 2 imaginary
    roots. We consider only real root.
67 printf(" (6).The molar volume using given virial
    equation of state is %e m^(3)/mol\n",V_6_5);

```

Scilab code Exa 2.8 Determination of second and third virial coefficients

```

1 clear;
2 clc;
3 funcprot(0);
4
5 //Example - 2.8
6 // Page number - 49
7 printf("Example - 2.8 and Page number - 49\n\n");
8
9 //Given
10 T = 0 + 273.15; // [K] - Temperature
11 R = 8.314; // [J/mol*K] - Universal gas constant
12
13 //Virial equation of state , Z=1+(B/V)+(C/V^(2))
14 //From above equation we get (Z-1)*V=B+(C/V)
15
16 P=[50,100,200,400,600,1000];
17 Z=[0.9846,1.0000,1.0365,1.2557,1.7559,2.0645];
18 V=zeros(6);
19 k=zeros(6);
20 t=zeros(6);
21 for i=1:6;
22     V(i)=(Z(i)*R*T)/(P(i)*101325); // [m^(3)/mol]
23     k(i)=(Z(i)-1)*V(i);
24     t(i)=1/V(i);

```

```

25 end
26 [C,B,sig]=reglin(t',k');
27
28 //From the regression , we get intercept=B and slope=
    C,and thus ,
29 printf(" The value of second virial coefficient (B)
    is %e m^(3)/mol\n",B);
30 printf(" The value of third virial coefficient (C)
    is %e m^(6)/mol^(2)",C);

```

Scilab code Exa 2.9 Estimation of second virial coefficient

```

1 clear;
2 clc;
3
4 //Example - 2.9
5 //Page number - 51
6 printf("Example - 2.9 and Page number - 51\n\n");
7
8 //Given
9 T = 444.3; //[K] - Temperature
10 R = 8.314; //[J/mol*K] - Universal gas constant
11 B_11 = -8.1; //[cm^(3)/mol]
12 B_11 = -8.1*10^(-6); //[m^(3)/mol]
13 B_22 = -293.4*10^(-6); //[m^(3)/mol]
14 y1 = 0.5; // mole fraction // equimolar mixture
15 y2 = 0.5;
16
17 // For component 1 (methane)
18 Tc_1 = 190.6; //[K] - cricritical temperature
19 Vc_1 = 99.2; //[cm^(3)/mol] - cricritical molar volume
20 Zc_1 = 0.288; // critical compressibility factor
21 w_1 = 0.012; // acentric factor
22
23 // For component 2 (n-butane)

```



```

24 Tc_2 = 425.2; // [K]
25 Vc_2 = 255.0; // [cm^(3)/mol]
26 Zc_2 = 0.274;
27 w_2 = 0.199;
28
29 //Using virial mixing rule ,we get
30 Tc_12 = (Tc_1*Tc_2)^(1/2); // [K]
31 w_12 = (w_1 + w_2)/2;
32 Zc_12 = (Zc_1+Zc_2)/2;
33 Vc_12 = (((Vc_1)^(1/3) + (Vc_2)^(1/3))/2)^(3); // [cm
      ^ (3)/mol]
34 Vc_12 = Vc_12*10^(-6); // [cm^(3)/mol]
35 Pc_12 = (Zc_12*R*Tc_12)/Vc_12; // [N/m^(2) ]
36 Tr_12 = T/Tc_12; //Reduced temperature
37 B_0 = 0.083 - (0.422/(Tr_12)^(1.6));
38 B_1 = 0.139 - (0.172/(Tr_12)^(4.2));
39
40 //We know ,(B_12*Pc_12)/(R*Tc_12) = B_0 + (w_12*B_1)
41 B_12 = ((B_0+(w_12*B_1))*(R*Tc_12))/Pc_12; // [m^(3)/
      mol] - Cross coefficient
42 B = y1^(2)*B_11+2*y1*y2*B_12+y2^(2)*B_22; // [m^(3)/
      mol] - Second virial coefficient for mixture
43 B = B*10^(6); // [cm^(3)/mol]
44 printf(" The second virial coefficient ,(B) for the
      mixture of gas is %f cm^(3)/mol",B);

```

Scilab code Exa 2.10 Estimation of molar volume

```

1 clear;
2 clc;
3
4 //Example - 2.10
5 //Page number - 52
6 printf("Example - 2.10 and Page number - 52\n\n");
7

```

```

8 //Given
9 T = 71+273.15; // [K] - Temperature
10 P = 69*10^(5); // [N/m^(2)] - Pressure
11 y1 = 0.5; // [mol] - mole fraction of equimolar
    mixture
12 y2 = 0.5;
13 R = 8.314; // [J/mol*K] - Universal gas constant
14
15 //For component 1 (methane)
16 Tc_1 = 190.6; // [K] - Critical temperature
17 Pc_1 = 45.99*10^(5); // [N/m^(2)] - Critical pressure
18 Vc_1 = 98.6; // [cm^(3)/mol] - Critical volume
19 Zc_1 = 0.286; // Critical compressibility factor
20 w_1 = 0.012; // acentric factor
21
22 //For component 2 (hydrogen sulphide)
23 Tc_2 = 373.5; // [K]
24 Pc_2 = 89.63*10^(5); // [N/m^(2)]
25 Vc_2 = 98.5; // [cm^(3)/mol]
26 Zc_2 = 0.284;
27 w_2 = 0.094;
28
29 //For component 1
30 Tr_1 = T/Tc_1; //Reduced temperature
31 //At reduced temperature
32 B1_0 = 0.083 - (0.422/(Tr_1)^(1.6));
33 B1_1 = 0.139 - (0.172/(Tr_1)^(4.2));
34 //We know, (B*Pc)/(R*Tc) = B_0+(w*B_1)
35 B_11 = ((B1_0+(w_1*B1_1))*(R*Tc_1))/Pc_1; // [m^(3)/
    mol]
36
37 //Similarly for component 2
38 Tr_2 = T/Tc_2; //Reduced temperature
39 //At reduced temperature Tr_2,
40 B2_0 = 0.083 - (0.422/(Tr_2)^(1.6));
41 B2_1 = 0.139 - (0.172/(Tr_2)^(4.2));
42 B_22 = ((B2_0+(w_2*B2_1))*(R*Tc_2))/Pc_2; // [m^(3)/
    mol]

```

```

43
44 //For cross coefficient
45 Tc_12 = (Tc_1*Tc_2)^(1/2); // [K]
46 w_12 = (w_1 + w_2)/2;
47 Zc_12 = (Zc_1 + Zc_2)/2;
48 Vc_12 = (((Vc_1)^(1/3) + (Vc_2)^(1/3))/2)^(3); // [cm
      ^ (3)/mol]
49 Vc_12 = Vc_12*10^(-6); // [m^(3)/mol]
50 Pc_12 = (Zc_12*R*Tc_12)/Vc_12; // [N/m^(2) ]
51
52 //Now we have ,(B_12*Pc_12)/(R*Tc_12) = B_0+(w_12*B_1
      )
53 //where B_0 and B_1 are to be evaluated at Tr_12
54 Tr_12 = T/Tc_12;
55 //At reduced temperature Tr_12
56 B_0 = 0.083 - (0.422/(Tr_12)^(1.6));
57 B_1 = 0.139 - (0.172/(Tr_12)^(4.2));
58 B_12=((B_0 + (w_12*B_1))*R*Tc_12)/Pc_12; // [m^(3)/mol
      ]
59
60 //For the mixture
61 B = y1^(2)*B_11+2*y1*y2*B_12 + y2^(2)*B_22; // [m^(3)/
      mol]
62
63 //Now given virial equation is , Z=1+(B*P)/(R*T)
64 Z = 1 + (B*P)/(R*T);
65
66 //Also Z = (P*V)/(R*T). Therefore ,
67 V = (Z*R*T)/P; // [m^(3)/mol]
68
69 printf(" The molar volume of the mixture is %e m^(3)
      /mol",V);
70 //The value obtained is near the experimental value
      of V_exp = 3.38*10^(-4) m^(3)/mol

```

Scilab code Exa 2.11 Calculation of maximum temperature

```
1 clear;
2 clc;
3 funcprot(0);
4
5 // Example - 2.11
6 // Page number - 53
7 printf("Example - 2.11 and Page number - 53\n\n");
8
9 // Given
10 P = 6*10^(6); // [Pa] - Pressure
11 P_max = 12*10^(6); // [Pa] - Max pressure to which
    cylinder may be exposed
12 T = 280; // [K] - Temperature
13 R = 8.314; // [J/mol*K] - Universal gas constant
14
15 // (1). Assuming ideal gas behaviour ,
16 V_ideal = (R*T)/P; // [m^(3)/mol]
17 // Now when temperature and pressure are increased ,
    the molar volume remains same, as total volume and
    number of moles are same.
18 // For max pressure of 12 MPa, temperature is
19 T_max_ideal = (P_max*V_ideal)/R;
20 printf(" (1). The maximum temperature assuming ideal
    behaviour is %f K\n", T_max_ideal);
21
22 // (2). Assuming virial equation of state
23 // For component 1 (methane), at 280 K
24 Tc_1 = 190.6; // [K]
25 Pc_1 = 45.99*10^(5); // [N/m^(2)]
26 Vc_1 = 98.6; // [cm^(3)/mol]
27 Zc_1 = 0.286;
28 w_1 = 0.012;
29 Tr_1 = T/Tc_1; // Reduced temperature
30 B1_0 = 0.083 - (0.422/(Tr_1)^(1.6));
31 B1_1 = 0.139 - (0.172/(Tr_1)^(4.2));
32
```

```

33 //We know , (B*Pc)/(R*Tc) = B_0+(w*B_1)
34 B_11 = ((B1_0 + (w_1*B1_1))*(R*Tc_1))/Pc_1; // [m^(3)/
    mol]
35
36 //For component 2 (Propane)
37 Tc_2 = 369.8; // [K]
38 Pc_2 = 42.48*10^(5); // [N/m^(2) ]
39 Vc_2 = 200; // [cm^(3)/mol]
40 Zc_2 = 0.276;
41 w_2 = 0.152;
42 Tr_2 = T/Tc_2; // Reduced temperature
43 B2_0 = 0.083 - (0.422/(Tr_2)^(1.6));
44 B2_1 = 0.139 - (0.172/(Tr_2)^(4.2));
45 B_22 = ((B2_0 + (w_2*B2_1))*(R*Tc_2))/Pc_2; // [m^(3)/
    mol]
46
47 //For cross coeffcient
48 y1 = 0.8; //mole fraction of component 1
49 y2 = 0.2; //mole fraction of component 2
50 Tc_12 = (Tc_1*Tc_2)^(1/2); // [K]
51 w_12 = (w_1 + w_2)/2;
52 Zc_12 = (Zc_1 + Zc_2)/2;
53 Vc_12 = (((Vc_1)^(1/3) + (Vc_2)^(1/3))/2)^(3); // [cm
    ^ (3)/mol]
54 Vc_12 = Vc_12*10^(-6); // [m^(3)/mol]
55 Pc_12 = (Zc_12*R*Tc_12)/Vc_12; // [N/m^(2) ]
56 Tr_12 = T/Tc_12;
57
58 //At reduced temperature , Tr_12 ,
59 B_0 = 0.083 - (0.422/(Tr_12)^(1.6));
60 B_1 = 0.139 - (0.172/(Tr_12)^(4.2));
61 B_12 = ((B_0 + (w_12*B_1))*R*Tc_12)/Pc_12; // [m^(3)/
    mol]
62
63 //For the mixture
64 B = y1^(2)*B_11+2*y1*y2*B_12 + y2^(2)*B_22; // [m^(3)/
    mol]
65

```

```

66 //Now given virial equation is ,  $Z=1+(B*P)/(R*T)$ 
67  $Z = 1 + (B*P)/(R*T)$ ;
68 //Also  $Z = (P*V)/(R*T)$ . Therefore ,
69  $V_{real} = (Z*R*T)/P$ ; // [m3/mol]
70
71 // This molar volume remains the same as the volume
    and number of moles remains fixed.
72 // Since Z is a function of pressure and temperature ,
    we shall assume a temperature , calculate Z and
    again calculate temperature , till convergence is
    obtained.
73 // We will use the concept of iteration to compute
    the convergent value of temperature
74 // Let us start with the temperature at ideal
    conditions i.e T = 560 K,
75
76 T_prime = 560; // [K]
77 fault = 10;
78
79 while(fault > 1)
80 T_prime_r1 = T_prime/Tc_1;
81 B_prime1_0 = 7.7674*10(-3);
82 B_prime1_1 = 0.13714;
83 B_prime_11 = ((B_prime1_0 + (w_1*B_prime1_1))*(R*
    Tc_1))/Pc_1; // [m3/mol]
84
85 //Similarly for component 2,
86 T_prime_r2 = T_prime/Tc_2;
87 B_prime2_0 = -0.1343;
88 B_prime2_1 = 0.10887;
89 B_prime_22 = ((B_prime2_0 + (w_2*B_prime2_1))*(R*
    Tc_2))/Pc_2; // [m3/mol]
90
91 //For cross coefficient (assuming k12=0)
92 //Tc_12 , w_12 , Zc_12 , Vc_12 and Pc_12 have
    already been calculated above , now
93 T_prime_r12 = T_prime/Tc_12; //
94 //At reduced temperature , T_prime_r12 ,

```

```

95 B_prime_0 = 0.083 - (0.422/(T_prime_r12)^(1.6));
96 B_prime_1 = 0.139 - (0.172/(T_prime_r12)^(4.2));
97 B_prime_12 = ((B_prime_0+(w_12*B_prime_1))*R*Tc_12)/
    Pc_12; //[m^(3)/mol]
98
99 //For the mixture
100 B_prime = y1^(2)*B_prime_11 + 2*y1*y2*B_prime_12 +
    y2^(2)*B_prime_22; //[m^(3)/mol]
101 Z_prime = 1 + (B_prime*P_max)/(R*T_prime);
102 T_new = (P_max*V_real)/(Z_prime*R);
103 fault = abs(T_prime - T_new);
104 T_prime = T_new;
105 end
106
107 printf(" (2).The maximum temperature assuming the
    gas to follow virial equation of stste is %f K\n"
    ,T_new);

```

Scilab code Exa 2.12 Calculation of pressure

```

1 clear;
2 clc;
3
4 // Example - 2.12
5 // Page number - 64
6 printf("Example - 2.12 and Page number - 64\n\n");
7
8 //Given
9
10 V_vessel = 0.1; //[m^(3)]// Volume of vessel
11 T = 25 + 273.15; //[K] - Temperature
12 R = 8.314; //[J/mol*K] - Universal gas constant
13 m = 25*1000; //[g]// Mass of ethylene
14 Tc = 282.3; //[K] - Critical temperature
15 Pc = 50.40; //[bar] - Critical pressure

```

```

16 Pc = Pc*10^(5); // [N/m^(2)]
17 Zc = 0.281; // Critical compressibility factor
18 Vc = 131; // [cm^(3)/mol] - Critical volume
19 Vc = Vc*10^(-6); // [m^(3)/mol]
20 w = 0.087; // Acentric factor
21 M = 28.054; // Molecular weight of ethylene
22
23 n = m/M; // [mole] - No. of moles of ethylene
24 V = V_vessel/n; // [m^(3)/mol] - Molar volume
25
26 //Under Redlich Kwong equation of state, we have
27 a = (0.42748*(R^(2))*(Tc^(2.5)))/Pc; // [Pa*m^(6)*K
    ^ (1/2)/mol]
28 b = (0.08664*R*Tc)/Pc; // [m^(3)/mol]
29 P = ((R*T)/(V-b)) - (a/(T^(1/2)*V*(V+b))); // [N/m^(2)]
30 printf(" The required pressure using Redlich Kwong
    equation of state is %e N/m^(2)\n",P);
31
32 //For ideal gas equation of state,
33 P_ideal = (R*T)/V; // [N/m^(2)]
34 printf(" For ideal gas equation of state, the
    required pressure is %e N/m^(2)\n",P_ideal);

```

Scilab code Exa 2.13 Calculation of pressure

```

1 clear;
2 clc;
3
4 // Example - 2.13
5 // Page number - 65
6 printf("Example - 2.13 and Page number - 65\n\n");
7
8 //Given
9
10 V_vessel = 360*10^(-3); // [m^(3)] - volume of vessel

```



```

11 T = 62+273.15; // [K] - Temperature
12 R = 8.314; // [J/mol*K] - Universal gas constant
13 m = 70*1000; // [g] - Mass of carbon dioxide
14
15 // For carbon dioxide
16 Tc = 304.2; // [K] - Critical temperature
17 Pc = 73.83; // [bar] - Critical pressure
18 Pc = Pc*10^(5); // [N/m^(2)]
19 Zc = 0.274; // Critical compressibility factor
20 Vc = 94.0; // [cm^(3)/mol]
21 Vc = Vc*10^(-6); // [m^(3)/mol]
22 w = 0.224; // Acentric factor
23 M = 44.01; // Molecular weight of carbon dioxide
24
25 n = m/M; // [mol] - No. of moles
26 V = V_vessel/n; // [m^(3)/mol] // molar volume
27
28 // (1)
29 // Ideal gas behaviour
30 P_1 = (R*T)/V; // [N/m^(2)]
31 printf(" (1).The required pressure using ideal
    equation of state is %e N/m^(2)\n", P_1);
32
33 // (2)
34 // Virial equation of state , Z = 1 + (B*P)/(R*T)
35 // (P*V)/(R*T) = 1 + (B*P)/(R*T) , and thus P = (R*T)
    /(V - B). Now
36 Tr = T/Tc; // Reduced temperature
37 // At reduced temperature Tr,
38 B_0 = 0.083 - (0.422/(Tr)^(1.6));
39 B_1 = 0.139 - (0.172/(Tr)^(4.2));
40 B = ((B_0 + (w*B_1))*(R*Tc))/Pc; // [m^(3)/mol]
41 P_2 = (R*T)/(V - B); // [N/m^(2)]
42 printf(" (2).The required pressure using given
    virial equation of state is %e N/m^(2)\n", P_2);
43
44 // (3)
45 // Virial equation of state , Z = 1 + (B/V)

```

```

46 // (P*V)/(R*T) = 1 + (B/V)
47 P_3 = ((R*T)/V) + (B*R*T)/(V^(2)); // [N/m^(2)]
48 printf(" (3).The required pressure using given
    virial equation of state is %e N/m^(2)\n",P_3);
49
50 // (4)
51 // Van der Walls equation of state ,P = ((R*T)/(V-b))
    - a/(V^(2))
52 a = (27*(R^(2))*(Tc^(2)))/(64*Pc); // [Pa*m^(6)/mol
    ^ (2)]
53 b = (R*Tc)/(8*Pc); // [m^(3)/mol]
54 P_4 = ((R*T)/(V-b)) - a/(V^(2)); // [N/m^(2)]
55 printf(" (4).The required pressure using van der
    Walls equation of state is %e N/m^(2)\n",P_4);
56
57 // (5)
58 // Redlich Kwong equation of state ,
59 a_1 = (0.42748*(R^(2))*(Tc^(2.5)))/Pc; // [Pa*m^(6)*K
    ^ (1/2)/mol]
60 b_1 = (0.08664*R*Tc)/Pc; // [m^(3)/mol]
61 P_5 = ((R*T)/(V - b_1)) - (a_1/(T^(1/2)*V*(V + b_1))
    ); // [N/m^(2)]
62 printf(" (5).The required pressure using Redlich
    Kwong equation of state is %e N/m^(2)\n",P_5);

```

Scilab code Exa 2.14 Determination of compressibility factor

```

1 clear;
2 clc;
3 funcprot(0);
4
5 // Example - 2.14
6 // Page number - 66
7 printf("Example - 2.14 and Page number - 66\n\n");
8

```

```

9 //Given
10 T = 500+273.15; //[K] - Temperature
11 R = 8.314; //[J/mol*K] - Universal gas constant
12 P = 325*1000; //[Pa] - Pressure
13 Tc = 647.1; //[K] - Cricritical temperature
14 Pc = 220.55; //[bar] - Cricritical pressure
15 Pc = Pc*10^(5); //[N/m^(2)]
16
17 //(1)
18 // Van der Walls equation of state ,
19 a = (27*(R^(2))*(Tc^(2)))/(64*Pc); //[Pa*m^(6)/mol
    ^ (2)]
20 b = (R*Tc)/(8*Pc); //[m^(3)/mol]
21 // The cubic form of van der Walls equation of state
    is given by,
22 // V^(3)-(b+(R*T)/P)*V^(2)+(a/P)*V-(a*b)/P=0
23 // Solving the cubic equation
24 def f(' [y]=f(V) ', 'y=V^(3)-(b+(R*T)/P)*V^(2)+(a/P)*V-(
    a*b)/P');
25 V_1 = fsolve(1,f);
26 V_2 = fsolve(10,f);
27 V_3 = fsolve(100,f);
28 // The above equation has 1 real and 2 imaginary
    roots. We consider only real root ,
29 Z_1 = (P*V_1)/(R*T); //compressibility factor
30 printf(" (1).The compressibility factor of steam
    using van der Walls equation of state is %f\n",
    Z_1);
31
32 //(2)
33
34 //Redlich Kwong equation of state ,
35 a_1 = (0.42748*(R^(2))*(Tc^(2.5)))/Pc; //[Pa*m^(6)*K
    ^ (1/2)/mol]
36 b_1 = (0.08664*R*Tc)/Pc; //[m^(3)/mol]
37 // The cubic form of Redlich Kwong equation of state
    is given by,
38 // V^(3)-((R*T)/P)*V^(2)-((b_1^(2)))+((b_1*R*T)/P)-(a

```

```

    /(T^(1/2)*P))*V-(a*b)/(T^(1/2)*P)=0
39 //Solving the cubic equation
40 deff(' [y]=f1(V) ', 'y=V^(3) -((R*T)/P)*V^(2) -((b_1^(2))
    +((b_1*R*T)/P)-(a_1/(T^(1/2)*P)))*V-(a_1*b_1)/(T
    ^ (1/2)*P) ');
41 V_4=fsolve(1,f1);
42 V_5=fsolve(10,f1);
43 V_6=fsolve(100,f1);
44 // The above equation has 1 real and 2 imaginary
    roots. We consider only real root ,
45 // Thus compressibility factor is
46 Z_2 = (P*V_4)/(R*T); //compressibility factor
47 printf(" (2).The compressibility factor of steam
    using Redlich Kwong equation of state is %f\n",
    Z_2);

```

Scilab code Exa 2.15 Determination of molar volume

```

1 clear;
2 clc;
3
4 // Example - 2.15
5 // Page number - 67
6 printf("Example - 2.15 and Page number - 67\n\n");
7
8 //Given
9 T = 250+273.15; // [K]
10 R = 8.314; // [J/mol*K]
11 P = 39.76; // [bar] Vapour pressure of water at T
12 P = P*10^(5); // [N/m^(2)]
13 Tc = 647.1; // [K] - Cricritical temperature
14 Pc = 220.55*10^(5); // [N/m^(2)] - Cricritical pressure
15 w = 0.345; // Acentric factor
16 M = 18.015; // Molecular weight of water
17

```

```

18 // Using peng-Robinson equation of stste
19 m = 0.37464 + 1.54226*w - 0.26992*w^(2);
20 Tr = T/Tc;
21 alpha = (1 + m*(1 - Tr^(1/2)))^(2);
22 a = ((0.45724*(R*Tc)^(2))/Pc)*alpha; // [Pa*m^(6)/mol
    ^ (2)]
23 b = (0.07780*R*Tc)/Pc; // [m^(3)/mol]
24 // Cubic form of Peng-Robinson equation of stste is
    given by
25 // V^(3) + (b-(R*T)/P)*V^(2) - ((3*b^(2)) + ((2*R*T*
    b)/P) - (a/P))*V+b^(3) + ((R*T*(b^(2)))/P) - ((a*b
    )/P) = 0;
26 // Solving the cubic equation
27 def f(' [y]=f(V) ', 'y=V^(3)+(b-(R*T)/P)*V^(2)-((3*b^(2)
    )+((2*R*T*b)/P)-(a/P))*V+b^(3)+((R*T*(b^(2)))/P)
    -((a*b)/P) ');
28 V_1 = fsolve(-1,f);
29 V_2 = fsolve(0,f);
30 V_3 = fsolve(1,f);
31 //The largest root is for vapour phase ,
32 V_vap = V_3; // [m^(3)/mol] - Molar volume (saturated
    vapour)
33 V_vap = V_vap*10^(6)/M; // [cm^(3)/g]
34
35 printf(" The moar volume of saturated water in the
    vapour phase (V_vap) is %f cm^(3)/g\n",V_vap);
36
37 //The smallest root is for liquid phase ,
38 V_liq = V_1; // [m^(3)/mol] - molar volume (saturated
    liquid)
39 V_liq = V_liq*10^(6)/M; // [cm^(3)/g]
40 printf(" The moar volume of saturated water in the
    liquid phase (V_liq) is %f cm^(3)/g\n",V_liq);
41
42 //From steam table at 250 C, V_vap = 50.13 [cm^(3)/g
    ] and V_liq = 1.251 [cm^(3)/g].
43 printf(" From steam table at 250 C, V_vap = 50.13 [
    cm^(3)/g] and V_liq = 1.251 [cm^(3)/g]");

```

Scilab code Exa 2.16 Calculation of volume

```
1 clear;
2 clc;
3 funcprot(0);
4
5 // Example - 2.16
6 // Page number - 68
7 printf("Example - 2.16 and Page number - 68\n\n");
8
9 //Given
10 T = 500+273.15; // [K] - Temperature
11 P = 15; // [atm] - Pressure
12 P = P*101325; // [N/m^(2)]
13 R = 8.314; // [J/mol*K] - Universal gas constant
14 Tc = 190.6; // [K] - Cricritical temperature
15 Pc = 45.99*10^(5); // [N/m^(2)] - Cricritical pressure
16 Vc = 98.6; // [cm^(3)/mol] - Cricritical molar volume
17 Zc = 0.286; // Critical compressibility factor
18 w = 0.012; // Acentric factor
19
20 //(1)
21 //Virial equation of state ,Z = 1 + (B*P)/(R*T)
22 Tr_1 = T/Tc; //Reduced temperature
23 B_0 = 0.083-(0.422/(Tr_1)^(1.6));
24 B_1 = 0.139-(0.172/(Tr_1)^(4.2));
25 // We know ,(B*Pc)/(R*Tc) = B_0+(w*B_1)
26 B = ((B_0+(w*B_1))*(R*Tc))/Pc; // [m^(3)/mol]// second
    virial coefficient
27 Z = 1 + (B*P)/(R*T); //compressibility factor
28 //(P*V)/(R*T)=1+(B*P)/(R*T), and thus ,
29 V_1 = (Z*R*T)/P; // [m^(3)/mol]
30 printf(" (1).The molar volume of methane using given
    virial equation is %e m^(3)/mol\n",V_1);
```

```

31
32 // (2).
33 // Virial equation of state,  $Z = 1 + (B/V)$ 
34 // Also,  $Z = (P*V)/(R*T)$ . Substituting the value of Z,
    we get
35 //  $V^2 - ((R*T)/P)*V - ((B*R*T)/P) = 0$ . Solving the
    quadratic equation
36 deff ( ' [y]=f(V) ', 'y= $V^2 - ((R*T)/P)*V - ((B*R*T)/P)$  ');
37 V2_1=fsolve(0,f);
38 V2_2=fsolve(1,f);
39 // Out of two roots, we will consider only positive
    root
40 printf (" (2). The molar volume of methane using given
    virial equation is %e m(3)/mol\n", V2_2);
41
42 // (3)
43 // Van der Waals equation of state,
44 //  $(P + (a/V^2))*(V - b) = R*T$ 
45 a_3 = (27*(R(2))*(Tc(2)))/(64*Pc); // [Pa*m(6)/mol
    (2)]
46 b_3 = (R*Tc)/(8*Pc); // [m(3)/mol]
47 // The cubic form of van der Waals equation of state
    is given by,
48 //  $V^3 - (b + (R*T)/P)*V^2 + (a/P)*V - (a*b)/P =
    0$ 
49 // Solving the cubic equation
50 deff ( ' [y]=f1(V) ', 'y= $V^3 - (b_3 + (R*T)/P)*V^2 + (a_3/P)
    *V - (a_3*b_3)/P$  ');
51 V3_1=fsolve(1,f1);
52 V3_2=fsolve(10,f1);
53 V3_3=fsolve(100,f1);
54 // The above equation has 1 real and 2 imaginary
    roots. We consider only real root.
55 printf (" (3). The molar volume of methane using van
    der Waals equation of state is %e m(3)/mol\n",
    V3_1);
56
57 // (4)

```

```

58 // Redlich Kwong equation of state
59 a_4 = (0.42748*(R^(2))*(Tc^(2.5)))/Pc; // [Pa*m^(6)*K
      ^ (1/2)/mol]
60 b_4 = (0.08664*R*Tc)/Pc; // [m^(3)/mol]
61 // The cubic form of Redlich Kwong equation of state
      is given by,
62 //  $V^3 - ((R*T)/P)*V^2 - ((b_1^2)/P) + ((b_1*R*T)/P) - (a/(T^{1/2}*P))*V - (a*b)/(T^{1/2}*P) = 0$ 
63 // Solving the cubic equation
64 deff(' [y]=f2(V) ', 'y=V^3 -((R*T)/P)*V^2 -((b_4^2)/P)
      +((b_4*R*T)/P) - (a_4/(T^(1/2)*P))*V - (a_4*b_4)/(T
      ^ (1/2)*P) ');
65 V4_1=fsolve(1,f2);
66 V4_2=fsolve(10,f2);
67 V4_3=fsolve(100,f2);
68 //The above equation has 1 real and 2 imaginary
      roots. We consider only real root.
69 printf(" (4).The molar volume of methane using
      Redlich Kwong equation of state is %e m^(3)/mol\n
      ",V4_1);
70
71 // (5)
72 // Using Peng–Robinson equation of state
73 m = 0.37464 + 1.54226*w - 0.26992*w^(2);
74 Tr_5 = T/Tc;
75 alpha = (1 + m*(1 - Tr_5^(1/2)))^(2);
76 a = ((0.45724*(R*Tc)^(2))/Pc)*alpha; // [Pa*m^(6)/mol
      ^ (2)]
77 b = (0.07780*R*Tc)/Pc; // [m^(3)/mol]
78 // Cubic form of Peng–Robinson equation of stste is
      given by
79 //  $V^3 + (b - (R*T)/P)*V^2 - ((3*b^2)/P) + ((2*R*T*b)/P) - (a/P)*V + b^3 + ((R*T*(b^2))/P) - ((a*b)/P) = 0;$ 
80 // Solving the cubic equation
81 deff(' [y]=f3(V) ', 'y=V^3 + (b - (R*T)/P)*V^2 - ((3*b
      ^ (2)) + ((2*R*T*b)/P) - (a/P))*V + b^3 + ((R*T*(b^2))/
      /P) - ((a*b)/P) ');
82 V5_1=fsolve(-1,f3);

```



```

83 V5_2=fsolve(0,f3);
84 V5_3=fsolve(1,f3);
85 //The largest root is for vapour phase,
86 //The largest root is only considered as the
    systemis gas
87 printf(" (5).The molar volume of methane using Peng-
    Robinson equation of state is %e m^(3)/mol\n",
    V5_3);

```

Scilab code Exa 2.17 Estimation of compressibility factor

```

1 clear;
2 clc;
3 funcprot(0);
4
5 // Example - 2.17
6 // Page number - 70
7 printf("Example - 2.17 and Page number - 70\n\n");
8
9 //Given
10 T = 310.93;//[K] - Temperature
11 P = 2.76*10^(6);//[N/m^(2)] - Pressure
12 R = 8.314;//[J/mol*K] - Universal gas constant
13 y1 = 0.8942; // Mole fraction of component 1 (methane
    )
14 y2 = 1-y1; // Mole fraction of component 2 (n-butane)
15
16 //For component 1 (methane)
17 Tc_1 = 190.58;//[K] - Cricritical temperature
18 Pc_1 = 46.05;//[bar] - Cricritical pressure
19 Pc_1 = Pc_1*10^(5);//[N/m^(2)]
20 Zc_1 = 0.288; // Critical compressibility factor
21 Vc_1 = 99.1;//[cm^(3)/mol]
22 Vc_1 = Vc_1*10^(-6);//[m^(3)/mol]
23 w_1 = 0.011; // Acentric factor

```

```

24
25 //For component 2 (n-butane)
26 Tc_2 = 425.18; // [K] - Ccritical temperature
27 Pc_2 = 37.97; // [bar] - Ccritical pressure
28 Pc_2 = Pc_2*10^(5); // [N/m^(2)]
29 Zc_2 = 0.274; // Critical compressibility factor
30 Vc_2 = 255.1; // [cm^(3)/mol]
31 Vc_2 = Vc_2*10^(-6); // [m^(3)/mol]
32 w_2 = 0.193; // Acentric factor
33
34 // (1)
35 // Virial equation of state , Z = 1 + (B*P)/(R*T)
36 // For component 1 (methane)
37 Tr_1 = T/Tc_1; //Reduced temperature
38 // At reduced temperature
39 B1_0 = 0.083 - (0.422/(Tr_1)^(1.6));
40 B1_1 = 0.139 - (0.172/(Tr_1)^(4.2));
41 // We know ,(B*Pc)/(R*Tc) = B_0+(w*B_1)
42 B_11 = ((B1_0+(w_1*B1_1))*(R*Tc_1))/Pc_1; // [m^(3)/
    mol]
43
44 //Similarly for component 2
45 Tr_2 = T/Tc_2; //Reduced temperature
46 //At reduced temperature Tr_2 ,
47 B2_0 = 0.083 - (0.422/(Tr_2)^(1.6));
48 B2_1 = 0.139 - (0.172/(Tr_2)^(4.2));
49 B_22 = ((B2_0 + (w_2*B2_1))*(R*Tc_2))/Pc_2; // [m^(3)/
    mol]
50
51 //For cross coeffcient
52 Tc_12 = (Tc_1*Tc_2)^(1/2); // [K]
53 w_12 = (w_1 + w_2)/2;
54 Zc_12 = (Zc_1 + Zc_2)/2;
55 Vc_12 = (((Vc_1)^(1/3)+(Vc_2)^(1/3))/2)^(3); // [m^(3)
    /mol]
56 Pc_12 =(Zc_12*R*Tc_12)/Vc_12; // [N/m^(2)]
57
58 //Now we have ,(B_12*Pc_12)/(R*Tc_12) = B_0+(w_12*B_1

```

```

)
59 //where B_0 and B_1 are to be evaluated at Tr_12
60 Tr_12 = T/Tc_12;
61 //At reduced temperature Tr_12
62 B_0 = 0.083 - (0.422/(Tr_12)^(1.6));
63 B_1 = 0.139 - (0.172/(Tr_12)^(4.2));
64 B_12 = ((B_0+(w_12*B_1))*R*Tc_12)/Pc_12; //[m^(3)/mol
    ]
65
66 //For the mixture
67 B = y1^(2)*B_11+2*y1*y2*B_12+y2^(2)*B_22; //[m^(3)/
    mol]
68 Z_1 = 1+(B*P)/(R*T); //compressibility factor
69 printf(" (1).The compressibility factor of mixture
    using Virial equation of state is %f\n",Z_1);
70
71 // (2)
72 // Pseudo reduced method.
73 T_pc = (y1*Tc_1)+(y2*Tc_2); //[K] - Cricritical
    temperature
74 P_pc = (y1*Pc_1)+(y2*Pc_2); //[N/m^(2)] - Cricritical
    pressure
75 w = (y1*w_1)+(y2*w_2); // Acentric factor
76 T_pr = T/T_pc; // Reduced temperature
77 P_pr = P/P_pc; // Reduced pressure
78 //At this value of Tpr,
79 B0 = 0.083 - (0.422/(T_pr)^(1.6));
80 B1 = 0.139 - (0.172/(T_pr)^(4.2));
81 Z0 = 1 + B0*(P_pr/T_pr);
82 Z1 = B1*(P_pr/T_pr);
83 Z = Z0 + w*Z1;
84 printf(" (2).The compressibility factor of mixture
    using pseudo reduced method is %f\n",Z);
85
86 // (3)
87 // Redlich Kwong equation of state is given by
88 //  $P = ((R*T)/(V-b)) - (a/(T^{1/2}*V*(V+b)))$ 
89 // For methane ,component 1

```

```

90 a_1 = (0.42748*(R^(2))*(Tc_1^(2.5)))/Pc_1; // [Pa*m
      ^ (6)*K^(1/2)/mol]
91 b_1 = (0.08664*R*Tc_1)/Pc_1; // [m^(3)/mol]
92 //For n-butane , component 2
93 a_2 = (0.42748*(R^(2))*(Tc_2^(2.5)))/Pc_2; // [Pa*m
      ^ (6)*K^(1/2)/mol]
94 b_2 = (0.08664*R*Tc_2)/Pc_2; // [m^(3)/mol]
95 //For the mixture
96 a_12 = (a_1*a_2)^(1/2); // [Pa*m^(6)*K^(1/2)/mol]
97 a = y1^(2)*a_1 + 2*y1*y2*a_12 + y2^(2)*a_2; // [Pa*m
      ^ (6)*K^(1/2)/mol]
98 b = (y1*b_1) + (y2*b_2); // [m^(3)/mol]
99 // The cubic form of Redlich Kwong equation of state
      is given by,
100 // V^(3) - ((R*T)/P)*V^(2) - ((b_1^(2)) + ((b_1*R*T)
      /P) - (a/(T^(1/2)*P))*V - (a*b)/(T^(1/2)*P) = 0
101 // Solving the cubic equation
102 deff(' [y]=f(V) ', 'y=V^(3) -((R*T)/P)*V^(2) -((b^(2))+((
      b*R*T)/P)-(a/(T^(1/2)*P)))*V-(a*b)/(T^(1/2)*P) ');
103 V_1=fsolve(1,f);
104 V_2=fsolve(10,f);
105 V_3=fsolve(100,f);
106 // Thus compressibility factor is
107 Z_3 = (P*V_1)/(R*T); //compressibility factor
108 printf(" (3).The compressibility factor of mixture
      using Redlich Kwong equation of state is %f\n",
      Z_3);

```

Chapter 3

The First Law and Its Applications

Scilab code Exa 3.1 Calculation of temperature

```
1 clear;
2 clc;
3 funcprot(0);
4
5 //Example - 3.1
6 //Page number - 80
7 printf("Example - 3.1 and Page number - 80\n\n");
8
9 // Given
10 V_vessel = 4*10^(-3);//[m^(-3)] - Volume of vessel
11 T = 200+273.15;//[K] - Temperature
12 R = 8.314;//[J/mol*K] - Universal gas constant
13 P = 1.5*10^(6);//[Pa] - Pressure
14 Q = 40*1000;//[J] - Heat input
15 // From steam table at 200 C, Psat=1.55549 MPa,
    therefore the steam is superheated.
16
17 // (1)
18 // Using steam table ,at 1.5 MPa and 200 C,
```

```

19 V_1 = 0.1325; //[m^(3)/mol] - Specific volume
20 U_1 = 2598.1; //[kJ/kg] - Specific internal energy
21 // From first law under constant pressure ,
22 // Q - m*P*(V2 - V1) = m*(U2 - U1)
23 m = V_vessel/V_1; //[kg] - Mass of system
24 // Putting the values ,the above equation becomes
25 // 45283*(V2 - 0.1325) + 30.1887*(U2 - 2598.1) =
    40000
26 // From steam table at 700 C LHS is 33917.0 and at
    800 C, it is 40925.3.
27 // Therefore the final temperature lies between 700
    and 800 C
28 printf(" (1).From steam table the final temperature
    lies between 700 and 800 C\n");
29
30 // Alternate method
31 // Here we use first law at constant pressure ,
32 // Q = m*(H_2 - H_1)
33 H_1 = 2796.8; //[kJ/kg]
34 // Substituting the values ,
35 // 40 = 0.0301887*(H_2 - 2796.8)
36 H_2 = (40/0.0301887) + 2796.9; //[kJ/kg]
37 // Therefore ,final enthalpy is (H2) 4121.8 [kJ/kg]
    and pressure is 1.5 [MPa].
38 // From steam table at 1.5 [MPa]and 700 C, enthalpy
    is 3920.3 [kJ/kg] and at 1.5 [MPa]and 800 C,
    enthalpy is 4152.6 [kJ/kg]
39 printf("\tAlternate method\n");
40 printf("\tBy linear interpolation we get the
    temperature at which enthalpy is 4121.8 kJ/kg to
    be 786.74 C\n\n");
41
42 // (2)
43 // Assuming ideal behaviour .
44 n = (P*V_vessel)/(R*T); //[mol] - No of moles
45 M = 18.015; // Molecular weight of water
46 m_2 = n*M; //[g] - mass of moles
47 Cp_1 = 7.7 + 0.04594*10^(-2)*T + 0.2521*10^(-5)*T

```

```

    ^ (2) - 0.8587*10^(-9)*T^(3); // [cal/mol*K] - Heat
    capacity at constant pressure
48 R0 = 1.987; // [cal/mol*K] - universal gas constant
49 Cv_1 = Cp_1 - R0; // [cal/mol*K] - Heat capacity at
    constant volume
50 Cv_1 = Cv_1*4.184/M; // [J/g*K]
51 T1 = T;
52 // From 1st law energy balance for constant pressure
    , we have  $Q = W = m(\Delta U)$ 
53 //  $Q = P(V_2 - V_1)m = mCv(T_2 - T_1)$ 
54 //  $Q = P((T_2/T_1) - 1)V_1m = mCv(T_2 - T_1)$ 
55 // But,  $(V_1Cv) =$  initial total volume of system =
    V_vessel
56 //  $Q = P((T_2/T_1) - 1)V_vessel = m_2Cv_0(T_2 - T_1)$ ;
57 def ( ' [y]=f(T2) ', 'y=Q-P*((T2/T1)-1)*V_vessel-m_2*
    Cv_1*(T2-T1)' );
58 T2_1 = fsolve(1,f);
59 //The heat capacity should be evaluated at mean
    temperature
60 T_mean = (T1 + T2_1)/2;
61 Cp_2 = 7.7 + 0.04594*10^(-2)*T_mean+0.2521*10^(-5)*
    T_mean^ (2) - 0.8587*10^(-9)*T_mean^ (3); // [cal/mol
    *K] - Heat capacity at constant pressure
62 Cv_2 = Cp_2-R0; // [cal/mol*K] - Heat capacity at
    constant volume
63 Cv_2 = Cv_2*4.184/M; // [J/g*K]
64 //Now again solving the equation  $Q = P((T_2/T_1) - 1)V_1m$ 
    =  $mCv(T_2 - T_1)$ , for  $Cv = Cv_2$ 
65 def ( ' [y]=f1(T2) ', 'y=Q-P*((T2/T1)-1)*V_vessel-m_2*
    Cv_2*(T2-T1)' );
66 T2_2 = fsolve(1,f1);
67 printf(" (2).The temperature assuming ideal
    behaviour is %f K\n", T2_2);
68
69 // Alternate method
70 // From 1st law at constant pressure , we have  $Q = mCp(T_2 - T_1)$ 
71 T2_3 = Q/(m_2*(Cp_1*4.184/M))+T1;

```

```

72 //We can calculate the mean temperature as done
    above
73 T_mean1 = (T1 + T2_3)/2; //[J/g*K]
74 //The heat capacity should be evaluated at mean
    temperature
75 Cp_3 = 7.7 + 0.04594*10(-2)*T_mean1 +
    0.2521*10(-5)*T_mean1(2) - 0.8587*10(-9)*T_mean1
    (3); //[cal/mol*K] - Heat capacity at constant
    presure
76 Cp_3 = Cp_3*4.184/M; //[J/g*K]
77 // Again solving the equation Q = m*Cp(T2-T1), for
    Cp=Cp_3
78 T2_4 = Q/(m_2*Cp_3) + T1;
79 printf("\tAlternate method\n");
80 printf("\tThe temperature assuming ideal behaviour (
    alternate method) is %f K\n",T2_4);

```

Scilab code Exa 3.2 Calculation of heat required

```

1 clear;
2 clc;
3
4 //Example - 3.2
5 //Page number - 83
6 printf("Example - 3.2 and Page number - 83\n\n");
7
8 //Given
9 V_tank = 1; //[m(3)] - Volume of the tank
10 V_liq = 0.05; //[m(3)] - Volume of saturated water
11 V_vap = 0.95; //[m(3)] - Volume of saturated vapour
12 P = 1; //[bar] - Pressure
13 V_liq_sat = 0.001043; //[m(3)/kg] - Specific volume
    of saturated water
14 V_vap_sat = 1.6940; //[m(3)/kg] - Specific volume of
    saturatd vapour

```



```

15 U_liq_sat = 417.4;//[kJ/kg] - Saturated liquid
    internal energy
16 U_vap_sat = 2506.1;//[kJ/kg] - Saturated vapour
    internal energy
17 m = (V_liq/V_liq_sat) + (V_vap/V_vap_sat);//[kg] -
    Total mass of water
18 U_1 = (V_liq/V_liq_sat)*U_liq_sat + (V_vap/V_vap_sat
    )*U_vap_sat;//[kJ] - Total internal energy
19
20 // At final state ,which is saturated vapour
21 V = V_tank/m;//[m^(3)/kg] - Molar volume
22 // From saturated steam table at 8 MPa,as reported
    in the book V_vap = 0.02352[m^(3)/kg] and U_vap =
    2569.8[kJ/kg]
23 // At 9 MPa, Vv = 0.02048[m^(3)/kg] and Uv = 2557.8[
    kJ/kg]
24 // Therefore final state pressure of the system (
    from interpolation) is 8.954 [MPa] and internal
    energy of saturated vapour is 2558.35 [kJ/kg]
25 U_2 = m*2558.35;//[kJ] - Final total internal energy
26 del_Ut = U_2 - U_1;//[kJ]
27 //we have , del_U = Q - W
28 //Here work done is zero because volume is rigid .
29 Q = del_Ut;//[kJ]
30 Q = del_Ut*10^(-3);//[MJ]
31 printf(" The amount of heat to be added is %f MJ" , Q
    );

```

Scilab code Exa 3.3 Calculation of temperature internal energy and enthalpy

```

1 clear;
2 clc;
3
4 //Example - 3.3

```

```

5 //Page number - 83
6 printf("Example - 3.3 and Page number - 83\n\n");
7
8 //Given
9 M_vap_sat = 0.22; //[kg] - mass of saturated vapour
10 M_liq_sat = 1.78; //[kg] - mass of saturated liquid
11 P = 700; //[kPa] - Pressure
12
13 //At P=700 kPa, the system is saturated, from steam
    table as reported in the book
14 T_sat1 = 164.97; //[C]
15 V_liq_1 = 0.001108; //[m^(3)/kg]
16 V_vap_1 = 0.2729; //[m^(3)/kg]
17 Vt_1 = V_liq_1*M_liq_sat + V_vap_1*M_vap_sat; //[m
    ^{(3)}] - total volume
18
19 //At final state, P = 8 MPa
20 T_sat2 = 295.06; //[C]
21 V_liq_2 = 0.001384; //[m^(3)/kg]
22 V_vap_2=0.02352; //[m^(3)/kg]
23 Vt_2 = Vt_1; // Since the volume is rigid.
24 // Since the volume of 2 kg of vapour is 0.062 [m
    ^{(3)}]
25 V = Vt_2/2; //[m^(3)/kg] - specific volume
26
27 // (a)
28 // From steam table at 8 [MPa] and 350 [C], V=0.02995[
    m^(3)/kg];
29 V_1 = 0.02995; //[m^(3)/kg]
30 // And at 8 [MPa] and 400 [C],
31 V_2 = 0.03432; //[m^(3)/kg]
32 // By interpolation,
33 T = ((V-V_1)/(V_2 - V_1))*(400-350)+350;
34 printf(" (a). The final temperature is %f c\n", T);
35
36 // (b)
37 // From steam table
38 U_1 = 2747.7; //[kJ/kg]

```

```

39 H_1 = 2987.3; //[kJ/kg]
40 // And at 8 [MPa] and 400 C,
41 U_2 = 2863.8; //[kJ/kg]
42 H_2 = 3138.3; //[kJ/kg]
43 // Therefore at T = 362.01 C
44 U = U_1 + ((U_2 - U_1) / (400 - 350)) * (T - 350);
45 printf(" (b). The internal energy is %f kJ/kg\n", U);
46
47 // (c)
48 H = H_1 + ((H_2 - H_1) / (400 - 350)) * (T - 350);
49 printf(" (b). The enthalpy is %f kJ/kg\n", H);

```

Scilab code Exa 3.4 Calculation of work done

```

1 clear;
2 clc;
3
4 // Example - 3.4
5 // Page number - 85
6 printf(" Example - 3.4 and Page number - 85\n\n");
7
8 // Given
9 T = 300; //[K] - Temperature
10 P1 = 1; //[bar] - Initial pressure
11 P1 = P1 * 10^(5); //[N/m^(2)]
12 P2 = 8; //[bar] - Final pressure
13 P2 = P2 * 10^(5); //[N/m^(2)]
14 R = 8.314; //[J/mol*K] - Universal gas constant
15 Tc = 126.2; //[K] - Critical temperature
16 Pc = 34; //[bar] - Critical pressure
17 Pc = Pc * 10^(5); //[N/m^(2)]
18 w = 0.038; // Acentric factor
19
20 // w = integral(Pdv)
21 // Z = 1 + (B/V)

```

```

22 // (P*V)/(R*T) = 1 + (B/V)
23 // P = (R*T)/V + (B*R*T)/V^(2)
24 // w = integrate(' (R*T/V) + (B*R*T)/V^(2) ', 'V', V1, V2
    )
25 // Under isothermal conditions ,
26 // w = R*T*log(V2/V1) - B*R*T*((1/V2) - (1/V1));
27 // The second virial coefficient at state 1 is same
    as at state 2, as the temperature is the same i.e.,
    T=300 [K]
28 Tr = T/Tc;
29 B_0 = 0.083 - (0.422/(Tr)^(1.6));
30 B_1 = 0.139 - (0.172/(Tr)^(4.2));
31 B = ((B_0+(w*B_1))*(R*Tc))/Pc; // [m^(3)/mol]
32
33 // Now we have to calculate molar volume i.e V1 and
    V2 at given conditions
34 // At state 1,
35 deff(' [y]=f(V) ', 'y=V^(2)-(R*T/P1)*V-(B*R*T)/P1 ');
36 V_1 = fsolve(-1,f);
37 V_2 = fsolve(1,f);
38 // We will take root near to (R*T)/P, i.e V_2
39 V1 = V_2;
40
41 // At state 2,
42 deff(' [y]=f1(V) ', 'y=V^(2)-(R*T/P2)*V-(B*R*T)/P2 ');
43 V_3=fsolve(-1,f1);
44 V_4=fsolve(1,f1);
45 V2 = V_4;
46 // The work done is thus ,
47 w = R*T*log(V2/V1) - B*R*T*((1/V2) - (1/V1)); // [J]
48 w = w*10^(-3); // [kJ]
49
50 printf(" The work done is %f kJ/mol\n",w);
51 printf(" Negative sign indicates that work is done
    on the gas");

```

Scilab code Exa 3.5 Calculation of work done

```
1 clear;
2 clc;
3
4 //Example - 3.5
5 //Page number - 86
6 printf("Example - 3.5 and Page number - 86\n\n");
7
8 //Given
9
10 T = 300; // [K] - Temperature
11 P1 = 1; // [bar] - Initial pressure
12 P1 = P1*10^(5); // [N/m^(2)]
13 P2 = 8; // [bar] - Final pressure
14 P2 = P2*10^(5); // [N/m^(2)]
15 R = 8.314; // [J/mol*K] - Universal gas constant
16 y1 = 0.21; // Mole fraction of component 1 (oxygen)
17 y2 = 0.79; // Mole fraction of component 1 (nitroen)
18
19 // For component 1 (Oxygen)
20 Tc_1 = 154.6; // [K]
21 Pc_1 = 50.43*10^(5); // [N/m^(2)]
22 Vc_1 = 73.4; // [cm^(3)/mol]
23 Zc_1 = 0.288;
24 w_1 = 0.022;
25
26 //For component 2 (Nitrogen)
27 Tc_2 = 126.2; // [K]
28 Pc_2 = 34*10^(5); // [N/m^(2)]
29 Vc_2 = 89.2; // [cm^(3)/mol]
30 Zc_2 = 0.289;
31 w_2 = 0.038;
32
```

```

33 //For component 1
34 Tr_1 = T/Tc_1;//Reduced temperature
35 //At reduced temperature
36 B1_0 = 0.083 - (0.422/(Tr_1)^(1.6));
37 B1_1 = 0.139 - (0.172/(Tr_1)^(4.2));
38 // We know ,(B*Pc)/(R*Tc) = B_0+(w*B_1)
39 B_11 = ((B1_0+(w_1*B1_1))*(R*Tc_1))/Pc_1;// [m^(3)/
      mol]
40
41 // Similarly for component 2
42 Tr_2 = T/Tc_2;//Reduced temperature
43 // At reduced temperature Tr_2 ,
44 B2_0 = 0.083 - (0.422/(Tr_2)^(1.6));
45 B2_1 = 0.139 - (0.172/(Tr_2)^(4.2));
46 B_22 = ((B2_0 + (w_2*B2_1))*(R*Tc_2))/Pc_2;// [m^(3)/
      mol]
47
48 //For cross coeffcient
49 Tc_12 = (Tc_1*Tc_2)^(1/2);// [K]
50 w_12 = (w_1 + w_2)/2;
51 Zc_12 = (Zc_1+Zc_2)/2;
52 Vc_12 = (((Vc_1)^(1/3)+(Vc_2)^(1/3))/2)^(3);// [cm
      ^3)/mol]
53 Vc_12 = Vc_12*10^(-6);// [m^(3)/mol]
54 Pc_12 = (Zc_12*R*Tc_12)/Vc_12;// [N/m^(2) ]
55
56 // Now we have ,(B_12*Pc_12)/(R*Tc_12) = B_0 + (w_12*
      B_1)
57 // where B_0 and B_1 are to be evaluated at Tr_12
58 Tr_12 = T/Tc_12;
59 // At reduced temperature Tr_12
60 B_0 = 0.083 - (0.422/(Tr_12)^(1.6));
61 B_1 = 0.139 - (0.172/(Tr_12)^(4.2));
62 B_12 = ((B_0+(w_12*B_1))*R*Tc_12)/Pc_12;// [m^(3)/mol
      ]
63
64 // For the mixture
65 B = y1^(2)*B_11 + 2*y1*y2*B_12+y2^(2)*B_22;// [m^(3)/

```

```

mol]
66 // Now we have to calculate molar volume i.e V1 and
    V2 at given conditions
67
68 // At state 1,
69 deff(' [y]=f(V) ', 'y=V^(2)-(R*T/P1)*V-(B*R*T)/P1 ');
70 V_1=fsolve(-1,f);
71 V_2=fsolve(1,f);
72 // We will take root near to (R*T)/P, i.e V_2
73 V1 = V_2; // [m^(3)/mol]
74
75 // At state 2,
76 deff(' [y]=f1(V) ', 'y=V^(2)-(R*T/P2)*V-(B*R*T)/P2 ');
77 V_3=fsolve(-1,f1);
78 V_4=fsolve(1,f1);
79 V2 = V_4; // [m^(3)/mol]
80
81 // Work done per mole of air is given by, w=integral
    (Pdv)
82 // Z = 1 + (B/V)
83 // (P*V)/(R*T) = 1 +( B/V)
84 // P = (R*T)/V+(B*R*T)/V^(2)
85 // w = integrate(' (R*T/V)+(B*R*T)/V^(2) ', 'V', V1, V2)
86 // Under isothermal conditions,
87 w = R*T*log(V2/V1)-B*R*T*((1/V2)-(1/V1));
88 w = w*10^(-3); // [kJ/mol]
89 printf(" The work done is %f kJ/mol", w);

```

Scilab code Exa 3.6 Calculation of work done

```

1 clear;
2 clc;
3 funcprot(0);
4
5 //Example - 3.6

```

```

6 //Page number - 88
7 printf("Example - 3.6 and Page number - 88\n\n");
8
9
10 //Given
11 T = 125+273.15; // [K] - Temperature
12 P1 = 1; // [bar] - Initial pressure
13 P1 = P1*10^(5); // [N/m^(2)]
14 P2 = 60; // [bar] - Final pressure
15 P2 = P2*10^(5); // [N/m^(2)]
16 R = 8.314; // [J/mol*K] - Universal gas constant
17 Tc = 416.3; // [K] - Critical temperature
18 Pc = 66.80*10^(5); // [N/m^(2)] - Critical pressure
19
20 // (1)
21 // Virial equation of state ,  $Z = 1 + (B/V)+(C/V^2)$ 
22 //  $(P*V)/(R*T) = 1 + (B/V)+(C/V^2)$ 
23 //  $P = (R*T)/V+(B*R*T)/V^2+(C*R*T)/V^3$ 
24 //  $w = \int PdV = R*T*\log(V2/V1) - (B*R*T)*(1/V2 - 1/V1) - (C*R*T/2)*(1/V2^2 - 1/V1^2)$ 
25
26 B = -207.5; // [cm^3/mol] - Second virial
    coefficient
27 B = -207.5*10^(-6); // [m^3/mol]
28 C = 18200; // [cm^6/mol^2] - Third virial
    coefficient
29 C = 18200*10^(-12); // [m^6/mol^2]
30
31 // We need to calculate molar volume at state 1 and
    2,
32 // At state 1, P = P1,
33 //  $V^3 - (R*T/P)*V^2 - (B*R*T/P)*V - (C*R*T/P) = 0$ 
34 // Solving the cubic equation
35 def f(y)=f1(V) , 'y=V^3 - (R*T/P1)*V^2 - (B*R*T/P1)*
    V - (C*R*T/P1)';
36 V_1=fsolve(-1, f1);
37 V_2=fsolve(0, f1);
38 V_3=fsolve(10, f1);

```



```

39 // The cubic equation has only 1 real root ,other two
    roots are imaginary.
40 V1 = V_3;
41
42 // Similarly at state 2,P=P2
43 //  $V^3 - (R*T/P)*V^2 - (B*R*T/P)*V - (C*R*T/P) = 0$ 
44 // Solving the cubic equation
45 def ( ' [y]=f2 (V) ', 'y= $V^3 - (R*T/P2)*V^2 - (B*R*T/P2)*$ 
     $V - (C*R*T/P2)$  ');
46 V_4=fsolve(-1,f2);
47 V_5=fsolve(0,f2);
48 V_6=fsolve(1,f2);
49 V2 = V_6;
50 // Finally work done is given by,
51 w1 = R*T*log(V2/V1)-(B*R*T)*(1/V2-1/V1)-(C*R*T/2)
    *(1/V2^2)-1/V1^2);//[J/mol]
52 w1 = w1*10^(-3);//[kJ/mol]
53 printf(" (1).The work done using given virial
    equation of state is %f kJ/mol\n",w1);
54
55 // (2)
56 // Virial equation of state ,  $Z = 1+(B*P)/(R*T)+((C-B$ 
     $^2)/(R*T)^2)*P^2$ 
57 //  $(P*V)/(R*T) = 1+(B*P)/(R*T)+((C-B^2)/(R*T)^2)*$ 
     $P^2$ 
58 //  $V = (R*T)/P+B+((C-B^2)/(R*T))*P$ 
59 // Differentiating both sides by P and integrating
    we get ,
60 //  $w = \text{integral}(PdV) = -(R*T)*\log(P2/P1) + ((C-B^2)$ 
     $/(2*R*T))*(P2^2-P1^2)$ 
61 w2 = -(R*T)*log(P2/P1) + ((C-B^2)/(2*R*T))*(P2^2)
    -P1^2);//[J/mol]
62 w2 = w2*10^(-3);//[kJ/mol]
63 printf(" (2).The work done using given virial
    equation of state is %f kJ/mol\n",w2);
64
65 // (3)

```

```

66 // Van der Walls equation of state is given by,
67 a = (27*(R^(2))*(Tc^(2)))/(64*Pc); // [Pa*m^(6)/mol
    ^ (2)]
68 b = (R*Tc)/(8*Pc); // [m^(3)/mol]
69 // P = ((R*T)/(V-b))-a/(V^(2)); // [N/m^(2)]
70 // w = integral(PdV)=R*T*log((V2-b)/(V1-a))+a*(1/V2
    -1/V1)
71 // The cubic form of van der Walls equation of state
    is given by,
72 // V^(3) - (b+(R*T)/P)*V^(2) + (a/P)*V - (a*b)/P = 0
73 // Solving the cubic equation for P=P1
74 deff(' [y]=f3(V)', 'y=V^(3)-(b+(R*T)/P1)*V^(2)+(a/P1)*
    V-(a*b)/P1');
75 V2_1=fsolve(1,f3);
76 V2_2=fsolve(10,f3);
77 V2_3=fsolve(100,f3);
78 // The above equation has 1 real and 2 imaginary
    roots. We consider only real root (V2_3).
79
80 // Similarly at state 2, for P=P2,
81 deff(' [y]=f4(V)', 'y=V^(3)-(b+(R*T)/P2)*V^(2)+(a/P2)*
    V-(a*b)/P2');
82 V2_4=fsolve(1,f4);
83 V2_5=fsolve(10,f4);
84 V2_6=fsolve(100,f4);
85 // The above equation has 1 real and 2 imaginary
    roots. We consider only real root (V2_6).
86 // Finally work done is given by
87 w3 = R*T*log((V2_6-b)/(V2_3-b))+a*(1/V2_6-1/V2_3); //
    [J/mol]
88 w3 = w3*10^(-3); // [kJ/mol]
89 printf(" (3).The work done using van der Walls
    equation of state is %f kJ/mol\n",w3);
90
91 // (4)
92 // Redlich Kwong equation of state ,
93 a_1 = (0.42748*(R^(2))*(Tc^(2.5)))/Pc; // [Pa*m^(6)*K
    ^ (1/2)/mol]

```

```

94 b_1 = (0.08664*R*Tc)/Pc; // [m3/mol]
95 // P = ((R*T)/(V-b_1))-(a_1/(T1/2*V*(V+b_1))); // [
    N/m2]
96 // Work done is given by
97 // w = R*T*log((V2-b)/(V1-b))-a/T1/2*integrate
    ('1/V*(V+b)', V', V1, V2)
98 // Using the factorization 1/(V*(V+b))=(1/b)*((1/V)
    -(1/V+b)), we get
99 // w = R*T*log((V2-b)/(V1-b))-(a/(b*T1/2))*(log(
    V2/V1)-log((V2+b)/(V1+b)))
100 // Now we have calculate V1 and V2,
101 // The cubic form of Redlich Kwong equation of state
    is given by,
102 // V3 - ((R*T)/P)*V2 - ((b_12)) + ((b_1*R*T)
    /P) - (a/(T1/2*P))*V - (a*b)/(T1/2*P) = 0
103 // Solving the cubic equation at state 1,
104 deff ('[y]=f5(V)', 'y=V3 -((R*T)/P1)*V2 -((b_12)
    )+((b_1*R*T)/P1)-(a_1/(T1/2*P1))*V-(a_1*b_1)
    /(T1/2*P1)');
105 V3_1=fsolve(1, f5);
106 V3_2=fsolve(10, f5);
107 V3_3=fsolve(100, f5);
108 // The above equation has 1 real and 2 imaginary
    roots. We consider only real root (V3_3).
109
110 // Similarly at state 2, for P = P2,
111 deff ('[y]=f6(V)', 'y=V3 -((R*T)/P2)*V2 -((b_12)
    )+((b_1*R*T)/P2)-(a_1/(T1/2*P2))*V-(a_1*b_1)
    /(T1/2*P2)');
112 V3_4=fsolve(1, f6);
113 V3_5=fsolve(10, f6);
114 V3_6=fsolve(100, f6);
115 // The above equation has 1 real and 2 imaginary
    roots. We consider only real root (V3_6).
116 // Finally work done is given by
117 w4 = R*T*log((V3_6-b_1)/(V3_3-b_1))-(a_1/(b_1*T
    1/2))*(log(V3_6/V3_3)-log((V3_6+b_1)/(V3_3+b_1)
    ))); // [J/mol]

```

```
118 w4 = w4*10^(-3);//[kJ/mol]
119 printf(" (3).The work done using Redlich Kwong
    equation of state is %f kJ/mol\n",w4);
```

Scilab code Exa 3.7 Proving a mathematical relation

```
1 clear;
2 clc;
3
4 //Example - 3.7
5 //Page number - 92
6 printf("Example - 3.7 and Page number - 92\n\n");
7
8 //This problem involves proving a relation in which
   no numerical components are involved.
9 //For prove refer to this example 3.7 on page number
   92 of the book.
10 printf(" This problem involves proving a relation in
    which no numerical components are involved.\n\n"
    );
11 printf(" For prove refer to this example 3.7 on page
    number 92 of the book.");
```

Scilab code Exa 3.8 Calculation of work done

```
1 clear;
2 clc;
3 funcprot(0);
4
5 //Example - 3.8
6 //Page number - 93
7 printf("Example - 3.8 and Page number - 93\n\n");
8
```

```

9
10 // Given
11 T = 20 + 273.15; // [K] - Temperature
12 P_1 = 140; // [kPa] - Initial pressure
13 P_1 = P_1*10^(3); // [Pa]
14 P_2 = 560; // [kPa] - Final pressure
15 P_2 = P_2*10^(3); // [Pa]
16 R = 1.987; // [cal/mol*K] - Universal gas constant
17
18 // Cp_0 = 1.648+4.124*10^(-2)*T - 1.53*10^(-5)*T^(2)
    + 1.74*10^(-9)*T^(3)
19 // Using adiabatic compression, P*V^(Y)=constant.
    For ideal gases
20 // P*(R*T/P)^(Y) = constant
21 // P^(1-Y)*T^(Y) = constant or, P1^(1-Y)*T1^(Y)=P2
    ^^(1-Y)*T2^(Y)
22 // Now, at state 1, i.e at T=20[C]
23 Cp_1 = 1.648+4.124*10^(-2)*T-1.53*10^(-5)*T^(2)
    +1.74*10^(-9)*T^(3); // [cal/mol*K] - Heat capacity
    at constant pressure
24 Cv_1 = Cp_1 - R; // [cal/mol*K] - Heat capacity at
    constant volume
25 Y1 = Cp_1/Cv_1; // Ratio of heat capacities
26
27 // Now calculating the temperature at state 2 (T2)
28 // (T2/T1)=(P1/P2)^((1-Y1)/Y1)
29 T_1 = T;
30 T_2 = ((P_1/P_2)^((1-Y1)/Y1))*T_1; // [K]
31
32 // Now calculating the mean temperature
33 T_mean = (T_1 + T_2)/2; // [K]
34 // At mean temperature
35 Cp_2 = 1.648+4.124*10^(-2)*T_mean - 1.53*10^(-5)*
    T_mean^(2) + 1.74*10^(-9)*T_mean^(3); // [cal/mol*K]
    ] - Heat capacity at constant pressure
36 Cv_2 = Cp_2 - R; // [cal/mol*K] - Heat capacity at
    constant volume
37 Y2 = Cp_2/Cv_2;

```

```

38
39 // Calculating exit temperature
40 // Again using the realation ,(T2/T1)=(P1/P2)^((1-Y1)
//Y1)
41 T_exit = ((P_1/P_2)^((1-Y2)/Y2))*T_1; //[K]
42 // Since value of mean temperature has not changed
// much the molar heat capacity ratio can be assumed
// to be same. Therefore
43 // w = -delta(U)=Cv_0*(T2-T1)
44 w = Cv_2*(T_1 - T_exit); //[cal/mol]
45 w = w*4.184; //[J/mol]
46
47 printf(" The work done for adiabatic compression is
// %f J/mol\n",w);

```

Scilab code Exa 3.9 Calculation of final temperature

```

1 clear;
2 clc;
3
4 //Example - 3.9
5 //Page number - 93
6 printf("Example - 3.9 and Page number - 93\n\n");
7
8 //Given
9 m_ice = 1000; //[g] - Mass of ice
10 m_water = 1000; //[g] - Mass of water
11 T_ice = 273.15; //[K] - Temperature of ice
12 T_water = 373.15; //[K] - Temperature of water
13 L = 79.71; //[cal/g] - Latent heat of melting of ice.
14
15 //(1)
16 Cp_1 = 1; //[cal/g-K] - Heat capacity at constant
// pressure
17 // Let the final temperature be T

```

```

18 // We assume that all of the ice melts. Energy taken
    up by ice is
19 //  $E_1 = L \cdot m_{\text{ice}} + m_{\text{ice}} \cdot C_{p,1} \cdot (T - T_{\text{ice}})$ 
20 // Energy given by hot water is ,
21 //  $E_2 = m_{\text{water}} \cdot C_{p,1} \cdot (T_{\text{water}} - T)$ 
22 // No heat exchange with surrounding. Solving for T
23  $T_1 = (m_{\text{ice}} \cdot C_{p,1} \cdot T_{\text{ice}} + m_{\text{water}} \cdot C_{p,1} \cdot T_{\text{water}} - L \cdot$ 
     $m_{\text{ice}}) / (m_{\text{ice}} \cdot C_{p,1} + m_{\text{water}} \cdot C_{p,1}); // [K]$ 
24  $T_1 = T_1 - 273.15; // [C]$ 
25
26 printf(" (1). The final temperature (taking  $C_{p,\text{water}}$ 
    = 1 cal/g-K) is %f C\n", T_1);
27 // Since the final temperature is greater than 273.15
    K, so our assumption that all of ice melts is
    correct
28
29 // (2)
30 //  $C_{p,2} = 1.00874 - 0.7067 \cdot 10^{-3} \cdot T + 15.93 \cdot 10^{-6} \cdot T$ 
     $^2 - 83.8 \cdot 10^{-9} \cdot T^3$ ;
31 // From energy balance, we get  $L \cdot m_{\text{ice}} + m_{\text{ice}} \cdot$ 
     $\text{integrate}('C_{p,2}', 'T', 0, T) + m_{\text{water}} \cdot \text{integrate}('$ 
     $C_{p,2}', 'T', 100, T) = 0$ ; (where T is in C)
32 // On putting the values and then simplifying we get
33 //  $2.01748 \cdot T - 0.0007067 \cdot T^2 + 1.062 \cdot 10^{-5} \cdot T^3$ 
     $- 4.19 \cdot 10^{-8} \cdot T^4 - 20.8455 = 0$ 
34 // Solving the above equation we get
35 def ('[y]=f1(T)', 'y = 2.01748 * T - 0.0007067 * T^2 +
    1.062 * 10^(-5) * T^3 - 4.19 * 10^(-8) * T^4 -
    20.8455');
36 T_0 = fsolve(1, f1); // [C]
37 printf(" (2). The final temperature using specific
    heat capacity equation is %f C\n", T_0);

```

Scilab code Exa 3.10 Finding expressions for temperature and pressure

```

1 clear;
2 clc;
3
4 //Example - 3.10
5 //Page number - 95
6 printf("Example - 3.10 and Page number - 95\n\n");
7
8 //This problem involves proving a relation in which
   no numerical components are involved.
9 //For prove refer to this example 3.10 on page
   number 95 of the book.
10 printf(" This problem involves proving a relation in
   which no numerical components are involved.\n\n"
   );
11 printf(" For prove refer to this example 3.10 on
   page number 95 of the book.");

```

Scilab code Exa 3.11 Calculation of final pressure

```

1 clear;
2 clc;
3
4 //Example - 3.11
5 //Page number - 97
6 printf("Example - 3.11 and Page number - 97\n\n");
7
8 //Given
9 n = 1.5; // - ratio of heat capacities
10 T_1 = 500; // [K] - Initial temperature
11 T_2 = 1000; // [K] - Final temperature
12 P_1 = 1; // [bar] - Initial pressure
13 P_1 = P_1*10^(5); // [Pa]
14 R = 8.314; // [J/mol*K] - Universal gas constant
15
16 // The compression path is given by,  $P*V^{(1.5)} =$ 

```



```

    constant
17 //  $P*(R*T/P)^{(1.5)} = \text{constant}$ 
18 //  $P_1^{(-0.5)}*T_1^{(1.5)} = P_2^{(-0.5)}*T_2^{(1.5)}$ 
19 P_2 = P_1*(T_1/T_2)^(-3);//[Pa]
20 P_2_final = P_2*10^(-5);//[bar] – Final pressure in
    bar
21 printf(" The final pressure is %f bar\n",P_2_final);
22
23 // From first law  $q - w = \text{delta}(U)$ .
24 // First  $w$  and  $\text{delta}(U)$  are calculated and
    thereafter heat exchange is determined.
25 V_1 = R*T_1/P_1;//[m^(3)/mol] – Initial volume
26 V_2 = R*T_2/P_2;//[m^(3)/mol] – Final volume
27 w = ((P_1*V_1)/(n - 1))*(1 - (P_2/P_1)^(1 - 1/n));//
    [J/mol] – work done
28
29 // Mean temperature is given by,
30 T_mean = (T_1 + T_2)/2;//[K]
31
32 //Now, heat capacity at T_mean is given by,
33 Cp_0 = R*(3.3 + 0.63*10^(-3)*T_mean);//[J/mol*K]
34 Cv_0 = Cp_0 - R;//[J/mol*K]
35 //Therefore  $\text{delta}(U)$  is given by
36 del_U = Cv_0*(T_2 - T_1);//[J/mol] – Change in
    internal energy
37 q = w + del_U;//[J/mol] – heat change
38 printf(" The amount of heat supplied to the system
    is %f J/mol\n",q);

```

Scilab code Exa 3.12 Calculation of slope and work done

```

1 clear;
2 clc;
3
4 //Example – 3.12

```

```

5 //Page number - 99
6 printf("Example - 3.12 and Page number - 99\n\n");
7
8 //Given
9 P_1 = 150*10^(3);//[Pa] - Initial pressure
10 V_1 = 0.001;//[m^(3)] - Initial volume
11 P_2 = 1000*10^(3);//[Pa] - Final pressure
12 V_2 = 0.003;//[m^(3)] - Final volume
13
14 // At x = 0, Vt(total volume) = 0.001 m^(3),
    // therefore x = (V_t - V_1)/A; where A is area of
    // cross section and x is length
15 // Force exerted b sprig is given by, F = Ps*A = k*x
    // = k*(V_t - V_1)/A
16 // Ps = (k/A^(2))*(V_t - V_1)
17 // Total pressure = Initial pressre + Pressre due to
    // spring
18 // P = P_1 + (k/A^(2))*(V_t - V_1)
19 // Let (k/A^(2)) = t (say)
20 // At state 2, i.e at P2 and V_t = V_2.
21 deff(' [y]=f(t) ', 'y=P_2-P_1 - t*(V_2-V_1) ');
22 t = fsolve(1000,f);
23 // Therefore ,pressure is related to total volume as
    // P = P_1-t*(V_t - V_1)
24
25 // (a)
26 //slope = (k/A^(2))
27 printf(" (a).The slope of the line on P-Vt diagram
    // is %e N/m^(5)\n",t);
28
29 // (b)
30 // Work done by the gas is given by w=integral(
    // PdVt)
31 w = integrate('P_1+t*(V_t-V_1)', 'V_t',V_1,V_2);//[J]
32 w = w*10^(-3);//[kJ]
33 printf(" (b).The work done by gas is %f kJ\n",w);

```

Scilab code Exa 3.13 Calculation of work done and final temperature

```
1 clear;
2 clc;
3
4 //Example - 3.13
5 //Page number - 99
6 printf("Example - 3.13 and Page number - 99\n\n");
7
8 //Given
9 V = 36; //[L] - Vol of gas on each side
10 P_1 = 1; //[atm] - pressure on left side of the
    piston
11 P_1 = P_1*101325; //[Pa]
12 T = 273.15; //[K]
13 P_2 = 3.375; //[atm] - Pressure on right side of the
    piston
14 P_2 = P_2*101325; //[Pa]
15 Y = 1.44; // Ratio of heat capacities
16 R = 8.314; //[J/mol*K] - Universal gas constnt
17
18 // (a)
19 // For total system, del(U_total) = Q.
20 // Onto gas on right hand side no heat is supplied,
    as the piston is non conducting. Therefore, for
    gas on the right hand side, del(U) = -W.
21 // As heat is slowly supplied to the left hand side,
    expansion on right hand side is slow and process
    is adiabatic.
22 // For gas on right hand side,  $PV^Y = \text{constant}$ .
23 //  $T_2/T_1 = (P_2/P_1)^{((Y - 1)/Y)}$ 
24 T_right = T*(P_2/P_1)^{((Y - 1)/Y)}; //[K]
25
26 Cv_0 = R/(Y-1); //[J/mol*K] - Heat capacity at
```

```

    constant volume.
27 // Now work done on the gas on right hand side is
    given by
28 //  $W = (P_1*V_1 - P_2*V_2)/(Y - 1) = R*(T_2 - T_1)/($ 
     $Y - 1) = C_{v,0}*(T_1 - T_2)$ 
29  $W_{left} = C_{v,0}*(T - T_{right});$ //[J/mol]
30 // Negative sign for the work done on LHS gas
    implies work is done on the gas
31
32 // For right hand side of the gas
33 //  $P*V_t = n*R*T$ 
34  $n = P_1*(V*10^{(-3)})/(R*T);$ // number of moles
35  $W_{right} = (-W_{left})*n;$ //[J] - We used negative sign
    for 'W_left' because it is negative in magnitude.
36  $W_{right} = W_{right}/1000;$ //[kJ]
37 printf(" (a).Total work done on gas on the right
    hand side is %f kJ\n",W_right);
38
39 //(b)
40 printf(" (b).The final temperature of the gas on
    right side is %f K\n",T_right);
41
42 //(c)
43 // Pressure is same on both sides as piston is
    frictionless.
44 // The number of moles on both sides are also same
    as they have same temperature, pressure and volume.
45 // We have  $(P_{left}*V_{left})/T_{left} = (P_{right}*V_{right})/T_{right}$ .
46 // Since  $P_{left} = P_{right}$ ,  $(V_{left}/T_{left}) = ($ 
     $V_{right}/T_{right})$  and also  $P*V^Y = \text{constant}$ .
47  $V_{right} = V*(P_1/P_2)^{(1/Y});$ //[L] - The total volume
    on right side
48
49 // The total volume on right side can also be
    calculated using  $P_2*V_2 = n*R*T_2$ .
50 // Since total volume = 72 [L], therefore volume of
    left side is

```

```

51 V_left = 2*V - V_right;//[L]
52 T_left = T_right*(V_left/V_right);
53 printf(" (c).Final temperature of the gas on the
    left side is %f K\n",T_left);
54
55 //(d)
56 //The first law applied to the total system (left
    side and right side) gives.
57 //Q - W = del(U_left) + del(U_right)
58 //There is no net work done by the total system as
    the cylinder is closed at both ends.
59 Q = n*Cv_0*(T_left-T) + n*Cv_0*(T_right-T);//[J]
60 Q = Q/1000;//[kJ]
61 printf(" (d).Amount of heat added to the gas on the
    left side is %f kJ",Q);

```

Scilab code Exa 3.14 Calculation of power and discharge head

```

1 clear;
2 clc;
3
4 //Example - 3.14
5 //Page number - 105
6 printf("Example - 3.14 and Page number - 105\n\n");
7
8
9 //Given
10 P_2 = 0.2;//[bar]
11 P_2 = P_2*10^(5);//[Pa]
12 int_dia_2 = 2.4*10^(-2);//[m] - internal diameter at
    state 2.
13 Q = 5*10^(-3);//[cubic metre/s] - Flow rate at point
    2.
14 den = 1000;//[kg/cubic metre] - density
15 delta_z = 1;//[m] - Difference in height

```

```

16 g = 9.81; //[m/s^(2)] - Acceleration due to gravity
17
18 // (1)
19 // Pressure at state 1 is atmospheric pressure and
    at state 2 is gauge pressure at state 2 +
    atmospheric pressure ,thus
20 // (delta(P)/den) = (P2-P1)/den = P2/den
21 Vel_2 = Q/(3.14*(int_dia_2/2)^(2)); //[m/s] -
    Velocity of water at state 2.
22 // Velocity at state 1 is negligible as compared to
    velocity at state 2, because the diameter of
    reservoir is very large as compared to diameter
    of pipe at state 2
23
24 // From Bernoulli equation we get ,
25 //  $-w = (\Delta P)/\rho + \Delta(v^2)/2 + g\Delta z$ 
26 w = -((P_2/den) + (Vel_2^2)/2 + (g*delta_z)); //[J/
    kg]
27 // w multiplied by m = (den*Q), will give the fluid
    power.
28 m = den*Q; //[kg/s]
29 W_net = m*w; //[Watt]
30 printf(" (1).The fluid power is %f Watt\n",W_net);
31
32 // (2)
33 // Total discharge head developed by the pump is
    given by
34 //  $h = (\Delta P)/(\rho g) + (V_2^2)/(2g) + \Delta z$ 
35 h = (P_2/(den*g)) + (Vel_2^2)/(2*g) + delta_z; //[m
    ]
36 printf(" (2).Total discharge head developed by the
    pump is given by h = %f m',h);

```

Scilab code Exa 3.15 Calculation of discharge velocity

```

1 clear;
2 clc;
3
4 //Example - 3.15
5 //Page number - 106
6 printf("Example - 3.15 and Page number - 106\n\n");
7
8 //Given
9 T_1 = 1000; //[K] - Temperature at entry
10 P_1 = 0.6; //[MPa] - Pressure at entry
11 P_2 = 0.2; //[MPa] - Exit pressure
12 Vel_1 = 50; //[m/s] - Entry velocity
13 Y = 1.4; // Ratio of heat capacities
14 Mol_wt = 28; //[g/mol] - Molecular weight of air
15 Cp = 29.099; //[J/mol-K] - Specific heat capacity at
    constant pressure
16 Cp = (Cp/Mol_wt)*1000; //[J/kg-K]
17
18 // We know that for a flow process
19 //  $\Delta H + \Delta V^2/2 + \Delta(gz) = q - w$ 
20 // Since process is adiabatic, therefore  $q = 0$  and
    since no work is done by the gas, therefore  $w = 0$ 
21 // Assuming there is no change in the potential
    energy between entry and exit, we have
22 //  $\Delta H + \Delta V^2/2 = 0$ 
23
24 // For a reversible process  $P*V^Y = \text{constant}$  and
    thus  $(T_2/T_1) = (P_2/P_1)^{(Y-1)/Y}$ 
25 T_2 = T_1*(P_2/P_1)^{(Y-1)/Y}; //[K] - Exit
    temperature
26
27 //  $\Delta H + \Delta V^2/2 = 0$ 
28 //  $Vel_2^2/2 - Vel_1^2/2 - (H_1 - H_2) = 0$ 
29 //  $Vel_2^2/2 - Vel_1^2/2 - Cp*(T_1 - T_2) = 0$ 
30 Vel_2_square = 2*(Vel_1^2/2 + Cp*(T_1 - T_2)); //[m
    ^2/s^2]
31 Vel_2 = (Vel_2_square)^(1/2); //[m/s]
32

```

```
33 printf(" The discharge velocity is %f m/s\n",Vel_2);
```

Scilab code Exa 3.16 Calculation of change in enthalpy

```
1 clear;
2 clc;
3
4 //Example - 3.16
5 //Page number - 107
6 printf("Example - 3.16 and Page number - 107\n\n");
7
8 //Given
9 P_entry = 10;//[bar] - Pressure at entry
10 V_entry = 200;//[m/s] - Velocity at entry
11 P_exit = 1;//[bar] - Vressure at exit
12 V_exit = 800;//[m/s] - Velocity at exit
13 g = 9.81;//[m/s^(2)] - Acceleration due to gravity
14
15 //Heat balance gives
16 //  $\Delta H + (\Delta V^2)/2 + g \cdot \Delta z = q - w$ 
17 //  $\Delta H = q - w - (\Delta V^2)/2$ 
18 //From nozzle no work is extracted, therefore
19  $\Delta H = -(V_{exit}^2 - V_{entry}^2)/2$ ;//[J/kg]
20  $\Delta H = \Delta H \cdot 10^{-3}$ ;//[kJ/kg]
21
22 printf(" The change in enthalpy per kg of steam is
    %f kJ/kg",delta_H);
```

Scilab code Exa 3.17 Calculation of work done and change in enthalpy

```
1 clear;
2 clc;
3
```



```

4 //Example - 3.17
5 //Page number - 111
6 printf("Example - 3.17 and Page number - 111\n\n");
7
8
9 //Given
10 T_1 = 280; //[K] - Temperature at entry
11 P_1 = 100; //[kPa] - Pressure at entry
12 T_2 = 400; //[K] - Temperature at exit
13 P_2 = 600; //[kPa] - Pressure at exit
14 m = 0.02; //[kg/s] - Mass flow rate
15 m = m*10^(3); //[g/s]
16 heat_loss = 16; //[kJ/kg]
17
18 //Cp_0 = 28.11 + 0.1967*10^(-2)*T + 0.4802*10^(-5)*T
    ^ (2) - 1.966*10^(-9)*T^(3)
19 //delta_H = q - w (neglecting kinetic and potential
    changes)
20 //delta_H = integral(Cp_0*dT)
21 delta_H = integrate('28.11 + 0.1967*10^(-2)*T +
    0.4802*10^(-5)*T^(2) - 1.966*10^(-9)*T^(3)', 'T',
    T_1, T_2); //[J/mol - Enthalpy change]
22 printf(" Change in enthalpy is %f J/mol\n", delta_H);
23
24 //Molecular weight of air(21 vol% O2 and 79 vol% N2)
    =(0.21*32)+(0.79*28)= 28.84 g/mol
25 Mol_wt = 28.84; //[g/mol]
26 q = - (heat_loss*Mol_wt); //[J/mol]
27 w = q - delta_H; //[J/mol]
28 printf(" The work done per mole of air is %f J/mol\n
    ", w);
29 //the negative sign implies that work is done on the
    compressor.
30
31 n = m/Mol_wt; //[mol/s] - Mole flow rate
32 W_net = delta_H*n; //[W]
33 W_net = -W_net*10^(-3); //[kW]
34 printf(" And the necessary power input to the

```

```
compressor is %f kW\n",W_net);
```

Scilab code Exa 3.18 Calculation of work done per unit mass

```
1 clear;
2 clc;
3
4 //Example - 3.18
5 //Page number - 112
6 printf("Example - 3.18 and Page number - 112\n\n");
7
8
9 //Given
10 T_1 = 300; // [K] - Temperature at entry
11 P_1 = 100; // [kPa] - Pressure at entry
12 P_2 = 900; // [kPa] - Pressure at exit
13 R = 8.314; // [J/mol*K] - Universal gas constant
14
15 // (a)
16 // Reversible adiabatic compression
17 Y = 1.4; // Ratio of specific heat capacities
18 // For ideal gas,  $P*V^Y = \text{constant}$  and it takes the
19 // form of  $(T_2/T_1) = (P_2/P_1)^{(Y-1)/Y}$ 
20 T_2 = T_1*(P_2/P_1)^((Y - 1)/Y); // [K]
21 // The work exchange for adiabatic process is given
22 // by
23 //  $W_{\text{adia}} = -\Delta H = -C_p*(T_2-T_1) = C_p*(T_1-T_2) = ((Y*R)/(Y-1))*(T_1-T_2)$ 
24 W_adia = ((Y*R)/(Y - 1))*(T_1 - T_2); // [J/mol] -work
25 // done
26 // Molecular weight of air (21 vol% O2 and 79 vol% N2)
27 //  $= (0.21*32) + (0.79*28) = 28.84 \text{ g/mol}$ 
28 Mol_wt = 28.84; // [g/mol]
29 W_adia = W_adia/Mol_wt; // [J/g]
30 printf(" (a).The compressor work done for reversible
```

```

        adiabatic compression is %f J/g\n",W_adia);
27
28 //(b)
29 //Isothermal compression
30 //W_iso = -integral(V*dP) = -integral((R*T/P)*dP) =
        R*T*ln(P_2/P_1)
31 W_iso = -R*T_1*log(P_2/P_1);//[J/mol]
32 W_iso = W_iso/Mol_wt;//[J/g]
33 printf(" (b).The compressor work done for isothermal
        compression is %f J/g\n",W_iso);
34 //Note that in isothermal compression between the
        same states work done is less as compared to
        reversible adiabatic compression.
35
36 //(c)
37 //Ideal two-stage compression
38 n = 1.3; //Polytropic exponent.
39 //Minimum work done in two stage compression is
        given by
40 //W_comp = ((2*n*R*T_1)/(n-1))*[1-(P_x/P_1)^(n-1)/n]
41 //where for minimum work, (P_x/P_1) = (P_x/P_2), and
        thus
42 P_x = (P_1*P_2)^(1/2);//[kPa]
43 //therefore, work done is given by,
44 W_comp = ((2*n*R*T_1)/(n-1))*[1-(P_x/P_1)^((n-1)/n)
        ];//[J/mol]
45 W_comp = W_comp/Mol_wt;//[J/g]
46 printf(" (c).The compressor work done for ideal two-
        stage compression is %f J/g\n",W_comp);

```

Scilab code Exa 3.19 Calculation of inlet and outlet velocity and power

```

1 clear;
2 clc;
3

```

```

4 //Example - 3.19
5 //Page number - 113
6 printf("Example - 3.19 and Page number - 113\n\n");
7
8
9 //Given
10 T_1 = 600; //[C] - Temperature at entry
11 P_1 = 15; //[MPa] - Pressure at entry
12 T_2 = 400; //[K] - Temperature at exit
13 P_2 = 100; //[kPa] - Pressure at exit
14 A_in = 0.045; //[metre square] - flow in area
15 A_out = 0.31; //[metre square] - flow out area
16 m = 30; //[kg/s] - mass flow rate.
17
18 //At 15 MPa and 600 C, it has been reported in the
    book that the properties of steam are,
19 Vol_1 = 0.02491; //[m^(3)/kg] - Specific volume
20 H_1 = 3582.3; //[kJ/kg] - Enthalpy
21 // m = den*vel*A = (Vel*A)/Vol, substituting the
    values
22 vel_1 = (m*Vol_1)/A_in; //[m/s] - Velocity at point
    1.
23 printf(" The inlet velocity is %f m/s\n", vel_1);
24
25 //At 100 MPa (saturated vapour), it has been reported
    in the book that the properties of steam are,
    T_sat = 99.63 C, and
26 Vol_vap_2 = 1.6940; //[m^(3)/kg] - specific volume of
    saturated vapour.
27 H_vap_2 = 2675.5; //[kJ/kg] - Enthalpy os saturated
    vapour.
28 vel_2 = (m*Vol_vap_2)/A_out; //[m/s] - Velocity at
    point 2.
29 printf(" The exit velocity is %f m/s\n", vel_2);
30
31 //From first law we get , q - w =delta_H + delta_V
    ^ (2)/2
32 //q = 0, therefore , -w = delta_H + delta_V ^ (2)/2

```

```

33 delta_H = H_vap_2 - H_1;//[kJ/kg] - change in
    enthalpy.
34 delta_V_square = (vel_2^(2) - vel_1^(2))/2;//[J/kg]
35 delta_V_square = delta_V_square*10^(-3);//[kJ/kg]
36 w = -(delta_H + delta_V_square);//[J/kg]
37 W_net = w*m;//[kW]
38 W_net = W_net*10^(-3);//[MW] - power produced.
39 printf(" The power that can be produced by the
    turbine is %f MW",W_net);

```

Scilab code Exa 3.20 Proving a mathematical relation

```

1  clear;
2  clc;
3
4  //Example - 3.20
5  //Page number - 117
6  printf("Example - 3.20 and Page number - 117\n\n");
7
8  //(1)
9  //This part involves no numerical components
10 //For prove refer to this example 3.20 on page
    number 117 of the book.
11
12 //(2)
13 //Given ,
14 R = 8.314;//[J/mol-K] - Universal gas constant
15 Cp_0 = 2.5*R;//[J/mol-K] - Specific heat capacity
    at constant pressure
16 Cv_0 = 1.5*R;//[J/mol-K] - Specific heat capacity
    at constant volume
17 T_L = 300;//[K] - Temperature at which port
    properties are constant.
18
19 Y = Cp_0/Cv_0;// Ratio of heat capacities.

```

```

20 //From part(1) we obtained the relation ,
21 // T_2 = 1/(((P_2-P_1)/(Y*T_L*P_2))+(P_1/(P_2*T_1)))
22 // Not that when P_2 >> P_1 ,T_2 approaches Y*T_L
    and thus
23 T_2 = Y*T_L;//[K]
24 printf(" (b).The final temperature is %f K",T_2);

```

Scilab code Exa 3.21 Determination of equilibrium temperature

```

1 clear;
2 clc;
3
4 //Example - 3.21
5 //Page number - 119
6 printf("Example - 3.21 and Page number - 119\n\n");
7
8 //Given
9 T_1 = 40 + 273.15;//[K] - Initial temperature.
10 P_1 = 1;//[bar] - Initial pressure.
11 P_1 = P_1*10^(5);//[Pa]
12 Vol_1 = 0.01;//[cubic metre] - Initial volume of the
    cylinder.
13 T_2 = 100 + 273.15;//[K] - Final temperature.
14 P_2 = 100;//[kPa] - Final pressure.
15 P_2 = P_2*10^(5);//[Pa]
16 Vol_2 = 0.02;//[cubic metre] - Final volume of the
    cylinder.
17 Cp = 1.005;//[J/g-K] - Specific heat capacity at
    constant pressure.
18 Cv = 0.718;//[J/g-K] - Specific heat capacity at
    constant volume.
19 Mol_wt = 28.84;//[g/mol] - Molecular weight of air.
20 R = 8.314;//[J/mol-K] - universal gas constant
21
22 delta_Vol = Vol_2 - Vol_1;//[cubic metre] - Change

```

```

    in volume.
23 // Assuming ideal gas  $P*V = R*T$ 
24  $V_1 = (R*T_1)/P_1$ ; // [ $m^3/mol$ ] - Initial specific
    volume.
25 // Therefore, the total number of moles initially in
    the system is ,
26  $n_1 = (Vol_1/V_1)$ ; // [mol]
27  $m_1 = n_1*Mol\_wt$ ; // [g] - Initial mass of the system
    .
28  $Y = Cp/Cv$ ; // Ratio of heat capacities
29
30 // The energy balance equation is given by
31 //  $-P*\Delta Vol + H_{liq}*(m_2 - m_1) = m_2*Cv*(P*V_2)/$ 
     $R - m_1*Cv*T_1$ 
32 //  $m_2*Cv*(P*V_2)/R = (Cv*P_1*Vol_2)/R$ 
33 //  $Cv/R = 1/(Y-1)$ 
34 // Since pressure of the gas in system is assumed
    constant, therefore it remains at 1 bar and thus  $P$ 
    =  $P_1$  ,
35  $H_{liq} = Cp*T_2$ ; // [J/g] - Enthalpy of liquid
36  $m_2 = (P_1*\Delta Vol + ((P_1*Vol_2)/(Y-1)) + H_{liq}*$ 
     $m_1 - m_1*Cv*T_1)/H_{liq}$ ; // [g]
37
38 //The mass entering the assembly during the filling
    process is given by
39  $m = m_2 - m_1$ ; // [g]
40  $n_2 = m_2/Mol\_wt$ ; // [mol] - Number of moles in the
    final state.
41  $V_2 = Vol_2/n_2$ ; // [ $m^3/mol$ ] - Final specific
    volume.
42 // Therefore, final temperature is given by,
43  $T_2 = (P_1*V_2)/R$ ; // [K] - Final temperature.
44
45 printf(" The final equilibrium temperature is %f K\n
    ",  $T_2$ );
46 printf(" The mass entering through the valve is %f g
    \n",  $m$ );

```

Scilab code Exa 3.22 Determination of mass

```
1 clear;
2 clc;
3
4 //Example - 3.22
5 //Page number - 122
6 printf("Example - 3.22 and Page number - 122\n\n");
7
8 //Given
9 V_total = 5; //[L] - Volume of pressure cooker.
10 V_total = V_total*10^(-3); //m^(3)
11 P_gauze = 15; //[psi] - Operating pressure (gauze) of
    pressure cooker.
12 P_gauze = (P_gauze/14.5)*10^(5); // [N/m^(2)]
13 P_atm = 0.966*10^(5); // [N/m^(2)] - Atmospheric
    pressure.
14 m_1 = 1; //[kg] - Initial mass.
15 t = 30*60; //[s] - Total time.
16 J = 500; //[W] - Rate of heat supply
17
18 P_abs = P_gauze + P_atm; // [N/m^(2)] - Absolute
    pressure.
19 //The energy balance equation gives,
20 //  $Q = m_e H_e + (m_2 U_2 - m_1 U_1)$ , where 'm_e' is
    the mass exit from the system and 'H_e' is
    enthalpy at exit conditions.
21
22 //It has been reported in the book that from steam
    table at P_abs,
23 T_sat = 120.23; //[K] - Saturated temperature
24 V_liq = 0.001061; //[m^(3)/kg] - specific volume of
    liquid.
25 V_vap = 0.8857; //[m^(3)/kg] - specific volume of
```



```

    vapour .
26 U_liq = 504.49;//[kJ/kg] - specific internal energy
    of liquid .
27 U_vap = 2529.5;//[kJ/kg] - specific internal energy
    of vapour .
28 H_liq = 504.70;//[kJ/kg] - specific enthalpy of
    liquid .
29 H_vap = 2706.7;//[kJ/kg] - specific internal energy
    of vapour .
30
31 //We know that total volume occupied by 1 kg of
    fluid is
32 //V_total = (1-x)*V_liq + x*V_vap
33 x1 = (V_liq - V_total)/(V_liq - V_vap);//[g]
34
35 //Internal energy at this state is
36 U_1 = (1-x1)*U_liq + x1*U_vap;//[kJ/kg] - specific
    internal energy
37 U_1_net = m_1*U_1;//[kJ] - Internal energy
38
39 //The amount of heat supplied is given by,
40 J_net = J*t;//[J] - Net heat supplied .
41 J_net = J_net*10(-3);//[kJ]
42
43 //Let the dryness factor at the end of the process
    be x
44 //The specific properties of the liquid and vapour
    remain same as P and T_sat are the same in the
    cooker .
45 //Let the total mass of H2O (liquid + vapour) at the
    end of the process be 'm' kg.
46 // V_total/m = (1-x)*(V_liq) + x*V_vap .....
    equqtion (1)
47
48 //the specific internal energy at the end of process
    is
49 //U = (1-x)*U_liq + x*U_vap
50 //The total internal energy at the end of the

```

```

        process is
51 //U_net = m*U = x*[(1-x)*U_liq + x*U_vap]
52
53 //The energy balance equation gives ,
54 // Q = m_e*H_e +(m_2*U_2 - m_1*U_1), where 'm_e' is
        the mass exit from the system and 'H_e' is
        enthalpy at exit conditions.
55 //Since the vapour which exits out have enthalpy
        equal to that of saturated vapour,we get on
        simplification
56 // 900 = (1-m)*(2706.7) + m*((1-x)*504.49 + x
        *2529.5) - 513.5.....equation(2)
57 // The second equation on simplification becomes
58 // x = ((0.005/m) - 0.001061)/0.884639
59
60 // Putting the expression of x in first equation and
        then simplifying , we get
61 // - 1293.2 = -2202.21*m + 11.445 - 2.429*m
62 m = (11.445+1293.2)/(2202.21+2.429);//[kg]
63
64 // Therefore x can be calculated as
65 x = ((0.005/m) - 0.001061)/0.884639;
66
67 // Therefore total water (liquid + vapour) present in
        the pressure cooker at the end of the process is
        m kg.
68 m_vapour = x*m;//[kg] - Mass of vapour
69 m_liquid = (1-x)*m;//[kg] - Mass of vapour
70
71 printf(" Total water (liquid + vapour) present in
        the pressure cooker at the end of the process is
        %f kg\n",m);
72 printf(" The mass of vapour is %f kg\n",m_vapour);
73 printf(" The mass of liquid is %f kg\n",m_liquid);

```

Chapter 4

The Second Law and Its Applications

Scilab code Exa 4.1 Calculation of entropy change

```
1 clear;
2 clc;
3
4 //Example - 4.1
5 //Page number - 148
6 printf("Example - 4.1 and Page number - 148\n\n");
7
8 //Given
9 n = 1000; //[mol]
10 T = 400; //[K]
11 P_1 = 100; //[kPa]
12 P_2 = 1000; //[kPa]
13 R = 8.314; //[J/mol*K] - Universal gas constant
14
15 //(a)
16 T_surr = 400; //[K] - surrounding temperature
17 // Total change in entropy of the system is given by
18 //  $\Delta S_{sys} = n(C_{p,0} \log(T_2/T_1) - R \log(P_2/P_1))$ 
```

```

19 // The process being isothermal the first term is
    zero and the total entropy change of the system
    is
20 delta_S_sys_a = - n*R*log(P_2/P_1); // [J/K]
21 delta_S_sys_a = delta_S_sys_a*10^(-3); // [kJ/K]
22
23 // Since the process is reversible therefore
24 Q_sys = T_surr*delta_S_sys_a; // [kJ] - Heat change in
    the system
25 // Negative sign in the value of Q_sys implies that
    heat is released from the system and is released
    to the surroundings, therefore
26 Q_surr = - Q_sys; // [kJ] - Heat change in the
    surrounding
27 delta_S_surr_a = Q_surr/T_surr; // [kJ/K]
28
29 delta_S_univ_a = delta_S_sys_a + delta_S_surr_a; // [
    kJ/K]
30 // We get delta_S_univ = 0, which is true for a
    reversible process
31
32 printf(" (a).The entropy change of the gas is given
    by delta_S_sys = %f kJ/K \n", delta_S_sys_a);
33 printf("      The entropy change of the surrounding
    is, delta_S_surr = %f kJ/K \n", delta_S_surr_a);
34 printf("      The total entropy change of the gas is,
    delta_S_univ = %f kJ/K \n\n", delta_S_univ_a);
35
36 //(b)
37 T_surr_b = 300; // [K] - surrounding temperature
38 // Since the initial and final states are fixed
    therefore the entropy change of the system is
    same whether the process is carried out
    reversibly or irreversibly.
39 delta_S_sys_b = delta_S_sys_a;
40
41 // Work done under reversible condition is given by
42 // W = integral(P*dV) = integral(((R*T)/V)*dV) = R*T

```

```

    *log(V_2/V_1)
43 // For ideal gas we have,  $P_1V_1/T_1 = P_2V_2/T_2$  or,  $V_2/V_1 = P_1/P_2$  (for isothermal conditions)
44 W = R*T*log(P_1/P_2);//[J/mol]
45 W = W*10^(-3);//[kJ/mol]
46 // 20% extra work has to be done for the system to
    reach the same final state as under reversible
    conditions. Therefore
47 W = W*(120/100);//[kJ/mol]
48 W = W*n;//[kJ] – Total work done for n moles
49
50 // Using the first law we have  $\Delta U = Q - W$ . Now
    under isothermal conditions for ideal gas,
     $\Delta U = 0$ . Therefore,
51 Q = -W;
52 // It implies that whatever work is done on the
    system is lost as heat to the surroundings.
53 // Since heat is gained by the surroundings
    therefore
54 delta_S_surr_b = Q/T_surr_b;//[kJ/K]
55
56 delta_S_univ_b = delta_S_sys_b + delta_S_surr_b;//[
    kJ/K]
57
58 printf(" (b).The entropy change of the gas is given
    by  $\Delta S_{sys} = \text{\%f kJ/K \n}$ ",delta_S_sys_b);
59 printf("      The entropy change of the surrounding
    is,  $\Delta S_{surr} = \text{\%f kJ/K \n}$ ",delta_S_surr_b);
60 printf("      The total entropy change of the gas is,
     $\Delta S_{univ} = \text{\%f kJ/K \n\n}$ ",delta_S_univ_b);

```

Scilab code Exa 4.2 Determination of whether the process is reversible or not

```
1 clear;
```

```

2  clc;
3
4  //Example - 4.1
5  //Page number - 148
6  printf("Example - 4.1 and Page number - 148\n\n");
7
8  // Given
9  T = 400;//[K] - Temperature
10 P_1 = 500*10^(3);//[Pa] - Initial pressure
11 P_2 = 100*10^(3);//[Pa] - Final pressure
12 V_1 = 750*10^(-6);//[m^(3)] - Initial volume
13 W_actual = 0.55*10^(3);//[J] - Actual work done
14 R = 8.314;//[J/mol*K] - Universal gas constant
15
16 // Suppose that the surroundings are at 400 K.
17 // Therefore the process is externally reversible as
    temperature of the surroundings is same as
    system temperature.
18 // The number of moles is given by
19 n = (P_1*V_1)/(R*T);//[mol]
20 // The entropy change of ideal gas under isothermal
    condition is given by
21 delta_S_sys = - n*R*log(P_2/P_1);//[J/mol]
22
23 // The heat supplied to the system in the internally
    reversible process is
24 Q_theot = T*delta_S_sys;//[J]
25 // Since the process is isothermal therefore ,
    workdone is given by
26 W_theot = Q_theot;//[J] - Theoretical work done
27 // Since actual work done by the gas is 0.55 kJ
    therefore actual heat supplied is also 0.55 kJ
    because under isothermal conditions delta_U = 0
28 Q_actual = W_actual;
29
30 // Since Q_theot > Q_actual , so the process is
    irreversible
31 printf(" Since , Q_theot = %f J is greater than

```

```

    Q_actual = %f J\n",Q_theot,Q_actual);
32 printf(" Therefore ,the process is internally
    irreversible");
33
34 // Moreover delta_S_sys is same whether the process
    is reversible or irreversible as the initial and
    final states is the same.
35 // In the reversible process higher amount of heat
    is supplied (as compared to irreversible) due to
    which delta_S_sys take place.
36 // In the irreversible process the entropy of system
    increases due two reasons : heat supplied and
    entropy generation
37 // So in the irreversible case amount of heat
    supplied is less as compared to reversible case
    as entropy generation term also adds to the
    entropy change of system
38 // delta_S_sys = Q/T_b + S_gen
39 S_gen = delta_S_sys - (Q_theot/T);//[J/K]
40 // The entropy generated may be due to friction and
    other dissipative effects or due to non-quasi-
    static expansion

```

Scilab code Exa 4.3 Calculation of final pressure temperature and increase in entropy

```

1 clear;
2 clc;
3
4 //Example - 4.3
5 //Page number - 150
6 printf("Example - 4.3 and Page number - 150\n\n");
7
8 // Given
9 R = 8.314;//[J/mol*K] - Universal gas constant

```

```

10 // For side A
11 V_A = 1; //[L] - Volume
12 V_A = V_A*10^(-3); //[m^(3)]
13 T_A = 300; //[K] - Temperature
14 P_A = 2; //[atm] - Pressure
15 P_A = P_A*101325; //[Pa]
16
17 // For side B
18 V_B = 1; //[L] - volume
19 V_B = V_B*10^(-3); //[m^(3)]
20 T_B = 300; //[K] - Temperature
21 P_B = 1; //[atm] - Pressure
22 P_B = P_B*101325; //[Pa]
23
24 // From first law final temperature and pressure are
    given by (example 3.30)
25 //  $T = ((n_A*T_A) + (n_B*T_B))/(n_A + n_B)$ 
26 //  $P = ((P_A*V_A) + (P_B*V_B))/(V_A + V_B)$ 
27
28 // Since in this case  $T_A = T_B$ , therefore final
    pressure is given by
29  $P = ((P_A*V_A) + (P_B*V_B))/(V_A + V_B)$ ; //[Pa]
30  $P = P/101325$ ; //[atm]
31
32 printf(" The final temperature is %f K\n",T_A);
33 printf(" The final pressure is %f atm\n",P);
34
35 // The number of moles of air on each side are
36  $n_A = (P_A*V_A)/(R*T_A)$ ; //[mol]
37  $n_B = (P_B*V_B)/(R*T_B)$ ; //[mol]
38
39  $\text{delta}_S_A = -n_A*R*\log((P*101325)/P_A)$ ; //[J/K] -
    Entropy change on side A
40  $\text{delta}_S_B = -n_B*R*\log((P*101325)/P_B)$ ; //[J/K] -
    Entropy change on side B
41  $\text{delta}_S_{\text{sys}} = \text{delta}_S_A + \text{delta}_S_B$ ; //[J/K] - Total
    entropy change of system
42

```



```

43 // Since the system is insulated there is no heat
    exchange with the surroundings , therefore entropy
    change of surrounding is zero
44 delta_S_surr = 0;//[J/K]
45 delta_S_univ = delta_S_sys + delta_S_surr;//[J/K]
46 printf(" The total increase in entropy is %f J/K" ,
    delta_S_univ);
47
48 // The entropy change of the system can also be
    writtten as
49 // delta_s_sys = Q/T_b + S_gen
50 // Since there is no heat transfer , therefore
51 S_gen = delta_S_univ;//[J/K]
52 // The process is reversible because of entropy
    generation due to spontaneous release of piston .

```

Scilab code Exa 4.4 CA Calculation of final temperature heat transfer and change of entropy

```

1 clear;
2 clc;
3
4 //Example - 4.4
5 //Page number - 151
6 printf("Example - 4.4 and Page number - 151\n\n");
7
8 // Given
9 V_vessel = 0.2;//[m^(3)] - Volume of the vessel
10 P_1 = 10;//[bar] - Initial pressure inside the
    vessel
11 P_1 = P_1*10^(5);//[Pa]
12 P_2 = 3.5;//[bar] - Final pressure inside the vessel
13 P_2 = P_2*10^(5);//[Pa]
14 T_1 = 250 + 273.15;//[K] - Initial temperature of
    the vessel

```

```

15 R = 8.314; //[J/mol*K] - Universal gas constant
16
17 // (a)
18 // At 10 bar and 250 C the steam is superheated.
    From steam table as reported in book we have
19 V_1 = 0.2327; //[m^(3)/kg] - specific volume
20 U_1 = 2709.9; //[kJ/kg] - specific internal energy
21 H_1 = 2942.6; //[kJ/kg] - Specific enthalpy
22 S_1 = 6.9247; //[kJ/kg-K] - Specific entropy
23 // the quantity of steam is given by
24 m = V_vessel/V_1; //[kg]
25
26 // At final state
27 V_2 = 0.2327; //[m^(3)/kg] - Molar volume
28 V_liq_2 = 0.001079; //[m^(3)/kg]
29 V_vap_2 = 0.5243; //[m^(3)/kg]
30 // Since overall volume lies between saturated
    liquid and saturated vapour therefore the steam
    is saturated and its dryness fraction at final
    state is given by
31 x = (V_2 - V_liq_2)/(V_vap_2 - V_liq_2);
32 // Final temperature = T_sat (at 3 bar) from steam
    table
33 T_final = 138.88; //[C]
34
35 // At 3.5 bar saturated conditions
36 S_liq = 1.7275; //[kJ/kg-K] - Entropy of saturated
    liquid
37 S_vap = 6.9405; //[kJ/kg-K] - Entropy of saturated
    vapour
38 U_liq = 583.95; //[kJ/kg] - Internal energy of
    saturated liquid
39 U_vap = 2548.9; //[kJ/kg] - Internal energy of
    saturated vapour
40 // Therefore at final state
41 U_2 = U_liq*(1 - x) + x*U_vap; //[kJ/kg]
42 S_2 = S_liq*(1 - x) + x*S_vap; //[kJ/kg-K]
43 Q_1 = m*(U_2 - U_1); //[kJ]

```

```

44 delta_S_1 = m*(S_2 - S_1);//[kJ/kg-K]
45
46 printf(" (a).The final temperature is %f C\n",
         T_final);
47 printf("      The amount of heat transfer is %f kJ\n"
         ,Q_1);
48 printf("      The change of entropy is %f kJ/kg-K\n\n"
         ",delta_S_1);
49
50 // (b)
51 Y = 1.4;// Ratio of heat capacities for air
52 // (P_1*V_1)/T_1 = (P_2*V_2)/T_2 and since V_1 =
         V_2
53 T_2 = (P_2/P_1)*T_1;//[K]
54
55 // Since the volume is fixed therefore work done (W)
         = 0 and from first law we get
56 // Q = delta_U = n*Cv_0*(T_2 - T_1)
57 Cv_0 = R/(Y - 1);//[J/mol-K] - Heat capacity at
         constant volume
58 Cp_0 = (Y*R)/(Y - 1);//[J/mol-K] - Heat capacity at
         constant pressure
59 n = (P_1*V_vessel)/(R*T_1);//[mol] - No. of moles
60 Q_2 = n*Cv_0*(T_2 - T_1);//[J] - Heat change
61 Q_2 = Q_2*10^(-3);//[kJ]
62
63 delta_S_2 = Cp_0*log(T_2/T_1) - R*log(P_2/P_1);//[J/
         mol-K]
64 delta_S_2 = n*delta_S_2*10^(-3);//[kJ/K]
65
66 printf(" (b).The final temperature is %f C\n",T_2);
67 printf("      The amount of heat transfer is %f kJ\n"
         ,Q_2);
68 printf("      The change of entropy is %f kJ/K\n",
         delta_S_2);

```

Scilab code Exa 4.5 Calculation of final temperature work and heat transfer

```

1  clear;
2  clc;
3
4  //Example - 4.5
5  //Page number - 153
6  printf(" Example - 4.5 and Page number - 153\n\n");
7
8  // Given
9  m = 1000; //[g] - Mass of fluid
10 P_1 = 20; //[bar] - Initial pressure
11 P_1 = P_1*10^(5); //[Pa]
12 P_2 = 2; //[bar] - Final pressure
13 P_2 = P_2*10^(5); //[Pa]
14 T_1 = 250 + 273.15; //[K] - Initial temperature
15 n = 1.25;
16 R = 8.314; //[J/mol*-] - Universal gas constant
17 Y = 1.4; // Index of expansion
18 Cv_0 = R/(Y- 1); //[J/mol-K]
19 Cp_0 = R + Cv_0; //[J/mol-K]
20
21 //(a)
22 // For steam at 20 bar and 250 C, from steam table
    as reported in the book
23 V_1 = 0.11144; //[m^(3)/kg]
24 U_1 = 2679.6; //[kJ/kg]
25 S_1 = 6.5453; //[kJ/kg-K]
26 // P_1*V_1^(n) = P_2*V_2^(n)
27 V_2 = ((P_1*V_1^(n))/P_2)^(1/n); //[m^(3)/kg]
28
29 // At 2 bar under saturated conditions ,from steam
    table as reported in the book

```

```

30 V_liq = 0.001061; // [m3/kg]
31 V_vap = 0.8857; // [m3/kg]
32 x = (V_2 - V_liq)/(V_vap - V_liq); // Dryness
    fraction
33 T_sat = 120.23; // [C] - The final temperature
34 U_liq = 504.49; // [kJ/kg] - Internal energy of
    saturate liquid
35 U_vap = 2529.5; // [kJ/kg] - Internal energy of
    saturate vapour
36 // Therefore, internal energy at state 2 is given by
37 U_2 = U_liq*(1 - x) + x*U_vap; // [kJ/kg]
38
39 // Work transfer is given by
40 W = (P_1*V_1 - P_2*V_2)/(n - 1); // [J/kg]
41 W = W*10(-3); // [kJ/kg]
42 delta_U = U_2 - U_1; // [kJ/kg]
43
44 // From first law, q - W = delta_U
45 q = W + delta_U; // [kJ/kg]
46
47 // At final state (2 bar saturated), as reported in
    the book
48 S_liq = 1.5301; // [kJ/kg-K] - Entropy of saturated
    liquid
49 S_vap = 7.1271; // [kJ/kg-K] - Entropy of saturated
    vapour
50 // Therefore, entropy at state 2 is given by
51 S_2 = S_liq*(1 - x) + x*S_vap; // [kJ/kg-K]
52 delta_S = S_2 - S_1; // [kJ/kg-K]
53
54 printf(" (a).The final temperature is %f C\n",T_sat)
    ;
55 printf("      The work done is equal to %f kJ/kg\n",W
    );
56 printf("      The heat change is equal to %f kJ/kg\n"
    ,q);
57 printf("      The entropy change is equal to %f kJ/kg
    -K\n\n",delta_S);

```

```

58
59 //(b)
60 // P*V^(n) = constant
61 // Since the gas behaves as ideal we can write
62 // P_1^(1-n)*T_1^(n) = P_2^(1-n)*T_2^(n)
63 T_2 = T_1*(P_1/P_2)^((1-n)/n);//[K]
64
65 // Molar volume is given by
66 V_2_1 = (R*T_1)/P_1;//[m^(3)/mol] - At state 1
67 V_2_2 = (R*T_2)/P_2;//[m^(3)/mol] - At state 2
68
69 // Work transfer is given by
70 w_2 = ((P_1*V_2_1) - (P_2*V_2_2))/(n-1);//[J/mol]
71 Mol_wt_air = 0.21*32 + 0.79*28;//[g/mol] - Molecular
    weight of air
72 n_mole = m/Mol_wt_air;
73 // Total work transfer is given by
74 W_2 = w_2*n_mole*10^(-3);//[kJ]
75 // Internal energy change is given by
76 delta_U = n_mole*Cv_0*(T_2 - T_1)*10^(-3);//[kJ]
77
78 // Heat transfer is given by
79 Q = W_2 + delta_U;//[kJ]
80
81 // Entropy change is given by
82 delta_S_2 = Cp_0*log(T_2/T_1) - R*log(P_2/P_1);//[J/
    mol-K]
83 delta_S_2 = delta_S_2*n_mole;//[J/mol]
84
85 printf(" (b).The final temperature is %f C\n",T_2);
86 printf("      The work done is equal to %f kJ/kg\n",
    W_2);
87 printf("      The total heat change is equal to %f kJ
    \n",Q);
88 printf("      The entropy change is equal to %f kJ/kg
    -K\n\n",delta_S_2);

```

Scilab code Exa 4.6 Calculation of final temperature and work done

```
1 clear;
2 clc;
3
4 //Example - 4.6
5 //Page number - 154
6 printf("Example - 4.6 and Page number - 154\n\n");
7
8 //Given
9 m = 1000; //[g] - Mass of fluid
10 P_1 = 20; //[bar] - Initial pressure
11 P_2 = 2; //[bar] - Final pressure
12 T_1 = 250 + 273.15; //[K] - Initial temperature
13 R = 8.314; //[J/mol*K] - Universal gas constant
14
15 // (a).
16 // At 20 bar and 250 C as reported in the book
17 V_1 = 0.11144; //[m^3/kg] - Specific volume
18 U_1 = 2679.6; //[kJ/kg] - Specific internal energy
19 S_1 = 6.5453; //[kJ/kg-K] - Specific entropy
20 S_2 = S_1; // Isentropic expansion
21
22 // At 2 bar under saturated conditions
23 S_liq = 1.5301; //[kJ/kg-K]
24 S_vap = 7.1271; //[kJ/kg-K]
25 U_liq = 504.49; //[kJ/kg-K]
26 U_vap = 2529.5; //[kJ/kg-K]
27 // Therefore dryness factor can be determined as
28 x = (S_1 - S_liq)/(S_vap - S_liq);
29 U_2 = U_liq*(1 - x) + x*U_vap; //[kJ/kg] - Specific
    internal energy at final state
30 delta_U = U_2 - U_1; //[kJ/kg] - change in internal
    energy
```

```

31 W = - delta_U; // - Work done
32
33 // The final saturated temperature at 2 bar from
    steam table is
34 T_2 = 120.23; // [C]
35
36 printf(" (a).The final temperature is %f C\n",T_2);
37 printf("      The work done is equal to %f kJ/kg\n\n"
    ,W);
38
39 // (b).
40 Y = 1.4; // Index of expansion for air
41 Cv_0 = R/(Y-1); // [J/mol*K] - Specific heat capacity
    at constant volume
42 // Ideal gas under isentropic expansion      P_1^(1-Y)
    *T_1^(Y) =P_2^(1-Y)*T_2^(Y)
43 T_2_prime = T_1*(P_1/P_2)^((1-Y)/Y); // [K] - Final
    temperature
44 delta_U_prime = Cv_0*(T_2_prime - T_1); // [J/mol] -
    change in internal energy
45
46 // Number of moles is given by
47 n = m/28.84; // [mol]
48 delta_U_prime = delta_U_prime*n*10^(-3); // [kJ]
49 W_prime = - delta_U_prime; // Work done
50
51 printf(" (b).The final temperature is %f C\n",
    T_2_prime);
52 printf("      The work done is equal to %f kJ/kg\n",
    W_prime);

```

Scilab code Exa 4.7 Determination of index of isentropic expansion

```

1 clear;
2 clc;

```



```

3
4 //Example - 4.7
5 //Page number - 155
6 printf("Example - 4.7 and Page number - 155\n\n");
7
8 //Given
9 P_1 = 15;//[bar] - Initial pressure
10 P_2 = 0.15;//[bar] - Final pressure
11
12 // We know that during isentropic expansion
13 //  $W = ((P_1*V_1) - (P_2*V_2))/(Y - 1)$ 
14
15 // At 15 bar (saturated vapour), from steam table as
    reported in the book
16 V_1 = 0.13177;//[m3/kg]
17 U_1 = 2594.5;//[kJ/kg]
18 S_1 = 6.4448;//[kJ/kg-K]
19
20 // Now at state 2 (P_2 = 0.15 bar),from steam table
    as reported in the book
21 S_2 = S_1;// Isentropic expansion
22 S_liq = 0.7549;//[kJ/kg-K]
23 S_vap = 8.0085;//[kJ/kg-K]
24 U_liq = 225.92;//[kJ/kg]
25 U_vap = 2448.7;//[kJ/kg]
26 V_liq = 0.001014;//[m3/kg]
27 V_vap = 10.02;//[m3/kg]
28
29 // Therefore dryness factor can be calculated as
30 x = (S_1 - S_liq)/(S_vap - S_liq);
31 U_2 = U_liq*(1 - x) + x*U_vap;//[kJ/kg] - Specific
    internal energy at final state
32 delta_U = U_2 - U_1;//[kJ/kg] - change in internal
    energy
33 W = - delta_U;// - Work done
34
35 // The specific volume at the final state is
36 V_2 = V_liq*(1 - x) + x*V_vap;//[m3/kg]

```

```

37
38 // From work done under adiabatic conditions we get
39 //  $W = ((P_1*V_1) - (P_2*V_2))/(Y - 1)$ 
40 Y = (((P_1*V_1) - (P_2*V_2))/W) + 1;
41
42 printf(" The index of expansion is given by Y = %f\n
      ",Y);

```

Scilab code Exa 4.8 Determination of entropy production

```

1 clear;
2 clc;
3
4 //Example - 4.8
5 //Page number - 157
6 printf("Example - 4.8 and Page number - 157\n\n");
7
8 //Given
9 P_1 = 40; //[bar] - Initial pressure
10 T_1 = 500; //[C] - Initial temperature
11 Vel_1 = 140; //[m/s] - Initial velocity
12 T_2 = 100; //[C] - Final temperature
13 Vel_2 = 80; //[m/s] - Final velocity
14 W = 746.0; //[kJ/kg] - Work output
15
16 // (a).
17 // From steam table as reported in the book
18 H_1 = 3445.3; //[kJ/kg]
19 H_2 = 2676.1; //[kJ/kg]
20 S_1 = 7.0901; //[kJ/kg-K]
21 S_2 = 7.3549; //[kJ/kg-K]
22
23 // The temperature at which heat exchange take place
   is given by
24 T_b =(T_1 + T_2)/2 + 273.15; //[K]

```

```

25
26 // From first law in a control volume
27 //  $q - W = \Delta H + (\Delta V^2)/2$  , therefore
28  $q = W*10^3 + (H_2 - H_1)*10^3 + (Vel_2^2 -$ 
     $Vel_1^2)/2$ ; // [J/kg]
29  $q = q*10^{-3}$ ; // [kJ/kg]
30
31  $S_{gen} = (S_2 - S_1) - (q/T_b)$ ; // [kJ/kg-K]
32
33 printf(" (a).The specific entropy production within
    turbine is %f kJ/kg-K\n", S_gen);
34
35 //(b)
36 // If control volume is too large to include the
    turbine and the environment then T_b becomes
    equal to 289 K. In this case
37 T_b_prime = 298; // [K]
38
39 // The entropy change of the system is given by
40 //  $\Delta S = q/T_b + S_{gen}$ 
41  $S_{gen} = (S_2 - S_1) - (q/T_{b\_prime})$ ; // [kJ/kg-K]
42
43 printf(" (b).The specific entropy production within
    turbine is %f kJ/kg-K", S_gen);
44
45 // In the first part only irreversibilities within
    the turbine are evaluated
46 // whereas in part (2) irreversible heat transfer
    between the turbine cover and environment are
    also included.

```

Scilab code Exa 4.9 Determination of work required and exit temperature

```

1 clear;
2 clc;

```

```

3
4 //Example - 4.9
5 //Page number - 160
6 printf("Example - 4.9 and Page number - 160\n\n");
7
8 //Given
9 P_1 = 1; //[MPa] - Initial pressure
10 T_1 = 200 + 273.15; //[K] - Initial temperature
11 P_2 = 8; //[MPa] - Final pressure
12 Y = 1.4; // Index of expansion of gas
13 R = 8.314; //[J/mol-K] - Universal gas constant
14
15 //(1)
16 // The exit temperature for ideal gas under
    isentropic conditions is given by
17 T_2 = T_1*((P_2/P_1)^((Y-1)/Y)); //[K] - Exit
    temperature
18 Cp_0 = Y*R/(Y-1); //[J/mol-K] - Specific heat
    capacity at constant pressure
19 // For isentropic conditions the enthalpy change for
    ideal gas is given by
20 delta_H_s = Cp_0*(T_2 - T_1); //[J/mol]
21 // Therefore work is given by
22 W = - delta_H_s; //[J/mol]
23
24 printf(" (1).The exit temperature of steam is %f K\n
    ",T_2);
25 printf("      The required work is %f J/mol\n\n",W);
26
27 //(2)
28 eff = 0.8; // Adiabatic efficiency
29 // delta_H_s/delta_H_a = 0.8
30 delta_H_a = delta_H_s/eff; //[J/mol] - Actual
    enthalpy change
31 W_a = - delta_H_a; //[J/mol]
32
33 // The ideal gas enthalpy is a function only of
    temperature, therefore actual exit temperature

```

```

    T_2a is given by
34 // delta_H_a = Cp_0*(T_2a - T_1)
35 T_2a = (delta_H_a/Cp_0) + T_1;
36
37 printf(" (2).The exit temperature of steam is %f K\n
    ",T_2a);
38 printf("      The required work is %f J/mol\n\n",W_a)
    ;

```

Scilab code Exa 4.10 Determination of work required and exit temperature

```

1 clear;
2 clc;
3
4 //Example - 4.10
5 //Page number - 161
6 printf("Example - 4.10 and Page number - 161\n\n");
7
8 //Given
9 P_1 = 1; //[MPa] - Initial pressure
10 T_1 = 200 + 273.15; //[K] - Initial temperature
11 P_2 = 8; //[MPa] - Final pressure
12 Y = 1.4; // Index of expansion of gas
13 R = 1.987; //[cal/mol*K] - Universal gas constant
14 // Cp_0 = 7.7 + 0.04594*10^(-2)*T + 0.2521*10^(-5)*T
    ^ (2) - 0.8587*10^(-9)*T^ (3), here T is in K and
    Cp_0 is in cal/mol-K
15 a = 7.7;
16 b = 0.04594*10^(-2);
17 c = 0.2521*10^(-5);
18 d = - 0.8587*10^(-9);
19
20 // delta_S = integral((Cp_0/T)*dT) - R*log(P_2/P_1)
    = 0

```

```

21 // delta_S = integral(((a + b*T + c*T^(2) + d*T^(3))
// /T)*dT) - R*log(P_2/P_1) = 0
22 // delta_S = a*log(T_2/T_1) + b*(T_2 - T_1) + (c/2)
// *(T_2^(2) - T_1^(2)) + (d/3)*(T_2^(3) - T_1^(3))
// - R*log(P_2/P_1) = 0
23 // Solving for T_2 in the above equation we get
24 def f(' [y]=f(T_2)', 'y=a*log(T_2/T_1)+b*(T_2-T_1)+(c
// /2)*(T_2^(2)-T_1^(2))+(d/3)*(T_2^(3)-T_1^(3))-R*
// log(P_2/P_1)');
25 T_2 = fsolve(100,f);
26
27 // Now let us calculate the enthalpy change under
// these conditions
28 delta_H_s = integrate('7.7+0.04594*10^(-2)*T
// +0.2521*10^(-5)*T^(2)-0.8587*10^(-9)*T^(3)', 'T',
// T_1,T_2);//[cal/mol]
29 delta_H_s = delta_H_s*4.184;//[J/mol]
30 // Therefore isentropic work done is
31 W = - delta_H_s;
32
33 printf(" (1).The exit temperature of steam is %f K\n
// ",T_2);
34 printf(" The required work is %f J/mol\n\n",W);
35
36 //(2)
37 eff = 0.8;
38 delta_H_a = delta_H_s/eff;//[J/mol] - Actual
// enthalpy change
39 // Therefore actual work done is given by
40 W_a = - delta_H_a;//[J/mol]
41
42 // Now we have to determine the exit temperature
// under actual conditions
43 // delta_H_a = integral(Cp_0*dT) from limit T_1 =
// 473.15 K to T_2
44 // On putting the values and simplifying we get
45 // 7.7*T_2 + 0.02297*10^(-2)*T_2^(2) + 0.084*10^(-5)
// *T_2^(3) - 0.214675*10^(-9)*T_2^(4) - 6907.106 =

```

```

0
46
47 def f1(T_2_prime) = a*(T_2_prime-T_1)+(b/2)
    *(T_2_prime^2-T_1^2)+(c/3)*(T_2_prime^3-T_1
    ^3)+(d/4)*(T_2_prime^4-T_1^4)-(delta_H_a
    /4.184)');
48 T_2_prime = fsolve(100,f1);
49
50 printf(" (2).The exit temperature of steam is %f K\n
    ",T_2_prime);
51 printf(" The required work is %f J/mol\n\n",W_a)
    ;

```

Scilab code Exa 4.11 Determination of work required and exit temperature

```

1 clear;
2 clc;
3
4 //Example - 4.11
5 //Page number - 162
6 printf("Example - 4.11 and Page number - 162\n\n");
7
8 //Given
9 P_1 = 1; //[MPa] - Initial pressure
10 T_1 = 200 + 273.15; //[K] - Initial temperature
11 P_2 = 8; //[MPa] - Final pressure
12 Y = 1.4; // Index of expansion of gas
13
14 // At state 1 (1 MPa and 200 C) from steam table as
    reported in the book
15 H_1 = 2827.9; //[kJ/kg]
16 S_1 = 6.694; //[kJ/kg]
17 // At state 2 (8 MPa)
18 S_2 = S_1; // Isentropic process

```

```

19 // From steam table at 8 MPa and 450 C
20 S_21 = 6.5551; //[kJ/kg-K]
21 // From steam table at 8 MPa and 500 C
22 S_22 = 6.7240; //[kJ/kg-K]
23 // Therefore temperature at which entropy of steam
    is 6.694 kJ/kg-K is given by
24 T_2 = 450 + (500-450)/(S_22-S_21)*(S_2-S_21); //[C]
25 T_2 = T_2 + 273.15; //[K]
26
27 // Enthalpy of steam at 8 MPa and 450 C from steam
    table as reported in the book
28 H_21 = 3272.0; //[kJ/kg]
29 // And at 8 MPA and 500 C
30 H_22 = 3398.3; //[kJ/kg]
31 // Therefore enthalpy of steam at 8 MPa and T_2
32 H_2 = H_21 + ((H_22-H_21)/(500-450))*((T_2-273.15) -
    450);
33 // Work done is given by
34 // W = - delta_H_s
35 W = - (H_2 - H_1); //[J/g]
36 W = W*18.015; //[J/mol]
37 delta_H_s = - W;
38
39 printf(" (1).The exit temperature of steam is %f K\n
    ",T_2);
40 printf("      The required work is %f J/mol\n\n",W);
41
42 //(2)
43 eff = 0.8; // Adiabatic efficiency
44 // delta_H_s/delta_H_a = 0.8
45 delta_H_a = delta_H_s/eff; //[J/mol] - Actual
    enthalpy change
46 // Therefore actual work done
47 W_a = - delta_H_a; //[J/mol]
48 // Enthalpy at actual exit conditions is
49 H_2_a = H_1 + delta_H_a/18.015; //[kJ/kg]
50
51 // Enthalpy of steam at 8 MPa and 500 C from steam

```



```

        table as reported in the book
52 H_21_a = 3398.3; //[kJ/kg]
53 // And at 8 MPA and 550 C
54 H_22_a = 3521.0; //[kJ/kg]
55 // Therefore temperature at H_22_a is given by
56 T_2_a = 500 + ((550-500)*(H_2_a - H_21_a))/(H_22_a -
        H_21_a); //[C]
57 T_2_a = T_2_a + 273.15; //[K]
58
59 printf(" (2).The exit temperature of steam is %f K\n
        ",T_2_a);
60 printf("          The required work is %f J/mol\n\n",W_a)
        ;

```

Scilab code Exa 4.12 Determination of work required and discharge temperature

```

1 clear;
2 clc;
3
4 //Example - 4.12
5 //Page number - 163
6 printf("Example - 4.12 and Page number - 163\n\n");
7
8 //Given
9 P_1 = 140; //[kPa] - Initial pressure
10 T_1 = 20 + 273.15; //[K] - Initial temperature
11 P_2 = 560; //[kPa] - Final pressure
12 eff = 0.75; // Compressor efficiency
13 R = 1.987; //[cal/mol*K] - Universal gas constant
14 // Cp_0 = 4.750 + 1.200*10^(-2)*T + 0.3030*10^(-5)*T
        ^(-2) - 2.630*10^(-9)*T^(3), here T is in K and
        Cp_0 is in cal/mol-K
15 a = 7.7;
16 b = 0.04594*10^(-2);

```

```

17 c = 0.2521*10(-5);
18 d = - 0.8587*10(-9);
19
20 // At 20 C, as reported in the book
21 Cp_0 = 8.46; // [cal/mol-K] - Specific heat capacity
    at constant pressure
22 Cv_0 = Cp_0 - R; // [cal/mol-K] - Specific heat
    capacity at constant volume
23 Y = Cp_0/Cv_0; // Index of expansion
24
25 // Assuming 100 % efficiency ,for reversible and
    adiabatic process the final temperature is given
    by
26 //  $P*V^{(Y)} = \text{constant}$  or ,  $P*((R*T)/P)^{(Y)} = \text{constant}$ 
27 T_2 = ((P_1/P_2)((1-Y)/Y))*T_1; // [K]
28
29 // Since at final temperature the value of heat
    capacity ratio would have changed
30 // So let us determine Y at mean temperature and
    then calculating final temperature
31 T_mean = (T_1 + T_2)/2; // [K]
32
33 // At T_mean, as reported in the book
34 Cp_0_new = 9.153; // [cal/mol-K]
35 Cv_0_new = Cp_0_new - R; // [cal/mol-K]
36 Y_new = Cp_0_new/Cv_0_new;
37 T_2_new = T_1*((P_1/P_2)((1-Y_new)/Y_new)); // [K]
38
39 // The enthalpy change is given by
40 delta_H = integrate('4.750+1.200*10(-2)*T
    +0.3030*10(-5)*T(2)-2.630*10(-9)*T(3)', 'T',
    T_1, T_2_new); // [cal/mol]
41
42 //For adiabatic process
43 W = - delta_H; // [cal/mol]
44 // Now actual work done on the system is given by
45 W_a = W/eff; // [cal/mol]
46

```

```

47 // Since the actual process is adiabatic the work
    done is change in negative of enthalpy
48 // Therefore actual change in enthalpy is - W_a, or
49 // - W_a = 4.750*(T_2-T_1) + (1.2*10^(-2)/2)*(T_2
    ^ (2)-T_1 ^ (2)) + (0.3030*10^(-5)/3)*(T_2 ^ (3)-T_1
    ^ (3)) - (2.63*10^(-9)/4)*(T_2 ^ (4)-T_1 ^ (4));
50 // Solving for T_2 in the above equation
51 deff(' [y]=f1(T_2_prime)', 'y=4.750*(T_2_prime-T_1)
    +((1.2*10^(-2))/2)*(T_2_prime ^ (2)-T_1 ^ (2))
    +((0.3030*10^(-5))/3)*(T_2_prime ^ (3)-T_1 ^ (3))
    -((2.63*10^(-9))/4)*(T_2_prime ^ (4)-T_1 ^ (4))+W_a')
    ;
52 T_2_prime=fsolve(100,f1);
53
54 printf(" The required work is %f cal/mol\n",W_a);
55 printf(" The discharge temperature of methane is %f
    K\n",T_2_prime);

```

Scilab code Exa 4.13 Determination of power output entropy and exit temperature

```

1 clear;
2 clc;
3
4 //Example - 4.13
5 //Page number - 164
6 printf("Example - 4.13 and Page number - 164\n\n");
7
8 //Given
9 P_1 = 10; //[bar] - Initial pressure
10 T_1 = 500 + 273.15; //[K] - Initial temperature
11 P_2 = 2; //[psia] - Final pressure
12 P_2 = P_2/14.5; //[bar]
13 P_2 = P_2*10^(2); //[kPa]
14 m = 1.8; //[kg/s] - Mass flux

```

```

15 eff = 0.8; // Efficiency
16
17 // At state 1, from steam table
18 H_1 = 3478.5; // [kJ/kg]
19 S_1 = 7.7622; // [kJ/kg-K]
20 S_2 = S_1; // Adiabatic process
21
22 // From saturated steam table at 10 kPa
23 S_liq_1 = 0.6493; // [kJ/kg-K]
24 S_vap_1 = 8.1502; // [kJ/kg-K]
25 // From saturated steam table at 15 kPa
26 S_liq_2 = 0.7549; // [kJ/kg-K]
27 S_vap_2 = 8.0085; // [kJ/kg-K]
28 // Therefore at P_2
29 S_liq = S_liq_1 + ((S_liq_2-S_liq_1)/(15-10))*(P_2
    -10);
30 S_vap = S_vap_1 + ((S_vap_2-S_vap_1)/(15-10))*(P_2
    -10);
31
32 // The dryness fraction at exit state is
33 x_2 = (S_1-S_liq)/(S_vap-S_liq);
34 // The enthalpy at exit to be determined. At 10 kPa
35 H_liq_1 = 191.83; // [kJ/kg]
36 H_vap_1 = 2584.7; // [kJ/kg]
37 // At 15 kPa
38 H_liq_2 = 225.94; // [kJ/kg]
39 H_vap_2 = 2599.1; // [kJ/kg]
40 // Therefore at P_2
41 H_liq = H_liq_1 + ((H_liq_2-H_liq_1)/(15-10))*(P_2
    -10);
42 H_vap = H_vap_1 + ((H_vap_2-H_vap_1)/(15-10))*(P_2
    -10);
43
44 // Enthalpy at state 2
45 H_2_s = H_liq*(1-x_2) + x_2*H_vap; // [kJ/kg]
46 W = m*(H_1 - H_2_s); // [kW]
47
48 printf(" (1).The power output is %f kW\n\n",W);

```

```

49
50 //(2)
51 // If the process is 80 % efficient the enthalpy
    change is
52 //  $H_1 - H_{2_a} = \text{eff}*(H_1 - H_{2_s})$ 
53  $H_{2_a} = H_1 - (0.8*(H_1 - H_{2_s}))$ ;
54
55 // Now under these conditions temperature and
    entropy have to be determined. From superheated
    steam tables ,as reported in the book
56 // At 10 kPa and 100 C
57  $H_{2_1} = 2687.5$ ;//[kJ/kg]
58  $S_{2_1} = 8.4479$ ;//[kJ/kg-k]
59 // At 10 kPa and 150 C
60  $H_{2_2} = 2783.0$ ;//[kJ/kg]
61  $S_{2_2} = 8.6882$ ;//[kJ/kg-k]
62 // At 50 kPa and 100 C
63  $H_{3_1} = 2682.5$ ;//[kJ/kg]
64  $S_{3_1} = 7.6947$ ;//[kJ/kg-k]
65 // At 50 kPa and 150 C
66  $H_{4_1} = 2780.1$ ;//[kJ/kg]
67  $S_{4_1} = 7.9401$ ;//[kJ/kg-k]
68 // Therefore at P_2 and 100 C
69  $H_{\text{prime}_1} = H_{2_1} + ((H_{3_1}-H_{2_1})/(50-10))*(P_2-10)$ 
    ;//[kJ/kg]
70  $S_{\text{prime}_1} = S_{2_1} + ((S_{3_1}-S_{2_1})/(50-10))*(P_2-10)$ 
    ;//[kJ/kg-K]
71 // Therefore at P_2 and 150 C
72  $H_{\text{prime}_2} = H_{2_2} + ((H_{4_1}-H_{2_2})/(50-10))*(P_2-10)$ 
    ;//[kJ/kg]
73  $S_{\text{prime}_2} = S_{2_2} + ((S_{4_1}-S_{2_2})/(50-10))*(P_2-10)$ 
    ;//[kJ/kg-K]
74
75 // Enthalpy at exit is H_2_a. So at this condition
    temperature can be nom be determined
76  $T_{\text{exit}} = ((H_{2_a} - H_{\text{prime}_1})/(H_{\text{prime}_2} - H_{\text{prime}_1})$ 
    )/(150-100) + 100;//[C]
77 // The entropy at exit is

```

```

78 S_exit = ((H_2_a - H_prime_1)/(H_prime_2 -
           H_prime_1))/(S_prime_2 - S_prime_1) + S_prime_1;
           //[kJ/kg-K]
79
80 printf(" (2).The entropy at exit is %f kJ/kg-K\n",
           S_exit);
81 printf("           The temperature of the exit state is %f
           C\n\n",T_exit);
82
83 printf("           The irreversibility is advatageous
           because the exit steam is superheated and
           therefore ,\n");
84 printf("           the blades of the turbine are not
           eroded by water droplets which get formed when
           the process is isentropic");

```

Scilab code Exa 4.14 Calculation of work output per unit mass

```

1  clear;
2  clc;
3
4  //Example - 4.14
5  //Page number - 166
6  printf("Example - 4.14 and Page number - 166\n\n");
7
8  //Given
9  P_1 = 6; //[MPa] - Initial pressure
10 T_1 = 500 + 273.15; //[K] - Initial temperature
11 P_2 = 10; //[kPa] - Final pressure
12 out_qlty = 0.9; // Output quality
13
14 // At 6 MPa and 500 C, from steam table as reported
   in the book
15 H_1 = 3422.2; //[kJ/kg]
16 S_1 = 6.8803; //[kJ/kg-K]

```

```

17 S_2 = S_1; // Adiabatic reversible conditions
18 // At 10 kPa saturated
19 H_liq = 191.83; // [kJ/kg]
20 H_vap = 2584.7; // [kJ/kg]
21 S_liq = 0.6493; // [kJ/kg-K]
22 S_vap = 8.1502; // [kJ/kg-K]
23
24 // The dryness fraction is given by
25 x = (S_1-S_liq)/(S_vap-S_liq);
26
27 // Now the exit enthalpy is given by
28 H_2 = H_liq*(1-x) + x*H_vap; // [kJ/kg]
29 W = - (H_2 - H_1); // [kJ/kg] - Under isentropic
    conditions
30
31 // We know that,  $\Delta S = q/T_b + S_{gen}$ 
32 // Since  $\Delta S = 0$ , therefore under isentropic
    conditions
33 S_gen = 0; // [kJ/kg-K]
34
35 // Now for output quality 0.9
36 H_2_a = H_liq*(1-out_qlty) + out_qlty*H_vap; // [kJ/kg
    ]
37 S_2_a = S_liq*(1-out_qlty) + out_qlty*S_vap; // [kJ/kg
    ]
38 W_a = - (H_2_a - H_1); // [kJ/kg]
39 delta_S_a = S_2_a - S_1; // [kJ/kg-k]
40 // Again,  $\Delta S = q/T_b + S_{gen}$ 
41 // Since  $q = 0$ , therefore under isentropic
    conditions
42 S_gen_a = delta_S_a; // [kJ/kg-K]
43 // Now efficiency is given by  $eff = \Delta H_a / \Delta H_s$ 
44 eff = W_a/W;
45
46 printf(" For output quality = 0.9\n");
47 printf(" The work output per unit mass is %f kJ/kg\n
    ",W_a);

```

```

48 printf(" The entropy generation is given by S_gen =
    %f kJ/kg-K\n",S_gen_a);
49 printf(" The efficiency with respect to reversible
    adiabatic case is given by eff = %f\n\n",eff);
50
51 // Now for output quality 1
52 out_qlty_1 = 1;
53 H_2_a_1 = H_liq*(1-out_qlty_1) + out_qlty_1*H_vap;//
    [kJ/kg]
54 S_2_a_1 = S_liq*(1-out_qlty_1) + out_qlty_1*S_vap;//
    [kJ/kg]
55 W_a_1 = - (H_2_a_1 - H_1);//[kJ/kg]
56 delta_S_a_1 = S_2_a_1 - S_1;//[kJ/kg-k]
57 // Again, delta_S = q/T_b + S_gen
58 // Since q = 0, therefore under isentropic
    conditions
59 S_gen_a_1 = delta_S_a_1;//[kJ/kg-K]
60 // Now efficiency is given by eff = delta_H_a/
    delta_H_s
61 eff_1 = W_a_1/W;
62
63 printf(" For output quality = 1.0\n");
64 printf(" The work output per unit mass is %f kJ/kg\n
    ",W_a_1);
65 printf(" The entropy generation is given by S_gen =
    %f kJ/kg-K\n",S_gen_a_1);
66 printf(" The efficiency with respect to reversible
    adiabatic case is given by eff = %f\n",eff_1);

```

Scilab code Exa 4.15 Estimation of final velocity

```

1 clear;
2 clc;
3
4 //Example - 4.15

```



```

5 //Page number - 168
6 printf("Example - 4.15 and Page number - 168\n\n");
7
8 //Given
9 P_1 = 3; //[bar] - Initial pressure
10 T_1 = 150 + 273.15; //[K] - Initial temperature
11 Vel_1 = 90; //[m/s] - Initial velocity
12 P_2 = 1; //[bar] - Final pressure
13 eff = 0.95; // Adiabatic efficiency of the nozzle
14 R = 8.314; //[J/mol*-] - Universal gas constant
15
16 // At 3 bar and 150 C, from steam table
17 S_1 = 7.0778; //[kJ/kg-K]
18 H_1 = 2761.0; //[kJ/kg]
19 S_2 = S_1; //
20
21 // At 1 bar saturated
22 S_liq = 1.3026; //[kJ/kg-K]
23 S_vap = 7.3594; //[kJ/kg-K]
24 H_liq = 417.46; //[kJ/kg]
25 H_vap = 2675.5; //[kJ/kg]
26 // The dryness factor of exit steam can be
    determined as
27 x = (S_1-S_liq)/(S_vap-S_liq);
28 // Enthalpy of exit steam is given by
29 H_2 = H_liq*(1-x) + x*H_vap; //[kJ/kg]
30 delta_H_s = H_2 - H_1; //[kJ/kg] - Enthalpy change
31 delta_H_a = eff*delta_H_s; //[kJ/kg]
32
33 // Assuming no heat exchange with surroundings and
    since no work is done
34 //  $\Delta H + (\Delta V^2)/2 = 0$ 
35 delta_Vel_square = 2*(-delta_H_a)*1000; //[m^2)/s
    ^2)]
36 Vel_2 = (delta_Vel_square + Vel_1^2)^(1/2); //[m/s]
37
38 printf(" (1).The final velocity (when fluid is steam
    ) is %f m/s\n\n",Vel_2);

```

```

39
40 // (2)
41 Y = 1.4; // Index of expansion
42 Cp_0 = (Y*R)/(Y-1); // [J/mol-K] - Specific heat
    capacity at constant pressure
43 // The final temperature has to be determined such
    that entropy change is zero. Under isentropic
    conditions
44 //  $P_1^{(1-Y)} * T_1^Y = P_2^{(1-Y)} * T_2^Y$ 
45 T_2 = T_1*(P_1/P_2)^((1-Y)/Y); // [K]
46 delta_H_s_2 = Cp_0*(T_2 - T_1); // [J/mol]
47 delta_H_a_2 = eff*delta_H_s_2; // [J/mol]
48 delta_H_a_2 = (delta_H_a_2*1000)/28.84; // [J/kg]
49
50 delta_Vel_square_2 = 2*(-delta_H_a_2); // [m^2/s^2]
    ]
51 Vel_2_2 = (delta_Vel_square_2 + Vel_1^2)^(1/2); // [
    m/s]
52
53 printf(" (2).The final velocity (when fluid is air
    which behaves as an ideal gas) is %f m/s\n\n",
    Vel_2_2);

```

Scilab code Exa 4.16 Calculation of final velocity and increase in entropy

```

1 clear;
2 clc;
3
4 //Example - 4.16
5 //Page number - 169
6 printf("Example - 4.16 and Page number - 169\n\n");
7
8 //Given
9 P_1 = 300; // [kPa] - Initial pressure
10 T_1 = 450; // [K] - Initial temperature

```

```

11 Vel_1 = 90; //[m/s] - Initial velocity
12 P_2 = 180; //[kPa] - Final pressure
13 eff = 0.95; // Adiabatic efficiency of the nozzle
14 R = 8.314; //[J/mol*-] - Universal gas constant
15 Cp = 5.19; //[kJ/kg-K] - Specific heat capacity at
    constant pressure
16
17 //(a)
18 // Exit velocity is highest when drop in enthalpy is
    highest or when isentropic conditions are
    maintained
19
20 Mol_wt_He = 4; //[g/mol] - Molecular weight of helium
21 R_He = R/Mol_wt_He; // 'R' for helium
22 Y = Cp/(Cp - R_He);
23
24 // Now temperature at exit to be determined
25 T_2s = T_1*(P_1/P_2)^((1-Y)/Y); //[K]
26 delta_H_2s = Cp*(T_2s - T_1); //[kJ/kg]
27
28 // Since no work is done and heat exchange is zero ,
    from first law we get
29 // delta_H + delta_Vel^(2)/2 = 0
30 delta_Vel_square = 2*(-delta_H_2s)*1000; //[m^(2)/s
    ^ (2)]
31 Vel_2 = (delta_Vel_square)^(1/2); //[m/s] - ( as
    Vel_1 << Vel_2)
32
33 printf(" (a).The maximum exit velocity is %f m/s\n\n
    ",Vel_2);
34
35 //(b)
36 T_2a = 373; //[K] - Measured temperature of helium
37 delta_H_a = Cp*(T_2a - T_1); //[kJ/kg]
38 delta_Vel_square_a = 2*(-delta_H_a)*1000; //[m^(2)/s
    ^ (2)]
39 Vel_2a = (delta_Vel_square_a)^(1/2); //[m/s] - ( as
    Vel_1 << Vel_2a)

```

```

40
41 printf(" (b).The actual exit velocity is %f m/s\n\n"
    ,Vel_2a);
42
43 //(c)
44 delta_S_sys = Cp*log(T_2a/T_1) - R_He*log(P_2/P_1);
45 // we know that delta_S_sys = q/T_b + S_gen and
    since q = 0, therefore
46 S_gen = delta_S_sys;//[kJ/kg-K]
47
48 printf(" (c).The increase in entropy per unit mass
    is %f kJ/kg-K",S_gen);
49
50 // The source of irreversibility is friction in the
    nozzle.

```

Scilab code Exa 4.17 Calculation of work done and heat transfer

```

1 clear;
2 clc;
3
4 //Example - 4.17
5 //Page number - 170
6 printf("Example - 4.17 and Page number - 170\n\n");
7
8 //Given
9 P_1 = 1;//[bar] - Initial pressure
10 T_1 = 150 + 273.15;//[K] - Initial temperature
11 V_2 = 0.28;//[m^(3)/kg] - Final specific volume
12 T_2 = T_1;//[K] - Isothermal process
13 R = 8.314;//[J/mol-K] - Universal gas constant
14
15 // At 1 bar and 150 C, from steam table
16 S_1 = 7.6134;//[kJ/kg-K]
17 H_1 = 2776.4;//[kJ/kg]

```

```

18
19 // At 150 C saturated
20 V_liq = 0.001091; // [m^(3)/kg]
21 V_vap = 0.3928; // [m^(3)/kg]
22 H_liq = 632.2; // [kJ/kg]
23 H_vap = 2746.5; // [kJ/kg]
24 S_liq = 1.8418; // [kJ/kg-K]
25 S_vap = 6.8379; // [kJ/kg-K]
26
27 // The dryness factor of exit steam can be
    determined as
28 x = (V_2 - V_liq)/(V_vap - V_liq);
29 S_2 = S_liq*(1-x) + x*S_vap; // [kJ/kg-K] -Entropy
30 H_2 = H_liq*(1-x) + x*H_vap; // [kJ/kg] -Enthalpy
31 delta_H = H_2 - H_1; // [kJ/kg] - Enthalpy change
32 delta_S = S_2 - S_1; // [kJ/kg]
33
34 // Since the compression is reversible
35 q = T_1*delta_S; // [kJ/kg] - Heat transfer
36 // From first law q - W = delta_H
37 W = q - delta_H; // [kJ/kg]
38
39 printf(" (1).The amount of heat transfer (when fluid
    is steam) is %f kJ/kg\n",q)
40 printf("      The amount of work transfer (when fluid
    is steam) is %f kJ/kg\n\n",W)
41
42 // (2)
43 V_2 = V_2*(28.84/1000); // [m^(3)/mol] - Molar volume
    at exit
44 // Pressure at exit is given by
45 P_2 = ((R*T_2)/V_2); // [N/m^(2)]
46 P_2 = P_2*10^(-5); // [bar]
47
48 // Entropy change is given by, delta_S_2 = Cp*log(
    T_2/T_1) - R*log(P_2/P_1), but since T_1 = T_2,
    therefore
49 delta_S_2 = - R*log(P_2/P_1); // [J/mol-K]

```

```

50
51 q_2 = T_1*delta_S_2;//[J/mol] - Heat change
52 q_2 = q_2/28.84;//[kJ/kg]
53
54 // Enthalpy change is given by,  delta_H_2 = Cp*(T_2
    - T_1) = 0 (as T_1 = T_2)
55 delta_H_2 = 0;//[kJ/kg]
56
57 // From first law  q - W = delta_H , therefore
58 W_2 = q_2 - delta_H_2;//[kJ/kg]
59
60 printf(" (2).The amount of heat transfer (when fluid
    is ideal gas) is %f kJ/kg\n",q_2)
61 printf("      The amount of work transfer (when fluid
    is ideal gas) is %f kJ/kg\n",W_2)

```

Scilab code Exa 4.18 Calculation of air velocity and change in entropy

```

1 clear;
2 clc;
3
4 //Example - 4.18
5 //Page number - 171
6 printf("Example - 4.18 and Page number - 171\n\n");
7
8 //Given
9 P_1 = 7*10^(5);//[Pa] - Initial pressure
10 T_1 = 95 + 273.15;//[K] - Initial temperature
11 P_2 = 3.5*10^(5);//[Pa] - Final pressure
12 dia = 15*10^(-2);//[m] - Diameter of pipe
13 m = 2;//[kg/s] - Mass flow rate
14 R = 8.314;//[J/mol-K] - Universal gas constant
15 Y = 1.4; // Index of expansion
16 Cp_0 = (R*Y)/(Y-1);//[J/mol-K] - Specific heat
    capacity at constant pressure

```

```

17 Cp_0 = (Cp_0/28.84)*1000; //[J/kg-K]
18 rho_1 = 6.6; //[kg/m^(3)] - Density
19
20 // velocity before throttling is to be determined m
    = rho*Vol*Area
21 V_1 = (R*T_1)/P_1; //[m^(3)/mol] - Specific volume
22 V_1 = (V_1/28.84)*1000; //[m^(3)/kg]
23 Vel_1 = m/(rho_1*3.14*(dia/2)^(2)); //[m/s] -
    Velocity before throttling
24
25 // Let the temperature after throttling be T_2, then
26 // V_2 = (((R*T_2)/P_2)/28.84)*1000
27 // Vel_2 = m/(rho_2*Area) = (m*V_2)/(3.14*(dia/2)
    ^ (2))
28 // From first law, since q = W = 0, we get
29 // delta_H + (delta_V^(2))/2 = 0
30 // Cp_0*(T_2 - T_1) + ((Vel_2)^(2) - (Vel_1)^(2))/2
    = 0
31 //Cp_0*(T_2 - T_1) + (((m*(((R*T_2)/P_2)/28.84)
    *1000))/(3.14*(dia/2)^(2)))^(2) - (Vel_1)^(2))/2
    = 0
32 // Solving the above equation for T_2, we get
33 deff(' [y]=f1(T_2)', 'y=Cp_0*(T_2 - T_1) + (((m*(((R
    *T_2)/P_2)/28.84)*1000))/(3.14*(dia/2)^(2)))^(2)
    - (Vel_1)^(2))/2');
34 T_2 =fsolve(100,f1); //[K] - Temperature after
    throttling
35 // Therefore velocity of air downstream of
    restriction is given by
36 Vel_2 = ((m*(((R*T_2)/P_2)/28.84)*1000))/(3.14*(dia
    /2)^(2)); //[m/s]
37
38 printf(" The velocity of air downstream of
    restriction is %f m/s\n",Vel_2);
39
40 delta_T = (T_2 - T_1);
41 // Since temperature difference (delta_T) is very
    small, therefore enthalpy change is also very

```

```
    small
42
43 // Entropy change is given by,  $\Delta S = C_{p,0} \ln(T_2/T_1) - R \ln(P_2/P_1)$ , but since  $T_1$  and  $T_2$ 
    are almost equal
44  $\Delta S = - R \ln(P_2/P_1)$ ; // [J/mol-K]
45
46 printf(" The change in entropy is %f kJ/mol-k",
     $\Delta S$ );
```

Chapter 5

Exergy

Scilab code Exa 5.1 Determination of fraction of the availability loss

```
1 clear;
2 clc;
3
4 //Example - 5.1
5 //Page number - 184
6 printf("Example - 5.1 and Page number - 184\n\n");
7
8
9 //Given
10 T_1 = 500+273.15;//[C] - Condensation temperature
11 T_2 = 250+273.15;//[C] - Temperature at which
    vaporization takes place.
12
13 T_3 = 25+273.15;//[C] - Ambient atmospheric
    temperature.
14
15 Q = 1; //We are taking a fictitious value of Q, its
    value is not given. But we need to initialize it
    with some value, so we are taking its value as Q=1.
16
17 //The exergy content of the vapour at 500 C,
```

```

18 Ex_T_1 = Q*(1-(T_3/T_1));
19 Ex_T_2 = Q*(1-(T_3/T_2));
20 //Therefore, loss in exergy is given by
21 Ex_loss = Ex_T_1 - Ex_T_2;
22 //Fraction of exergy lost due to irreversible
    process is ,
23 Ex_fraction =(Ex_loss/Ex_T_1);
24 printf(" The fraction of exergy lost due to
    irreversible process is %f",Ex_fraction);

```

Scilab code Exa 5.2 Determination of availability change and irreversibility

```

1 clear;
2 clc;
3
4 //Example - 5.2
5 //Page number - 188
6 printf("Example - 5.2 and Page number - 188\n\n");
7
8 //Given
9 T_1 = 300; //[K] - Initial temperature.
10 P_1 = 100; //[kPa] - Initial pressure.
11 T_2 = 500; //[K] - Final temperature.
12 T_0 = 300; //[K] - Environment temperature.
13 P_0 = 1; //[atm] - Environment pressure.
14 R = 8.314; //[J/mol*K]
15 //(Cp_0/R)= 3.626
16 Cp_0 = 3.626*R; //[J/mol-K] - Heat capacity at
    constant pressure
17
18
19 //(1).
20 //The availability change is given by, (phi_1 -
    phi_2) = U_1 - U_2 + P_0*(V_1 - V_2) - T_0*(S_1 -

```

```

    S_2)
21 //Let us determine the change in internal energy
22 //For ideal gas the molar internal energy change is
    given by  $\Delta U = C_{v,0}(T_2 - T_1)$ 
23 //For ideal gas  $C_{p,0} - C_{v,0} = R$ , and therefore
24  $C_{v,0} = ((C_{p,0}/R) - 1)*R$ ; // [J/mol-K] - Heat capacity
    at constant volume
25  $\Delta U = C_{v,0}(T_2 - T_1)$ ; // [J/mol]
26 // $\Delta U = -w$  (from energy balance). Therefore,  $U_1 -$ 
     $U_2 = -\Delta U$ .
27 //The entropy change of ideal gas is given by
28 // $\Delta S = C_{p,0} \log(T_2/T_1) - R \log(P_2/P_1)$ , but
    ,  $(P_1 V_1 / T_1) = (P_2 V_2 / T_2)$  and therefore  $(P_2/P_1) =$ 
     $(T_2/T_1)$ 
29  $\Delta S = C_{p,0} \log(T_2/T_1) - R \log(T_2/T_1)$ ; // [J/
    mol-K]
30 //The exergy change is given by,  $(\phi_1 - \phi_2) =$ 
     $U_1 - U_2 + P_0(V_1 - V_2) - T_0(S_1 - S_2)$ 
31 //  $(V_1 - V_2) = 0$ , because the tank is rigid and so
    the volume is constant
32  $\Delta \phi = (-\Delta U) - T_0(-\Delta S)$ ; // [J/mol]
33 printf(" (1).The change in exergy is %f J/mol\n\n",
     $\Delta \phi$ );
34
35 //(2).
36 //Entropy change of the system is given by,
     $\Delta S_{sys} = q/T_b + S_{gen}$ 
37 //Since the system is adiabatic therefore ,
     $\Delta S_{sys} = S_{gen}$ 
38  $S_{gen} = \Delta S$ ;
39 //Irreversibility is given by
40  $i = T_0 S_{gen}$ ; // [J/mol]
41 printf(" (2).The value of irreversibility is %f J/
    mol", i);
42 //Irreversibility can also be determined using
43 //  $i = (W_{rev,use} - W_{use})$ 

```

Scilab code Exa 5.3 Determination of availability change and irreversibility

```
1 clear;
2 clc;
3
4 //Example - 5.3
5 //Page number - 190
6 printf("Example - 5.3 and Page number - 190\n\n")
7
8 //Given
9 P_1 = 15;//[bar] - Initial pressure
10 P_1 = P_1*10^(5);//[Pa]
11 T_1 = 300+273.15;//[K] - Initial temperature
12 T_0 = 298.15;//[K]
13 T_R = 1200;//[K] - Reservoir temperature.
14 P_0 = 1;//[bar]
15 P_0 = P_0*10^(5);//[Pa]
16 n = 1;//[mol] - No of moles
17 R = 8.314;//[J/mol*K]
18 Y = 1.4;// - Ratio of heat capacities.
19 Cv_0 = R/(Y-1);//[J/mol-K] - Heat capacity at
    constant volume
20 Cp_0 = Cv_0 + R;//[J/mol-K] - Heat capacity at
    constant pressure
21
22 //(1)
23 //V_2 = 2*V_1 and since pressure is constant,we get
    (V_1/T_1) = (2*V_1/T_2), or, T_2 = 2*T_1.
24 T_2 = 2*T_1;//[K]
25 W = P_1*((R*T_2)/P_1)-((R*T_1)/P_1);//[J/mol] -
    Actual work done
26 delta_U = Cv_0*(T_2-T_1);//[J/mol] - Change in
    internal energy.
```

```

27 q = W + delta_U;//[J/mol] - Heat change
28 //Now the availability change is given by, (phi_1 -
    phi_2) = U_1 - U_2 + P_0*(V_1 - V_2) - T_0*(S_1 -
    S_2) + q*(1-(T_0/T_R))
29 //delta_S = Cp_0*log(T_2/T_1) - R*log(P_2/P_1), and
    P_1 = P_2, Therefore
30 delta_S = Cp_0*log(T_2/T_1);;//[J/mol-K] - Entropy
    change.
31 //Substituting expressions for delta_phi calculation
    . Decrease in availability is given by,
32 delta_phi = (-delta_U) + P_0*(((R*T_1)/P_1)-((R*T_2)
    /P_1)) - T_0*(-delta_S) + q*(1-(T_0/T_R));//[J/
    mol]
33 //Actual work done is given by, W = P_1*(V_2-V_1)
34 //Work done to displace the atmosphere is given by,
    W = P_0*(V_2-V_1)
35 //Therefore, W_use = (P_1*(V_2-V_1) - P_0*(V_2-V_1))
36 W_use = (P_1-P_0)*(((R*T_2)/P_1)-((R*T_1)/P_1));//[J
    /mol] - useful work done
37 W_rev_use = delta_phi;// reversible useful work done
38 //Irreversibility is given by,
39 i = W_rev_use - W_use;//[J/mol]
40 printf(" (a).The irreversibility value is %f J/mol\n\
    n",i);
41
42 //The irreversibility can also be calculated using
43 // i = T_0*S_gen
44 //S_gen = delta_S - (q/T_R)
45
46 //(b)
47 //V_2 = 2*V_1 and therefore T_2 = 2*T_1, as P_2 = P_1
48 //Actual work done is same as before
49 //Let work done on stirrer be W_stir. Thus net work
    done by the system is W - W_stir.Fron energy
    balance we get,
50 W_stir = W + delta_U;
51 //Initially the exergy is due to that of the system
    at state 1 and stirrer work,'W_stir' and finally

```

```

    we have the exergy due to system at state 2, the
    stirrer work is spent, thus availability is given
    by
52 delta_phi_b = (-delta_U) + P_0*((R*T_1)/P_1)-((R*
    T_2)/P_1)) - T_0*(-delta_S) + W_stir; //[J/mol]
53 W_rev_use_b = delta_phi_b; // reversible useful work
    done
54 W_use_b = W_use; // useful work done
55 //Now the irreversibility is given by,
56 i_b = W_rev_use_b - W_use_b; //[J/mol]
57 printf(" (b).The irreversibility value is %f J/mol\n\
    n", i_b);
58
59 //The irreversibility can also be calculated using
60 // i_b = T_0*S_gen
61 //S_gen = delta_S - (q/T_R) and here, q = 0
62
63 //(c)
64 P_2_c = 10; //[bar] - Final pressure, (Given)
65 P_2_c = P_2_c*10^(5); //[Pa]
66 ((P_1^(1-Y))*(T_1^Y)) = (P_2^(1-Y))*(T_2^Y)
67 T_2_c = T_1*((P_1/P_2_c)^((1-Y)/Y)); //[K]
68 //Work done is given by, W = -delta_U = -Cv_0*(T_2_c
    - T_1)
69 W_c = -Cv_0*(T_2_c - T_1); //[J/mol]
70 //The final molar volume is calculated using P_1*V_1
    ^Y = P_2*V_2^Y
71 //V_2 = V_1*((P_1/P_2_c)^(1/Y))
72 V_1 = (R*T_1)/P_1; //[cubic metre/mol] - Initial
    molar volume
73 V_2 = V_1*((P_1/P_2_c)^(1/Y)); //[cubic metre/mol] -
    Final molar volume
74 //Now let us determine the work done to displace the
    atmosphere,
75 W_atm_c = P_0*(V_2 - V_1); //[J/mol] - work done to
    displace the atmosphere
76 //Thus useful work is given by,
77 W_use_c = W - W_atm_c; //[J/mol] - useful work done

```

```

78 //Here delta_S = 0,for reversible adiabatic process.
    Therefore ,
79 W_rev_use_c = W_use_c;
80 //Now finally the irreversibility is given by,
81 i_c = W_rev_use_c - W_use_c;//[J/mol]
82 printf(" (c).The ireversibility value is %f J/mol\n\
    n",i_c);
83
84 //(d)
85 //Here temperature is constant ,but V_2 = 2*V_1,
    therefore P_2 = P_1/2
86 V_2_d = 2*V_1;
87 P_2_d = P_1/2;
88 //Under isothermal conditions work done is
89 W_d = R*T_1*log(V_2_d/V_1);//[J/mol]
90 //Work done to displace the atmosphere is given by,
91 W_atm_d = P_0*(V_2_d - V_1);//[J/mol] - work done to
    displace the atmosphere
92 //Thus useful work is given by,
93 W_use_d = W_d - W_atm_d;//[J/mol] - useful work done
94 delta_U_d = 0;//isothermal conditions
95 q_d = W_d;// since , delta_U_d = 0
96 //delta_S_d = Cp_0*log(T_2/T_1) - R*log(P_2/P_1),
    and T_1 = T_2, Therefore
97 delta_S_d = -R*log(P_2_d/P_1);//[J/mol-K] - Entropy
    change
98 //The reversible useful work is given by,
99 W_rev_use_d = P_0*(V_1 - V_2_d) - T_0*(-delta_S_d) +
    q_d*(1-(T_0/T_R));//[J/mol] - Reversible useful
    work done.
100 //The irreversibility is given by,
101 i_d = W_rev_use_d - W_use_d;//[J/mol]
102 printf(" (d).The ireversibility value is %f J/mol\n\
    n",i_d);
103
104 //(e)
105 P_2_e = 10;//[bar] - Final pressure , (Given)
106 P_2_e = P_2_e*10^(5);//[Pa]

```

```

107 //During the expansion of an ideal gas in into
    vacuum the temperature of the gas remains the
    same,
108 T_2_e = T_1; // Final temperature
109 //Since boundary of the system is fixed so no net
    work is done, W = 0 and thus
110 W_use_e = 0; // [J/mol] - Useful work done
111 //Here, delta_U = 0, as temperature is same and
112 //(V_1-V_2) = 0, as for overall system there is no
    change in volume
113 delta_S_e = - R*log(P_2_e/P_1); // [J/mol-K] - Entropy
    change
114 //The reversible useful work is given by,
115 W_rev_use_e = - T_0*(-delta_S_e); // [J/mol] -
    Reversible useful work done.
116 //The irreversibility is given by,
117 i_e = W_rev_use_e - W_use_e; // [J/mol]
118 printf(" (e).The irreversibility value is %f J/mol\n\n\
    n", i_e);

```

Scilab code Exa 5.4 Determination of useful work and irreversibility

```

1 clear;
2 clc;
3
4 //Example - 5.4
5 //Page number - 194
6 printf("Example - 5.4 and Page number - 194\n\n")
7
8
9 //Given
10 T_1 = 150+273.15; // [K] - Initial temperature.
11 m = 4.6; // [kg] - mass of water
12 P_1 = 1; // [MPa] - Initial pressure
13 Q = 11000; // [kJ] - Heat transferred to the system.

```



```

14 T_R = 600+273.15;//[K] - Temperature of the
    reservior.
15 T_0 = 298;//[K] - Temperature of the environment
16 P_0 = 100;//[kPa] - Pressure of the environment
17
18 //(1)
19 //The entropy change of an isothermal system
    undergoing an internally reversible process is
    given by,
20 delta_S_t = (Q/T_1);//[kJ] - Entropy change
21 delta_S = delta_S_t/m;//[kJ/kg-K] -
22
23 //At 150 C, it has been reported in the book that,
    P_sat = 0.4758 kPa, V_liq = 0.001091 m^(3)/kg,
    U_liq = 631.68 kJ/kg, S_liq = 1.8418 kJ/kg-K,
    S_vap = 6.8379 kJ/kg-K
24 V_1 = 0.001091;//[m^(3)/kg] - initial specific
    volume
25 U_1 = 631.68;//[kJ/kg] - initial specific internal
    energy
26 S_1 = 1.8418;//[kJ/kg-K] - initial entropy
27 //The initial state of the water is a compressed
    liquid state, and S_1 is therefore equal to the
    entropy of the saturated liquid of the saturated
    liquid at the same temperature.
28 S_2 = S_1 + delta_S;//[kJ/kg-K] - Final entropy
29
30 //At the final state the temperature is 150 C and S
    = 7.499 kJ/kg-K which is more than S_vap
    therefore it is superheated steam.
31 S_final = 7.494;//[kJ/kg-K]
32 //At 150 C, and 0.1 MPa: V = 1.9364 m^(3)/kg, U =
    2582.8 kJ/kg, S = 7.6134 kJ/kg-K
33 //At 150 C, and 0.2 MPa: V = 0.9596 m^(3)/kg, U =
    2576.9 kJ/kg, S = 7.2795 kJ/kg-K
34 U_t_1 = 2582.8;//[kJ/kg] - Internal energy
35 U_t_2 = 2576.9;//[kJ/kg]
36 V_t_1 = 1.9364;//[m^(3)/kg] - Specific volume

```

```

37 V_t_2 = 0.9596; // [m^(3)/kg]
38 S_t_1 = 7.6134; // [kJ/kg-K] - Entropy
39 S_t_2 = 7.2795; // [kJ/kg-K]
40 //The pressure at exit is given by,
41 P_2 = ((S_final - S_t_1)/(S_t_2 - S_t_1))*(0.2 -
        0.1) + 0.1; // [Mpa] - Final pressure
42 //At final state
43 U_2 = U_t_1 + (U_t_2 - U_t_1)*((S_final - S_t_1)/(
        S_t_2 - S_t_1)); // [kJ/kg] - Final specific
        internal energy
44 V_2 = V_t_1 + (V_t_2 - V_t_1)*((S_final - S_t_1)/(
        S_t_2 - S_t_1)); // [m^(3)/kg] - Final specific
        volume
45
46 q = Q/m; // [kJ/kg] - Heat supplied per unit kg of
        mass.
47 W_rev_use = U_1 - U_2 + P_0*(V_1 - V_2) - T_0*(S_1 -
        S_2) + q*(1 - (T_0/T_R)); // [kJ/kg] - Reversible
        useful work done.
48
49 //Now let us calculate the actual work done. We know
        q - W = delta_U, therefore
50 W = q - (U_2 - U_1); // [kJ/kg] - Work done
51 W_use = W - P_0*(V_2 - V_1); // [kJ/kg]
52 i = W_rev_use - W_use; // [kJ/kg] - Irreversibility
53 //Since the system contains 4.6 g therefore,
54 W_use_new = W_use*m; // [kJ]
55 W_rev_use_new = W_rev_use*m; // [kJ]
56 I = W_rev_use_new - W_use_new; // [kJ]
57
58 printf(" (1).The useful work obtained is %f kJ\n\n",
        W_use_new);
59 printf(" (2).The reversible usefuk work done is %f
        kJ\n\n",W_rev_use_new);
60 printf(" (3).The irreversibility is %f kJ\n\n",I);

```

Scilab code Exa 5.5 Determination of reversible work and irreversibility

```
1 clear;
2 clc;
3
4 //Example - 5.5
5 //Page number - 197
6 printf("Example - 5.5 and Page number - 197\n\n")
7
8 //Given
9 T_1 = 700+273.15; //[K] - Initial temperature.
10 P_1 = 12; //[MPa] - Initial pressure
11 P_2 = 0.6; //[MPa] - Final pressure
12 //At 12 MPa and 700 C,
13 H_1 = 3858.4; //[kJ/kg] - initial enthalpy
14 S_1 = 7.0757; //[kJ/kg-K] - initial entropy
15
16 //At 0.6 MPa and 200 C,
17 H_2 = 2850.1; //[kJ/kg]
18 S_2 = 6.9673; //[kJ/kg-K]
19
20 //At 0.6 MPa and 250 C,
21 H_3 = 2957.2; //[kJ/kg]
22 S_3 = 7.1824; //[kJ/kg-K]
23
24 //At 0.6 MPa and 300 C,
25 H_4 = 3061.6; //[kJ/kg]
26 S_4 = 7.3732; //[kJ/kg-K]
27
28 //(1)
29 //In the case of ideal turbine the entropy change
    does not take place, therefore the exit conditions
    are
30 P_exit = P_2; //[MPa] - exit pressure
```

```

31 T_exit = ((S_1 - S_2)/(S_3 - S_2))*(250 - 200) +
    200; //[C] - exit temperature
32 H_exit = ((S_1 - S_2)/(S_3 - S_2))*(H_3 - H_2) + H_2
    ;//[kJ/kg] - exit enthalpy
33
34 //Snce it is a flow pocess ,therfore
35 //W_rev = H_1 - H_exit - T_0*(S_1 - S_2)
36 //As S_1 = S_2 ,the above equation becomes
37 W_rev_1 = H_1 - H_exit; //[kJ/kg] - reversible work
    done
38
39 //From the first law the actual work done can be
    calculated using , delta_H = q - W
40 //Since the turbine does not exchange heat ,therefore
    W = - delta_H.
41 W_1 = - (H_exit - H_1); //[kJ/kg]
42
43 printf(" (1).The reversible work done is %f kJ/kg\n"
    ,W_1);
44 printf("      And since the maximum work is same as
    the actual work ,therefore irreversibility is zero
    \n\n");
45
46 //(2)
47 //Given
48 T_0 = 298.15; //[K] - Environment temperature
49 P_0 = 1; //[atm] - Environment pressure
50 adi_eff = 0.88; //adiabatc efficiency
51
52 //(H_1 - H_exit_actual)/(H_1 - H_exit) = 0.88,
    therefore
53 H_exit_actual = H_1 - 0.88*(H_1 - H_exit); // -
    Actual exit enthalpy
54
55 //Now two properties i.e pressure = 0.6 MPa and
    enthalpy = H_exit_actual is fixed at the exit.
    The exit temperature is given by,
56 T_exit_actual = ((H_exit_actual - H_3)/(H_4 - H_3))

```

```

    *(300 - 250) + 250; //[C]
57 S_exit_actual = ((H_exit_actual - H_3)/(H_4 - H_3))
    *(S_4 - S_3) + S_3; //[kJ/kg]
58
59 //Now reversible work done is given by,
60 W_rev_2 = H_1 - H_exit_actual - T_0*(S_1 -
    S_exit_actual); //[kJ/kg]
61 printf(" (2).The reversible work done is %f kJ/kg\n"
    ,W_rev_2);
62
63 //The actual work is given by the first law,
64 W_2 = H_1 - H_exit_actual; //[kJ/kg] - Actual work
    done
65 i = W_rev_2 - W_2; //[kJ/kg] - irreversibility
66 printf("    The value of irreversibility is %f kJ/
    kg\n",i);
67
68 //The irreversibility can also be determined using
69 // i = T_0*S_gen, and S_gen is given by
70 // S_gen = (q/T_R) - delta_S
71
72 //The second law efficiency of the turbine is actual
    work done divided by reversible work, therefore
73 sec_eff = W_2/W_rev_2;
74 printf("    The second law efficiency of the
    turbine is %f\n",sec_eff);

```

Scilab code Exa 5.6 Determination of maximum obtainable work and efficiency

```

1 clear;
2 clc;
3
4 //Example - 5.6
5 //Page number - 198

```

```

6 printf("Example - 5.6 and Page number - 198\n\n")
7
8 //Given
9 P_1 = 8;//[bar] - Initial pressure
10 T_1 = 93 + 273.15;//[C] - Initial temperature
11 V_1 = 100;//[m/s] - Initial velocity
12 P_2 = 1.25;//[bar] - Exit pressure
13 T_2 = 27 + 273.15;//[C] - Exit temperature
14 V_2 = 60;//[m/s] - Exit velocity
15 Y = 1.4; //Ratio of specific heat capacities
16 T_0 = 298.15;//[K] - surrounding temperature
17 P_0 = 1;//[bar] - surrounding pressure
18 R = 8.314;//[J/mol*K] - Gas constant
19 Cp_0 = (R*Y)/(Y-1);//[J/mol-K] - Heat capacity at
    constant pressure
20
21 //Since the amount of heat transfer is negligible ,
    therefore from first law the actual work done is
    given by,
22 //W = delta_H + (delta_V_square)/2
23 delta_H = Cp_0*(T_2 - T_1);//[J/mol] - enthalpy
    change
24 delta_H = (delta_H/28.84);//[kJ/kg] - (1 mole =
    28.84 g).
25 delta_V_square = V_2^(2) - V_1^(2);
26
27 W = - delta_H - ((delta_V_square)/2)/1000;//[kJ/kg]
    - Actual work done
28 printf(" The actual work done is %f kJ/kg\n\n",W);
29
30 //Now let us calculate the maximum work that can be
    obtained
31 //W_rev = (H_1 + (V_1^(2))/2) - (H_2 + (V_2^(2))/2)
    - T_0*(S_1 - S_2)
32 delta_S = Cp_0*log(T_2/T_1) - R*log(P_2/P_1);//[J/
    mol-K] - Entropy change
33 delta_S = delta_S/28.84;//[kJ/kg-K]
34 W_rev = -delta_H - ((delta_V_square/2)/1000) + T_0*

```

```

    delta_S; //[kJ/kg]
35 printf(" The maximum work obtainable per kg of air
    is %f kJ/kg\n\n",W_rev);
36
37 //The second law efficiency of the turbine is actual
    work done divided by reversible work,therefore
38 sec_eff = W/W_rev;
39 printf(" The second law efficiency of the turbine is
    %f\n\n",sec_eff);

```

Scilab code Exa 5.7 Determination of entropy generation rate and irreversibility

```

1 clear;
2 clc;
3
4 //Example - 5.7
5 //Page number - 200
6 printf("Example - 5.7 and Page number - 200\n\n")
7
8 //Given
9 m_cold_water = 60; //[kg/s] - mass flow rate of cold
    water
10 P_1 = 50; //[kPa]
11 T_2 = 250; //[C]
12 T_water_1 = 1000 + 273.15; //[K] - Entering
    temperature of water
13 T_water_2 = 450 +273.15; //[K] - Exit temperature of
    water
14 T_0 = 298.15; //[K] - surrounding temperature
15 P_0 = 1; //[atm] - surrounding pressure
16 Cp_0 = 1.005; //[kJ/kg-K]
17
18 //For water at 50 kPa under saturated conditions ,
    T_sat = 81.33 C,

```

```

19 H_liq_1 = 340.49;//[kJ/kg] - Enthalpy
20 S_liq_1 = 1.0910;//[kJ/kg-K] - Entropy
21
22 //For steam at 50 kPa and 250 C,
23 H_2 = 2976.0;//[kJ/kg]
24 S_2 = 8.3556;//[kJ/kg-K]
25
26 //The cold stream is water which enters as saturated
    liquid at 50 kPa and exits as superheated vapour
    at 50 kPa and 250 C, since pressure drop is
    neglected.
27 //The mass flow rate of hot stream can be obtained
    from energy balance
28 m_hot_water = (m_cold_water*(H_2 - H_liq_1))/(Cp_0*(
    T_water_1 - T_water_2));//[kg/s] - mass flow rate
    of hot water
29
30 //Since there is no heat exchange with the
    surrounding therefore the total entropy
    generation is given by
31 //S_gen = delta_S_hot + delta_S_cold
32 delta_S_cold = S_2 - S_liq_1;//[kJ/kg-K] - change of
    entropy of cold water
33 //delta_S_hot = Cp_0*log(T_2/T_1)-R*log(P_2/P_1),
    But pressure drop is zero, therefore
34 delta_S_hot = Cp_0*log(T_water_2/T_water_1);//[kJ/kg
    -K] - change of entropy of hot water
35
36 S_gen = m_cold_water*delta_S_cold + m_hot_water*
    delta_S_hot;//[kW/K] - Entropy generated
37 printf(" The entropy generation rate is %f kW/K\n\n"
    ,S_gen);
38
39 //The irreversibility rete is given by
40 I = T_0*S_gen;//[kW]
41 printf(" The irreversibility rate of the heat
    exchanger is %f kW\n",I);
42

```



```

43 //The irreversibility can also be determined using
    the exergy approach
44 //We know that,  $I = W_{rev} -$ , but since actual work
    done zero in heat exchangers, therefore  $I = W_{rev}$ 
    = exergy change
45 // $(si_1 - si_2)_{cold} = H_1 - H_2 - T_0*(S_1 - S_2)$ 
46 // $(si_1 - si_2)_{hot} = Cp_0*(T_1 - T_2) - T_0*(S_1 -$ 
     $S_2)$ 
47 //  $I = (si_1 - si_2)_{cold} - (si_1 - si_2)_{hot}$ .

```

Scilab code Exa 5.8 Calculation of exit temperature entropy and irreversibility rate

```

1  clear;
2  clc;
3
4  //Example - 5.8
5  //Page number - 201
6  printf("Example - 5.8 and Page number - 201\n\n")
7
8  //Given
9  m_water = 10000; //[kg/h] - Mass flow rate of cold
    water
10 m_water = m_water/3600; //[kg/s]
11 T_1_water = 30 + 273.15; //[K] - Cold water entering
    temperature
12 m_HC = 5000; //[kg/h] - mass flow rate of hot
    hydrocarbon
13 m_HC = m_HC/3600; //[kg/s]
14 T_1_HC = 200 + 273.15; //[K] - Hot hydrocarbon
    entering temperature
15 T_2_HC = 100 + 273.15; //[K] - Hot hydrocarbon
    leaving temperature
16 Cp_0_water = 1.0; //[kcal/kg-K] - Mean heat capacity
    of cooling water

```

```

17 Cp_0_HC = 0.6; // [kcal/kg-K] - Mean heat capacity of
    hydrocarbon
18
19 //(1)
20 //Applying energy balance to the heat exchanger, we
    get
21 //m_water*Cp_0_water*(T - T_1_water) = m_HC*Cp_0_HC
    *(T_1_HC - T_2_HC)
22 T_2_water = ((m_HC*Cp_0_HC*(T_1_HC - T_2_HC))/(
    m_water*Cp_0_water)) + T_1_water; // [K]
23 T_2 = T_2_water - 273.15; // [C]
24 printf(" (1).The exit temperature of the cooling
    water is %f C\n\n", T_2);
25
26 //(2)
27 //delta_S_hot_HC = Cp_0*log(T_2/T_1)-R*log(P_2/P_1),
    But pressure drop is zero, therefore
28 delta_S_hot_HC = (Cp_0_HC*4.184)*log(T_2_HC/T_1_HC);
    // [kW/K] - change of entropy of hot hydrocarbon
29 delta_S_HC = m_HC*delta_S_hot_HC; // [kW/K] - Entropy
    change for hudrocarbon liquid
30 printf(" (2).Entropy change rate of hydrocarbon
    liquid is %f kW/K\n", delta_S_HC);
31
32 delta_S_cold_water = (Cp_0_water*4.184)*log(
    T_2_water/T_1_water); // [kW/K] - change of entropy
    of cooling water
33 delta_S_water = m_water*delta_S_cold_water; // [kW/K]
    - Entropy change for water
34 printf(" And entropy change rate of water is %f
    kW/K\n\n", delta_S_water);
35
36 //(3)
37 T_0 = 298.15; // [K] - Surrounding temperature
38 //S_gen = delta_S_cold_water + delta_S_hot_HC =
    m_water*delta_S_cold_water + m_HC*delta_S_hot_HC
    ; // [kW/K] - Entropy generated
39 S_gen = delta_S_water + delta_S_HC; // [kW/K]

```

```

40 I = T_0*S_gen; // [kW]
41 printf(" (3).The irreversibility rate of the heat
    exchanger is %f kW\n",I);

```

Scilab code Exa 5.9 Determinatio of exit temperature availability change and irreversibility

```

1 clear;
2 clc;
3
4 //Example - 5.9
5 //Page number - 202
6 printf("Example - 5.9 and Page number - 202\n\n");
7
8 //Given
9 T_1_hotgas = 800; // [K]
10 P_1_hotgas = 1; // [bar]
11 T_2_hotgas = 700; // [K]
12 P_2_hotgas = 1; // [bar]
13 T_1_air = 470; // [K]
14 P_1_air = 1; // [bar]
15 P_2_air = 1; // [bar]
16 Cp_0_hotgas = 1.08; // [kJ/kg-K] - Mean heat capacity
    of hot gas
17 Cp_0_air = 1.05; // [kcal/kg-K] - Mean heat capacity
    of air
18 T_0 = 298.15; // [K] - surrounding temperature
19 P_0 = 1; // [bar] - surrounding pressure
20 //m_air = 2*m_hotgas
21
22 //(1)
23 //Assuming heat exchange only takes places in-
    between the streams,from energy balance we get ,
24 //m_gas*Cp_0_hotgas*(T_2_hotgas - T_1_hotgas) + 2*
    m_gas*Cp_0_air*(T - T_1.air) ,

```

```

25 T_2_air = T_1_air - ((Cp_0_hotgas*(T_2_hotgas -
    T_1_hotgas))/(2*Cp_0_air));//[K] - Temp of
    emerging air
26 printf(" (1).The temperature of emerging air is %f K
    \n\n",T_2_air);
27
28 //(2)
29 //Availability change of hot gas is given by,
30 //(si_1 - si_2)_hot = H_1 - H_2 - T_0*(S_1 - S_2)
31 delta_H_hotgas = (Cp_0_hotgas*(T_2_hotgas -
    T_1_hotgas));//[kJ/kg] - change in enthalpy of
    hotgas
32 //delta_S_hotgas = Cp_0_hotgas*log(T_2_hotgas/
    T_1_hotgas)- R*log(P_2/P_1), But pressure drop is
    zero (P_1 = P_2), therefore
33 delta_S_hotgas = Cp_0_hotgas*log(T_2_hotgas/
    T_1_hotgas);//[kJ/kg-K] - change of entropy of
    hot gas
34 delta_si_hotgas = (-delta_H_hotgas) - (-T_0*
    delta_S_hotgas);//[kJ/kg]
35 printf(" (2).The availability change of hot gas is
    %f kJ/kg\n\n",delta_si_hotgas);
36
37 //(3)
38 //Availability change of air is given by,
39 //(si_1 - si_2)_air = H_1 - H_2 - T_0*(S_1 - S_2)
40 delta_H_air = (Cp_0_air*(T_2_air - T_1_air));//[kJ/
    kg] - change in enthalpy of air
41 //delta_S_air = Cp_0_air*log(T_2_air/T_1_air)- R*log
    (P_2/P_1), But pressure drop is zero (P_1 = P_2),
    therefore
42 delta_S_air = Cp_0_air*log(T_2_air/T_1_air);//[kJ/kg
    -K] - change of entropy of air
43 delta_si_air = (-delta_H_air) - (-T_0*delta_S_air);
    //[kJ/kg]
44 printf(" (3).The availability change of air is %f kJ
    /kg\n\n",delta_si_air);
45

```

```

46 //(4)
47 //For the heat exchanger (Q = 0, W = 0)
48 //Basis : 1 kg of hot gas flowing through heat
      exchanger
49 S_gen = delta_S_hotgas + 2*delta_S_air;//[kJ/kg-K] -
      (as m_air = 2*m_hotgas)
50 I = T_0*S_gen;//[kJ/kg]
51 printf(" (4).The irreversibility of thr exchanger
      per kg of hot gas flowing is %f kJ/kg\n",I);
52
53 //Irreversibility can also be obtained using
54 //I = 2*(si_1 - si_2)_air + (si_1 - si_2)_hotgas

```

Chapter 6

Chemical reactions

Scilab code Exa 6.1 Determination of enthalpy entropy and Gibbs free energy change of reaction

```
1 clear;
2 clc;
3
4 //Example - 6.1
5 //Page number - 217
6 printf("Example - 6.1 and Page number - 217\n\n");
7
8
9 //Given
10 T_1 = 298.15; // [K] - Standard temperature
11 T_2 = 880; // [K] - Reaction temperature
12
13 a_S02 = 6.157;
14 a_S03 = 3.918;
15 a_O2 = 6.732;
16 b_S02 = 1.384*10^(-2);
17 b_S03 = 3.483*10^(-2);
18 b_O2 = 0.1505*10^(-2);
19 c_S02 = -0.9103*10^(-5);
20 c_S03 = -2.675*10^(-5);
```

```

21 c_O2 = -0.01791*10^(-5);
22 d_SO2 = 2.057*10^(-9);
23 d_SO3 = 7.744*10^(-9);
24
25 delta_H_rkn_298 = -23.45*10^(3);//[cal] - Rkn
    enthalpy at 298.15 K
26 delta_H_SO2_for_298 = -70.94*10^(3);//[cal/mol] -
    Enthalpy of formation of SO2 at 298.15 K
27 delta_H_SO3_for_298 = -94.39*10^(3);//[cal/mol] -
    Enthalpy of formation of SO3 at 298.15 K
28 delta_G_SO2_for_298 = -71.68*10^(3);//[cal/mol] -
    Gibbs free energy change for formation of SO2 at
    298.15 K
29 delta_G_SO3_for_298 = -88.59*10^(3);//[cal/mol] -
    Gibbs free energy change for formation of SO3 at
    298.15 K
30
31 //(1)
32 //Standard enthalpy change of reaction at
    temperature T is given by,
33 //delta_H_rkn_T = delta_rkn_298 + delta_Cp_0*delta_T
34 delta_a = a_SO3 - a_SO2 - (a_O2/2);
35 delta_b = b_SO3 - b_SO2 - (b_O2/2);
36 delta_c = c_SO3 - c_SO2 - (c_O2/2);
37 delta_d = d_SO3 - d_SO2;
38
39 //Cp_0 = delta_a + (delta_b*T) + (delta_c*T^(2)) + (
    delta_d*T^(3));
40 //Therefore we get,
41 delta_H_rkn_880 = delta_H_rkn_298 + integrate('
    delta_a+(delta_b*T)+(delta_c*T^(2))+(delta_d*T
    ^ (3))', 'T', T_1, T_2);
42
43 //On manual simplification of the above expression ,
    we will get the expression for 'delta_H_rkn_880 '
    as a function of T,
44
45 printf(" (1).The expression for standard enthalpy

```

```

    change of reaction as a function of temperature
    is given by\n");
46 printf("      delta_H_rkn_880 = -22534.57 - 5.605*T
    + 1.012*10^(-2)*T^(2) - 0.585*10^(-5)*T^(3) +
    1.422*10^(-9)*T^(4)\n\n")
47
48 printf(" (2).Standard enthalpy change of reaction at
    880 K is %f cal\n\n",delta_H_rkn_880);
49
50 //(3)
51 //Let us determine the standard entropy change of
    reaction at 298.15 K
52 delta_S_S02_298 = (delta_H_S02_for_298 -
    delta_G_S02_for_298)/298.15; //[ cal/mol-K]
53 delta_S_S03_298 = (delta_H_S03_for_298 -
    delta_G_S03_for_298)/298.15; //[ cal/mol-K]
54 delta_S_02_298 = 0; //[ cal/mol-K]
55
56 delta_S_rkn_298 = delta_S_S03_298 - delta_S_S02_298
    - (delta_S_02_298/2); //[ cal/K]
57 delta_S_rkn_880 = delta_S_rkn_298 + integrate('(
    delta_a+delta_b*T+delta_c*T^(2)+delta_d*T^(3))/T'
    , 'T',T_1,T_2); //[ cal/K]
58
59 printf(" (3).Standard entropy change of reaction at
    880 K is %f cal/K\n\n",delta_S_rkn_880);
60
61 //(4)
62 delta_G_rkn_880 = delta_H_rkn_880 - 880*
    delta_S_rkn_880; //[ cal]
63
64 printf(" (4).Standard Gibbs free energy change of
    reaction at 880 K is %f cal\n\n",delta_G_rkn_880)
    ;

```

Scilab code Exa 6.2 Determination of standard enthalpy and Gibbs free energy change of reaction

```
1 clear;
2 clc;
3
4 //Example - 6.2
5 //Page number - 219
6 printf("Example - 6.2 and Page number - 219\n\n");
7
8 //Given
9 T_1 = 298.15; // [K] - Standard temperature
10 T_2 = 400; // [K] - Reaction temperature
11
12 a_CH3OH = 4.55;
13 a_CO = 6.726;
14 a_H2 = 6.952;
15 b_CH3OH = 2.186*10^(-2);
16 b_CO = 0.04001*10^(-2);
17 b_H2 = -0.04576*10^(-2);
18 c_CH3OH = -0.291*10^(-5);
19 c_CO = 0.1283*10^(-5);
20 c_H2 = 0.09563*10^(-5);
21 d_CH3OH = -1.92*10^(-9);
22 d_CO = -0.5307*10^(-9);
23 d_H2 = -0.2079*10^(-9);
24
25 delta_H_rkn_298 = -21.6643*10^(3); // [cal] - Reaction
    enthalpy at 298.15 K
26 delta_H_CO_for_298 = -26.4157*10^(3); // [cal/mol] -
    Enthalpy of formation of CO at 298.15 K
27 delta_H_CH3OH_for_298 = -48.08*10^(3); // [cal/mol] -
    Enthalpy of formation of CH3OH at 298.15 K
28 delta_G_CO_for_298 = -32.8079*10^(3); // [cal/mol] -
    Gibbs free energy change for formation of CO at
    298.15 K
29 delta_G_CH3OH_for_298 = -38.69*10^(3); // [cal/mol] -
    Gibbs free energy change for formation of CH3OH
```

```

    at 298.15 K
30
31 //Standard enthalpy change of reaction at
    temperature T is given by,
32 //delta_H_rkn_T = delta_rkn_298 + delta_Cp_0*delta_T
33 delta_a = a_CH3OH - a_CO - 2*(a_H2);
34 delta_b = b_CH3OH - b_CO - 2*(b_H2);
35 delta_c = c_CH3OH - c_CO - 2*(c_H2);
36 delta_d = d_CH3OH - d_CO - 2*(d_H2);
37
38 //Cp_0 = delta_a + (delta_b*T) + (delta_c*T^(2)) + (
    delta_d*T^(3));
39 //Therefore we get ,
40 delta_H_rkn_400 = delta_H_rkn_298 + integrate('
    delta_a+(delta_b*T)+(delta_c*T^(2))+(delta_d*T
    ^3)', 'T', T_1, T_2);
41
42 printf(" Standard enthalpy change of reaction at 400
    K is %f cal\n\n", delta_H_rkn_400);
43
44 //Let us determine the standard Gibbs free energy
    change of reaction at 298.15 K
45 delta_G_rkn_298 = delta_G_CH3OH_for_298 -
    delta_G_CO_for_298; //[cal]
46
47 //Now determining the standard entropy change of
    reaction at 298.15 K
48 delta_S_rkn_298 = (delta_H_rkn_298 - delta_G_rkn_298
    )/298.15; //[cal/mol-K]
49
50 delta_S_rkn_400 = delta_S_rkn_298 + integrate('(
    delta_a+delta_b*T+delta_c*T^(2)+delta_d*T^(3))/T'
    , 'T', T_1, T_2); //[cal/K]
51 //Therefore, the standard Gibbs free energy change of
    the reaction is given by,
52 delta_G_rkn_400 = delta_H_rkn_400 - 400*
    delta_S_rkn_400; //[cal]
53

```

```
54 printf(" Standard Gibbs free energy change of
    reaction at 400 K is %f cal\n",delta_G_rkn_400);
```

Scilab code Exa 6.3 Determination of standard enthalpy and Gibbs free energy change of reaction

```
1 clear;
2 clc;
3
4 //Example - 6.3
5 //Page number - 220
6 printf("Example - 6.3 and Page number - 220\n\n");
7
8 //Given
9 T_1 = 298.15; //[K] - Standard temperature
10 T_2 = 1200; //[K] - Reaction temperature
11
12
13 a_CO2 = 5.316;
14 a_H2 = 6.952;
15 a_CO = 6.726;
16 a_H2O = 7.700;
17 b_CO2 = 1.4285*10^(-2);
18 b_H2 = -0.04576*10^(-2);
19 b_CO = 0.04001*10^(-2);
20 b_H2O = 0.04594*10^(-2);
21 c_CO2 = -0.8362*10^(-5);
22 c_H2 = 0.09563*10^(-5);
23 c_CO = 0.1283*10^(-5);
24 c_H2O = 0.2521*10^(-5);
25 d_CO2 = 1.784*10^(-9);
26 d_H2 = -0.2079*10^(-9);
27 d_CO = -0.5307*10^(-9);
28 d_H2O = -0.8587*10^(-9);
29
```

```

30 delta_H_rkn_298 = -9.8382*10^(3);//[cal] - Reaction
    enthalpy at 298.15 K
31 delta_H_CO2_for_298 = -94.0518*10^(3);//[cal/mol-K]
    - Enthalpy of formation of CO2 at 298.15 K
32 delta_H_CO_for_298 = -26.4157*10^(3);//[cal/mol-K] -
    Enthalpy of formation of CO at 298.15 K
33 delta_H_H2O_for_298 = -57.7979*10^(3);//[cal/mol-K]
    - Enthalpy of formation of H2O at 298.15 K
34 delta_G_CO2_for_298 = -94.2598*10^(3);//[cal/mol] -
    Gibbs free energy change for formation of CO at
    298.15 K
35 delta_G_CO_for_298 = -32.8079*10^(3);//[cal/mol] -
    Gibbs free energy change for formation of CH3OH
    at 298.15 K
36 delta_G_H2O_for_298 = -54.6357*10^(3);//[cal/mol] -
    Gibbs free energy change for formation of H2O at
    298.15 K
37
38 //Standard enthalpy change of reaction at
    temperature T is given by,
39 //delta_H_rkn_T = delta_rkn_298 + delta_Cp_0*delta.T
40 delta_a = a_CO2 + a_H2 - a_CO - a_H2O;
41 delta_b = b_CO2 + b_H2 - b_CO - b_H2O;
42 delta_c = c_CO2 + c_H2 - c_CO - c_H2O;
43 delta_d = d_CO2 + d_H2 - d_CO - d_H2O;
44
45 //Cp_0 = delta_a + (delta_b*T) + (delta_c*T^(2)) + (
    delta_d*T^(3));
46 //Therefore we get,
47 delta_H_rkn_1200 = delta_H_rkn_298 + integrate('
    delta_a+(delta_b*T)+(delta_c*T^(2))+(delta_d*T
    ^{3})', 'T', T_1, T_2);
48
49 printf(" Standard enthalpy change of reaction at
    1200 K is %f cal\n\n", delta_H_rkn_1200);
50
51 //Let us determine the standard Gibbs free energy
    change of reaction at 298.15 K

```

```

52 delta_G_rkn_298 = delta_G_CO2_for_298 -
    delta_G_CO_for_298 - delta_G_H2O_for_298;//[cal]
53
54 //Now determining the standard entropy change of
    reaction at 298.15 K
55 delta_S_rkn_298 = (delta_H_rkn_298 - delta_G_rkn_298
    )/298.15;//[cal/mol-K]
56
57 delta_S_rkn_1200 = delta_S_rkn_298 + integrate('(
    delta_a+delta_b*T+delta_c*T^(2)+delta_d*T^(3))/T'
    , 'T',T_1,T_2);//[cal/K]
58 //Therefore, the standard Gibbs free energy change of
    the reaction is given by,
59 delta_G_rkn_1200 = delta_H_rkn_1200 - 1200*
    delta_S_rkn_1200;//[cal]
60
61 printf(" Standard Gibbs free energy change of
    reaction at 1200 K is %f cal",delta_G_rkn_1200);

```

Scilab code Exa 6.4 Determination of standard enthalpy and Gibbs free energy change of reaction

```

1 clear;
2 clc;
3
4 //Example - 6.4
5 //Page number - 221
6 printf("Example - 6.4 and Page number - 221\n\n");
7
8 //Given
9 T_1 = 298.15;//[K] - Standard temperature
10 T_2 = 500;//[K] - Reaction temperature
11
12 a_NH3 = 6.5846;
13 a_N2 = 6.903;

```

```

14 a_H2 = 6.952;
15 b_NH3 = 0.61251*10^(-2);
16 b_N2 = -0.03753*10^(-2);
17 b_H2 = -0.04576*10^(-2);
18 c_NH3 = 0.23663*10^(-5);
19 c_N2 = 0.1930*10^(-5);
20 c_H2 = 0.09563*10^(-5);
21 d_NH3 = -1.5981*10^(-9);
22 d_N2 = -0.6861*10^(-9);
23 d_H2 = -0.2079*10^(-9);
24
25 delta_H_rkn_298 = -22.08*10^(3);//[cal] - Reaction
    enthalpy at 298.15 K
26 delta_H_NH3_for_298 = -11.04*10^(3);//[cal/mol] -
    Enthalpy of formation of NH3 at 298.15 K
27 delta_G_NH3_for_298 = -3.976*10^(3);//[cal/mol] -
    Gibbs free energy change for formation of NH3 at
    298.15 K
28
29 //Standard enthalpy change of reaction at
    temperature T is given by,
30 //delta_H_rkn_T = delta_rkn_298 + delta_Cp_0*delta_T
31 delta_a = 2*a_NH3 - a_N2 - 3*a_H2;
32 delta_b = 2*b_NH3 - b_N2 - 3*b_H2;
33 delta_c = 2*c_NH3 - c_N2 - 3*c_H2;
34 delta_d = 2*d_NH3 - d_N2 - 3*d_H2;
35
36 //Cp_0 = delta_a + (delta_b*T) + (delta_c*T^(2)) + (
    delta_d*T^(3));
37 //Therefore we get,
38 delta_H_rkn_500 = delta_H_rkn_298 + integrate(
    delta_a+(delta_b*T)+(delta_c*T^(2))+(delta_d*T
    ^ (3))', 'T', T_1, T_2);
39
40 printf(" Standard enthalpy change of reaction at 500
    K is %f cal\n\n", delta_H_rkn_500);
41
42 //Let us determine the standard Gibbs free energy

```

```

    change of reaction at 298.15 K
43 delta_G_rkn_298 = 2*delta_G_NH3_for_298;//[cal]
44
45 //Now determining the standard entropy change of
    reaction at 298.15 K
46 delta_S_rkn_298 = (delta_H_rkn_298 - delta_G_rkn_298
    )/298.15;//[cal/mol-K]
47
48 delta_S_rkn_500 = delta_S_rkn_298 + integrate('(
    delta_a+delta_b*T+delta_c*T^(2)+delta_d*T^(3))/T'
    , 'T',T_1,T_2);//[cal/K]
49 //Therefore, the standard Gibbs free energy change of
    the reaction is given by,
50 delta_G_rkn_500 = delta_H_rkn_500 - 500*
    delta_S_rkn_500;//[cal]
51
52 printf(" Standard Gibbs free energy change of
    reaction at 500 K is %f cal",delta_G_rkn_500);

```

Scilab code Exa 6.5 Determination of standard enthalpy and Gibbs free energy change of reaction

```

1 clear;
2 clc;
3
4 //Example - 6.5
5 //Page number - 222
6 printf("Example - 6.5 and Page number - 222\n\n");
7
8 //Given
9 //Cp_0 = 7.7 + 0.04594*10^(-2)*T + 0.2521*10^(-5)*T
    ^(-2) - 0.8587*10^(-9)*T^(3)
10
11 delta_H_rkn_298 = -57.7979*10^(3);//[cal/mol] -
    Reaction enthalpy at 298.15 K

```

```

12 delta_G_rkn_298 = -54.6351*10^(3);//[cal/mol] -
    Gibbs free energy change for formation of H2O at
    298.15 K
13
14 //Standard enthalpy change of reaction at
    temperature T is given by,
15 //delta_H_rkn_T = delta_rkn_298 + delta_Cp_0*delta_T
16 T_1 = 298.15;//[K] - Standard temperature
17 T_2_1 = 873.15;//[K] - Reaction temperature
18 T_2_2 = 1000;//[K] - Reaction temperature
19
20 //Therefore we get ,
21 delta_H_rkn_873 = delta_H_rkn_298 + integrate('
    7.7+0.04594*10^(-2)*T+0.2521*10^(-5)*T^(2)
    -0.8587*10^(-9)*T^(3)', 'T', T_1, T_2_1);;//[cal/mol
    ]
22 delta_H_rkn_1000 = delta_H_rkn_298 + integrate('
    7.7+0.04594*10^(-2)*T+0.2521*10^(-5)*T^(2)
    -0.8587*10^(-9)*T^(3)', 'T', T_1, T_2_2);//[cal/mol]
23
24 printf(" Standard enthalpy change of reaction at 873
    K is %f cal/mol\n\n", delta_H_rkn_873);
25 printf(" Standard enthalpy change of reaction at
    1000 K is %f cal/mol\n\n", delta_H_rkn_1000);
26
27 //Now determining the standard entropy change of
    reaction at 298.15 K
28 delta_S_rkn_298 = (delta_H_rkn_298 - delta_G_rkn_298
    )/298.15;//[cal/mol-K]
29
30 delta_S_rkn_873 = delta_S_rkn_298 + integrate('
    (7.7+0.04594*10^(-2)*T+0.2521*10^(-5)*T^(2)
    -0.8587*10^(-9)*T^(3))/T', 'T', T_1, T_2_1);//[cal/
    mol-K]
31 delta_S_rkn_1000 = delta_S_rkn_298 + integrate('
    (7.7+0.04594*10^(-2)*T+0.2521*10^(-5)*T^(2)
    -0.8587*10^(-9)*T^(3))/T', 'T', T_1, T_2_2);//[cal/
    mol-K]

```



```

32 //Therefore ,the standard Gibbs free energy change of
    the reaction is given by,
33 delta_G_rkn_873 = (delta_H_rkn_873 - 873.15*
    delta_S_rkn_873)*10^(-3);//[kcal/mol]
34 delta_G_rkn_1000 = (delta_H_rkn_1000 - 1000*
    delta_S_rkn_1000)*10^(-3);//[kcal/mol]
35
36 printf(" Standard Gibbs free energy change of
    reaction at 873 K is %f kcal/mol\n\n",
    delta_G_rkn_873);
37 printf(" Standard Gibbs free energy change of
    reaction at 1000 K is %f kcal/mol\n",
    delta_G_rkn_1000);

```

Scilab code Exa 6.6 Calculation of heat exchange

```

1 clear;
2 clc;
3
4 //Example - 6.6
5 //Page number - 223
6 printf("Example - 6.6 and Page number - 223\n\n");
7
8 //Given
9 T_1 = 298.15;//[K] - Standard temperature
10 T_2 = 500;//[K] - Reaction temperature
11
12 a_C2H6 = 1.648;
13 a_O2 = 6.085;
14 a_CO2 = 5.316;
15 a_H2O = 7.700;
16 b_C2H6 = 4.124*10^(-2);
17 b_O2 = 0.3631*10^(-2);
18 b_CO2 = 1.4285*10^(-2);
19 b_H2O = 0.04594*10^(-2);

```

```

20 c_C2H6 = -1.530*10^(-5);
21 c_O2 = -0.1709*10^(-5);
22 c_CO2 = -0.8362*10^(-5);
23 c_H2O = 0.2521*10^(-5);
24 d_C2H6 = 1.740*10^(-9);
25 d_O2 = 0.3133*10^(-9);
26 d_CO2 = 1.784*10^(-9);
27 d_H2O = -0.8587*10^(-9);
28
29 //Since excess is entering and leaving at the same
    temperature, therefore it does not take or give
    any heat to the system.
30 //Therefore the heat exchange is only due to heat of
    reaction at temperature T, or Q = delta_H_rkn_T
31
32 delta_H_C2H6_for_298 = -20.236*10^(3);//[cal/mol] -
    Enthalpy of formation of C2H6 at 298.15 K
33 delta_H_CO2_for_298 = -94.0518*10^(3);//[cal/mol] -
    Enthalpy of formation of CO2 at 298.15 K
34 delta_H_H2O_for_298 = -57.7979*10^(3);//[cal/mol] -
    Enthalpy of formation of H2O at 298.15 K
35
36 delta_H_rkn_298 = 2*delta_H_CO2_for_298 + 3*
    delta_H_H2O_for_298 - delta_H_C2H6_for_298;//[cal
    ] - Reaction enthalpy at 298.15 K
37
38 //Standard enthalpy change of reaction at
    temperature T is given by,
39 //delta_H_rkn_T = delta_rkn_298 + delta_Cp_0*delta_T
40 delta_a = 2*a_CO2 + 3*a_H2O - a_C2H6 - 7/2*(a_O2);
41 delta_b = 2*b_CO2 + 3*b_H2O - b_C2H6 - 7/2*(b_O2);
42 delta_c = 2*c_CO2 + 3*c_H2O - c_C2H6 - 7/2*(c_O2);
43 delta_d = 2*d_CO2 + 3*d_H2O - d_C2H6 - 7/2*(d_O2);
44
45 //Cp_0 = delta_a + (delta_b*T) + (delta_c*T^(2)) + (
    delta_d*T^(3));
46 //Therefore we get,
47 delta_H_rkn_500 = delta_H_rkn_298 + integrate('

```

```

        delta_a+(delta_b*T)+(delta_c*T^(2))+(delta_d*T
        ^ (3))', 'T', T_1, T_2); // [cal]
48 delta_H_rkn_500 = delta_H_rkn_500*10^(-3); // [kcal]
49
50 printf(" The heat exchange of the reaction at 500 K
        is %f kcal", delta_H_rkn_500);

```

Scilab code Exa 6.7 Calculation of change in entropy

```

1 clear;
2 clc;
3
4 //Example - 6.7
5 //Page number - 224
6 printf("Example - 6.7 and Page number - 224\n\n");
7
8 //Given
9 T_1 = 298.15; // [K] - Standard temperature
10 T_2 = 600; // [K] - Reaction temperature
11
12 a_C2H6 = -8.65;
13 a_H2O = 7.700;
14 a_CH4 = 4.750;
15 a_O2 = 6.085;
16 b_C2H6 = 11.578*10^(-2);
17 b_H2O = 0.04594*10^(-2);
18 b_CH4 = 1.200*10^(-2);
19 b_O2 = 0.3631*10^(-2);
20 c_C2H6 = -7.540*10^(-5);
21 c_H2O = 0.2521*10^(-5);
22 c_CH4 = 0.3030*10^(-5);
23 c_O2 = -0.1709*10^(-5);
24 d_C2H6 = 18.54*10^(-9);
25 d_H2O = -0.8587*10^(-9);
26 d_CH4 = -2.630*10^(-9);

```

```

27 d_O2 = 0.3133*10^(-9);
28
29 delta_S_CH4_for_298 = 44.50; //[cal/mol-K] - Entropy
    of formation of CH4 at 298.15 K
30 delta_S_O2_for_298 = 49.00; //[cal/mol-K] - Entropy
    of formation of O2 at 298.15 K
31 delta_S_C2H6_for_298 = 64.34; //[cal/mol-K] - Entropy
    of formation of C2H6 at 298.15 K
32 delta_S_H2O_for_298 = 45.11; //[cal/mol-K] - Entropy
    of formation of C2H6 at 298.15 K
33
34 //Cp_0 = delta_a + (delta_b*T) + (delta_c*T^(2)) + (
    delta_d*T^(3));
35
36 //Standard entropy change of reaction at temperature
    T is given by,
37 //delta_S_rkn_T = delta_rkn_298 + delta_Cp_0*delta_T
38 delta_a = 1/6*(a_C2H6) + 3/2*(a_H2O) - a_CH4 - 3/4*(
    a_O2);
39 delta_b = 1/6*(b_C2H6) + 3/2*(b_H2O) - b_CH4 - 3/4*(
    b_O2);
40 delta_c = 1/6*(c_C2H6) + 3/2*(c_H2O) - c_CH4 - 3/4*(
    c_O2);
41 delta_d = 1/6*(d_C2H6) + 3/2*(d_H2O) - d_CH4 - 3/4*(
    d_O2);
42
43 delta_S_rkn_298 = 1/6*(delta_S_C2H6_for_298) + 3/2*(
    delta_S_H2O_for_298) - delta_S_CH4_for_298 -
    3/4*(delta_S_O2_for_298); //[cal/K]
44 delta_S_rkn_600 = delta_S_rkn_298 + integrate('(
    delta_a+delta_b*T+delta_c*T^(2)+delta_d*T^(3))/T'
    , 'T', T_1, T_2); //[cal/K]
45
46 printf(" Change in entropy of the reaction at 298.15
    K is %f cal/K\n\n", delta_S_rkn_298);
47 printf(" Standard entropy change of reaction at 600
    K is %f cal/K", delta_S_rkn_600);

```

Scilab code Exa 6.8 Calculation of standard enthalpy change and Gibbs free energy change

```
1 clear;
2 clc;
3
4 //Example - 6.8
5 //Page number - 225
6 printf("Example - 6.8 and Page number - 225\n\n");
7
8 //Given
9 T_1 = 298.15;//[K] - Standard temperature
10 T_2 = 973.15;//[K] - Reaction temperature
11
12 //At 298.15 K
13 delta_H_CH4_for_298 = -17.889*10^(3);//[cal/mol] -
    Enthalpy of formation of CH4 at 298.15 K
14 delta_H_C_for_298 = 0.00;//[cal/mol] - Enthalpy of
    formation of C (s, graphite) at 298.15 K
15 delta_H_H2_for_298 = 0.00;//[cal/mol] - Enthalpy of
    formation of H2 at 298.15 K
16 delta_G_CH4_for_298 = -12.140*10^(3);//[cal/mol] -
    Gibbs free energy change for formation of H2 at
    298.15 K
17 delta_G_C_for_298 = 0.00;//[cal/mol] - Gibbs free
    energy change for formation of C (s, graphite) at
    298.15 K
18 delta_G_H2_for_298 = 0.00;//[cal/mol] - Gibbs free
    energy change for formation of H2 at 298.15 K
19
20 ///Standaerd heat capacity data in cal/mol-K are
    given below, T is in K
21 //Cp_0_CH4 = 4.75 + 1.2*10^(-2)*T + 0.303*10^(-5)*T
    ^2 - 2.63*10^(-9)*T^(3)
```

```

22 //Cp_0_C = 3.519 + 1.532*10(-3)*T - 1.723*10(5)*T
    ^(-2)
23 //Cp_0_H2 = 6.952 - 0.04576*10(-2)*T +
    0.09563*10(-5)*T(2) - 0.2079*10(-9)*T(3)
24
25 //Therefore standard heat capacity of reaction is
    given by,
26 //Cp_0_rkn = 2*Cp_0_H2 + Cp_0_C - Cp_0_CH4
27 //On simplification ,we get the relation
28 //Cp_0_rkn = 12.673 - 0.0113832*T - 1.1174*10(-6)*T
    ^(-2) + 2.2142*10(-9)*T(3) - 1.723*10(5)*T(-2)
29
30 delta_H_rkn_298 = -delta_H_CH4_for_298;//[cal] -
    Enthalpy of reaction at 298.15 K
31 delta_G_rkn_298 = -delta_G_CH4_for_298;//[cal] -
    Gibbs free energy of the reaction at 298.15 K
32
33 delta_H_rkn_973 = delta_H_rkn_298 + integrate('
    12.673-0.0113832*T-1.1174*10(-6)*T(2)
    +2.2142*10(-9)*T(3)-1.723*10(5)*T(-2)', 'T',
    T_1,T_2);//[cal]
34
35 printf(" Standard enthalpy change of reaction at
    973.15 K is %f cal\n\n",delta_H_rkn_973);
36
37 //Now determining the standard entropy change of
    reaction at 298.15 K
38 delta_S_rkn_298 = (delta_H_rkn_298 - delta_G_rkn_298
    )/298.15;//[cal/K]
39 delta_S_rkn_973 = delta_S_rkn_298 + integrate('
    (12.673-0.0113832*T-1.1174*10(-6)*T(2)
    +2.2142*10(-9)*T(3)-1.723*10(5)*T(-2))/T', 'T',
    ,T_1,T_2);//[cal/K]
40
41 //Therefore ,the standard Gibbs free energy change of
    the reaction is given by,
42 delta_G_rkn_973 = delta_H_rkn_973 - 973.15*
    delta_S_rkn_973;//[cal]

```

43

```
44 printf(" Standard Gibbs free energy change of  
    reaction at 973 K is %f cal\n",delta_G_rkn_973);
```

Scilab code Exa 6.9 Calculation of standard enthalpy change and Gibbs free energy change

```
1 clear;  
2 clc;  
3  
4 //Example - 6.9  
5 //Page number - 226  
6 printf("Example - 6.9 and Page number - 226\n\n");  
7  
8 //Given  
9 T_1 = 298.15;//[K] - Standard temperature  
10 T_2 = 1000;//[K] - Reaction temperature  
11  
12 //At 298.15 K  
13 delta_H_C_for_298 = 0.00;//[cal/mol] - Enthalpy of  
    formation of C(s, graphite) at 298.15 K  
14 delta_H_H2O_for_298 = -57.7979*10^(3);//[cal/mol] -  
    Enthalpy of formation of H2O at 298.15 K  
15 delta_H_CO_for_298 = -26.4157*10^(3);//[cal/mol] -  
    Enthalpy of formation of CO at 298.15 K  
16 delta_H_H2_for_298 = 0.00;//[cal/mol] - Enthalpy of  
    formation of H2 at 298.15 K  
17 delta_G_C_for_298 = 0.00;//[cal/mol] - Gibbs free  
    energy change for formation of C(s, graphite) at  
    298.15 K  
18 delta_G_H2O_for_298 = -54.6357*10^(3);//[cal/mol] -  
    Gibbs free energy change for formation of H2O at  
    298.15 K  
19 delta_G_CO_for_298 = -32.8079*10^(3);//[cal/mol] -  
    Gibbs free energy change for formation of CO at
```

```

208.15 K
20 delta_G_H2_for_298 = 0.00; //[cal/mol] - Gibbs free
    energy change for formation of H2 at 298.15 K
21
22 ///Standaerd heat capacity data in cal/mol-K are
    given below, T is in K
23 //Cp_0_C = 3.519 + 1.532*10^(-3)*T - 1.723*10^(5)*T
    ^(-2)
24 //Cp_0_H2O = 7.7 + 0.04594*10^(-2)*T +
    0.2521*10^(-5)*T^(2) - 0.8587*10^(-9)*T^(3)
25 //Cp_0_CO = 6.726 + 0.04001*10^(-2)*T +
    0.1283*10^(-5)*T^(2) - 0.5307*10^(-9)*T^(3)
26 //Cp_0_H2 = 6.952 - 0.04576*10^(-2)*T +
    0.09563*10^(-5)*T^(2) - 0.2079*10^(-9)*T^(3)
27
28 //Therefore standard heat capacity of reaction is
    given by,
29 //Cp_0_rkn = Cp_0_H2 + Cp_0_CO - Cp_0_C - Cp_0_H2O
30 //On simplification ,we get the relation
31 //Cp_0_rkn = 2.459 - 2.0489*10^(-3)*T -
    2.817*10^(-7)*T^(2) + 1.201*10^(-10)*T^(3) +
    1.723*10^(5)*T^(-2)
32
33 delta_H_rkn_298 = delta_H_CO_for_298 +
    delta_H_H2_for_298 - delta_H_C_for_298 -
    delta_H_H2O_for_298; //[cal] - Enthalpy of
    reaction at 298.15 K
34 delta_G_rkn_298 = delta_G_CO_for_298 +
    delta_G_H2_for_298 - delta_G_C_for_298 -
    delta_G_H2O_for_298; //[cal] - Gibbs free energy
    of the reaction at 298.15 K
35
36 delta_H_rkn_1000 = delta_H_rkn_298 + integrate('
    2.459-2.0489*10^(-3)*T-2.817*10^(-7)*T^(2)
    +1.201*10^(-10)*T^(3)+1.723*10^(5)*T^(-2)', 'T',
    T_1,T_2); //[cal]
37
38 printf(" Standard enthalpy change of reaction at

```



```

    1000 K is %f cal\n\n",delta_H_rkn_1000);
39
40 //Now determining the standard entropy change of
    reaction at 298.15 K
41 delta_S_rkn_298 = (delta_H_rkn_298 - delta_G_rkn_298
    )/298.15; //[cal/K]
42 delta_S_rkn_1000 = delta_S_rkn_298 + integrate('
    (2.459-2.0489*10^(-3)*T-2.817*10^(-7)*T^(2)
    +1.201*10^(-10)*T^(3)+1.723*10^(5)*T^(-2))/T', 'T'
    ,T_1,T_2); //[cal/K]
43
44 //Therefore ,the standard Gibbs free energy change of
    the reaction is given by,
45 delta_G_rkn_1000 = delta_H_rkn_1000 - 1000*
    delta_S_rkn_1000; //[cal]
46
47 printf(" Standard Gibbs free energy change of
    reaction at 1000 K is %f cal\n\n",delta_G_rkn_1000)
    ;

```

Scilab code Exa 6.10 Determination of standard enthalpy change and Gibbs free energy change

```

1 clear;
2 clc;
3
4 //Example - 6.10
5 //Page number - 228
6 printf("Example - 6.10 and Page number - 228\n\n");
7
8 //Given
9 T_1 = 298.15; //[K] - Standard temperature
10 T_2 = 1042; //[K] - Reaction temperature
11
12 //At 298.15 K

```

```

13 delta_H_CaCO3_for_298 = -289.5*10^(3);//[cal/mol] -
    Enthalpy of formation of CaCO3 at 298.15 K
14 delta_H_CaO_for_298 = -151.7*10^(3);//[cal/mol] -
    Enthalpy of formation of CaO at 298.15 K
15 delta_H_CO2_for_298 = -94.052*10^(3);//[cal/mol] -
    Enthalpy of formation of CO2 at 298.15 K
16 delta_G_CaCO3_for_298 = -270.8*10^(3);//[cal/mol] -
    Gibbs free energy change for formation of CaCO3
    at 298.15 K
17 delta_G_CaO_for_298 = -144.3*10^(3);//[cal/mol] -
    Gibbs free energy change for formation of CaO at
    298.15 K
18 delta_G_CO2_for_298 = -94.260*10^(3);//[cal/mol] -
    Gibbs free energy change for formation of CO2 at
    298.15 K
19
20 ///Standaerd heat capacity data in cal/mol-K are
    given below, T is in K
21 //Cp_0_CO2 = 5.316 + 1.4285*10^(-2)*T -
    0.8362*10^(-5)*T^(2) + 1.784*10^(-9)*T^(3)
22 //Cp_0_CaO = 12.129 + 0.88*10^(-3)*T + 2.08*10^(5)*T
    ^(-2)
23 //Cp_0_CaCO3 = 24.98 + 5.240*10^(-3)*T +
    6.199*10^(5)*T^(-2)
24
25 //Therefore standard heat capacity of reaction is
    given by,
26 //Cp_0_rkn = Cp_0_CO2 + Cp_0_CaO - Cp_0_CaCO3
27 //On simplification ,we get the relation
28 //Cp_0_rkn = -7.535 + 9.925*10^(-3)*T -
    0.8362*10^(-5)*T^(2) + 1.784*10^(-9)*T^(3) +
    4.119*10^(5)*T^(-2)
29
30 delta_H_rkn_298 = delta_H_CaO_for_298 +
    delta_H_CO2_for_298 - delta_H_CaCO3_for_298;//[
    cal] - Enthalpy of reaction at 298.15 K
31 delta_G_rkn_298 = delta_G_CaO_for_298 +
    delta_G_CO2_for_298 - delta_G_CaCO3_for_298;//[

```

```

    cal] – Gibbs free energy of the reaction at
    298.15 K
32
33 delta_H_rkn_1042 = delta_H_rkn_298 + integrate('
    -7.535+9.925*10^(-3)*T-0.8362*10^(-5)*T^(2)
    +1.784*10^(-9)*T^(3)+4.119*10^(5)*T^(-2)', 'T', T_1
    , T_2); // [cal]
34
35 printf(" Standard enthalpy change of reaction at
    1042 K is %f cal\n\n", delta_H_rkn_1042);
36
37 //Now determining the standard entropy change of
    reaction at 298.15 K
38 delta_S_rkn_298 = (delta_H_rkn_298 - delta_G_rkn_298
    )/298.15; // [cal/K]
39 delta_S_rkn_1042 = delta_S_rkn_298 + integrate('
    (-7.535+9.925*10^(-3)*T-0.8362*10^(-5)*T^(2)
    +1.784*10^(-9)*T^(3)+4.119*10^(5)*T^(-2))/T', 'T',
    T_1, T_2); // [cal/K]
40
41 //Therefore, the standard Gibbs free energy change of
    the reaction is given by,
42 delta_G_rkn_1042 = delta_H_rkn_1042 - 1042*
    delta_S_rkn_1042; // [cal]
43
44 printf(" Standard Gibbs free energy change of
    reaction at 1042 K is %f cal", delta_G_rkn_1042);

```

Chapter 7

Thermodynamic property relations of pure substance

Scilab code Exa 7.1 Proving a mathematical relation

```
1 clear;
2 clc;
3
4 //Example - 7.1
5 //Page number - 235
6 printf("Example - 7.1 and Page number - 235\n\n");
7
8 //This problem involves proving a relation in which
9 //no numerical components are involved.
10 //For prove refer to this example 7.1 on page number
11 //235 of the book.
12 printf(" This problem involves proving a relation in
13 which no numerical components are involved.\n\n"
14 );
15 printf(" For prove refer to this example 7.1 on page
16 number 235 of the book.");
```

Scilab code Exa 7.2 Proving a mathematical relation

```
1 clear;
2 clc;
3
4 //Example - 7.2
5 //Page number - 236
6 printf("Example - 7.2 and Page number - 236\n\n");
7
8 //This problem involves proving a relation in which
9 //no numerical components are involved.
10 //For prove refer to this example 7.2 on page number
11 //236 of the book.
12 printf(" This problem involves proving a relation in
13 //which no numerical components are involved.\n\n"
14 //);
15 printf(" For prove refer to this example 7.2 on page
16 //number 236 of the book.")
```

Scilab code Exa 7.3 Proving a mathematical relation

```
1 clear;
2 clc;
3
4 //Example - 7.3
5 //Page number - 236
6 printf("Example - 7.2 and Page number - 236\n\n");
7
8 //This problem involves proving a relation in which
9 //no numerical components are involved.
10 //For prove refer to this example 7.3 on page number
11 //236 of the book.
12 printf(" This problem involves proving a relation in
13 //which no numerical components are involved.\n\n"
14 //);
```

```
11 printf(" For prove refer to this example 7.3 on page
    number 236 of the book.")
```

Scilab code Exa 7.4 Proving a mathematical relation

```
1 clear;
2 clc;
3
4 //Example - 7.4
5 //Page number - 240
6 printf("Example - 7.4 and Page number - 240\n\n");
7
8 //This problem involves proving a relation in which
    no numerical components are involved.
9 //For prove refer to this example 7.4 on page number
    240 of the book.
10 printf(" This problem involves proving a relation in
    which no numerical components are involved.\n\n"
    );
11 printf(" For prove refer to this example 7.4 on page
    number 240 of the book.")
```

Scilab code Exa 7.5 Proving a mathematical relation

```
1 clear;
2 clc;
3
4 //Example - 7.5
5 //Page number - 240
6 printf("Example - 7.5 and Page number - 240\n\n");
7
8 //This problem involves proving a relation in which
    no numerical components are involved.
```

```

9 //For prove refer to this example 7.5 on page number
   240 of the book.
10 printf(" This problem involves proving a relation in
   which no numerical components are involved.\n\n"
   );
11 printf(" For prove refer to this example 7.5 on page
   number 240 of the book.");

```

Scilab code Exa 7.6 Estimation of entropy change

```

1 clear;
2 clc;
3
4 //Example - 7.6
5 //Page number - 241
6 printf("Example - 7.6 and Page number - 241\n\n");
7
8 //Given
9 P_1 = 1; // [MPa] - Initial pressure
10 P_2 = 1.4; // [MPa] - Final pressure
11
12 //We know that
13 //  $dS = (C_p/T)*dT - (dV/dT)*dP$ 
14 // Along an isothermal path, integration of above
   expression between states 1 and 2 yields
15 //  $S_2 - S_1 = - \int_{P_1}^{P_2} (dV/dT)*dP$ 
16 // An estimate can be made by assuming that  $(dV/dT)$ 
    $_P$  remains constant over the range of pressure
   from  $P_1$  to  $P_2$  and evaluating the derivative at
   average pressure of 1.2 MPa
17 P_avg = P_2;
18 //  $S_2 - S_1 = - \int_{P_1}^{P_2} (dV/dT)*dP$   $_P$   $_{avg}*(P_2 - P_1)$ 
   )
19 //  $(dV/dT)_P=1.2MPa = ((V_{350} - V_{250})/(350 - 250))$ 
20 dV_dT = (0.2345 - 0.19234)/100; //  $[m^3/kg-K]$ 

```

```

21 //Therefore
22 delta_S = -dV_dT*(P_2 - P_1)*1000; //[kJ/kg-K] -
    Entropy change
23
24 printf("The change in entropy is given by\n S_2-S_1
    = %f kJ/kg-K",delta_S);
25
26 //Let us compare this tabulated values. At 1MPa and
    300 C, S_1 = 7.1229 kJ/kg-K. At 1.4 MPa and 300 C
    , S_2 = 6.9534 kJ/kg-K.
27 //Therefore S_2 - S_1 = -0.1695 kJ/kg-K

```

Scilab code Exa 7.7 Determination of work done

```

1 clear;
2 clc;
3
4 //Example - 7.7
5 //Page number - 241
6 printf("Example - 7.7 and Page number - 241\n\n");
7
8 //Given
9 T = 25 + 273.15; //[K] - Temperature of the
    surrounding
10 P_1 = 1; //[atm] - Initial pressure
11 P_2 = 1000; //[atm] - Final pressure
12
13 // V = 18.066 - 7.15*10^(-4)*P + 4.6*10^(-8)*P^(2)
    where, V is in 'cm^(3)/mol' and P is in 'atm
14 // (dV/dT)_P = 0.0045 + 1.4*10^(-6)*P ;//cm^(3)/
    mol-K
15
16 // The work done by 1 mol is given by W = integral(P
    *dV)
17 // Differentiating both sides of the expression for

```



```

V, we get
18 // dV = -7.15*10(-4)*dP + 9.2*10(-8)*(P*dP)
19 // P*dV = -7.15*10(-4)*P*dP + 9.2*10(-8)*(P(2)*dP
    )
20
21 // The work done is given by
22 W = integrate( '-7.15*10(-4)*P + 9.2*10(-8)*(P(2))
    ', 'P', P_1, P_2); // [atm-cm(3)/mol]
23 W = W*101325*10(-6); // [J/mol]
24
25 printf("The necessary work to compress water from 1
    atm to 1000 atm is %f J/mol\n\n", W);
26
27 //Let us calculate the amount of heat transfer
28 // q = integral(T*dS)
29 // dS = ((ds/dT)_P)*dT + ((dS/dP)_T)*dP
30 // Since the temperature is constant the first term
    is zero and
31 // dS = ((dS/dP)_T)*dP = -((dV/dT)_P)*dP
32 // Thus, q = integral(T*dS) = T*(integral(dS))      (
    as temperature is constant )
33 // or, q = -T*(integral((dV/dT)_P)*dP)
34
35 // Thus the heat transfer is given by
36 q = -T*integrate( '0.0045+1.4*10(-6)*P', 'P', P_1, P_2)
    ; // [atm-cm(3)/mol]
37 q = q*101325*10(-6); // [J/mol]
38
39 // q - W = delta_U
40 // Thus delta_U is given by
41 delta_U = q - W; // [J/mol]
42
43 printf("The change in internal energy is %f J/mol",
    delta_U);

```

Scilab code Exa 7.8 Proving a mathematical relation

```
1 clear;
2 clc;
3
4 //Example - 7.8
5 //Page number - 243
6 printf("Example - 7.8 and Page number - 243\n\n");
7
8 //This problem involves proving a relation in which
   no numerical components are involved.
9 //For prove refer to this example 7.8 on page number
   243 of the book.
10 printf(" This problem involves proving a relation in
   which no numerical components are involved.\n\n"
   );
11 printf(" For prove refer to this example 7.8 on page
   number 243 of the book.")
```

Scilab code Exa 7.9 Proving a mathematical relation

```
1 clear;
2 clc;
3
4 //Example - 7.9
5 //Page number - 244
6 printf("Example - 7.9 and Page number - 244\n\n");
7
8 //This problem involves proving a relation in which
   no numerical components are involved.
9 //For prove refer to this example 7.9 on page number
   244 of the book.
10 printf(" This problem involves proving a relation in
   which no numerical components are involved.\n\n"
   );
```

```
11 printf(" For prove refer to this example 7.9 on page
    number 244 of the book.");
```

Scilab code Exa 7.10 Proving a mathematical relation

```
1 clear;
2 clc;
3
4 //Example - 7.10
5 //Page number - 245
6 printf("Example - 7.10 and Page number - 245\n\n");
7
8 //This problem involves proving a relation in which
   no numerical components are involved.
9 //For prove refer to this example 7.10 on page
   number 245 of the book.
10 printf(" This problem involves proving a relation in
    which no numerical components are involved.\n\n"
    );
11 printf(" For prove refer to this example 7.10 on
    page number 245 of the book.");
```

Scilab code Exa 7.11 Proving a mathematical relation

```
1 clear;
2 clc;
3
4 //Example - 7.11
5 //Page number - 246
6 printf("Example - 7.11 and Page number - 246\n\n");
7
8 //This problem involves proving a relation in which
   no numerical components are involved.
```

```

9 //For prove refer to this example 7.11 on page
   number 246 of the book.
10 printf(" This problem involves proving a relation in
   which no numerical components are involved.\n\n"
   );
11 printf(" For prove refer to this example 7.11 on
   page number 246 of the book.");

```

Scilab code Exa 7.12 Evaluation of beta and K for nitrogen gas

```

1 clear;
2 clc;
3
4 //Example - 7.12
5 //Page number - 247
6 printf("Example - 7.9 and Page number - 244\n\n");
7
8 //given
9 T = 25+273.15; // [K] - Temperature
10 P = 1; // [atm] - Pressure
11 P = P*101325; // [Pa]
12 Tc = 126.2; // [K] - Critical temperature
13 Pc = 34; // [bar] - Critical pressure
14 Pc = Pc*10^(5); // [Pa]
15 R=8.314; // [J/mol*K] - Universal gas constant
16
17 a = (27*R^(2)*Tc^(2)/(64*Pc)); // [Pa-m^(6)/mol^(2)]
18 b = (R*Tc)/(8*Pc); // [m^(3)/mol]
19
20
21 // the cubic form of van der Walls equation of state
   is
22 // V^(3)-(b+(R*T)/P)*V^(2)+(a/P)*V-(a*b)/P=0
23 // Solving the cubic equation
24 def(' [y]=f(V) ', 'y=V^(3)-(b+(R*T)/P)*V^(2)+(a/P)*V-(

```

```

    a*b)/P');
25 V = fsolve(1,f);
26 //The above equation has 1 real and 2 imaginary
    roots. We consider only real root.
27
28 Beta = R/((P*V)-(a/V)+((2*a*b)/V^(2))); // [K^(-1)]
29
30 K_t = (V-b)/((P*V)-(a/V)+((2*a*b)/V^(2))); // [Pa^(-1)
    ]
31 K_t = K_t*101325; // [atm^(-1)]
32
33 printf(" Beta\t = \t %f K^(-1)\n",Beta);
34 printf(" K_t\t = \t %f atm^(-1)",K_t);
35
36 //For ideal gas, Beta = 1/T = 0.0033354 K^(-1)
    and K_t = 1/P = 1 atm^(-1)
37 // So results obtained are convergent with those
    obtained assuming ideal gas.

```

Scilab code Exa 7.13 Calculation of temperature change and entropy change of water

```

1 clear;
2 clc;
3
4 //Example - 7.13
5 //Page number - 248
6 printf("Example - 7.13 and Page number - 248\n\n");
7
8 //Given
9 T = 45+273.15; // [K]
10 P_1 = 10; // [kPa] - Initial pressure
11 P_2 = 8600; // [kPa] - Final pressure
12 V = 1010; // [cm^(3)/kg] - Specific volume for
    saturated liquid water at 45 C

```

```

13 V = V*10(-6); // [m(3)/kg]
14 // Beta = (1/V)*(dV/dT)_P
15 Beta = 4.25*10(-4); // [k(-1)]
16 Cp = 4.178; // [kJ/kg-K] - Specific heat at constant
    pressure
17 eff = 0.75; // Efficiency of the pump
18
19 //(1)
20 //when efficiency of the pump is 100% , W = -
    delta_Hs
21 // Now delta_H = T*dS + V*dP, therefore under
    isentropic conditions , dH = V*dP
22 // Since the fluid is liquid , therefore the specific
    volume can be taken to be constant and
    integrating the above equaton we get
23 // delta_Hs = V*dP
24 delta_Hs = V*(P_2 - P_1); // [kJ/kg]
25
26 //Actual pumps are not isentropic and therefore not
    100% efficient. Therefore actual work done by the
    pump is given by
27 W = -delta_Hs/eff; // [kJ/kg]
28
29 printf(" (1).The work done by the pump is %f kJ/kg\n
    ",W);
30
31 //(2)
32 // We know that dH = Cp*dT + (1 - Beta*T)*V*dP
33 // Beta and V are weak functions of pressure in the
    case of liquids.
34 // Integrating the above equation we get
35 // delta_H = Cp*delta_T + (1 - Beta*T)*V*(delta_P)
36 // Now from energy balance delta_H = q - W . But q =
    0. Therefore ,
37 delta_H = -W; // [kJ/kg]
38 // Solving for delta_T
39 delta_T = (delta_H - (1 - Beta*T)*V*(P_2-P_1))/Cp;
40

```

```

41 printf(" (2).The temperature of water change by,
    delta_T = %f K\n",delta_T);
42
43 //(3)
44 T_1 = T;//[K]
45 T_2 = T + delta_T;//[K]
46 // dS = (Cp/T)*dT - Beta*V*dP
47 // Beta and V are weak functions of pressure in the
    case of liquids. Integrating the above equation
    we get
48 delta_S = Cp*log(T_2/T_1) - Beta*V*(P_2-P_1);//[kJ/
    kg-K]
49
50 printf(" (3).The entropy change of water is given by
    delta_S = %f kJ/kg-K",delta_S);

```

Scilab code Exa 7.14 Estimation of change in entropy and enthalpy

```

1 clear;
2 clc;
3
4 //Example - 7.14
5 //Page number - 249
6 printf("Example - 7.14 and Page number - 249\n\n");
7
8 //Given
9 T = 270;//[K]
10 P_1 = 381;//[kPa] - Initial pressure
11 P_2 = 1200;//[kPa] - Final pressure
12 V_liq = 1.55*10^(-3);//[m^(3)/kg] - Specific volume
    for saturated water in liquid phase at 270 C
13 Beta = 2.095*10^(-3);//[K^(-1)]
14
15 //dH = Cp*dT + [V - T*(dV/dT)_P]*dP
16 // dS = (Cp/T)*dT - ((dV/dT)_P)*dP

```

```

17 // Since isothermal conditions are maintained we get
18 //  $dH = [V - T*(dV/dT)_P]*dP = V*(1 - Beta*T)*dP$ 
19 // For the liquid assuming V and Beta to remain
    constant during pressure change, and since
    temperature is constant we get
20 delta_H = V_liq*(1 - Beta*T)*(P_2 - P_1);//[kJ/kg]
21
22 printf("The enthalpy change is given by delta_H = %f
    kJ/kg\n",delta_H);
23
24 // Under isothermal conditions
25 //  $dS = -((dV/dT)_P)*dP = -Beta*V_liq*dP$ 
26 // If Beta*V is assumed to remain constant during
    pressure change we get
27 delta_S = -Beta*V_liq*(P_2-P_1);//[kJ/kg-K]
28
29 printf("The entropy change is given by delta_S = %e
    kJ/kg-K",delta_S);

```

Scilab code Exa 7.15 Calculation of percentage change in volume

```

1 clear;
2 clc;
3
4 //Example - 7.15
5 //Page number - 249
6 printf("Example - 7.15 and Page number - 249\n\n");
7
8 //Given
9 T_1 = 0;//[C] - Initial tempetaure
10 T_2 = 100;//[C] - Final temperature
11 //  $Beta = 1.0414*10^{-3} + 1.5672*10^{-6}*T +$ 
     $5.148*10^{-8}*T^2$ , where T is in C
12 // At constant pressure  $(1/V)*(dV/dT) = Beta$ 
13 // or,  $d(\log(V)) = Beta*dT$ 

```



```

14 // Integrating we get  $\log(V_2/V_1) = \int_{T_1}^{T_2} \text{Beta} \cdot dT$ 
    // from limit T_1 to T_2
15 integral = integrate('1.0414*10^(-3)+1.5672*10^(-6)*
    T+5.148*10^(-8)*T^(2)', 'T', T_1, T_2);
16
17 //  $\log(V_2/V_1) = \text{integral}$ 
18 //  $(V_2/V_1) = \exp(\text{integral})$ 
19 //  $(V_2 - V_1)/V_1 = \text{change} = \exp(\text{integral}) - 1$ ;
20 change = exp(integral) - 1;
21 per_change = 100*change;
22
23 printf("The percentage change in volume = %f %%",
    per_change);

```

Scilab code Exa 7.16 Determination of enthalpy and entropy change

```

1 clear;
2 clc;
3
4 //Example - 7.16
5 //Page number - 250
6 printf("Example - 7.16 and Page number - 250\n\n");
7
8 //Given
9 T_1 = 25 + 273.15; // [C] - Initial tempetaure
10 T_2 = 50 + 273.15; // [C] - Final temperature
11 P_1 = 1; // [bar] - Initial pressure
12 P_2 = 1000; // [bar] - Final pressure
13
14 Cp_T1_P1 = 75.305; // [J/mol-K]
15 Cp_T2_P1 = 75.314; // [J/mol-K]
16 V_T1_P1 = 18.071; // [cm^(3)/mol]
17 V_T1_P2 = 18.012; // [cm^(3)/mol]
18 V_T2_P1 = 18.234; // [cm^(3)/mol]
19 V_T2_P2 = 18.174; // [cm^(3)/mol]

```

```

20 Beta_T1_P1 = 256*10^(-6); // [K^(-1)]
21 Beta_T1_P2 = 366*10^(-6); // [K^(-1)]
22 Beta_T2_P1 = 458*10^(-6); // [K^(-1)]
23 Beta_T2_P2 = 568*10^(-6); // [K^(-1)]
24
25 // The entropy change is given by
26 // dS = (Cp/T)*dT - ((dV/dT)_P)*dP
27 // The mean Cp between 25 and 50 C is
28 Cp_mean = (Cp_T1_P1 + Cp_T1_P1)/2; // [J/mol-K]
29
30
31 // (dV/dT)_P=1bar = (V_T2_P1 - V_T1_P1)/(50 - 25)
32 dV_dT_P1 = ((V_T2_P1 - V_T1_P1)/(50 - 25))*10^(-6);
    // [m^(-3)/mol-K]
33 dV_dT_P2 = ((V_T2_P2 - V_T1_P2)/(50 - 25))*10^(-6);
    // [m^(-3)/mol-K]
34 // The mean value of (dV/dT)_P between 1 and 1000
    bar is
35 dV_dT_mean = (dV_dT_P1 + dV_dT_P2)/2; // [m^(-3)/mol-K
    ]
36 delta_S = Cp_mean*log(T_2/T_1) - dV_dT_mean*(P_2 -
    P_1)*10^5; // [J/mol-K]
37
38 printf(" The value of entropy change is given by,
    delta_S = %f J/mol-K\n", delta_S);
39
40 // Now let us determine the enthalpy change. We know
    that
41 // dH = Cp*dT + [V - T*(dV/dT)_P]*dP
42 // [V - T*(dV/dT)_P] = (V - T*V*Beta) = val (say)
43 // At state 1
44 val_1 = ((V_T1_P1)*10^(-6))*(1 - (T_1)*(Beta_T1_P1))
    ; // [m^3/mol]
45 // At state 2
46 val_2 = ((V_T2_P2)*10^(-6))*(1 - (T_2)*(Beta_T2_P2))
    ; // [m^3/mol]
47 val_mean = (val_1 + val_2)/2; // [m^3/mol]
48

```

```

49 delta_H = Cp_mean*(T_2 - T_1) + val_mean*(P_2-P_1)
    *10^(5);//[J/mol]
50
51 printf(" The value of enthalpy change is given by,
    delta_H = %f J/mol",delta_H);

```

Scilab code Exa 7.17 Proving a mathematical relation

```

1  clear;
2  clc;
3
4  //Example - 7.17
5  //Page number - 253
6  printf("Example - 7.17 and Page number - 253\n\n");
7
8  //This problem involves proving a relation in which
    no numerical components are involved.
9  //For prove refer to this example 7.17 on page
    number 253 of the book.
10 printf(" This problem involves proving a relation in
    which no numerical components are involved.\n\n"
    );
11 printf(" For prove refer to this example 7.17 on
    page number 253 of the book.");

```

Scilab code Exa 7.18 Proving a mathematical relation

```

1  clear;
2  clc;
3
4  //Example - 7.18
5  //Page number - 253
6  printf("Example - 7.18 and Page number - 253\n\n");

```

```

7
8 //This problem involves proving a relation in which
   no numerical components are involved.
9 //For prove refer to this example 7.18 on page
   number 253 of the book.
10 printf(" This problem involves proving a relation in
   which no numerical components are involved.\n\n"
   );
11 printf(" For prove refer to this example 7.18 on
   page number 253 of the book.");

```

Scilab code Exa 7.19 Proving a mathematical relation

```

1 clear;
2 clc;
3
4 //Example - 7.19
5 //Page number - 254
6 printf("Example - 7.19 and Page number - 254\n\n");
7
8 //This problem involves proving a relation in which
   no numerical components are involved.
9 //For prove refer to this example 7.19 on page
   number 254 of the book.
10 printf(" This problem involves proving a relation in
   which no numerical components are involved.\n\n"
   );
11 printf(" For prove refer to this example 7.19 on
   page number 254 of the book.");

```

Scilab code Exa 7.20 Proving a mathematical relation

```

1 clear;

```

```

2  clc;
3
4  //Example - 7.20
5  //Page number - 254
6  printf("Example - 7.20 and Page number - 254\n\n");
7
8  //This problem involves proving a relation in which
   no numerical components are involved.
9  //For prove refer to this example 7.20 on page
   number 254 of the book.
10 printf(" This problem involves proving a relation in
     which no numerical components are involved.\n\n"
   );
11 printf(" For prove refer to this example 7.20 on
     page number 254 of the book.");

```

Scilab code Exa 7.21 Proving a mathematical relation

```

1  clear;
2  clc;
3
4  //Example - 7.21
5  //Page number - 255
6  printf("Example - 7.21 and Page number - 255\n\n");
7
8  //This problem involves proving a relation in which
   no numerical components are involved.
9  //For prove refer to this example 7.21 on page
   number 255 of the book.
10 printf(" This problem involves proving a relation in
     which no numerical components are involved.\n\n"
   );
11 printf(" For prove refer to this example 7.21 on
     page number 255 of the book.");

```

Scilab code Exa 7.22 Calculation of volume expansivity and isothermal compressibility

```
1 clear;
2 clc;
3
4 //Example - 7.22
5 //Page number - 256
6 printf("Example - 7.22 and Page number - 256\n\n");
7
8 // Given
9 T = 100 + 273.15; // [K]
10 P = 10; // [MPa]
11
12 // The volume expansivity is defined as
13 //  $\beta = (1/V) * (\partial V / \partial T)_P = (1/V) * (dV/dT)_P$ 
14 // From compressed liquid water tables at 100 C and
15 // 10 MPa,
16 V = 0.0010385; // [m(3)/kg]
17 Beta = (1/V) * ((0.0010549 - 0.0010245) / (120 - 80)); //
18 // [K-1] // The values are obtained from the
19 // steam table as reported in the book.
20
21 printf("The value of volume expansivity is Beta =
22 %e K-1\n", Beta);
23
24 // Isothermal compressibility is defined as
25 //  $K_t = -(1/V) * (\partial V / \partial T)_T = -(1/V) * (dV/dT)_T$ 
26 K_t = -(1/V) * ((0.0010361 - 0.0010410) / (15 - 5)); // [
27 // MPa-1] // The values are obtained from the
28 // steam table as reported in the book.
29
30 K_t = K_t * 10-3; // [kPa]
```

```

26 printf("The value of isothermal compressibility is
      K_t = %e kPa(-1)\n",K_t);
27
28 // Cp - Cv = (T*V*(Beta^(2)))/K_t
29 R = (T*V*(Beta^(2)))/K_t; // [kJ/kg-K]
30
31 printf("The value of the difference between Cp and
      Cv is Cp-Cv = %f kJ/kg-K",R);

```

Scilab code Exa 7.23 Estimation of specific heat capacity

```

1 clear;
2 clc;
3
4 //Example - 7.23
5 //Page number - 257
6 printf("Example - 7.23 and Page number - 257\n\n");
7
8 // Given
9 T = 300 + 273.15; // [K]
10 P = 4; // [MPa]
11
12 Cp_0 = 7.7 + 0.04594*10(-2)*T + 0.2521*10(-5)*T
      ^2 - 0.8587*10(-9)*T(3); // [cal/mol-K]
13 Cp_0 = (Cp_0*4.186)/18.015; // [kJ/kg-K]
14
15 // Cp(T,P) = Cp_0(T,P=0) - T*integral((del^2 V/del T
      ^2)_P)*dP from limit 0 to P
16 // Cp = Cp_0 - T*((del^2 V/del T^2)_Pavg)*(P_2 - P_1
      )
17
18 P_avg = (0+4)/2; // [MPa]
19
20 //Using finite difference we get (del^2 V/del T^2)
      ) = ((V_(T+delta T) - 2*V_T + V_(T-delta T)))/(

```

```

    delta_T^(2))
21 //((del^2 V/del T^(2))_Pavg = (V_(350 C) + V_(250 C)
    - 2*V_(300 C))/(delta_T^(2)) = del_2 (say)
22 del_2 = (0.13857 + 0.11144 - 2*0.12547)/(50^(2));//[
    m^(3)/kg-K^2] // The values are obtained from the
    steam table as reported in the book.
23
24
25 Cp = Cp_0 - T*del_2*4000;//[kJ/kg-K]
26
27 printf(" The value of constant pressure specific
    heat capacity is , Cp = %f kJ/kg-K",Cp);
28
29 // At P = 4 MPa
30 // Cp = (del H/del T)_P = (H_350 C - H_250 C)/(350 -
    250.4)
31 // Cp = (3092.5 - 2801.4)/(350 - 250.4) = 2.923 [kJ/
    kg-K]

```

Scilab code Exa 7.24 Estimation of specific heat capacity

```

1 clear;
2 clc;
3
4 //Example - 7.24
5 //Page number - 257
6 printf("Example - 7.24 and Page number - 257\n\n");
7
8 // Given
9 T = 300 + 273.15;//[K]
10 P = 2.0;//[MPa]
11
12 // At 2 MPa and 250 C
13 H_1 = 2902.5;//[kJ/kg]
14 // At 2 MPa and 350 C

```



```

15 H_2 = 3137.0; // [kJ/kg]
16
17 Cp = (H_2 - H_1)/(350 - 250); // [kJ/kg-K]
18
19 printf(" The value of constant pressure specific
    heat capacity is , Cp = %f kJ/kg-K", Cp);

```

Scilab code Exa 7.25 Calculation of volume expansivity and isothermal compressibility

```

1 clear;
2 clc;
3
4 //Example - 7.25
5 //Page number - 258
6 printf("Example - 7.25 and Page number - 258\n\n");
7
8 // Given
9 T = 80 + 273.15; // [K]
10 P = 10; // [MPa]
11
12 // Beta = (1/V)*(del V/del T)_P
13
14 // Pressure is kept fixed at 10 MPa and (del V/del T
    )_P is evaluated. Looking in the compressed
    liquid water tables , at
15 // At 80 C and 10 MPa
16 V_1 = 0.0010245; // [m^(3)/kg]
17 // At 60 C and 10 MPa
18 V_2 = 0.0010127; // [m^(3)/kg]
19 // At 100 C and 10 MPa
20 V_3 = 0.0010385; // [m^(3)/kg]
21
22 Beta = (1/V_1)*((V_3 - V_2)/(100 - 60)); // [K^(-1)]
23

```

```

24 printf("The value of volume expansivity is Beta =
      %e K(-1)\n",Beta);
25
26 //Isothermal compressibility is given by
27 // K_t = -(1/V)*(del V/del P)_T
28
29 // Temperature is kept fixed at 80 C and different
      pressures are taken to calculate (del V/del P)_T
30 // At 80 C and 5 MPa
31 V_4 = 0.0010268; // [m(3)/kg]
32 // At 80 C and 10 MPa
33 V_5 = 0.0010245; // [m(3)/kg]
34 // At 80 C and 15 MPa
35 V_6 = 0.0010222; // [m(3)/kg]
36
37 // K_t = -(1/V)*(del V/del T)_P
38 K_t = -(1/V_1)*((V_4 - V_6)/(5 - 15)); // [MPa(-1)]
39 K_t = K_t*10(-6); // [Pa(-1)]
40
41 printf("The value of isothermal compressibility is
      K_t = %e Pa(-1)\n",K_t);
42
43 // Cp - Cv = (T*V*(Beta(2)))/K_t
44 R = (T*V_1*(Beta(2)))/K_t; // [J/kg-K]
45 R = R*10(-3); // [kJ/kg-K]
46
47 printf("The value of the difference between Cp and
      Cv is Cp-Cv = %f kJ/kg-K",R);

```

Scilab code Exa 7.26 Calculation of mean Joule Thomson coefficient

```

1 clear;
2 clc;
3
4 //Example - 7.26

```

```

5 //Page number – 260
6 printf("Example – 7.26 and Page number – 260\n\n");
7
8 // Given
9 P_1 = 150; //[bar]
10 P_2 = 1; //[bar]
11
12 T_1 = 300; //[K]
13 T_2 = 260; //[K]
14 T_3 = 280; //[K]
15 T_4 = 200; //[K]
16 T_5 = 120; //[K]
17 T_6 = 140; //[K]
18
19 H_P1_T1 = 271.8; //[kJ/kg]
20 H_P2_T2 = 260.0; //[kJ/kg]
21 H_P2_T3 = 280.2; //[kJ/kg]
22 H_P1_T4 = 129.2; //[kJ/kg]
23 H_P2_T5 = 118.8; //[kJ/kg]
24 H_P2_T6 = 139.1; //[kJ/kg]
25
26 //(a)
27 // During the Joule–Thomson expansion the enthalpy
    should remain constant
28 // Therefore at 1 bar the exit temperature is such
    that enthalpy is 271.8 kJ/kg
29 // The temperature at which enthalpy is 271.8 kJ/kg
    is given by,
30 T_new = ((H_P1_T1 - H_P2_T2)/(H_P2_T3 - H_P2_T2))*
    (T_3 - T_2) + T_2; //[K]
31
32 // Therefore Joule–Thomson coefficient is given by,
33 meu = (T_1 - T_new)/(P_1 - P_2); //[K/bar]
34
35 printf(" (a).The value of Joule–Thomson coefficient
    (for initial T = 300 K) is %f J/bar\n",meu);
36
37 //(b)

```

```

38 // During the Joule–Thomson expansion the enthalpy
    should remain constant
39 // Therefore at 1 bar the exit temperature is such
    that enthalpy is 129.2 kJ/kg
40 // The temperature at which enthalpy is 129.2 kJ/kg
    is given by,
41 T_new_prime = ((H_P1_T4 - H_P2_T5)/(H_P2_T6 -
    H_P2_T5))*(T_6 - T_5) + T_5; //[K]
42
43 // Therefore Joule–Thomson coefficient is given by,
44 meu_prime = (T_4 - T_new_prime)/(P_1 - P_2); //[K/bar
    ]
45
46 printf(" (b).The value of Joule–Thomson coefficient
    (for initial T = 200 K) is %f J/bar",meu_prime);
47
48 // Therefore the Joule–Thomson coefficient is higher
    for low initial temperatures and therefore the
    drop in temperature is more.

```

Scilab code Exa 7.27 Estimation of Joule Thomson coefficient

```

1  clear;
2  clc;
3
4  //Example – 7.27
5  //Page number – 261
6  printf("Example – 7.27 and Page number – 261\n\n");
7
8  //(a)
9  //This part involves proving a relation in which no
    numerical components are involved.
10 //For prove refer to this example 7.27 on page
    number 261 of the book.
11

```

```

12 //(b)
13 //This part involves proving a relation in which no
    numerical components are involved.
14 //For prove refer to this example 7.27 on page
    number 261 of the book.
15
16 //(c)
17 T = 300; //[K] - Temperature
18 P = 5; //[atm] - Pressure
19 P = P*101325; //[Pa]
20 Cp_0 = 35.78; //[J/mol-K] - Standard specific heat
    capacity at constant pressure
21 B = -50; //[cm^(3)/mol]
22 B = B*10^(-6); //[m^(3)/mol]
23
24 //(dB/dT) = 1.0 = dB_dT (say)
25 dB_dT = 1.0; //[cm^(3)/mol-K]
26 dB_dT = dB_dT*10^(-6); //[m^(3)/mol-K]
27
28 //(d^2 B/d T^2) = -0.01 = dB_dT_2 (say)
29 dB_dT_2 = -0.01; //[cm^(3)/mol-K^(2)]
30 dB_dT_2 = dB_dT_2*10^(-6); //[m^(3)/mol-K^(2)]
31
32 Cp = Cp_0 - P*T*(dB_dT_2); // [[J/mol-K]] - Specific
    heat capacity at constant pressure
33
34 //Therefore Joule-Thomson coefficient is given by,
35 meu = (1/Cp)*(-B + T*dB_dT); //[K/Pa]
36 meu = meu*10^5; //[K/bar]
37
38 printf(" (c).The value of Joule-Thomson coefficient
    is %f J/bar", meu);

```

Scilab code Exa 7.28 Proving a mathematical relation

```

1  clear;
2  clc;
3
4  //Example - 7.28
5  //Page number - 262
6  printf("Example - 7.28 and Page number - 262\n\n");
7
8  //This problem involves proving a relation in which
   no numerical components are involved.
9  //For prove refer to this example 7.28 on page
   number 262 of the book.
10 printf(" This problem involves proving a relation in
       which no numerical components are involved.\n\n"
       );
11 printf(" For prove refer to this example 7.28 on
       page number 262 of the book.");

```

Scilab code Exa 7.29 Proving a mathematical relation

```

1  clear;
2  clc;
3
4  //Example - 7.29
5  //Page number - 263
6  printf("Example - 7.29 and Page number - 263\n\n");
7
8  //This problem involves proving a relation in which
   no numerical components are involved.
9  //For prove refer to this example 7.29 on page
   number 263 of the book.
10 printf(" This problem involves proving a relation in
       which no numerical components are involved.\n\n"
       );
11 printf(" For prove refer to this example 7.29 on
       page number 263 of the book.");

```

Scilab code Exa 7.30 Calculation of pressure

```
1 clear;
2 clc;
3
4 //Example - 7.30
5 //Page number - 267
6 printf("Example - 7.30 and Page number - 267\n\n");
7
8 //Given
9 den_liq = 13690; //[kg/m^(3)] - Density of liquid
    mercury
10 den_solid = 14190; //[kg/m^(3)] - Density of solid
    mercury
11 mp = -38.87; //[C] - Melting point of mercury at
    pressure of 1 bar
12 mp = mp + 273.15; //[K]
13 T_req = 0; //[C] - Required temperature to which the
    melting point is to be raised
14 T_req = T_req + 273.15; //[K]
15 H_fus = 11.62; //[kJ/kg] - Latent heat of fusion of
    mercury
16
17 V_liq = (1/den_liq); //[m^(3)/kg] - Specific volume
    of liquid mercury
18 V_solid = (1/den_solid); //[m^(3)/kg] - Specific
    volume of solid mercury
19
20 // (delta P/delta T) = ((P - 1)*100)/(T_req - mp)
21 // delta H/(T*delta V) = (H_liq - H_solid)/(T*(V_liq
    - V_solid)) = del (say)
22 del = (H_fus)/(mp*(V_liq - V_solid)); //[kPa/K] -
    delta H/(T*delta V)
23
```

```

24 //Equating the two sides and then solving we get
25 P = (del*(T_req - mp))/100 + 1;//[bar]
26
27 printf(" The required pressure should be %f bar",P);

```

Scilab code Exa 7.31 Calculation of enthalpy change and entropy change

```

1 clear;
2 clc;
3
4 //Example - 7.31
5 //Page number - 268
6 printf("Example - 7.31 and Page number - 268\n\n");
7
8 // The clapeyron equation is
9 // (dP/dT)-sat = delta_H_fus/(T*delta_V_fus)
10
11 //(1)
12 // At 1 bar
13 // Considering the data given at pressure 1 and 1000
   bar , we have
14 delta_H_fus_1 = ((1000-1)*10^(5)*(273.15-22.6)
   *3.97*10^(-6))/(14.8+22.6);//[J/mol]
15 delta_S_fus_1 = delta_H_fus_1/(273.15-22.6);//[J/mol
   -K]
16
17 printf(" (1).The delta_H_fus at 1 bar is %f J/mol\n"
   ,delta_H_fus_1);
18 printf("      The delta_S_fus at 1 bar is %f J/mol-K\
   n\n",delta_S_fus_1);
19
20 //(2)
21 // At 6000 bar
22 T_mean = (128.8+173.6)/2;//[C] - Mean temperature
23 T_mean = T_mean + 273.15;//[K]

```



```

24 delta_V_fus_mean = (1.12+1.55)/2; //[cm^(3)/mol]
25
26 // Consider the data at pressure of 5000 and 7000
    bar we get ,
27 delta_H_fus_2 = ((7000-5000)*10^(5)*(T_mean*
    delta_V_fus_mean*10^(-6)))/(173.6-128.8); //[J/mol
    ]
28 delta_S_fus_2 = delta_H_fus_2/T_mean; //[J/mol-K]
29
30 printf(" (2).The delta_H_fus at 6000 bar is %f J/mol
    \n",delta_H_fus_2);
31 printf("      The delta_S_fus at 6000 bar is %f J/mol
    -K\n\n",delta_S_fus_2);

```

Scilab code Exa 7.32 Estimation of ratio of temperature change and pressure change

```

1 clear;
2 clc;
3
4 //Example - 7.32
5 //Page number - 268
6 printf("Example - 7.32 and Page number - 268\n\n");
7
8 //Given
9 H_fus = 80; //[cal/g] - Heat of fusion at 0 C and 1
    atm pressure
10 T = 0+273.15; //[K] - Temperature
11 vol_ratio = 1.091; // Ratio of the specific volume of
    ice and water.
12 sp_vol = 0.001; //[m^(3)/kg] - Specific volume of
    saturated liquid water.
13
14 // The clapeyron equation can be written as
15 // (dP/dT)_sat = T*delta V_LS/(delta H_LS) = (T*(

```

```

    (V_ice - V_water))/(H_ice - H_water)
16 dP_dT = (T*(vol_ratio - 1)*10^(-3))/(-H_fus*4.186);
    // [K/kPa]
17
18 printf("The value of (dT/dP)_sat is %e K/kPa",dP_dT)
    ;

```

Scilab code Exa 7.33 Determination of boiling point of water

```

1 clear;
2 clc;
3
4 //Example - 7.33
5 //Page number - 268
6 printf("Example - 7.33 and Page number - 268\n\n");
7
8 //Given
9 P = 2; //[atm] - Surrounding pressure
10 bp_water = 100 + 273.15; //[K] - Boiling point of
    water at 1 atm pressure
11 delta_H_vap = 2257; //[kJ/kg] - Enthalpy of
    vaporization
12 delta_H_vap = delta_H_vap*18.015; //[J/mol]
13 R = 8.314; //[J/mol*K] - Universal gas constant
14
15 // The clapeyron equation is given by
16 // log(P_2_sat/P_1_sat) = (-delta H_vap/R)*(1/T_2 -
    1/T_1)
17 P_1_sat = 1; //[atm]
18 P_2_sat = P;
19 T_1 = bp_water;
20
21 // Solving the above equation
22 T_2 = 1/((log(P_2_sat/P_1_sat))/(-delta_H_vap/R) +
    (1/T_1)); //[K]

```

```

23 T_2 = T_2 - 273.15; //[C]
24
25 printf(" The boiling point of water at a pressure of
      2 atm is %f C",T_2);

```

Scilab code Exa 7.34 Calculation of enthalpy and entropy of vaporization of water

```

1 clear;
2 clc;
3
4 //Example - 7.34
5 //Page number - 269
6 printf("Example - 7.34 and Page number - 269\n\n");
7
8 //Given
9 T_1 = 0.01 +273.15; //[K]
10 T_2 = 1 + 273.15; //[K]
11 P_sat_1 = 0.611; //[kPa] - P_sat at temperature T_1
12 P_sat_2 = 0.657; //[kPa] - P_sat at temperature T_2
13 Mol_wt = 18.015; //[g/mol] - Molecular weight of
      water
14 R = 8.314; //[J/mol*K] - Universal gas constant
15
16 // The clapeyron equation is given by
17 //  $\log(P_{sat\_2}/P_{sat\_1}) = (-\Delta H_{LV}/R)*(1/T_2 - 1/T_1)$ 
18
19 // Solving the above equation
20 delta_H = -(log(P_sat_2/P_sat_1)/(1/T_2 - 1/T_1))*R;
      //[J/mol]
21 delta_H = delta_H/Mol_wt; //[kJ/kg]
22
23 printf(" The enthalpy of vaporization is %f kJ/kg\n",
      ,delta_H);

```

```

24
25 // Entropy of vaporization is given by
26 S_vap = delta_H/T_2; // [kJ/kg-K]
27 printf(" The entropy of vaporization is %f kJ/kg-K",
        S_vap);

```

Scilab code Exa 7.35 Estimation of heat of vaporization of water

```

1 clear;
2 clc;
3
4 //Example - 7.35
5 //Page number - 269
6 printf("Example - 7.35 and Page number - 269\n\n");
7
8 //Given
9 T = 100 + 273.15; // [K]
10 // (dT/dP)_sat = (1/27.12) K/mm
11 dT_dP = (1/27.12); // [K/mm]
12 dT_dP = dT_dP*(760/101325); // [K/Pa]
13
14 // The clapeyron equation is given by
15 // (dP/dT)_sat = (-delta H_LV)/(T*delta V_LV)
16 // delta H_LV = T*delta V_LV*(dP/dT)_sat
17
18 // (dP/dT)_sat = 1/(dT/dP)_sat
19 dP_dT = 1/dT_dP; // [Pa/K]
20
21 // From saturated steam table at 100 C
22 V_vap = 1.6729; // [m^(3)/kg]
23 V_liq = 0.001044; // [m^(3)/kg]
24 delta_V = V_vap - V_liq; // [m^(3)/kg]
25
26 // Therefore delta_H_LV is given by
27 delta_H_LV = T*delta_V*(dP_dT); // [J/kg]

```

```

28 delta_H_LV = delta_H_LV*10^(-3);//[kJ/kg]
29
30 printf(" The heat of vaporization of water is %f kJ/
    kg\n",delta_H_LV);

```

Scilab code Exa 7.36 Calculation of latent heat of vaporization

```

1 clear;
2 clc;
3
4 //Example - 7.36
5 //Page number - 270
6 printf("Example - 7.36 and Page number - 270\n\n");
7
8 //Given
9 T_1 = 100 + 273.15;//[K]
10 P_1 = 1.01325;//[bar]
11 T_2 = 98 + 273.15;//[K]
12 P_2 = 0.943;//[bar]
13 V_vap = 1.789;//[m^(3)] - Volume in vapour phase
14 vessel_vol = 1.673;//[m^(3)] - Volume of the vessel
15 R = 8.314;//[J/mol*K] - Universal gas constant
16
17 // The total volume remains constant as the walls
    are rigid. At 98 C we get
18 // vessel_vol = V_liq*(1 - x) + V_vap*x
19 // Since V_liq is negligible as compared to V_vap,
    therefore
20 x = vessel_vol/V_vap;
21
22 // The quantity is given by x = m_vap/(m_liq + m_vap
    )
23 // Since (m_liq + m_vap) = 1, therefore at 98 C
    saturated vapour is x and saturated liquid is (1
    - x)

```

```

24 m_vap = x; //[kg] - Mass of saturated vapour
25 m_liq = (1 - x); //[kg] - Mass of saturated liquid
26
27 printf(" The amount of vapour condensed is %f kg\n",
        m_liq);
28
29 // The clapeyron equation is given by
30 //  $\log(P_{2\_sat}/P_{1\_sat}) = (-\Delta H_{LV}/R) * (1/T_2 - 1/T_1)$ 
31
32 // Solving the above equation
33 delta_H = -(log(P_2/P_1)/(1/T_2 - 1/T_1))*R;
34 delta_H = delta_H/18.015; //[kJ/kg]
35
36 printf(" The latent heat of vaporization is %f kJ/kg
        \n", delta_H);

```

Scilab code Exa 7.37 Determination of temperature dependence

```

1 clear;
2 clc;
3
4 //Example - 7.37
5 //Page number - 270
6 printf("Example - 7.37 and Page number - 270\n\n");
7
8 //Given
9 T_1 = 298.15; //[K] - Standard reaction temperature
10 delta_H_gas = -52.23; //[kcal/mol] - Enthalpy of
    formation of C2H5OH(gas)
11 delta_H_liq = -66.35; //[kcal/mol] - Enthalpy of
    formation of C2H5OH(liq)
12
13 // For ethanol(g) [T is in K and Cp_0 in cal/mol-K]
14 //  $Cp_0 = 4.75 + 5.006 * 10^{-2} * T - 2.479 * 10^{-5} * T^2$ 

```

```

    ^ (2) + 4.79*10^(-9)*T^(3)
15
16 // For ethanol(1) [T is in K and Cp_0 in cal/mol-K]
17 // Cp_0 = 67.29 - 0.343*T - 6.94*10^(-4)*T^(2)
18
19 // The vaporization of a liquid can be written as
    C2H5OH(liq) - C2H5OH(gas)
20 // Since the pressure is 1 atm therefore the
    standard data can be used
21 delta_H_298 = delta_H_gas - delta_H_liq;//[kcal/mol]
22 delta_H_298 = delta_H_298*1000;//[cal/mol]
23 delta_a = 4.75 - 67.29;
24 delta_b = 5.006*10^(-2) - (-0.343);
25 delta_c = -2.479*10^(-5) - 6.94*10^(-4);
26 delta_d = 4.79*10^(-9);
27
28 // The standard enthalpy of vaporization at a
    temperature T is given by
29 // delta_H_T = delta_H_298 + integrate('delta_a +
    delta_b*T + delta_c*T^(2) + delta_d*T^(3)', 'T',
    T_1, T);//[cal/mol]
30
31 // Therefore the standard enthalpy of vaporization
    at a temperature T = 283 K is given by
32 T_2 = 283;//[K]
33 delta_H_283 = delta_H_298 + integrate('delta_a+
    delta_b*T+delta_c*T^(2)+delta_d*T^(3)', 'T', T_1,
    T_2);//[cal/mol]
34
35 // Therefore the standard enthalpy of vaporization
    at a temperature T = 348 K is given by
36 T_3 = 348;//[K]
37 delta_H_348 = delta_H_298 + integrate('delta_a+
    delta_b*T+delta_c*T^(2)+delta_d*T^(3)', 'T', T_1,
    T_3);//[cal/mol]
38
39 // From the values of standard enthalpy of
    vaporization obtained above at 283, 298, and 348

```

```

    K, it is clear that enthalpy decreases with
    increase in temperature
40 printf(" The value of enthalpy of vaporization at
    283 K is %f cal/mol\n",delta_H_283);
41 printf(" The value of enthalpy of vaporization at
    298.15 K is %f cal/mol\n",delta_H_298);
42 printf(" The value of enthalpy of vaporization at
    348 K is %f cal/mol\n",delta_H_348);
43 printf(" Therefore standard enthalpy of vaporization
    decrease with the increase in temperature\n\n");
44
45 // Solving the above equatio manually we get ,
46 //  $\Delta H_{\text{vap}} = 1.1975 \times 10^{-9} T^4 -$ 
     $2.396 \times 10^{-4} T^3 + 0.1965 T^2 - 62.54 T +$ 
     $21639.54$ 
47 // Solving for 'T' at which ' $\Delta H_{\text{vap}}$ ' = 0
48 def (' [y]=f(T) ', 'y= $1.1975 \times 10^{-9} T^4 -$ 
     $2.396 \times 10^{-4} T^3 + 0.1965 T^2 - 62.54 T +$ 
     $21639.54$  ');
49 T_0 = fsolve(500,f);//[J/mol]
50
51 // We know that at critical point (critical
    temperature and critical pressure) the enthalpy
    of vaporization is zero.
52 // Here we have made the standard enthalpy of
    vaporization equal to zero which is at standard
    pressure of 1 atm.
53 // Therefore following conclusions can be drawn
54 printf(" The temperature obtained by equating
    standard enthalpy of vaporization equal to zero
    is %f K\n",T_0);
55 printf(" But the critical temperature of ethanol is
    513.9 K, which is far from the temperature
    obtained above\n")
56 printf(" Therefore the temperature obtained by
    equating standard enthalpy of vaporization equal
    to zero is not the critical temperature")

```

Scilab code Exa 7.38 Calculation of fugacity of water

```
1 clear;
2 clc;
3
4 //Example - 7.38
5 //Page number - 276
6 printf("Example - 7.38 and Page number - 276\n\n");
7
8 //Given
9 T = 300 + 273.15; // [K] - Temperature
10 P = 9000; // [kPa] - Pressure
11 P_sat = 8592.7; // [kPa] - Vapour pressure of
    saturated water at 300 C
12 f_sat = 6738.9; // [kPa] - Fugacity of saturated water
    at 300 C
13 V_liq = 25.28; // [cm^(3)/mol] - Molar volume of water
    in liquid phase
14 V_liq = V_liq*10^(-6); // [m^(3)/mol]
15 V_vap = 391.1; // [cm^(3)/mol] - Molar volume of water
    in vapour phase
16 V_vap = V_vap*10^(-6); // [m^(3)/mol]
17 R = 8.314; // [J/mol*K] - Universal gas constant
18
19 // At 300 C and 9000 kPa water is a compressed
    liquid and its fugacity is given by
20 //  $f = f_{sat} \exp[V_{liq}(P - P_{sat})/R*T]$ 
21 fugacity = f_sat*exp((V_liq*(P - P_sat)*1000)/(R*T))
    ;
22
23 printf(" The fugacity of water at 9000 kPa is %f kPa
    ", fugacity);
```

Scilab code Exa 7.39 Estimation of fugacity of saturated steam

```
1 clear;
2 clc;
3
4 //Example - 7.39
5 //Page number - 276
6 printf("Example - 7.39 and Page number - 276\n\n");
7
8 //Given
9 T = 200 + 273.15; //[K] - Temperature
10 R = 8.314; //[J/mol*K] - Universal gas constant
11
12 // From steam table at 200 C as reported in the book
13 P_sat = 1.5538; //[MPa] - Vapour pressure of
    saturated steam
14 H_vap = 2793.2; //[kJ/kg] - Enthalpy of saturated
    steam in vapour phase
15 S_vap = 6.4323; //[kJ/kg-K] - Entropy of saturated
    steam in vapour phase
16 G_sat = H_vap - T*S_vap; //[kJ/kg] - Gibbs free
    energy
17 G_sat = G_sat*18.015; //[J/mol]
18
19 // Now let us calculate the Gibbs free energy at the
    lowest pressure available in superheated steam
    tables at 200 C
20 // At 200 C and 0.01 MPa as reported in the book
21 H = 2879.5; //[kJ/kg] - Enthalpy
22 S = 8.9038; //[kJ/kg-K] - Entropy
23 G_ig = H - T*S; //[kJ/kg] - Gibbs free energy
24 G_ig = G_ig*18.015; //[J/mol]
25
26 // Integrating from ideal gas state at 200 C and
```

```

    0.01 MPa to saturated vapour at 200 C we get
27 //  $G_{\text{sat}} - G_{\text{ig}} = R \cdot T \cdot \log(f_{\text{sat}}/f_{\text{ig}})$ 
28
29 // Under the ideal gas condition the pressure is
    small therefore  $f_{\text{ig}} = P = 0.01$  MPa
30  $f_{\text{ig}} = 0.01$ ; // [MPa]
31
32 // Solving the above equation
33  $f_{\text{sat}} = f_{\text{ig}} \cdot (\exp((G_{\text{sat}} - G_{\text{ig}})/(R \cdot T)))$ ; // [MPa]
34
35 printf(" The fugacity of saturated steam at 200 C is
    %f MPa",  $f_{\text{sat}}$ );

```

Scilab code Exa 7.40 Estimation of fugacity of steam

```

1 clear;
2 clc;
3
4 // Example - 7.40
5 // Page number - 277
6 printf(" Example - 7.40 and Page number - 277\n\n");
7
8 // Given
9  $T = 320 + 273.15$ ; // [K]
10  $P_1 = 70$ ; // [bar]
11  $P_2 = 170$ ; // [bar]
12  $R = 8.314$ ; // [J/mol*K] - Universal gas constant
13
14 // (a)
15 //  $dG = R \cdot T \cdot d \log(f)$ 
16 //  $G - G_{\text{ig}} = R \cdot T \cdot \log(f/f_{\text{ig}})$ 
17
18 // From steam table the low pressure that is
    available is 1 kPa.
19  $f_{\text{ig}} = 1$ ; // [kPa] - Assuming ideal gas behaviour at

```

```

    such low pressure
20
21 // At 1 kPa (under saturated conditions)
22 P_sat = 112.891;//[bar]
23 // Therefore at both 1 kPa and 70 bar the stem is
    superheated and byond a pressure of 112.891 bar
    it is compressed liquid.
24
25 // For superheated steam table at 1 kPa and 320 C,
    as repoted in the book
26 H_1 = 3117.08;//[kJ/kg] - Enthalpy
27 S_1 = 10.41232;//[kJ/kg-K] - Entropy
28
29 // For superheated steam table at 70 bar and 320 C,
    as repoted in the book
30 H_2 = 2916.92;//[kJ/kg] - Enthalpy
31 S_2 = 6.0651;//[kJ/kg-K] - Entropy
32
33 // At 70 bar and 320 C,
34 G = H_2 - T*S_2;//[kJ/kg] - Gibbs free energy
35 // At 1 kPa and 320 C
36 G_ig = H_1 - T*S_1;//[kJ/kg] - Gibbs free energy
37
38 //  $\log(f/f_{ig}) = (G - G_{ig})/(R*T)$ 
39 f = f_ig*(exp((G - G_ig)*18/(R*T)));//[kPa]
40 f = f*10^(-2);//[bar]
41
42 // At 70 bar
43 phi = f/P_1;
44
45 printf(" (a).The fugacity of steam at 320 C and 70
    bar is %f bar\n",f);
46 printf("      The fugacity coefficient at 320 C and
    70 bar is , phi = %f\n\n",phi);
47
48 //(b)
49 // Now consider saturated steam at 320 C. We have
50 P_sat = 112.891;//[bar]

```

```

51 V_liquid = 1.5;//[cm^(3)/mol] - Molar vlume of
    saturated liquid
52 V_liquid = V_liquid*10^(-6);//[m^(3)/mol]
53 V_vapour = 15.48;//[cm^(3)/mol] - Molar vlume of
    saturated vapour
54 U_liquid = 1445.7;//[Kj/kg] - Internal energy of
    satuarted liquid
55 U_vapour = 2528.9;//[kJ/kg] - Internal energy of
    satuarted vapour
56 H_liquid = 1462.6;//[kJ/kg] - Enthalpy of saturated
    liquid
57 H_vapour = 2703.7;//[kJ/kg] - Enthalpy of saturated
    vapour
58 S_liquid = 3.45;//[kJ/kg-K] - Entropy of saturated
    liquid
59 S_vapour = 5.5423;//[kJ/kg-K] - Entropy of saturated
    vapour
60
61 // Now let us calculate Gibbs free energy of
    saturated liquid and saturated vapour
62 G_liquid = H_liquid - T*S_liquid;//[kJ/kg]
63 G_vapour = H_vapour - T*S_vapour;//[kJ/kg]
64 // Note that under saturated conditions
65 // G_sat = G_liquid = G_vapour
66 G_sat = G_liquid;//[kJ/kg]
67
68 // log(f_sat/f_ig) = (G_sat - G_ig)/(R*T)
69 f_sat = f_ig*(exp((G_sat - G_ig)*18/(R*T)));//[kPa]
70 f_sat = f_sat*10^(-2);//[bar]
71
72 phi_sat = f_sat/P_sat;
73
74 // And now the fugacity is to be determined at 320 C
    and P = 170 bar. We know the following relation
    for compressed liquid.
75 // f_CL = f_sat*exp(V_liquid*(P-P_sat)/(R*T))
76 f_CL = f_sat*exp(V_liquid*18*(P_2-P_sat)*10^(5)/(R*T
    ));//[bar]

```

```

77
78 // Therefore the fugacity coefficient at 170 bar and
    320 C is given by
79 phi_2 = f_CL/P_2;
80
81 printf(" (b).The fugacity of steam at 320 C and 170
    bar is %f bar\n",f_CL);
82 printf("      The fugacity coefficient at 320 C and
    170 bar is , phi = %f\n\n",phi_2);

```

Scilab code Exa 7.41 Determination of fugacities at two states

```

1 clear;
2 clc;
3
4 //Example - 7.41
5 //Page number - 278
6 printf("Example - 7.41 and Page number - 278\n\n");
7
8 //Given
9 T = 300 + 273.15; // [K]
10 P_1 = 12500*10^(3); // [Pa]
11 P_2 = 8581*10^(3); // [Pa]
12 P_3 = 300*10^(3); // [Pa]
13 V_liq = 1.404; // [cm^(3)/g] - Specific volume of
    liquid
14 V_liq = (V_liq/10^(6))*18.015; // [m^(3)/mol]
15 R = 8.314; // [J/mol*K] - Universal gas constant
16
17 // state 1: 300 C, 12500 kPa
18 // state 2: 300 C, 8581 kPa
19 // state 3: 300 C, 300 kPa
20
21 // From state 1 to state 2 the system is liquid
    water and if the molar volume of liquid is

```

```

    assumed constant we can write
22 //  $G_2 - G_1 = V_{liq}*(P_2 - P_1)$ 
23 //  $G_2 - G_1 = R*T*log(f_2/f_1)$ 
24 // Comparing the above two equations we get
25 //  $(f_2/f_1) = exp((V_{liq}*(P_2 - P_1)/(R*T))$ 
26 f2_f1 = exp((V_liq*(P_2 - P_1)/(R*T))); // (f_2/f_1)
    = f2_f1 (say)
27
28 // In state 2 the fugacity of liquid is same as that
    of saturated vapour and for the vapour phase
    change from state 2 to 3 the fugacity ratio is
    calculated using
29 //  $G_3 - G_2 = R*T*log(f_3/f_2)$ 
30
31 // At 300 C, 8581 kPa
32 H_liq_2 = 2749.0; // [kJ/kg]
33 S_vap_2 = 5.7045; // [kJ/kg-K]
34 G_vap_2 = -520.53; // [kJ/kg]
35 G_vap_2 = G_vap_2*18.015; // [J/mol]
36
37 // At 300 C, 300 kPa
38 H_3 = 3069.3; // [kJ/kg]
39 S_3 = 7.7022; // [kJ/kg-K]
40 G_3 = -1345.22; // [kJ/kg]
41 G_3 = G_3*18.015; // [J/mol]
42
43 // Substituting and solving the equation  $G_3 - G_2$ 
    =  $R*T*log(f_3/f_2)$ 
44 f3_f2 = exp((G_3 - G_vap_2)/(R*T)); // (f_3/f_2) =
    f3_f2 (say)
45
46 //  $(f_3/f_1) = (f_3/f_2)*(f_2/f_1)$ 
47 f3_f1 = f3_f2*f2_f1;
48
49 printf(" The ratio of fugacity in the final state to
    that in the initial state is given by f3/f2 = %f
    ", f3_f2);

```

Chapter 8

Thermodynamic Cycles

Scilab code Exa 8.1 Calculation of work done

```
1 clear;
2 clc;
3
4 //Example - 8.1
5 //Page number - 287
6 printf("Example - 8.1 and Page number - 287\n\n");
7
8 //Given
9 P_1 = 30;//[bar]
10 P_2 = 0.04;//[bar]
11
12 //(1).Carnot cycle
13 //It has been reported in the book that at 30 bar
    pressure (saturated) :
14 H_liq_1 = 1008.42;//[kJ/kg]
15 H_vap_1 = 2804.2;//[kJ/kg]
16 S_liq_1 = 2.6457;//[kJ/kg-K]
17 S_vap_1 = 6.1869;//[kJ/kg-K]
18 //Therefore , H_1 = H_liq_1 , H_2 = H_vap_1 , S_1 =
    S_liq_1 and S_2 = S_vap_1
19 H_1 = H_liq_1;
```



```

20 H_2 = H_vap_1;
21 S_1 = S_liq_1;
22 S_2 = S_vap_1;
23
24 //At 0.04 bar pressure (saturated) :
25 H_liq_2 = 121.46;//[kJ/kg]
26 H_vap_2 = 2554.4;//[kJ/kg]
27 S_liq_2 = 0.4226;//[kJ/kg-K]
28 S_vap_2 = 8.4746;//[kJ/kg-K]
29
30 //Dryness fraction at state 3 can be found the fact
    that S_3 = S_2
31 x_3 = (S_2 - S_liq_2)/(S_vap_2 - S_liq_2);
32 H_3 = H_liq_2*(1 - x_3) + x_3*H_vap_2;//[kJ/kg]
33
34 //Dryness fraction at state 4 can be found the fact
    that S_4 = S_1
35 x_4 = (S_1 - S_liq_2)/(S_vap_2 - S_liq_2);
36 H_4 = H_liq_2*(1 - x_4) + x_4*H_vap_2;//[kJ/kg]
37
38 //Work done by turbine W_tur = -delta.H = -(H_3 -
    H_2)
39 W_tur = H_2 - H_3;//[kJ/kg]
40
41 //Work supplied by boiler ,
42 q_H = H_2 - H_1;//[kJ/kg]
43
44 //Work transfer in compressor is given by
45 W_com = -(H_1 - H_4);//[kJ/kg]
46
47 //Efficiency can now be calculated as
48 //n = (Net work done/Work supplied by boiler)
49 n_carnot = (W_tur + W_com)/q_H;
50
51 //Efficiency of the Carnot cycle can also be
    determined from the formula
52 // n = 1 - (T_L/T_H), Where T_L is saturated
    temperature at 0.04 bar and T_H is saturated

```

```

    temperature at 30 bar
53
54 printf("(1).Carnot cycle\n\n");
55 printf("The work done by the turbine is %f kJ/kg\n\n
    ",W_tur);
56 printf("The heat transfer in the boiler is %f kJ/kg\
    n\n",q_H);
57 printf("The cycle efficiency is %f\n\n\n",n_carnot);
58
59 //(2).Rankine cycle
60 //The enthalpies at state 2 and 3 remain as in the
    Carnot cycle
61 //Saturated liquid enthalpy at 0.04 bar is
62 H_4_prime = H_liq_2;
63
64 //Saturated liquid volume at 0.04 bar as reported in
    the book is
65 V_liq = 0.001004; //[m^(3)/kg]
66 //Work transfer in pump can be calculated as
67 W_pump = -V_liq*(P_1 - P_2)*100; //[kJ/kg]
68
69 //Work transfer around pump gives , W_pump = -delta_H
    = -(H_1_prime - H_4_prime);
70 H_1_prime = H_4_prime - W_pump; //[kJ/kg]
71
72 //Heat supplied to boiler is
73 q_H_prime = H_2 - H_1_prime; //[kJ/kg]
74
75 //Work done by turbine is
76 W_tur_prime = H_2 - H_3; //[kJ/kg]
77
78 //Efficiency can now be calculated as
79 //n = (Net work done/Heat input)
80 n_rankine = (W_tur_prime + W_pump)/q_H_prime; //
81
82 printf("(2).Rankine cycle\n\n");
83 printf("The work done by the turbine is %f kJ/kg\n\n
    ",W_tur_prime);

```

```

84 printf("The heat transfer in the boiler is %f kJ/kg\
    n\n",q_H_prime);
85 printf("The cycle efficiency is %f",n_rankine);

```

Scilab code Exa 8.2 Calculation of efficiency of Rankine cycle

```

1 clear;
2 clc;
3
4 //Example - 8.2
5 //Page number - 288
6 printf("Example - 8.2 and Page number - 288\n\n");
7
8 //Given
9 T_max = 700+273.15; //[K] - Maximum temperature.
10 P_boiler = 10*10^(6); //[Pa] - Constant pressure in
    the boiler
11 P_condenser = 10*10^(3); //[Pa] - Constant pressure
    in the condenser
12
13 //At state 2 i.e, at 700 C and 10 MPa, it has been
    reported in the book that from steam table
14 S_2 = 7.1687; //[kJ/kg-K] - Entropy
15 H_2 = 3870.5; //[kJ/kg] - Enthalpy
16
17 //At state 3 i.e, at 700 C and 10 KPa,
18 S_3 = S_2; //[kJ/kg-K] - Entropy
19
20 //For saturated steam at 10 kPa, it has been reported
    in the book that from steam table
21 S_liq = 0.6493; //[kJ/kg-K] - Entropy of saturated
    liquid
22 S_vap = 8.1502; //[kJ/kg-K] - Entropy of saturated
    liquid
23 //Therefore steam is saturated and its dryness

```

```

        factor can be calculated as
24 x = (S_2 - S_liq)/(S_vap - S_liq);
25
26 //The enthalpy at state 3 is now calculated. For
    steam at 10 kPa, it has been reported in the book
    that from steam table
27 H_liq = 191.83; // [kJ/kg]
28 H_vap = 2584.7; // [kJ/kg]
29 //Therefore enthalpy at state 3 is
30 H_3 = H_liq*(1-x) + H_vap*x; // [kJ/kg]
31
32 //Work done by the turbine
33 W_tur = -(H_3 - H_2); // [kJ/kg]
34
35 //Now we have to calculate work input to the pump
36 //State 4: Saturated liquid at 10 kPa
37 //State 4: Compressed liquid at 10 MPa
38 //Since volume of liquid does not get affected by
    pressure we take volume of saturated liquid at 10
    kPa,
39 V_liq = 0.001010; // [m3/kg]
40
41 //Work transfer in the pump is
42 W_pump = -V_liq*(P_boiler - P_condenser)*10(-3); // [
    kJ/kg]
43
44 //Energy balance around pump gives , W_pump = -
    delta_H = -(H_1 - H_4)
45 H_4 = H_liq; // Enthalpy at state 4 (saturated liquid
    at 10 kPa)
46 H_1 = H_4 - W_pump; // [kJ/kg]
47
48 //Heat supplied to boiler is
49 q_H = H_2 - H_1; // [kJ/kg]
50
51 //Efficiency can now be calculated as
52 //n = (Net work done/Heat input)
53 n_rankine = (W_tur + W_pump)/q_H;

```

```

54
55 printf("The efficiency of the Rankine cycle is found
        to be %f",n_rankine);
56
57 //Now let us determine the efficiency of Carnot
        cycle. The maximum temperature is 700 C and
        minimum temperature is that of saturated steam at
        10 kPa,
58 T_min = 45.81 + 273.15;//[K] - From steam table as
        reported in the book
59 n_carnot = 1-(T_min/T_max);
60 //Note that the efficiency of Rankine cycle is less
        than that of carnot cycle.

```

Scilab code Exa 8.3 Calculation of COP of carnot refrigerator and heat rejected

```

1 clear;
2 clc;
3
4 //Example - 8.3
5 //Page number - 291
6 printf("Example - 8.3 and Page number - 291\n\n");
7
8 //Given
9 W = 1.1;//[kW] - Work done per ton of refrigeration
10 //1 ton refrigeration = 3.517 kW, therefore
11 H = 3.517;//[kW] - Heat absorbed
12 T_low = -30 + 273.15;//[K] - Low temperature
        maintained
13
14 //COP can be calculated as
15 //COP = (Heat absorbed/Work done)
16 COP = H/W;
17

```

```

18 //For reversed carnot cycle , COP = T_low/(T_high -
    T_low). Solving this we get
19 T_high = (T_low/COP) + T_low;//[K] - Higher
    temperature
20
21 //Heat rejected is
22 H_rej = W + H;//[kW];
23
24 printf("The COP is %f\n\n",COP);
25 printf("The higher temperature of the cycle is %f K\
    n\n",T_high);
26 printf("The heat rejected per ton of refrigeration
    is %f kW\n\n",H_rej);

```

Scilab code Exa 8.4 Calculation of minimum power required

```

1 clear;
2 clc;
3
4 //Example - 8.4
5 //Page number - 292
6 printf("Example - 8.4 and Page number - 292\n\n");
7
8 //Given
9 T_high = 20 + 273.15;//[K] - High temperature
10 T_low = 0 + 273.15;//[K] - Low temperature
11 Q_H = 10;//[kW] - Heat supplied
12
13 //If 'Q_H' is the rate at which heat is taken from
    surrounding and 'W' is the rate at which work is
    done , then
14 // Q_H = W + Q_L
15 //(Q_H/Q_L) = (T_high/T_low)
16 //Also for a reversible cycle , (Q_H/Q_L) = 1 + (W/
    Q_L). Solving we get ,

```

```

17 Q_L = (T_low/T_high)*Q_H; // [kW]
18 W = (Q_H - Q_L) ; // [kW]
19
20 printf("The minimum power required is %f kW",W);

```

Scilab code Exa 8.5 Determination of COP and power required

```

1 clear;
2 clc;
3
4 //Example - 8.5
5 //Page number - 292
6 printf("Example - 8.5 and Page number - 292\n\n");
7
8 //Given
9 T_high = 40 + 273.15; // [K] - High temperature
10 T_low = -20 + 273.15; // [K] - Low temperature
11 C = 10; // [tons of refrigeration] - Capacity
12 //1 ton refrigeration = 3.517 kW, therefore
13 H = C*3.517; // [kW] - Heat absorbed
14
15 //For reversed carnot cycle , COP = T_low/(T_high -
    T_low)
16 COP = T_low/(T_high - T_low);
17
18 // COP = (Refrigerating effect)/(Work input),
    therefore power required is given by
19 P = (H/COP); // [kW]
20
21 printf("The COP is %f\n\n",COP);
22 printf("The power required is %f kW",P);

```

Scilab code Exa 8.6 Determination of maximum refrigeration effect

```

1 clear;
2 clc;
3
4 //Example - 8.6
5 //Page number - 292
6 printf("Example - 8.6 and Page number - 292\n\n");
7
8 //Given
9 COP = 4; //Coefficient of performance
10 P = 10; // [kW] - Work done on the cycle
11
12 //For reversed carnot cycle, COP = T_low/(T_high -
    T_low)
13 //ratio = (T_high/T_low), therefore
14 ratio = -1/(COP + 1);
15
16 // Refrigerating effect = (COP)*Work input,
    therefore refrigeration is given by
17 H = COP*P; // [kW]
18
19 //Maximum refrigearation in tons is given by
20 H_max = (H/3.517);
21
22 printf("The maximum refrigeration value is %f ton",
    H_max);

```

Scilab code Exa 8.7 Determination of refrigeration effect power consumed and COP of refrigerator

```

1 clear;
2 clc;
3
4 //Example - 8.7
5 //Page number - 292
6 printf("Example - 8.7 and Page number - 292\n\n");

```



```

7
8 //Given
9 m = 0.6;//[kg/s] - mass flow rate
10 T_low = -20+273.15;//[K] - Temperature at which
    vapour enters the compressor
11 T_high = 30+273.15;//[K] - Temperature at which
    vapour leaves the condenser
12
13 //From saturated refrigeration-12 tables we get, at
    -20 C
14 H_1 = 178.74;//[kJ/kg] - (H_1 = H_vap)
15 P_1 = 0.15093;//[MPa] - (P_1 = P_sat)
16 P_4 = P_1;
17 S_1 = 0.7087;//[kJ/kg-K] - (S_1 = S_vap)
18 S_2 = S_1;
19
20 //At 30 C
21 P_2 = 0.7449;//[MPa] - (P_2 = P_sat)
22 P_3 = P_2;
23 H_3 = 64.59;//[kJ/kg] - (H_3 = H_liq)
24 H_4 = H_3;
25 S_3 = 0.24;//[kJ/kg-K] - (S_3 = S_liq)
26
27 //It is assumed that pressure drop in the
    evaporator and condenser are negligible. The heat
    transfer rate in the evaporator is
28 Q_L = m*(H_1 - H_4);
29
30 printf("The heat transfer rate in the evaporator is
    %f kW\n\n",Q_L);
31
32 //At state 2 (P = 0.7449 MPa and S = 0.7087 kJ/kg-K)
    and looking in the superheated tables we have to
    calculate the enthalpy at state 2
33
34 //At P = 0.7 MPa and S = 0.6917 kJ/kg-K,
35 H_11 = 200.46;//[kJ/kg]
36

```

```

37 //At P = 0.7 MPa and S = 0.7153 kJ/kg-K,
38 H_12 = 207.73; // [kJ/kg]
39
40 //Thus at P = 0.7 MPa and S = 0.7087 kJ/kg-K,
    enthalpy is given by
41 H_13 = ((S_2 - 0.6917)/(0.7153 - 0.6917))*(H_12 -
    H_11) + H_11; // [kJ/kg]
42
43 //At P = 0.8 MPa and S = 0.7021 kJ/kg-K,
44 H_21 = 206.07; // [kJ/kg]
45
46 //At P = 0.8 MPa and S = 0.7253 kJ/kg-K,
47 H_22 = 213.45; // [kJ/kg]
48
49 //Thus at P = 0.8 MPa and S = 0.7087 kJ/kg-K,
    enthalpy is given by
50 H_23 = ((S_2 - 0.7021)/(0.7253 - 0.7021))*(H_22 -
    H_21) + H_21; // [kJ/kg]
51
52 //At P = 0.7449 MPa, S = 0.7087 kJ/kg-K, the
    enthalpy is
53 H_2 = ((0.7449 - 0.7)/(0.8 - 0.7))*(H_23 - H_13) +
    H_13; // [kJ/kg]
54
55 //Power consumed by the compressor is
56 W_comp = m*(H_2 - H_1); // [kW]
57
58 printf("The power consumed by the compressor is %f
    kW\n\n", W_comp);
59
60 //Heat removed in evaporator/work done on compressor
61 COP_R = Q_L/W_comp;
62
63 printf("The COP the refrigerator is %f kW\n\n", COP_R
    );
64
65
66 //At -20 C, saturated conditions

```

```

67 H_liq = 17.82; //[kJ/kg]
68 H_vap = 178.74; //[kJ/kg]
69 x_4 = (H_4 - H_liq)/(H_vap - H_liq);
70
71 printf("The dryness factor of refrigerant after the
        expansion valve is %f\n\n",x_4);
72
73 //The heat transfer rate in the condenser is
74 Q_H = m*(H_3 - H_2); //[kW]
75
76 printf("The heat transfer rate in the condenser is
        %f kW\n\n",Q_H);
77
78 //If the cycle would have worked as a pump then,
79 //COP_HP = (Heat supplied from condenser/Work done
        on compressor)
80 COP_HP = (-Q_H)/W_comp;
81
82 printf("The COP if cycle would work as a heat pump
        is %f kW\n\n",COP_HP);
83
84 //If the cycle would have been a reversed Carnot
        cycle then
85 COP_C = T_low/(T_high - T_low);
86
87 printf("The COP if cycle would run as reversed
        Carnot cycle is %f kW\n\n",COP_C);

```

Scilab code Exa 8.8 Calculation of amount of air

```

1 clear;
2 clc;
3
4 //Example - 8.8
5 //Page number - 300

```

```

6 printf("Example - 8.8 and Page number - 300\n\n");
7
8 //Given
9 //From compressor to the expansion valve the
    pressure is 200 bar and from expansion valve to
    the inlet of compressor the pressure is 1 bar.
10 //Point 5 is saturated liquid at 1 bar and point 6
    is saturated vapour at 1 bar
11
12 //It has been reported in the book that at state 1
    (310 K, 1 bar)
13 H_1 = 310.38;//[kJ/kg]
14 //At state 2 (310 K, 200 bar)
15 H_2 = 277.7;//[kJ/kg]
16 //At state 5 (1 bar, saturated liquid)
17 H_5 = -122.6;//[kJ/kg]
18 //At state 6 (1 bar, saturated vapour)
19 H_6 = 77.8;//[kJ/kg]
20
21 //The enthalpy at point 3 is same at point 4 as the
    expansion is isenthalpic
22
23 //The mass condensed is 1 kg and therefore m_1 = m_6
    + 1
24
25 //Enthalpy balance around heat exchanger
26 //m_2*H_2 + m_2*H_6 = m_3*H_3 + m_7*H_7
27
28 //Enthalpy balance around separator
29 //m_4*H_4 = m_5*H_5 + m_6*H_6
30 //It can be seen that m_1 = m_2 = m_3 = m_4
31 //and m_6 = m_7 = m_1 - 1
32
33 //Substituting the values for enthalpy balance
    around heat exchanger we get ,
34 //m_1*H_2 + (m_1 - 1)*(H_6) = m_1*H_3 + (m_1 - 1)*
    H_1
35 //and substituting the values for enthalpy balance

```

```

    around separator we get
36 //m_1*H_3 = (1)*(-122.6) + (m_1 - 1)*77.8
37 //H_3 = ((1)*(-122.6) + (m_1 - 1)*77.8)/m_1
38 //Substituting the expression for 'H_3' in the above
    equation and then solving for m_1, we get
39 def f(' [y]=f(m_1) ', 'y=m_1*H_2+(m_1-1)*(H_6)-m_1*((1)
    *(-122.6) + (m_1 - 1)*77.8)/m_1)-(m_1-1)*H_1 ');
40 m_1 = fsolve(4,f);//[kg]
41 //Thus to liquify 1 kg of air compression of m_1 kg
    of air is carried out.
42
43 //Now substituting this value of m_1 to get the
    value of H_3,
44 H_3 = ((1)*(-122.6) + (m_1 - 1)*77.8)/m_1;//[kJ/kg]
45
46 //From given compressed air table we see at 200 bar
    and 160 K,
47 H_3_1 = 40.2;//[kJ/kg]
48
49 //At 200 bar and 180 K,
50 H_3_2 = 79.8;//[kJ/kg]
51 //By interpolation we get,
52 T_3 = ((H_3 - H_3_1)*(180 - 160))/(H_3_2 - H_3_1) +
    160;//[K]
53
54 printf("Temperature before throttling is %f",T_3);

```

Scilab code Exa 8.9 Calculation of amount of air and temperature

```

1 clear;
2 clc;
3 funcprot(0);
4
5 //Example - 8.9
6 //Page number - 304

```

```

7 printf("Example - 8.9 and Page number - 304\n\n");
8
9 //Given
10 //At 1 bar, 310 K
11 H_1 = 310.38; //[kJ/kg]
12 //At 200 bar, 310 K
13 H_2 = 277.7; //[kJ/kg]
14 //At 1 bar, Saturated liquid
15 H_7 = -122.6; //[kJ/kg]
16 //At 1 bar, Saturated vapour
17 H_8 = 77.8; //[kJ/kg]
18 //At 200 bar, 200 K
19 H_3 = 117.6; //[kJ/kg]
20 //At 1 bar, 100 K
21 H_11 = 98.3; //[kJ/kg]
22
23 //(1)
24 //For 1 kg of liquid air obtained, the overall
    enthalpy balance is
25 //m_2*H_2 = W - 122.6 + (m_2 - 1)*H_1
26 //W = - 0.8*m_2*(H_11 - H_3)
27 //Overall enthalpy balance equation becomes
28 //H_2*m_2 = 15.44*m_2 - H_7 + (m_2 - 1)*H_1, solving
29 m_2_prime = (H_7 - H_1)/(H_2 - 15.44 - H_1);
30
31 printf("The number of kilograms of air compressed
    per kg of liquid air produced is %f kg\n\n",
    m_2_prime);
32
33 //(2)
34 //Enthalpy balance around separator is
35 //0.2*m_2*H_5 = -H_7 + (0.2*m_2 - 1)*H_8, solving
36 m_2 = m_2_prime;
37 H_5_prime = ((0.2*m_2-1)*H_8 - H_7)/(0.2*m_2);
38
39 //At point 5, P = 200 bar and enthalpy is
40 H_5_1 = -33.53; //[kJ/kg]
41 //From compressed air tables at 200 bar and 140 K,

```

```

42 H_5_2 = 0.2; // [kJ/kg]
43 //At 200 bar and 120 K,
44 H_5_3 = -38.0; // [kJ/kg]
45 //Solving by interpolation we get
46 T_5 = ((H_5_1 - H_5_3)*(140 - 120))/(H_5_2 - H_5_3)
      + 120; // [K]
47
48 printf("The temperature of air before throttling is
      %f K\n\n", T_5);
49
50 //(3)
51 //During mixing of streams 8 and 11 to produce
      stream 9, the enthalpy balance is
52 // (0.2*m_2 - 1)*H_8 + 0.8*m_2*H_11 = (m_2 - 1)*H_9,
      Solving for H_9
53
54 H_9_prime = ((0.2*m_2-1)*H_8+0.8*m_2*H_11)/(m_2 - 1)
      ;
55
56 //From given compressed air tables at 1 bar and 100
      K,
57 H_9_1 = H_11;
58 //At 1 bar and 90 K
59 H_9_2 = 87.9; // [kJ/kg]
60 //Solving by interpolation we get
61 T_9 = ((H_9_prime - H_9_2)*(100 - 90))/(H_9_1 -
      H_9_2) + 90; // [K]
62
63 printf("The temperature of stream entering second
      heat exchanger is %f K\n\n", T_9);
64
65 //(4)
66 //Enthalpy balance around first heat exchanger is
67 //H_2*m_2 + (m_2 - 1)*H_10 = H_3*m_2 + (m_2 - 1)*H_1
      , solving for H_10
68
69 H_10_prime = ((m_2 - 1)*H_1 + H_3*m_2 - H_2*m_2)/(
      m_2 - 1);

```

```

70
71 //From given compressed air tables at 1 bar and 140
    K,
72 H_10_1 = 139.1; // [kJ/kg]
73 //At 1 bar and 120 K
74 H_10_2 = 118.8; // [kJ/kg]
75 //Solving by interpolation we get
76 T_10 = ((H_10_prime - H_10_2)*(140 - 120))/(H_10_1 -
    H_10_2) + 120; // [K]
77
78 printf("The temperature of stream exiting second
    heat exchanger is %f K\n\n",T_10);

```

Scilab code Exa 8.10 Determination of temperature of air

```

1 clear;
2 clc;
3 funcprot(0);
4
5 //Example - 8.10
6 //Page number - 307
7 printf("Example - 8.10 and Page number - 307\n\n");
8
9 //Given
10 P_high = 40; // [bar]
11 P_low = 5; // [bar]
12 m_1 = 0.5; // [kg/s] - Rate of mass moving through the
    expander
13 m_2 = 0.1; // [kg/s] - Rate of mass of vapour mixing
    with air
14 e = 0.7; // Efficiency
15
16 //At state 3,(40 bar and 200 K),enthalpy and entropy
    is given by
17 H_3 = 179.7; // [kJ/kg]

```



```

18 S_3 = 5.330; //[kJ/kg-K]
19
20 //If isentropic conditions exists in the turbine then
    state 11 is at 5 bar
21 S_11 = 5.330; //[kJ/kg-K]
22 //From given compressed air tables at 5 bar and 120
    K,
23 H_11_1 = 113.6; //[kJ/kg]
24 S_11_1 = 5.455; //[kJ/kg-K]
25 //At 5 bar and 100 K
26 H_11_2 = 90.6; //[kJ/kg]
27 S_11_2 = 5.246; //[kJ/kg-K]
28 //The enthalpy has to be determined when S = S_3
29 //Solving by interpolation we get
30 H_11_s = ((H_11_1 - H_11_2)*(S_3 - S_11_2))/(S_11_1
    - S_11_2) + H_11_2; //[kJ/kg]
31
32 //The adiabatic efficiency of tyrbine is given by
33 //(H_3 - H_11_a)/(H_3 - H_11_s) = e
34 H_11_a = H_3 - e*(H_3 - H_11_s); //[kJ/kg] - Actual
    enthalpy
35
36 //At 5 bar, the saturated enthalpy is given to be
37 H_8 = 88.7; //[kJ/kg]
38 //From enthalpy balance during mixing we get ,
39 //0.1*H_8 + 0.5*H_11_a = 0.6*H_9
40 H_9 = (m_2*H_8 + m_1*H_11_a)/(m_1 + m_2); //[kJ/kg]
41
42 //From given compressed air tables at 5 bar and 140
    K,
43 H_9_1 = 135.3; //[kJ/kg]
44 //At 5 bar and 120 K
45 H_9_2 = 113.6; //[kJ/kg]
46 //By interpolation we get
47 T_9 = ((H_9 - H_11_1)*(140 - 120))/(H_9_1 - H_11_1)
    + 120; //[K]
48
49 printf(" The temperature of air entering the second

```

heat exchanger is %f K\n\n",T_9);

Chapter 10

Residual Properties by Equations of State

Scilab code Exa 10.1 Determination of expression for residual enthalpy internal energy and Gibbs free energy

```
1 clear;
2 clc;
3
4 //Example - 10.1
5 //Page number - 323
6 printf("Example - 10.1 and Page number - 323\n\n");
7
8 //This problem involves proving a relation in which
   no mathematical components are involved.
9 //For prove refer to this example 10.1 on page
   number 323 of the book.
10 printf("This problem involves proving a relation in
   which no mathematical components are involved.\n\n");
11 printf("For prove refer to this example 10.1 on page
   number 323 of the book.")
```

Scilab code Exa 10.2 Preparation of fugacity and fugacity coefficient

```
1 clear;
2 clc;
3
4 //Example - 10.2
5 //Page number - 334
6 printf("Example - 10.2 and Page number - 334\n\n");
7
8 // Given
9 T = 40 + 273.15; //[C] - Temperature
10 P_1 = 0; //[bar]
11 P_2 = 10; //[bar]
12 V_liq = 90.45; //[cm^(3)/mol]
13 V_liq = V_liq*10^(-6); //[m^(3)/mol]
14 P_sat = 4.287; //[bar]
15
16 // For butadiene
17 T_c = 425.0; //[K] - Critical temperature
18 P_c = 43.3; //[bar] - Critical pressure
19 P_c = P_c*10^(5); //[N/m^(2)]
20 w = 0.195; // Acentric factor
21 R = 8.314; //[J/mol*K] - Universal gas constant
22
23 // Let us calculate second virial coefficient at 40
   C
24 Tr = T/T_c; // Reduced temperature
25 B_0 = 0.083-(0.422/(Tr)^(1.6));
26 B_1 = 0.139-(0.172/(Tr)^(4.2));
27 //We know, (B*Pc)/(R*Tc) = B_0+(w*B_1)
28 B = ((B_0 + (w*B_1))*(R*T_c))/P_c; //[m^(3)/mol] -
   Second virial coefficient
29
30 // log(f/P) = (B*P)/(R*T)
```

```

31 // f = P*exp((B*P)/(R*T))
32
33 printf(" The table is as follows\n\n")
34 printf(" P(bar) \t\t f(bar) \t\t phi\n");
35
36 P = [1,2,3,4,4.287,5,6,8,10];
37 f = zeros(9);
38 phi = zeros(9);
39 for i=1:5;
40     f(i)=P(i)*(exp((B*P(i)*10^(5))/(R*T)));//[bar]
        // Pressure inside the exponential term has
        to be in N/m^(2)
41     phi(i)= (f(i)/P(i));
42     printf(" %f \t %f \t\t\t %f\n",P(i),f(i),phi(i));
43 end
44 f_sat = f(5);
45
46 // From pressure of 4.287 bar onwards the valid
    equation to compute fugacity of compressed liquid
    is given by
47 // f = f_sat*exp[V_liq*(P-P_sat)/(R*T)]
48
49 for j=6:9
50     f(j) = f_sat*exp((V_liq*(P(j)-P_sat)*10^(5))/(R*
        T));//[bar] // Pressure inside the
        exponential term has to be in N/m^(2)
51     phi(j) = f(j)/P(j);
52     printf(" %f \t %f \t\t\t %f\n",P(j),f(j),phi(j))
        ;
53 end

```

Scilab code Exa 10.3 Calculation of enthalpy entropy and internal energy change

```
1 clear;
```

```

2  clc;
3
4  //Example - 10.3
5  //Page number - 334
6  printf("Example - 10.3 and Page number - 334\n\n");
7
8  //Given
9  n = 100;//[mol] - No of moles
10 T_1 = 600;//[K] - Initial temperature
11 T_2 = 300;//[K] - Final temperature
12 P_1 = 10;//[atm] - Initial pressure
13 P_1 = P_1*101325;//[Pa]
14 P_2 = 5;//[atm] - Final pressure
15 P_2 = P_2*101325;//[Pa]
16 Tc = 369.8;//[K] - Critical temperature
17 Pc = 42.48;//[bar] - Critical pressure
18 Pc = Pc*10^(5);//[Pa]
19 w = 0.152;
20 R = 8.314;//[J/mol*K] - Universal gas constant
21
22 // At 600 K
23 Tr = T_1/Tc; // Reduced temperature
24 B_0 = 0.083-(0.422/(Tr)^(1.6));
25 B_1 = 0.139-(0.172/(Tr)^(4.2));
26 //We know, (B*Pc)/(R*Tc) = B_0 + (w*B_1)
27 B = ((B_0 + (w*B_1))*(R*Tc))/Pc; // [m^(3)/mol] -
    Second virial coefficient
28 dB0_dT = 0.422*1.6*Tc^(1.6)*T_1^(-2.6); // (dB_0/dT)
29 dB1_dT = 0.172*4.2*Tc^(4.2)*T_1^(-5.2); // (dB_1/dT)
30 dB_dT = ((R*Tc)/(Pc))*(dB0_dT + w*dB1_dT); // dB/dT
31
32 // Now let us calculate B and dB/dT at 300 K
33 Tr_prime = T_2/Tc; // Reduced temperature
34 B_0_prime = 0.083-(0.422/(Tr_prime)^(1.6));
35 B_1_prime = 0.139-(0.172/(Tr_prime)^(4.2));
36 //We know, (B*Pc)/(R*Tc) = B_0 + (w*B_1)
37 B_prime = ((B_0_prime + (w*B_1_prime))*(R*Tc))/Pc; //
    [m^(3)/mol] - Second virial coefficient

```

```

38 dB0_dT_prime = 0.422*1.6*Tc^(1.6)*T_2^(-2.6); // (
    dB_0/dT)
39 dB1_dT_prime = 0.172*4.2*Tc^(4.2)*T_2^(-5.2); // (
    dB_1/dT)
40 dB_dT_prime = ((R*Tc)/(Pc))*(dB0_dT_prime + w*
    dB1_dT_prime); // dB/dT
41
42 // The change in enthalpy for ideal gas is given by
43 delta_H_ig = integrate(' -0.966+7.279*10^(-2)*T
    -3.755*10^(-5)*T^(2)+7.58*10^(-9)*T^(3)', 'T', T_1,
    T_2); // [cal/mol]
44 delta_H_ig = delta_H_ig*4.184; // [J/mol]
45
46 // We know that delta_H_ig = delta_U_ig + R*delta_T.
    Therefore change in internal energy is given by
47 delta_U_ig = delta_H_ig - R*(T_2 - T_1); // [J/mol]
48
49 // The change in entropy of ideal gas is given by
50 // delta_S_ig = integrate(' Cp_0/T', 'T', T_1, T_2) - R*
    log(P_2/P_1);
51 delta_S_ig = integrate(' (-0.966+7.279*10^(-2)*T
    -3.755*10^(-5)*T^(2)+7.58*10^(-9)*T^(3))/T', 'T',
    T_1, T_2)*4.184 - R*log(P_2/P_1); // [J/mol-K]
52
53 // Now let us calculate the change in enthalpy of
    gas. We know that
54 // delta_H = delta_H_ig + delta_H_R
55 // delta_H_R = H_2_R - H_1_R
56 H_2_R = B_prime*P_2 - P_2*T_2*dB_dT_prime; // [J/mol]
57 H_1_R = B*P_1 - P_1*T_1*dB_dT; // [J/mol]
58 delta_H_R = H_2_R - H_1_R; // [J/mol]
59 delta_H = delta_H_ig + delta_H_R; // [J/mol]
60
61 // Let us calculate the residual entropy of gas
62 S_2_R = -P_2*dB_dT_prime; // [J/mol-K]
63 S_1_R = -P_1*dB_dT; // [J/mol-K]
64 delta_S = delta_S_ig + (S_2_R - S_1_R); // [J/mol-K]
65

```

```

66 // Let us calculate the residual internal energy of
    gas
67 U_2_R = -P_2*T_2*dB_dT_prime; // [J/mol-K]
68 U_1_R = -P_1*T_1*dB_dT; // [J/mol-K]
69 delta_U = delta_U_ig + (U_2_R - U_1_R); // [J/mol-K]
70
71 // For 100 mol sample ,
72 delta_H_ig = delta_H_ig*n*10^(-3); // [kJ/mol]
73 delta_H = delta_H*n*10^(-3); // [kJ/mol]
74
75 delta_U_ig = delta_U_ig*n*10^(-3); // [kJ/mol]
76 delta_U = delta_U*n*10^(-3); // [kJ/mol]
77
78 delta_S_ig = delta_S_ig*n*10^(-3); // [kJ/mol]
79 delta_S = delta_S*n*10^(-3); // [kJ/mol]
80
81 printf(" The value of delta_H = %f kJ/mol\n", delta_H
    );
82 printf(" The value of delta_H_ig (ideal gas)= %f kJ/
    mol\n\n", delta_H_ig);
83 printf(" The value of delta_U = %f kJ/mol\n", delta_U
    );
84 printf(" The value of delta_U_ig (ideal gas) = %f kJ
    /mol\n\n", delta_U_ig);
85 printf(" The value of delta_S = %f kJ/mol\n", delta_S
    );
86 printf(" The value of delta_S_ig (ideal gas) = %f kJ
    /mol\n\n", delta_S_ig);

```

Scilab code Exa 10.4 Calculation of molar heat capacity

```

1 clear;
2 clc;
3
4 //Example - 10.4

```



```

5 //Page number - 337
6 printf("Example - 10.4 and Page number - 337\n\n");
7
8 // Given
9 T = 35 + 273.15; // [K] - Temperature
10 P = 10; // [atm] - Pressure
11 P = P*101325; // [Pa]
12 // Methane obeys the equation of state
13 //  $Z = 1 + (P*B)/(R*T)$ 
14
15 // At 35 C,
16 B = -50; // [cm3/mol]
17 dB_dT = 1.0; // [cm3/mol-K] - dB/dT
18 dB_dT = dB_dT*10(-6); // [m3/mol-K]
19 d2B_dT2 = -0.01; // [cm3/mol-K2] - d2(B)/d(T2)
20 d2B_dT2 = d2B_dT2*10(-6); // [m3/mol-K2]
21
22 // Ideal gas molar heat capacity of methane is given
    by
23 //  $C_{p_0} = 4.75 + 1.2*10^{(-2)}*T + 0.303*10^{(-5)}*T^{(2)}$ 
    -  $2.63*10^{(-9)}*T^{(3)}$ 
24
25 // The molar heat capacity is given by
26 //  $C_p = C_{p_0} + C_{p_R}$ 
27 // For virial gas equation of state
28  $C_{p_R} = -P*T*d2B_dT2$ ; // [J/mol-K]
29
30 // thus heat capacity is given by
31 //  $C_p = a + b*T + c*T^{(2)} + d*T^{(3)} - P*T*d2B_dT2$ 
32 // Putting the values, we get
33  $C_p = (4.75 + 1.2*10^{(-2)}*T + 0.303*10^{(-5)}*T^{(2)} -$ 
     $2.63*10^{(-9)}*T^{(3)})*4.184 - P*T*d2B_dT2$ ; // [J/mol-
    K]
34
35 printf(" The molar heat capacity of methane is %f J/
    mol-K\n", Cp);

```

Scilab code Exa 10.5 Calculation of final temperature after expansion

```
1 clear;
2 clc;
3
4 //Example - 10.5
5 //Page number - 338
6 printf("Example - 10.5 and Page number - 338\n\n");
7
8 //Given
9 T_1 = 360; // [K] - Initial temperature
10 P_1 = 10; // [bar] - Initial pressure
11 P_1 = P_1*10^(5); // [Pa]
12 Tc = 408.1; // [K] - Critical temperature
13 Pc = 36.48; // [bar] - Critical pressure
14 Pc = Pc*10^(5); // [Pa]
15 w = 0.181;
16 R = 8.314; // [J/mol*K] - Universal gas constant
17 Cv_0 = 106.0; // [J/mol-K]
18
19 // At 360 K
20 Tr = T_1/Tc; // Reduced temperature
21 B_0 = 0.083-(0.422/(Tr)^(1.6));
22 B_1 = 0.139-(0.172/(Tr)^(4.2));
23 //We know, (B*Pc)/(R*Tc) = B_0 + (w*B_1)
24 B = ((B_0 + (w*B_1))*(R*Tc))/Pc; // [m^(3)/mol] -
    Second virial coefficient
25 dB0_dT = 0.422*1.6*Tc^(1.6)*T_1^(-2.6); // (dB_0/dT)
26 dB1_dT = 0.172*4.2*Tc^(4.2)*T_1^(-5.2); // (dB_1/dT)
27 dB_dT = ((R*Tc)/(Pc))*(dB0_dT + w*dB1_dT); // dB/dT
28
29 // Since system is adiabatic therefore no heat
    exchange will take place , q = 0
30 // and expansion takes place into vacuum, therefore W
```

```

= 0
31 // From first law  $\Delta U = 0$ . If the gas would have
    followed ideal gas equation of state the final
    temperature would have been the same as initial
    as  $\Delta U = 0$ 
32 // But for real gases
33 //  $\Delta U = \Delta U_{ig} + \Delta U_R$ 
34 //  $\Delta U = \Delta U_{ig} + U_{2,R} - U_{1,R}$ 
35 // For equation of state  $Z = 1 + (B*P)/(R*T)$ 
36 //  $V = B + (R*T)/P$ 
37 //  $U_R = -P*T*(dB/dT)$ 
38
39 //  $\Delta U_{ig} = C_{v,0}*(T_2 - T_1)$ 
40 //  $\Delta U = C_{v,0}*(T_2 - T_1) - P_2*T_2*(dB/dT)_2 +$ 
     $P_1*T_1*(dB/dT)_1$ 
41
42 // At state 1
43  $V_1 = B + (R*T_1)/P_1$ ; // [m3/mol] - Molar volume
44 // At state 2
45  $V_2 = 10*V_1$ ; // [m3/mol] - Molar volume
46
47 // From the equation  $\Delta U = 0$ 
48 //  $C_{v,0}*(T_2 - T_1) - ((R*T_2)/(V_2 - B_2))*T_2*(dB/dT)_2 +$ 
     $P_1*T_1*(dB/dT)_1 = 0$ 
49
50 // Now we need to solve the above equation to get
    the value of  $T_2$ 
51 // In above equation the magnitude of second term is
    much smaller as compared to the third term
    because the molar volume has become 10 times
52 // So neglecting second term, we have
53 //  $C_{v,0}*(T_2 - T_1) + P_1*T_1*(dB/dT)_1 = 0$ 
54  $T_2 = -(P_1*T_1*(dB/dT))/C_{v,0} + T_1$ ; // [K]
55
56 // For exact calculation of final temperature, let
    us start with a temperature, say
57  $T = 350$ ;
58

```

```

59 fault = 10;
60 while(fault>0.007)
61     Tr_prime = T/Tc; // Reduced temperature
62     B_0_prime = 0.083-(0.422/(Tr_prime)^(1.6));
63     B_1_prime = 0.139-(0.172/(Tr_prime)^(4.2));
64     //We know, (B*Pc)/(R*Tc) = B_0 + (w*B_1)
65     B_prime = ((B_0_prime + (w*B_1_prime))*(R*Tc))/
        Pc; // [m^3)/mol] - Second virial coefficient
66     dB0_dT_prime = 0.422*1.6*Tc^(1.6)*T_2^(-2.6); //
        (dB_0/dT)
67     dB1_dT_prime = 0.172*4.2*Tc^(4.2)*T_2^(-5.2); //
        (dB_1/dT)
68     dB_dT_prime = ((R*Tc)/(Pc))*(dB0_dT_prime + w*
        dB1_dT_prime); // dB/dT
69     deff(' [y]=f(T) ', 'y= 106*(T-T_1)+972.72 - ((R*T^(2)
        )/(V_2-B_prime))*dB_dT_prime ');
70     T_prime = fsolve(0.15,f);
71     fault=abs(T-T_prime);
72     T = T + 0.001;
73 end
74
75 printf(" The final temperature is %f K\n",T);

```

Scilab code Exa 10.6 Calculation of fugacity of liquid benzene

```

1 clear;
2 clc;
3
4 //Example - 10.6
5 //Page number - 339
6 printf("Example - 10.6 and Page number - 339\n\n");
7
8 //Given
9 T = 220 + 273.15; // [K] - Temperature
10 Tc = 562.2; // [K] - Critical temperature

```

```

11 Pc = 48.98;//[bar] - Critical pressure
12 Pc = Pc*10^(5);//[Pa]
13 w = 0.210;
14 R = 8.314;//[J/mol*K] - Universal gas constant
15 P_sat = 1912.86;//[kPa] - Saturation pressure at 220
    C
16 P_sat = P_sat*10^(3);//[Pa]
17 Mol_wt = 78.114;//[g/mol] - Molecular weight of
    benzene
18
19 //(1)
20 // Since liquid and vapour are in equilibrium the
    fugacity is saturated fugacity (f_sat) and can be
    calculated using virial gas equation of state
21 // At 220 C
22 Tr = T/Tc; // Reduced temperature
23 B_0 = 0.083-(0.422/(Tr)^(1.6));
24 B_1 = 0.139-(0.172/(Tr)^(4.2));
25 //We know, (B*Pc)/(R*Tc) = B_0 + (w*B_1)
26 B = ((B_0 + (w*B_1))*(R*Tc))/Pc; // [m^(3)/mol] -
    Second virial coefficient
27
28 // We know that log(f/P) = (B*P)/(R*T)
29 // Thus at saturated conditions
30 // log(f_sat/P_sat) = B*P_sat/(R*T)
31 f_sat = P_sat*(exp((B*P_sat)/(R*T))); // [Pa]
32 f_sat = f_sat*10^(-3); // [kPa]
33
34 printf(" (1).The fugacity of liquid benzene is %f
    kPa\n\n", f_sat);
35
36 //(2)
37 P = 2014.7; // [psia] - Total gauge pressure
38 P = 138.94; // [bar]
39 P = P*10^(5); // [Pa]
40 den = 0.63; // [g/cm^(3)] - density of benzene
41 den = den*10^(3); // [kg/m^(3)]
42

```

```

43 // Therefore specific volume is
44 V = 1/den; // [m^(3)/kg]
45 // Molar volume is given by
46 V = V*Mol_wt*10^(-3); // [m^(3)/mol]
47
48 // Thus fugacity at 220 C and pressure P is given by
49 f = f_sat*(exp((V*(P-P_sat))/(R*T)));
50
51 printf(" (2).The fugacity of liquid benzene is %f
        kPa\n\n",f);

```

Scilab code Exa 10.7 Calculation of molar enthalpy

```

1 clear;
2 clc;
3
4 //Example - 10.7
5 //Page number - 341
6 printf("Example - 10.7 and Page number - 341\n\n");
7
8 //Given
9 // C = -0.067 + 30.7/T
10 // D = 0.0012 - 0.416/T
11
12 T = 80 + 273.15; // [K]
13 P = 30; // [bar]
14 //P = P; // [N/m^(2)]
15 R = 8.314; // [J/mol*K] - Universal gas constant
16
17 // We have the relation derived in the book
18 // d(G/(R*T)) = (V/(R*T))dP - (H/(R*T^(2)))dT
19 // Writing the same equation for ideal gas and
        subtracting it from the above equation we get
20 // d(G_R/(R*T)) = (V_R/(R*T))dP - (H_R/(R*T^(2)))dT
21 // Therefore , H_R/(R*T^(2)) = -[del((G_R)/(R*T))/del

```

```

    (T)]_P
22
23 // Substituting the relation  $G_R/(R*T) = \log(f/P)$ ,
    we get
24 //  $H_R/(R*T^2) = -[\text{del}(\log(f/P))/\text{del}(T)]_P = -[\text{del}
    (-C*P - D*P^2)/\text{del}(T)]_P$ 
25 // or,  $H_R/(R*T^2) = P*(dC/dT) + P^2*dD/dT$ 
26 // Note that in the above equation the partial
    derivative is replaced by full derivative as C
    and D are functions of temperature. Therefore we
    get
27 //  $H_R/(R*T^2) = (30.7*P)/T^2 + (0.416*P^2)/T
    ^2$ 
28 //  $H_R/R = -30.7*P + 0.416*P^2$ 
29
30 // Substituting the given conditions we get
31 H_R = R*(-30.7*P + 0.416*P^2); // [J/mol]
32
33 printf(" The molar enthalpy of the gas relative to
    that of the ideal gas at 80 C and 30 bar pressure
    is , H_R = %f J/mol\n",H_R);

```

Scilab code Exa 10.8 Determination of second and third virial coefficients and fugacity

```

1 clear;
2 clc;
3 funcprot(0);
4
5 //Example - 10.8
6 //Page number - 341
7 printf("Example - 10.8 and Page number - 341\n\n");
8
9 //Given
10 // (1)

```

```

11 T = 311; //[K] - Temperature
12 R = 8.314; //[J/mol*K] - Universal gas constant
13 // Pressure in 'bar' is given below
14 P =
    [0.690,1.380,2.760,5.520,8.280,11.034,13.800,16.550];

15 // Molar volume in 'm^(3)/mol' is given below
16 V =
    [0.0373,0.0186,0.00923,0.00455,0.00298,0.00220,0.00175,0.00144];

17
18 // Z = 1 + (B/V) + (C/V^(2))
19 // (Z-1)*V = B + (C/V)
20
21
22 Z=zeros(8);
23 k=zeros(8);
24 t=zeros(8);
25 for i=1:8;
26     Z(i)=(P(i)*10^(5)*V(i))/(R*T);
27     k(i)=(Z(i)-1)*V(i);
28     t(i)=1/V(i);
29 end
30 [C,B,sig]=reglin(t',k');
31
32 //From the regression , we get intercept = B and
    slope = C,and thus ,
33
34 printf(" (1).The second virial coefficient of CO2 is
    given by B = %e m^(3)/mol\n",B);
35 printf("      The thied virial coefficient of CO2 is
    given by C = %e m^(6)/mol^(2)\n\n",C);
36
37 // (2)
38 P_final = 13.8; //[bar]
39 // We know that R*T*log(f/P) = integrate('V-(R*T)/P
    ','P',0,P)
40 // Therefore we have to plot V - (R*T)/P versus P

```



```

    and calculate the area beneath the curve from 0
    to 13.8 bar
41 // For this we need the value of the term  $V - (R*T)/P$ 
    at  $P = 0$ . At low pressure the virial equation
    becomes
42 //  $Z = 1 + (B/V)$ 
43 // and  $V - (R*T)/P = (Z*R*T)/P - (R*T)/P = (1 + (B/V))$ 
     $*$   $((R*T)/P) - (R*T)/P = (B*R*T)/(P*V) = (B/Z)$ 
44 // Thus  $\lim_{P \rightarrow 0} (V - (R*T)/P) = B$  (
    as  $P$  tend to zero,  $Z$  tend to 1 )
45
46 P_prime =
    [0.000,0.690,1.380,2.760,5.520,8.280,11.034,13.800];

47 V_prime =
    [0.000,0.0373,0.0186,0.00923,0.00455,0.00298,0.00220,0.00175];

48 summation = 0;
49 x=zeros(8);
50 y=zeros(8);
51 z=zeros(8);
52 for j=2:8;
53     x(j)=V_prime(j)-(R*T)/(P_prime(j)*10^(5)); // [m
        ^3)/mol]
54     y(j)=(x(j) + x(j-1))/2;
55     z(j)=y(j)*((P_prime(j)-P_prime(j-1)))*10^(5);
56     summation = summation + z(j) ; // [J/mol]
57 end
58
59 summation = summation + 2*z(2) - z(2); // Because in
    the above calculation ,in order to calculate the
    average a summation of z(2) is not included ,only
    half of it gets added
60
61 // Now we have, area = integrate('V - (R*T)/P', 'P
    ',0,13.8*10^(5)) = summation
62 //  $R*T*\log(f/P) = \text{summation}$ 
63 f = P_final*(exp(summation/(R*T))); // [bar]

```

```

64
65 printf(" (2).The fugacity of steam at 311 K and 13.8
    bar pressure is %f bar",f);

```

Scilab code Exa 10.9 Determination of second and third virial coefficients

```

1 clear;
2 clc;
3
4 //Example - 10.9
5 //Page number - 344
6 printf("Example - 10.9 and Page number - 344\n\n");
7
8 //Given
9 T = 0 + 273.15; // [K] - Temperature
10 R = 8.314; // [J/mol*K] - Universal gas constant
11 // Pressure in 'atm' is given below
12 P = [100,200,300,400,500,600,700,800,900,1000];
13 // The compressibility factor values are
14 Z =
    [1.069,1.138,1.209,1.283,1.356,1.431,1.504,1.577,1.649,1.720];
15
16 // Z = 1 + (B/V) + (C/V^(2))
17 // (Z-1)*V = B + (C/V)
18
19
20 V = zeros(1,10);
21 k = zeros(1,10);
22 t = zeros(1,10);
23 for i=1:10;
24     V(1,i)=Z(i)*R*T/(P(i)*101325); // [m^(3)/mol]
25     k(1,i)=(Z(i)-1)*V(i);
26     t(1,i)=1/V(i);
27 end

```

```

28 [C,B,sig]=reglin(t,k);
29
30 //From the regression , we get intercept = B and
    slope = C,and thus ,
31
32 printf(" (1).The second virial coefficient of H2 is
    given by B = %e m^(3)/mol\n",B);
33 printf("      The thied virial coefficient of H2 is
    given by C = %e m^(6)/mol^(2)\n\n",C);
34
35 // (2)
36 // We know that , limit P tending to zero (V-(R*T)/P)
    = B, therefore P = 0, V-(R*T)/P = B
37 // Now let us tabulate V-(R*T)/P and determine the
    integral integrate(' (V-(R*T)/P) ', 'P',0,1000)
38
39 P_prime =
    [0,100,200,300,400,500,600,700,800,900,1000];
40 Z_prime =
    [0,1.069,1.138,1.209,1.283,1.356,1.431,1.504,1.577,1.649,1.720];
41
42 summation = 0;
43 V_prime = zeros(1,11);
44 x = zeros(1,11);
45 y = zeros(1,11);
46 z = zeros(1,11);
47 for j=2:11;
48     V_prime(1,j)=Z_prime(j)*R*T/(P_prime(j)*101325);
        // [m^(3)/mol]
49     x(1,j)=V_prime(j)-(R*T)/(P_prime(j)*101325);
50     y(1,j)=(x(j) + x(j-1))/2;
51     z(1,j)=y(j)*((P_prime(j)-P_prime(j-1)))*101325;
52     summation = summation + z(j) ;//[J/mol]
53 end
54
55 summation = summation + 2*z(2) - z(2);// Because in
    the above calculation ,in order to calculate the

```

```

    average a summation of  $z(2)$  is not included , only
    half of it gets added
56
57 // Now we have
58 //  $R*T*\log(f/P)$  = summation
59 P_dash = 1000; //[atm] - Pressure at which fugacity
    is to be calculated
60 T_dash = 273.15; //[K] - Temperature at which
    fugacity is to be calculated
61 f = P_dash*exp(summation/(R*T_dash)); //[atm]
62
63 printf(" (2).The fugacity of H2 at 0 C and 1000 atm
    pressure is , f = %f atm\n",f);

```

Scilab code Exa 10.10 Determination of work done and the exit temperature

```

1 clear;
2 clc;
3
4 //Example - 10.10
5 //Page number - 345
6 printf("Example - 10.10 and Page number - 345\n\n");
7
8 //Given
9 P_1 = 1*10^(6); //[Pa] - Initial pressure
10 T_1 = 200 + 273.15; //[K] - Initial temperature
11 P_2 = 8*10^(6); //[Pa] - Final pressure
12 Tc = 647.1; //[K] - Critical temperature of water
13 Pc = 220.55; //[bar] - Critical pressure of water
14 Pc = Pc*10^(5); //[Pa]
15 w = 0.345;
16 R = 8.314; //[J/mol*K] - Universal gas constant
17
18 // For the virial gas the following are the

```

```

relations for residual enthalpy and entropy
19 // H_R = B*P - P*T*(dB/dT)
20 // S_R = -P*(dB/dT)
21 // Where, (dB/dT) = ((R*Tc)/Pc)*((dB_0/dT) + w*(dB_1
/dT))
22 // dB0_dT = 0.422*1.6*Tc^(1.6)*T^(-2.6);// (dB_0/dT)
23 // dB1_dT = 0.172*4.2*Tc^(4.2)*T^(-5.2);// (dB_1/dT)
24
25 // (1)
26 Cp_0 = 29.114;//[J/mol-K] - Specific heat capacity
at constant pressure
27 // For the isentropic process entropy change is zero
, thus
28 // delta_S = Cp_0*log(T_2/T_1) - P_2*(dB/dT)_2 + P_1
*(dB/dT)_1 = 0
29
30 // At state 1,
31 Tr_1 = T_1/Tc;
32 B0_1 = 0.083 - 0.422/(Tr_1^(1.6));
33 B1_1 = 0.139 - 0.172/(Tr_1^(4.2));
34 // (B*Pc)/(R*Tc) = B0 + w*B1
35 B_1 = ((B0_1 + w*B1_1)*(R*Tc))/Pc;//[m^(3)/mol] -
Second virial coefficient at state 1
36 dB0_dT_1 = 0.422*1.6*Tc^(1.6)*T_1^(-2.6);// (dB_0/dT
)
37 dB1_dT_1 = 0.172*4.2*Tc^(4.2)*T_1^(-5.2);// (dB_1/dT
)
38 dB_dT_1 = ((R*Tc)/Pc)*((dB0_dT_1) + w*(dB1_dT_1));//
(dB/dT)_1
39
40 // Now let us assume the exit temperature to be 870
K, at this temperature
41 // T_2 = 870;//[K] -
42 // At this temperature
43 // delta_S = Cp_0*log(T_2/T_1) - P_2*(dB/dT)_2 + P_1
*(dB/dT)_1 =
44
45

```

```

46 T_2 = 860; // [K] - Exit temperature
47 // Therefore at state 2, we have
48 Tr_2 = T_2/Tc;
49 B0_2 = 0.083 - 0.422/(Tr_2^(1.6));
50 B1_2 = 0.139 - 0.172/(Tr_2^(4.2));
51 // (B*Pc)/(R*Tc) = B0 + w*B1
52 B_2 = ((B0_2 + w*B1_2)*(R*Tc))/Pc; // [m^(3)/mol] -
    Second virial coefficient at state 2
53 dB0_dT_2 = 0.422*1.6*Tc^(1.6)*T_2^(-2.6); // (dB_0/dT
    )
54 dB1_dT_2 = 0.172*4.2*Tc^(4.2)*T_2^(-5.2); // (dB_1/dT
    )
55 dB_dT_2 = ((R*Tc)/Pc)*((dB0_dT_2) + w*(dB1_dT_2)); //
    (dB/dT)_2
56
57 delta_H_s = Cp_0*(T_2 - T_1) + B_2*P_2 - P_2*T_2*(
    dB_dT_2) - B_1*P_1 + P_1*T_1*(dB_dT_1); // [J/mol]
    - Enthalpy change
58
59 // As no heat exchange is assumed to take place with
    the surroundings, work transfer is given by
60 W_1 = - delta_H_s; // [J/mol]
61
62 printf(" (1).The exit temperature is %f K\n",T_2);
63 printf("      The required amount of work is %f J/mol
    \n\n",W_1);
64
65
66 // (2)
67 eff = 0.8; // Adiabatic efficiency
68 delta_H_a = delta_H_s/0.8; // Actual enthalpy change
69
70 // Now for calculating the value of T_exit
71 // delta_H_a = Cp_0*(T_exit - T_1) + B*P_2 - P_2*
    T_exit*(dB_dT) - B_1*P_1 + P_1*T_1*(dB_dT_1)
72 // On simplification we get
73 // 29.114*(T_2 - T_1)*B_2*8*10^(6) - 8*10^(6)*T_2*(dB/
    dT)_2 = 12643.77

```

```

74
75 // Let us assume a temperature of say
76 T = 900; // [K]
77 fault=10;
78
79 while(fault>0.3)
80     Tr = T/Tc;
81     B0 = 0.083 - 0.422/(Tr^(1.6));
82     B1 = 0.139 - 0.172/(Tr^(4.2));
83     // (B*Pc)/(R*Tc) = B0 + w*B1
84     B = ((B0 + w*B1)*(R*Tc))/Pc; // [m^(3)/mol] -
        Second virial coefficient at state 2
85     dB0_dT = 0.422*1.6*Tc^(1.6)*T^(-2.6); // (dB_0/dT
        )
86     dB1_dT = 0.172*4.2*Tc^(4.2)*T^(-5.2); // (dB_1/dT
        )
87     dB_dT = ((R*Tc)/Pc)*((dB0_dT) + w*(dB1_dT)); // (
        dB/dT)_1
88     deff(' [y]=f(T_exit)', 'y = delta_H_a - Cp_0*(
        T_exit - T_1) + B*P_2 - P_2*T_exit*(dB_dT) -
        B_1*P_1 + P_1*T_1*(dB_dT_1)');
89     T_exit = fsolve(900,f);
90     fault=abs(T-T_exit);
91     T = T + 0.2;
92 end
93 Texit = T;
94
95 // As no heat exchange is assumed to take place with
        the surroundings, work transfer is given by
96 W_2 = - delta_H_a; // [J/mol]
97
98 printf(" (2).The exit temperature is %f K\n",Texit);
99 printf("          The required amount of work is %f J/mol
        \n\n",W_2);
100
101 // (3)
102 // Cp_0 = 7.7 + 0.04594*10^(-2)*T + 0.2521*10^(-5)*T
        ^ (2) - 0.8587*10^(-9)*T^(3)

```

```

103 // The entropy change for a gas following the virial
      equation of state is given by
104 //  $\Delta S = \int_{T_1}^{T_2} \frac{C_{p,0}}{T} dT - R \log \left( \frac{P_2}{P_1} \right) - P_2 \left( \frac{dB}{dT} \right)_2 + P_1 \left( \frac{dB}{dT} \right)_1$ 
105 // For an isentropic process the entropy change is
      zero and substituting the various values in the
      above equation we get
106 //  $32.2168 \log(T_2) + 0.1922 \cdot 10^{-2} T_2 + 0.5274 \cdot 10^{-5} T_2^2 - 1.1976 \cdot 10^{-9} T_2^3 - 8 \cdot 10^6 \left( \frac{dB}{dT} \right)_2 - 216.64 = 0$ 
107
108 // Let us assume a temperature of say
109 T_prime = 700; // [K]
110 fault1=10;
111
112 while(fault1>0.5)
113     Tr_prime = T_prime/Tc;
114     B0_prime = 0.083 - 0.422/(Tr_prime^(1.6));
115     B1_prime = 0.139 - 0.172/(Tr_prime^(4.2));
116     //  $(B \cdot P_c)/(R \cdot T_c) = B_0 + w \cdot B_1$ 
117     B_prime = ((B0_prime + w*B1_prime)*(R*Tc))/Pc; //
      [m^3/mol] - Second virial coefficient at
      state 2
118     dB0_dT_prime = 0.422*1.6*Tc^(1.6)*T_prime^(-2.6)
      ;// (dB_0/dT)
119     dB1_dT_prime = 0.172*4.2*Tc^(4.2)*T_prime^(-5.2)
      ;// (dB_1/dT)
120     dB_dT_prime = ((R*Tc)/Pc)*((dB0_dT_prime) + w*(
      dB1_dT_prime)); // (dB/dT)_1
121     deff(' [y]=f1(T_out)', 'y = 32.2168*log(T_out) +
      0.1922*10^(-2)*T_out + 0.5274*10^(-5)*T_2^(2)
      - 1.1976*10^(-9)*T_out^(3) - 8*10^(6)*
      dB_dT_prime - 216.64');
122     T_out = fsolve(10,f1);
123     fault1=abs(T_prime-T_out);
124     T_prime = T_prime + 0.5;
125 end
126 T_out = T_prime;

```



```

127
128 // Now we have to calculate enthalpy change as  $W = -\Delta H$ 
129 delta_H_3 = integrate('(7.7 + 0.04594*10^(-2)*T +
    0.2521*10^(-5)*T^(2) - 0.8587*10^(-9)*T^(3))
    *4.184', 'T', T_1, T_out) + B_prime*P_2 - P_2*T_out*
    dB_dT_prime - B_1*P_1 + P_1*T_1*dB_dT_1; // [J/mol]
130
131 W_3 = - delta_H_3; // [J/mol]
132
133 printf(" (3).The exit temperature is %f K\n", T_out);
134 printf("      The required amount of work is %f J/mol
    \n\n", W_3);
135
136 //(4)
137 n = 0.8; // Adiabatic efficiency
138 delta_H_a_4 = delta_H_3/n; // [J/mol]
139 W_4 = -delta_H_a_4; // [J/mol]
140
141 // Now we have to determine the exit temperature
    when the enthalpy is delta_H_a_4
142 //  $7.7*4.184*(T_2-T_1) + ((0.04594*4.184*10^{(-2)})/2)
    *(T_2^{(2)}-T_1^{(2)}) + ((0.2521*4.184*10^{(-5)})/3)*(
    T_2^{(3)}-T_1^{(3)}) - ((0.8587*4.184*10^{(-9)})/4)*(
    T_2^{(4)}-T_1^{(4)}) + B_2*8*10^{(6)} - 8*10^{(6)}*T_2
    *(dB/dT)_2 + 191.7 + 496.81 = \Delta H_a_4$ 
143
144 // Let us assume a temperature of say
145 T_prime1 = 700; // [K]
146 fault2=10;
147
148 while(fault2>0.5)
149     Tr_prime1 = T_prime1/Tc;
150     B0_prime1 = 0.083 - 0.422/(Tr_prime1^(1.6));
151     B1_prime1 = 0.139 - 0.172/(Tr_prime1^(4.2));
152     //  $(B*Pc)/(R*Tc) = B0 + w*B1$ 
153     B_prime1 = ((B0_prime1 + w*B1_prime1)*(R*Tc))/Pc
        ; // [m^3/mol] - Second virial coefficient

```

```

        at state 2
154     dB0_dT_prime1 = 0.422*1.6*Tc^(1.6)*T_prime1
        ^(-2.6); // (dB_0/dT)
155     dB1_dT_prime1 = 0.172*4.2*Tc^(4.2)*T_prime1
        ^(-5.2); // (dB_1/dT)
156     dB_dT_prime1 = ((R*Tc)/Pc)*((dB0_dT_prime1) + w
        *(dB1_dT_prime1)); // (dB/dT)_1
157     deff(' [y]=f2(T_2)', 'y = 7.7*4.184*(T_2-T_1) +
        ((0.04594*4.184*10^(-2))/2)*(T_2^(2)-T_1^(2))
        + ((0.2521*4.184*10^(-5))/3)*(T_2^(3)-T_1
        ^3) - ((0.8587*4.184*10^(-9))/4)*(T_2^(4)-
        T_1^(4)) + B_prime1*8*10^(6) - 8*10^(6)*T_2*
        dB_dT_prime1 + 191.7 + 496.81 - delta_H_a_4')
        ;
158     T_out1 = fsolve(100,f2);
159     fault2=abs(T_prime1-T_out1);
160     T_prime1 = T_prime1 + 0.5;
161 end
162 T_out1 = T_prime1;
163
164 printf(" (4).The exit temperature is %f K\n",T_out1)
        ;
165 printf("          The required amount of work is %f J/mol
        \n\n",W_4);

```

Scilab code Exa 10.11 Calculation of temperature and pressure

```

1 clear;
2 clc;
3 funcprot(0);
4
5 //Example - 10.11
6 //Page number - 348
7 printf("Example - 10.11 and Page number - 348\n\n");
8

```

```

9 //Given
10 Vol = 0.15;//[m^(3)] - Volume of the cylinder
11 P_1 = 100;//[bar] - Initial pressure
12 P_1 = P_1*10^(5);//[Pa]
13 T_1 = 170;//[K] - Initial temperature
14 n_withdrawn = 500;//[mol] - Withdrawn number of
    moles
15 R = 8.314;//[J/mol*K] - Universal gas constant
16
17
18 //(1)
19 Y = 1.4;// Coefficient of adiabatic expansion
20 n_total = (P_1*Vol)/(R*T_1);//[mol] - Total number
    of moles
21 n_2 = n_total - n_withdrawn;//[mol] - Left number of
    moles
22 V_1 = Vol/n_total;//[m^(3)/mol] - Molar volume at
    initial state.
23 // At final state
24 V_2 = Vol/n_2;//[m^(3)/mol] - Molar volume at final
    state
25
26 // During discharging  $P_1*V_1^Y = P_2*V_2^Y$ ,
    therefore
27 P_2_1 = P_1*((V_1/V_2)^Y);//[Pa] - Final pressure
28 P_2_1 = P_2_1*10^(-5);//[bar]
29 T_2_1 = ((P_2_1*10^(5))*V_2)/R;//[K] - Final
    temperature
30
31 printf(" (1).The final temperature %f K\n",T_2_1);
32 printf("      The final pressure %f bar\n\n",P_2_1);
33
34 //(2)
35 // Cp_0 = 27.2 + 4.2*10^(-3)*T
36 // For a discharge process entropy per mol of the
    gas that remains in the cylinder is constant ,
    delta_S = 0
37 // Therefore for one mol of ideal gas integrate('

```

```

    Cp_0/T', 'T', T_1, T_2) - R*log(P_2/P_1) = 0
38 // Since the gas is assumed to be ideal, therefore
    P_2*Vol = n_2*R*T_2
39 // P_2 = (n_2*R*T_2)/V_2. Substituting in above
    equation after simplification we get
40 // 27.2*log(T_2/T_1) + 4.2*10^(-3)*(T_2 - T_1) - R*
    log(P_2/P_1)
41 // f(T) = 18.886*log(T) + 4.2*10^(-3)*T - 92.4
42 // f(T)_dash = 18.886/T + 4.2*10^(-3) // Derivative
    of the above function
43
44 // Starting with a temperature of 150 K
45 T_prime = 150; // [K]
46 error = 10;
47 while(error>1)
48     f_T = 18.886*log(T_prime) + 4.2*10^(-3)*T_prime
        - 92.4;
49     f_T_dash = 18.886/T_prime + 4.2*10^(-3);
50     T_new = T_prime - (f_T/f_T_dash);
51     error=abs(T_prime - T_new);
52     T_prime = T_new;
53 end
54
55 T_2_2 = T_prime; // [K] - Final temperature
56 P_2_2 = ((n_2*R*T_2_2)/Vol)*10^(-5); // [bar] - Final
    pressure
57
58 printf(" (2).The final temperature %f K\n",T_2_2);
59 printf("      The final pressure %f bar\n\n",P_2_2);
60
61 //(3)
62 Tc = 126.2; // [K] - Critical temperature of nitrogen
63 Pc = 34.0; // [bar] - Critical pressure of nitrogen
64 Pc = Pc*10^(5); // [Pa]
65 w = 0.038; // Acentric factor
66
67 // Virial equation of state , Z = 1 + (B*P)/(R*T)
68 // S_R = -P*(dB/dT)

```

```

69
70 dB0_dT = 0.422*1.6*Tc^(1.6)*T_1^(-2.6); // (dB_0/dT)
    at state 1
71 dB1_dT = 0.172*4.2*Tc^(4.2)*T_1^(-5.2); // (dB_1/dT)
    at state 1
72 dB_dT = ((R*Tc)/Pc)*((dB0_dT) + w*(dB1_dT)); // (dB/
    dT) at state 1
73 // The residual entropy at the initial state is
    given by
74 S_R_1 = -P_1*(dB_dT); // [J/mol-K]
75
76 // Now let us calculate molar volume at initial
    state
77 Tr = T_1/Tc; // Reduced temperature
78 B_0 = 0.083-(0.422/(Tr)^(1.6));
79 B_1 = 0.139-(0.172/(Tr)^(4.2));
80
81 //We know, (B*Pc)/(R*Tc) = B_0 + (w*B_1)
82 B = ((B_0+(w*B_1))*(R*Tc))/Pc; // [m^(3)/mol]
83
84 V_1_3 = B + (R*T_1)/P_1; // [m^(3)/mol]
85 // Therefore number of moles in the initial state is
86 n_1_3 = Vol/V_1_3; // [mol]
87 // Therefore final number of moles is
88 n_2_3 = n_1_3 - n_withdrawn;
89
90 // Therefore molar volume at final state is
91 V_2_3 = Vol/n_2_3; // [m^(3)/mol]
92
93 // Now let us determine the relation between
    pressure and temperature in the final state
94 // P_2_3 = (R*T_2_3)/(V_2_3 - B_2)
95 // delta_S = 0, thus delta_S_ig + delta_S_R = 0
96 delta_S_R = - S_R_1;
97 // integrate('Cp_0/T', 'T', T_1, T_2) - R*log(P_2/P_1)
    - P_2*(dB/dT)_2 + S_R_1
98 // On simplification,
99 // delta_S = 27.2*(log(T_2_prime/T_1)) + 4.2*10^(-3)

```

```

        *(T_2_prime - T_1) - R*(log(P_2_3/P_1)) - P_2_3*(
        dB_dT_3) + delta_S_R
100 // Starting with a temperature of 135 K
101
102 T_2_prime = 135; // [K]
103 delta = 0.1;
104 error = 10;
105 while(error>0.01)
106     T_r = T_2_prime/Tc; // Reduced temperature
107     B_0_3 = 0.083-(0.422/(T_r)^(1.6));
108     B_1_3 = 0.139-(0.172/(T_r)^(4.2));
109     B_3 = ((B_0_3+(w*B_1_3))*(R*Tc))/Pc; // [m^(3)/mol
        ]
110     dB0_dT_3 = 0.422*1.6*Tc^(1.6)*T_2_prime^(-2.6);
        // (dB_0/dT)
111     dB1_dT_3 = 0.172*4.2*Tc^(4.2)*T_2_prime^(-5.2);
        // (dB_1/dT)
112     dB_dT_3 = ((R*Tc)/Pc)*((dB0_dT_3) + w*(dB1_dT_3)
        ); // (dB/dT)
113     P_2_3 = (R*T_2_prime)/(V_2_3 - B_3);
114     delta_S = 27.2*(log(T_2_prime/T_1)) +
        4.2*10^(-3)*(T_2_prime - T_1) - R*(log(P_2_3/
        P_1)) - P_2_3*(dB_dT_3) + delta_S_R;
115     T_new = T_2_prime + delta;
116     error=abs(delta_S);
117     T_2_prime = T_new;
118 end
119
120 T_2_3 = T_2_prime; // [K] - Final temperature
121 // Therefore at T_2_3
122 P_2_3 = P_2_3*10^(-5); // [bar] - Final pressure
123
124 printf(" (3).The final temperature %f K\n",T_2_3);
125 printf("      The final pressure %f bar\n\n",P_2_3);

```

Scilab code Exa 10.12 Calculation of change of internal energy enthalpy entropy and exergy

```

1  clear;
2  clc;
3  funcprot(0);
4
5  //Example - 10.12
6  //Page number - 351
7  printf("Example - 10.12 and Page number - 351\n\n");
8
9  //Given
10 P_1 = 80; //[bar] - Initial pressure
11 P_1 = P_1*10^(5); //[Pa]
12 T_1 = 300 + 273.15; //[T] - Initial temperature
13 P_2 = 40; //[bar] - Final pressure
14 P_2 = P_2*10^(5); //[Pa]
15 T_2 = 300 + 273.15; //[K] - Final temperature
16 T_0 = 25 + 273.15; //[K] - Surrounding temperature
17 P_0 = 1; //[atm] - Surrounding pressure
18 P_0 = P_0*101325; //[Pa]
19 Tc = 647.1; //[K]
20 Pc = 220.55; //[bar]
21 Pc = Pc*10^(5); //[Pa]
22 R = 8.314; //[J/mol*K] - Universal gas constant
23
24 // For van der Walls equation of state
25 a = (27*R^(2)*Tc^(2))/(64*Pc); //[Pa-m^(6)/mol^(2)]
26 b = (R*Tc)/(8*Pc); //[m^(3)/mol]
27
28 // The cubic form of van der Walls equation of state
    is given by,
29 //  $V^3 - (b + (R*T)/P)*V^2 + (a/P)*V - (a*b)/P = 0$ 
30
31 // Solving the cubic equation
32 // At 80 bar and 300 K
33 def f(' [y]=f(V) ', 'y=V^3-(b+(R*T_1)/P_1)*V^2+(a/

```

```

    P_1)*V-(a*b)/P_1');
34 V_1_1=fsolve(0.1,f);
35 V_1_2=fsolve(10,f);
36 V_1_2=fsolve(100,f);
37 // The largest root is considered because of vapour
38 V_1 = V_1_1;
39
40 U_R_1 = -a/V_1;//[J/mol] - Internal energy
41 H_R_1 = P_1*V_1 - R*T_1 - a/V_1;//[J/mol] - Enthalpy
42 S_R_1 = R*log((P_1*(V_1-b))/(R*T_1));
43
44 // Now let us calculate the residual properties at
    state 2
45 // At 40 bar and 300 K
46 deff(' [y]=f1(V) ', 'y=V^(3)-(b+(R*T_2)/P_2)*V^(2)+(a/
    P_2)*V-(a*b)/P_2');
47 V_2_1 = fsolve(0.1,f1);
48 V_2_2 = fsolve(10,f1);
49 V_2_3 = fsolve(100,f1);
50 // The above equation has 1 real and 2 imaginary
    roots. We consider only real root.
51 V_2 = V_2_1;
52
53 U_R_2 = -a/V_2;//[J/mol] - Internal energy
54 H_R_2 = P_2*V_2 - R*T_2 - a/V_2;//[J/mol] - Enthalpy
55 S_R_2 = R*log((P_2*(V_2-b))/(R*T_2));
56
57 delta_U_R = U_R_2 - U_R_1;//
58 delta_H_R = H_R_2 - H_R_1;//
59 delta_S_R = S_R_2 - S_R_1;//
60
61 delta_U_ig = 0;//[J/mol] - As temperature is
    constant
62 delta_H_ig = 0;//[J/mol] - As temperature is
    constant
63 // delta_S_ig = Cp_0*log(T_2/T_1) - R*log(P_2/P_1)
    ;// [J/mol-K]
64 // Since T_1 = T_2

```



```

65 // Therefore
66 delta_S_ig = - R*log(P_2/P_1); // [J/mol-K]
67 delta_U = delta_U_R + delta_U_ig; // [J/mol]
68 delta_H = delta_H_R + delta_H_ig; // [J/mol]
69 delta_S = delta_S_R + delta_S_ig; // [J/mol-K]
70
71 // Change in exergy is given by
72 // delta_phi = phi_1 - phi_2 = U_1 - U_2 + P_0*(V_1
    - _V_2) - T_0*(S_1 - S_2)
73 delta_phi = - delta_U + P_0*(V_1 - V_2) - T_0*(-
    delta_S); // [J/mol]
74
75 printf(" The change in internal energy is %f J/mol\n
    ", delta_U);
76 printf(" The change in enthalpy is %f J/mol\n",
    delta_H);
77 printf(" The change in entropy is %f J/mol-K\n",
    delta_S);
78 printf(" The change in exergy is %f J/mol\n",
    delta_phi);

```

Scilab code Exa 10.13 Calculation of change in enthalpy

```

1 clear;
2 clc;
3
4 //Example - 10.13
5 //Page number - 353
6 printf("Example - 10.13 and Page number - 353\n\n");
7
8 //Given
9 T_1 = 500; // [K] - Initial temperature
10 P_1 = 30; // [atm] - Initial pressure
11 P_1 = P_1*101325; // [Pa]
12 P_2 = 1; // [atm] - Final pressure

```

```

13 P_2 = P_2*101325;//[Pa]
14 R = 8.314;//[J/mol*K] - Universal gas constant
15 // For chlorine
16 Tc = 417.2;//[K] - Critical temperature
17 Pc = 77.10;//[bar] - Critical pressure
18 Pc = Pc*10^(5);//[Pa]
19
20 //Redlich Kwong equation of state ,
21 a = (0.42748*(R^(2))*(Tc^(2.5)))/Pc; // [Pa*m^(6)*K
    ^ (1/2)/mol]
22 b = (0.08664*R*Tc)/Pc; // [m^(3)/mol]
23
24 // The cubic form of Redlich Kwong equation of state
    is given by,
25 //  $V^3 - ((R*T)/P)*V^2 - ((b_1^2) + ((b_1*R*T)/P) - (a
    / (T^{1/2}*P)))*V - (a*b) / (T^{1/2}*P) = 0$ 
26 //Solving the cubic equation
27 // At state 1 (500 K, 30 atm)
28 def ( ' [y]=f1 (V) ', 'y=V^3 - ((R*T_1)/P_1)*V^2 - ((b
    ^ (2)) + ((b*R*T_1)/P_1) - (a / (T_1^{1/2}*P_1))) *V - (a*b
    ) / (T_1^{1/2}*P_1) ');
29 V_1=fsolve(1, f1);
30 V_2=fsolve(10, f1);
31 V_3=fsolve(100, f1);
32 // The above equation has 1 real and 2 imaginary
    roots. We consider only real root ,
33 V = V_1; // [m^(3)/mol]
34
35 // Thus compressibility factor is
36 Z = (P_1*V_1)/(R*T_1); //compressibility factor
37
38 // The residual enthalpy at state 1 is given by
39 H_R_1 = (Z-1)*R*T_1 + ((3*a)/(2*b*T_1^(1/2)))*(log(V
    /(V+b))); // [J/mol]
40
41 // Since chlorine is assumed to behave ideally under
    the final condition , therefore
42 H_R_2 = 0; // Residual enthalpy at state 2

```

```

43 delta_H_R = H_R_2 - H_R_1;//[J/mol] - Residual
    enthalpy change
44 // and since isothermal conditions are maintained,
    therefore
45 delta_H_ig = 0;// Enthalpy change under ideal
    condition
46 delta_H = delta_H_R + delta_H_ig;//[J/mol]
47
48 printf(" The change in enthalpy is given by, delta_H
    = %f J/mol\n",delta_H);

```

Scilab code Exa 10.14 Calculation of final temperature

```

1 clear;
2 clc;
3
4 //Example - 10.14
5 //Page number - 353
6 printf("Example - 10.14 and Page number - 353\n\n");
7
8 //(1)
9 //This part involves proving a relation in which no
    mathematical components are involved.
10 //For prove refer to this example 10.14 on page
    number 354 of the book.
11 printf(" (1).This part involves proving a relation
    in which no mathematical components are involved
    .\n");
12 printf("      For prove refer to this example 10.14
    on page number 354 of the book.\n\n")
13
14 //(2)
15 //Given
16 Vol_1 = 0.1;//[m^(3)] - Initial volume of each
    compartment

```

```

17 n_1 = 400; //[mol] - Initial number of moles in
    compartment 1
18 V_1 = Vol_1/n_1; //[m^(3)/mol] - Molar volume at
    state 1
19 T_1 = 294; //[K]
20 Vol_2 = 0.2; //[m^(3)] - Final volume of the
    compartment after removing the partition.
21 n_2 = n_1; //[mol] - Number of moles remains the same
22 V_2 = Vol_2/n_2; //[m^(3)/mol] - Molar volume at
    state 2
23
24 // For argon
25 a = 0.1362; //[Pa-m^(6)/mol^(2)]
26 b = 3.215*10^(-5); //[m^(3)/mol]
27 Cv_0 = 12.56; //[J/mol-K] - Heat capacity in ideal
    gas state
28
29 // For overall system q = 0, and no work is done,
    therefore delta_U = 0
30 // Therefore from the relation proved in part (1), we
    have
31 T_2 = T_1 + (a/Cv_0)*(1/V_2 - 1/V_1); //[K]
32
33 printf(" (2).The final temperatutre is %f K\n",T_2)

```

Scilab code Exa 10.15 Proving a mathematical relation

```

1 clear;
2 clc;
3
4 //Example - 10.15
5 //Page number - 354
6 printf("Example - 10.15 and Page number - 354\n\n");
7
8 //This problem involves proving a relation in which

```

```

    no mathematical components are involved.
9 //For prove refer to this example 10.15 on page
    number 354 of the book.
10 printf(" This problem involves proving a relation in
    which no mathematical components are involved.\n
    \n");
11 printf(" For prove refer to this example 10.15 on
    page number 354 of the book.")

```

Scilab code Exa 10.16 Proving a mathematical relation

```

1 clear;
2 clc;
3
4 //Example - 10.16
5 //Page number - 355
6 printf("Example - 10.16 and Page number - 355\n\n");
7
8 //This problem involves proving a relation in which
    no mathematical components are involved.
9 //For prove refer to this example 10.16 on page
    number 355 of the book.
10 printf(" This problem involves proving a relation in
    which no mathematical components are involved.\n
    \n");
11 printf(" For prove refer to this example 10.16 on
    page number 355 of the book.")

```

Scilab code Exa 10.17 Determination of work done and the exit temperature

```

1 clear;
2 clc;

```

```

3 funcprot(0);
4
5 //Example - 10.17
6 //Page number - 356
7 printf("Example - 10.17 and Page number - 356\n\n");
8
9 //Given
10 P_1 = 1*10^(6);//[Pa] - Initial pressure
11 T_1 = 200 + 273.15;//[K] - Initial temperature
12 P_2 = 8*10^(6);//[Pa]
13 R = 8.314;//[J/mol*K] - Universal gas constant
14 Y = 1.4; // Index of expansion
15 Cp_0 = 29.114;//[J/mol-K]
16 // For H2O, the van der Walls constants are
17 a = 0.55366;//[Pa-m^(6)/mol^(2)]
18 b = 3.049*10^(-5);//[m^(3)/mol]
19
20 // At state 1 (200 C, 1 MPa)
21 // The molar volume of steam following van der Walls
    equation of state (as reported in the book) is
22 V_1 = 3.816*10^(-3);//[m^(3)/mol]
23 // And the compressibility factor is
24 Z_1 = (P_1*V_1)/(R*T_1);
25
26 // Assuming ideal gas behaviour the exit temperature
    is given by
27 T_2 = T_1*(P_2/P_1)^((Y-1)/Y);//[K]
28
29 // At 8 MPa and T_2,
30 // The molar volume of steam following van der Walls
    equation of state (as reported in the book) is
31 V_2 = 8.41*10^(-4);//[m^(3)/mol]
32 // And the compressibility factor is
33 Z_2 = (P_2*V_2)/(R*T_2);
34
35 // For van der Walls equation of state we know that
36 // delta_S_R/R = log(Z_2/Z_1) + log((V_2 - b)/V_2) -
    log((V_1 - b)/V_1)

```

```

37 delta_S_R = R*(log(Z_2/Z_1) + log((V_2 - b)/V_2) -
    log((V_1 - b)/V_1)); // [J/mol]
38
39 // delta_S_ig = Cp_0*log(T_2/T_1) - R*log(P_2/P_1)
40 // The entropy change is therefore
41 // delta_S = delta_S_ig + delta_S_R
42 // But during an isentropic process the total
    entropy change is zero
43 // Therefore we have to modify the exit temperature
    so that the entropy change is zero
44
45 // Let us assume a temperature, say T = 870 K
46 // At 870 K the molar volume of steam following van
    der Waals equation of state (as reported in the
    book) is
47 // V_3 = 8.57*10^(-4); // [m^(3)/mol]
48 // Therefore
49 // Z_3 = (P_2*V_3)/(R*T_2);
50 // At this temperature,
51 // delta_S = Cp_0*log(T/T_1) - R*log(P_2/P_1) + R*(
    log(Z/Z_1) + R*log((V - b)/V) - R*log((V_1 - b)/
    V_1))
52
53 T = 800; // [K]
54 fault=10;
55
56 while(fault>0.3)
57     // At T and 8 MPa
58     deff(' [y]=f1(V) ', 'y=V^(3)-(b+(R*T)/P_2)*V^(2)+(a
        /P_2)*V-(a*b)/P_2');
59     V = fsolve(1,f1);
60     Z = (P_2*V)/(R*T);
61
62     deff(' [y]=f1(T) ', 'y = Cp_0*log(T/T_1) - R*log(
        P_2/P_1) + R*(log(Z/Z_1) + R*(log((V - b)/V))
        - R*(log((V_1 - b)/V_1)))');
63     T_exit = fsolve(0.1,f1);
64     fault=abs(T-T_exit);

```

```

65     T = T + 0.5;
66 end
67 Texit = T;
68
69 // Now applying the first law to an adiabatic
    process we get
70 // W = - delta_H
71
72 // For van der Waals gas the enthalpy change is
    given by
73 delta_H_s = Cp_0*(T_exit - T_1) + (Z - 1)*R*T_exit -
    a/V - (Z_1-1)*R*T_1 + a/V_1;//[J/mol]
74 W = - delta_H_s;//[J/mol]
75
76 printf(" (1).The exit temperature is %f K\n",Texit);
77 printf("     The work required is given by, W = %f J
    /mol\n\n",W);
78
79 //(2)
80 eff = 0.8; // Adiabatic efficiency
81 delta_H_a = eff*delta_H_s;//[J/mol] - Actual
    enthalpy change
82 W_2 = - delta_H_a;
83
84 // Let us assume a temperature, say
85 T_prime= 900;//[K]
86 fault1=10;
87
88 while(fault1>0.3)
89     // At T_prime and 8 MPa
90     deff(' [y]=f2(V) ', 'y=V^(3)-(b+(R*T_prime)/P_2)*V
        ^ (2)+(a/P_2)*V-(a*b)/P_2 ');
91     V_prime=fsolve(1,f2);
92     Z_prime = (P_2*V_prime)/(R*T_prime);
93
94     deff(' [y]=f3(T_prime) ', 'y = Cp_0*(T_prime - T_1)
        + (Z_prime - 1)*R*T_prime - a/V_prime -
        13230.49 ');

```



```

95     T_exit1 = fsolve(100,f3);
96     fault1=abs(T_prime-T_exit1);
97     T_prime = T_prime + 0.2;
98 end
99 Texit1 = T_prime;
100
101 printf(" (2).The exit temperature is %f K\n",Texit1)
    ;
102 printf("      The work required is given by, W = %f J
    /mol\n\n",W_2);

```

Scilab code Exa 10.18 Proving a mathematical relation

```

1 clear;
2 clc;
3
4 //Example - 10.18
5 //Page number - 358
6 printf("Example - 10.18 and Page number - 358\n\n");
7
8 //This problem involves proving a relation in which
    no mathematical components are involved.
9 //For prove refer to this example 10.18 on page
    number 358 of the book.
10 printf(" This problem involves proving a relation in
    which no mathematical components are involved.\n
    \n");
11 printf(" For prove refer to this example 10.18 on
    page number 358 of the book.")

```

Scilab code Exa 10.19 Calculation of molar volume and fugacity

```

1 clear;

```

```

2  clc;
3  funcprot(0);
4
5  //Example - 10.19
6  //Page number - 359
7  printf("Example - 10.19 and Page number - 359\n\n");
8
9  //Given
10 T = 100 + 273.15; // [K] - Temperature
11 Tc = 647.1; // [K] - Critical temperature of water
12 Pc = 220.55; // [bar] - Critical pressure of water
13 Pc = Pc*10^(5); // [Pa]
14 R = 8.314; // [J/mol*K] - Universal gas constant
15
16 // For van der Walls equation of state
17 a = (27*R^(2)*Tc^(2))/(64*Pc); // [Pa-m^(6)/mol^(2)]
18 b = (R*Tc)/(8*Pc); // [m^(3)/mol]
19
20 // The cubic form of van der Walls equation of state
    is given by,
21 //  $V^3 - (b + (R*T)/P)*V^2 + (a/P)*V - (a*b)/P = 0$ 
22
23 // For water vapour at 100 C under saturated
    conditions pressure is 1 atm, therefore
24 P = 1; // [atm]
25 P = P*101325; // [Pa]
26
27 // At 100 C and 1 atm
28 deff(' [y]=f(V) ', 'y=V^(3)-(b+(R*T)/P)*V^(2)+(a/P)*V-(
    a*b)/P');
29 V_1 = fsolve(0.1,f);
30 V_1 = fsolve(10,f);
31 V_1 = fsolve(100,f);
32 // The largest root is considered because of molar
    volume of vapour phase is to determined
33 V = V_1; // [m^(3)/mol]
34

```

```

35 // Now the figacity is given by
36 //  $\log(f/P) = \log((R*T)/(P*(V-b))) + b/(V-b) - (2*a)/(R$ 
     $/(R*T*V);$ 
37 f = P*(exp(log((R*T)/(P*(V-b))) + b/(V-b) - (2*a)/(R
    *T*V))); // [Pa]
38 f = f/101325; // [atm]
39
40 printf(" The molar volume is %f m3/mol\n\n",V);
41 printf(" The fugacity is %f atm\n\n",f);

```

Scilab code Exa 10.20 Calculation of enthalpy and entropy change

```

1 clear;
2 clc;
3 funcprot(0);
4
5 //Example - 10.20
6 //Page number - 359
7 printf("Example - 10.20 and Page number - 359\n\n");
8
9 //Given
10 P_1 = 6; // [bar] - Initial pressure
11 P_1 = P_1*10(5); // [Pa]
12 T_1 = 100 + 273.15; // [T] - Initial temperature
13 P_2 = 12; // [bar] - Final pressure
14 P_2 = P_2*10(5); // [Pa]
15 T_2 = 500 + 273.15; // [K] - Final temperature
16 R = 8.314; // [J/mol*K] - Universal gas constant
17 Y = 1.126; // Index of expansion
18 Cp_0 = (R*Y)/(Y-1); // [J/mol-K]
19
20 // For propane
21 Tc = 369.8; // [K]
22 Pc = 42.48; // [bar]
23 Pc = Pc*10(5);

```

```

24 w = 0.152;
25
26 //(1)
27 // For van der Walls equation of state
28 a = (27*R^(2)*Tc^(2))/(64*Pc); // [Pa-m^(6)/mol^(2)]
29 b = (R*Tc)/(8*Pc); // [m^(3)/mol]
30
31 // The cubic form of van der Walls equation of state
    is given by,
32 // V^(3) - (b + (R*T)/P)*V^(2) + (a/P)*V - (a*b)/P =
    0
33
34 // At state 1 (100 C and 6 bar)
35 def f(' [y]=f(V) ', 'y=V^(3)-(b+(R*T_1)/P_1)*V^(2)+(a/
    P_1)*V-(a*b)/P_1 ');
36 V_1_1 = fsolve(1,f);
37 V_1_2 = fsolve(10,f);
38 V_1_3 = fsolve(100,f);
39 // The largest root is considered because of molar
    volume of vapour phase is to determined
40 V_1 = V_1_1; // [m^(3)/mol]
41 // Thus compressibility factor is
42 Z_1 = (P_1*V_1)/(R*T_1); // compressibility factor
43
44 H_R_1 = (Z_1 - 1)*R*T_1 - (a/V_1); // [J/mol]
45 S_R_1 = R*log((P_1*(V_1-b))/(R*T_1)); // [J/mol-K]
46
47 // At state 2 (500 C and 12 bar)
48 def f1(' [y]=f1(V) ', 'y=V^(3)-(b+(R*T_2)/P_2)*V^(2)+(a/
    P_2)*V-(a*b)/P_2 ');
49 V_2_1 = fsolve(1,f1);
50 V_2_2 = fsolve(10,f1);
51 V_2_3 = fsolve(100,f1);
52 // The largest root is considered because of molar
    volume of vapour phase is to determined
53 V_2 = V_2_1; // [m^(3)/mol]
54 // Thus compressibility factor is
55 Z_2 = (P_2*V_2)/(R*T_2); // compressibility factor

```

```

56
57 H_R_2 = (Z_2 - 1)*R*T_2 - (a/V_2); // [J/mol]
58 S_R_2 = R*log((P_2*(V_2-b))/(R*T_2)); // [J/mol-K]
59
60 // Ideal gas entropy change is given by
61 delta_S_ig = Cp_0*log(T_2/T_1) - R*log(P_2/P_1); // [J
    /mol-K]
62 // Entropy change is given by
63 delta_S = delta_S_ig + (S_R_2 - S_R_1); // [J/mol-k]
64
65 // Ideal gas enthalpy change is given by
66 delta_H_ig = Cp_0*(T_2 - T_1); // [J/mol]
67 // Enthalpy change is given by
68 delta_H = delta_H_ig + (H_R_2 - H_R_1); // [J/mol]
69
70 printf("(1).The change in enthalpy is %f J/mol\n",
    delta_H);
71 printf("    The change in entropy is %f J/mol-K\n\n"
    ,delta_S);
72
73 //(2)
74 // Virial equation of state
75
76 // At state 1 (372.15 K, 6 bar) let us calculate B
    and dB/dT
77 Tr = T_1/Tc; // Reduced temperature
78 B_0 = 0.083-(0.422/(Tr)^(1.6));
79 B_1 = 0.139-(0.172/(Tr)^(4.2));
80
81 //We know, (B*Pc)/(R*Tc) = B_0 + (w*B_1)
82 B = ((B_0+(w*B_1))*(R*Tc))/Pc; // [m^3/mol]
83 dB0_dT = 0.422*1.6*Tc^(1.6)*T_1^(-2.6); // (dB_0/dT)
    at state 1
84 dB1_dT = 0.172*4.2*Tc^(4.2)*T_1^(-5.2); // (dB_1/dT)
    at state 1
85 dB_dT = ((R*Tc)/Pc)*((dB0_dT) + w*(dB1_dT)); // (dB/
    dT) at state 1
86

```

```

87 H_R_1_2 = B*P_1 - P_1*T_1*dB_dT; //[J/mol] - Residual
    enthalpy at state 1
88 S_R_1_2 = -P_1*(dB_dT); //[J/mol-K] - Residual
    entropy at state 1
89
90 // At state 2 (773.15 K, 12 bar)
91 Tr_2 = T_2/Tc; // Reduced temperature
92 B_0_2 = 0.083-(0.422/(Tr_2)^(1.6));
93 B_1_2 = 0.139-(0.172/(Tr_2)^(4.2));
94
95 //We know, (B*Pc)/(R*Tc) = B_0 + (w*B_1)
96 B_2 = ((B_0_2+(w*B_1_2))*(R*Tc))/Pc; // [m^(3)/mol]
97 dB0_dT_2 = 0.422*1.6*Tc^(1.6)*T_2^(-2.6); // (dB_0/dT
    ) at state 1
98 dB1_dT_2 = 0.172*4.2*Tc^(4.2)*T_2^(-5.2); // (dB_1/dT
    ) at state 1
99 dB_dT_2 = ((R*Tc)/Pc)*((dB0_dT_2) + w*(dB1_dT_2)); //
    (dB/dT) at state 1
100
101 H_R_2_2 = B_2*P_2 - P_2*T_2*dB_dT_2; //[J/mol] -
    Residual enthalpy at state 1
102 S_R_2_2 = -P_2*(dB_dT_2); //[J/mol-K] - Residual
    entropy at state 1
103
104 delta_H_2 = delta_H_ig + (H_R_2_2 - H_R_1_2); //[J/
    mol]
105 delta_S_2 = delta_S_ig + (S_R_2_2 - S_R_1_2); //[J/
    mol]
106
107 printf("(2).The change in enthalpy is %f J/mol\n",
    delta_H_2);
108 printf("    The change in entropy is %f J/mol-K\n",
    delta_S_2);

```

Scilab code Exa 10.21 Calculation of fugacity

```

1 clear;
2 clc;
3
4 //Example - 10.21
5 //Page number - 362
6 printf("Example - 10.21 and Page number - 362\n\n");
7
8 //Given
9 P = 2.76*10^(6); // [N/m^(2)] - Pressure
10 T = 310.93; // [K] - Temperature
11 R = 8.314; // [J/mol*K] - Universal gas constant
12
13 // For n-butane
14 Tc = 425.18; // [K] - Critical temperature
15 Pc = 37.97; // [bar] - Critical pressure
16 Pc = Pc*10^(5); // [Pa]
17 w = 0.193;
18 den = 0.61; // [g/cm^(3)]
19 mol_wt = 58; // [g/mol] - Molecular weight of butane
20
21 // log(P_sat) = 15.7374 - 2151.63/(T-36.24)
22 P_sat = exp(15.7374 - 2151.63/(T-36.24)); // [mm Hg]
23 P_sat = (P_sat/760)*101325; // [N/m^(2)]
24
25 //(1)
26 // Let us determine the second virial coefficient at
    310.93 K
27 Tr = T/Tc; // Reduced temperature
28 B_0 = 0.083-(0.422/(Tr)^(1.6));
29 B_1 = 0.139-(0.172/(Tr)^(4.2));
30 //We know, (B*Pc)/(R*Tc) = B_0 + (w*B_1)
31 B = ((B_0+(w*B_1))*(R*Tc))/Pc; // [m^(3)/mol]
32
33 // Fugacity under saturated conditions is given by
34 // log(f_sat/P_sat) = (B*P_sat)/(R*T)
35 f_sat = P_sat*(exp((B*P_sat)/(R*T))); // [N/m^(2)]
36
37 // The molar volume is given by

```

```

38 V_liq = (1/(den*1000))*(mol_wt/1000); // [m^(3)/mol]
39
40 f = f_sat*exp(V_liq*(P-P_sat)/(R*T));
41
42 printf(" (1).The fugacity of n-butane is %e N/m^(2)\
      n\n",f);
43
44 //(2)
45 // For van der Walls equation of state
46 a = (27*R^(2)*Tc^(2))/(64*Pc); // [Pa-m^(6)/mol^(2)]
47 b = (R*Tc)/(8*Pc); // [m^(3)/mol]
48
49 // The cubic form of van der Walls equation of state
      is given by,
50 // V^(3) - (b + (R*T)/P)*V^(2) + (a/P)*V - (a*b)/P =
      0
51
52 // At 100 C and 1 atm
53 def f(' [y]=f(V) ', 'y=V^(3)-(b+(R*T)/P)*V^(2)+(a/P)*V-(
      a*b)/P');
54 V_1 = fsolve(0.1,f);
55 V_1 = fsolve(10,f);
56 V_1 = fsolve(100,f);
57 // The above equation has only 1 real root , other
      two roots are imaginary
58 V = V_1; // [m^(3)/mol]
59
60 // log(f/P) = log((R*T)/(P*(V-b))) + b/(V-b) -(2*a)
      /(R*T*V)
61 f_2 = P*(exp(log((R*T)/(P*(V-b))) + b/(V-b) -(2*a)/(
      R*T*V)));
62
63 printf(" (2).The fugacity of n-butane is %e N/m^(2)\
      n\n",f_2);

```

Scilab code Exa 10.22 Calculation of enthalpy change

```
1 clear;
2 clc;
3
4 //Example - 10.22
5 //Page number - 363
6 printf("Example - 10.22 and Page number - 363\n\n");
7
8 //Given
9 T = 50+273.15; // [K] - Temperature
10 P = 25*10^(3); // [Pa] - Pressure
11 y1 = 0.5; // [mol] - mole fraction of equimolar
    mixture
12 y2 = 0.5;
13 R = 8.314; // [J/mol*K] - Universal gas constant
14
15 //For component 1 (methyl ethyl ketone)
16 Tc_1 = 535.5; // [K] - Critical temperature
17 Pc_1 = 41.5*10^(5); // [N/m^(2)] - Critical pressure
18 Vc_1 = 267; // [cm^(3)/mol] - Critical volume
19 Zc_1 = 0.249; // Critical compressibility factor
20 w_1 = 0.323; // acentric factor
21
22 //For component 2 (toluene)
23 Tc_2 = 591.8; // [K]
24 Pc_2 = 41.06*10^(5); // [N/m^(2)]
25 Vc_2 = 316; // [cm^(3)/mol]
26 Zc_2 = 0.264;
27 w_2 = 0.262;
28
29 // For equation of state  $Z = 1 + B/V$ 
30 //For component 1, let us calculate B and dB/dT
31 Tr_1 = T/Tc_1; //Reduced temperature
32 //At reduced temperature
33 B1_0 = 0.083-(0.422/(Tr_1)^(1.6));
34 B1_1 = 0.139-(0.172/(Tr_1)^(4.2));
35 //We know,  $(B*Pc)/(R*Tc) = B_0+(w*B_1)$ 
```

```

36 B_11 = ((B1_0+(w_1*B1_1))*(R*Tc_1))/Pc_1; // [m^(3)/
      mol-K]
37 dB0_dT_1 = 0.422*1.6*Tc_1^(1.6)*T^(-2.6); // [m^(3)/
      mol-K] - (dB_0/dT)
38 dB1_dT_1 = 0.172*4.2*Tc_1^(4.2)*T^(-5.2); // [m^(3)/
      mol-K] - (dB_1/dT)
39 dB_dT_1 = ((R*Tc_1)/Pc_1)*((dB0_dT_1) + w_1*(
      dB1_dT_1)); // [m^(3)/mol-K] - (dB/dT) -
40
41 //Similarly for component 2
42 Tr_2 = T/Tc_2; //Reduced temperature
43 //At reduced temperature Tr_2 ,
44 B2_0 = 0.083 - (0.422/(Tr_2)^(1.6));
45 B2_1 = 0.139 - (0.172/(Tr_2)^(4.2));
46 B_22 = ((B2_0+(w_2*B2_1))*(R*Tc_2))/Pc_2; // [m^(3)/
      mol]
47 dB0_dT_2 = 0.422*1.6*Tc_2^(1.6)*T^(-2.6); // [m^(3)/
      mol-K] - (dB_0/dT)
48 dB1_dT_2 = 0.172*4.2*Tc_2^(4.2)*T^(-5.2); // [m^(3)/
      mol-K] - (dB_1/dT)
49 dB_dT_2 = ((R*Tc_2)/Pc_2)*((dB0_dT_2) + w_2*(
      dB1_dT_2)); // [m^(3)/mol-K] - (dB/dT) -
50
51 //For cross coefficient , let us calculate B and dB/dT
52 Tc_12 = (Tc_1*Tc_2)^(1/2); // [K]
53 w_12 = (w_1 + w_2)/2;
54 Zc_12 = (Zc_1 + Zc_2)/2;
55 Vc_12 = (((Vc_1)^(1/3) + (Vc_2)^(1/3))/2)^(3); // [cm
      ^3)/mol]
56 Vc_12 = Vc_12*10^(-6); // [m^(3)/mol]
57 Pc_12 = (Zc_12*R*Tc_12)/Vc_12; // [N/m^(2) ]
58
59 //Now we have ,(B_12*Pc_12)/(R*Tc_12) = B_0+(w_12*B_1
      )
60 //where B_0 and B_1 are to be evaluated at Tr_12
61 Tr_12 = T/Tc_12;
62 //At reduced temperature Tr_12
63 B_0 = 0.083 - (0.422/(Tr_12)^(1.6));

```

```

64 B_1 = 0.139 - (0.172/(Tr_12)^(4.2));
65 B_12 = ((B_0 + (w_12*B_1))*R*Tc_12)/Pc_12; // [m^(3)/
      mol]
66 dB0_dT_12 = 0.422*1.6*Tc_12^(1.6)*T^(-2.6); // [m^(3)
      /mol-K] - (dB_0/dT)
67 dB1_dT_12 = 0.172*4.2*Tc_12^(4.2)*T^(-5.2); // [m^(3)
      /mol-K] - (dB_1/dT)
68 dB_dT_12 = ((R*Tc_12)/Pc_12)*((dB0_dT_12) + w_12*(
      dB1_dT_12)); // [m^(3)/mol-K] - (dB/dT)_12
69
70 //For the mixture
71 B = y1^(2)*B_11 + 2*y1*y2*B_12 + y2^(2)*B_22; // [m
      ^ (3)/mol]
72
73 // The equation of state can be written as
74 // V^(2) - ((R*T)/P) - (B*R*T)/P = 0
75 // V^(2) - 0.1075*V + 1.737*10^(-4) = 0
76 deff(' [y]=f(V) ', 'y=V^(2) - 0.1075*V + 1.737*10^(-4) ',
      );
77 V1 = fsolve(0.1,f);
78 V2 = fsolve(1,f);
79 // We will consider the root which is near to R*T/P
80 V = V1;
81 // dB/dT = y_1^(2)*dB_11/dT + y_2^(2)*dB_22/dT + 2*
      y_1*y_2*dB_12/dT
82 dB_dT = y1^(2)*dB_dT_1 + y2^(2)*dB_dT_2 + 2*y1*y2*
      dB_dT_12; // [m^(3)/mol-K]
83
84 // For equation of state Z = 1 + B/V
85 H_R = (B*R*T)/V - ((R*T^(2))/V)*dB_dT; // [J/mol]
86
87 printf(" (1).The value of H_R for the mixture using
      virial equation of state is %f J/mol\n\n",H_R);
88
89 //(2)
90 // For van der Walls equation of state
91 a_11 = (27*R^(2)*Tc_1^(2))/(64*Pc_1); // [Pa-m^(6)/mol
      ^ (2)]

```

```

92 a_22 = (27*R^(2)*Tc_2^(2))/(64*Pc_2); // [Pa-m^(6)/mol
    ^ (2)]
93 a_12 = (a_11*a_22)^(1/2);
94 b_1 = (R*Tc_1)/(8*Pc_1); // [m^(3)/mol]
95 b_2 = (R*Tc_2)/(8*Pc_2); // [m^(3)/mol]
96
97 // For the mixture
98 a = y1^(2)*a_11 + y2^(2)*a_22 + 2*y1*y2*a_12; // [Pa-m
    ^ (6)/mol^(2)]
99 b = y1*b_1 + y2*b_2; // [m^(3)/mol]
100
101 // From the cubic form of van der Walls equation of
    state
102 def f(' [y]=f1 (V) ', 'y=V^(3)-(b+(R*T)/P)*V^(2)+(a/P)*V
    -(a*b)/P');
103 V2_1 = fsolve(0.1, f1);
104 V2_2 = fsolve(10, f1);
105 V2_3 = fsolve(100, f1);
106 // The largest root is considered
107 V_2 = V2_1;
108
109 // The residual enthalpy is given by
110 H_R_2 = P*V_2 - R*T - a/V_2; // [J/mol]
111
112 printf(" (2).The value of H_R for the mixture using
    van der Walls equation of state is %f J/mol\n\n",
    H_R_2);

```

Scilab code Exa 10.23 Calculation of fugacity of water vapour

```

1 clear;
2 clc;
3
4 //Example - 10.23
5 //Page number - 366

```

```

6 printf("Example - 10.23 and Page number - 366\n\n");
7
8 //Given
9 T = 320 + 273.15; // [K]
10 R = 8.314; // [J/mol*K] - Universal gas constant
11
12 // For water
13 Tc = 647.1; // [K]
14 Pc = 220.55; // [bar]
15 Pc = Pc*10^(5); // [Pa]
16
17 // The cubic form of Redlich Kwong equation of state
    is given by,
18 //  $V^3 - ((R*T)/P)*V^2 - ((b_1^2)) + ((b_1*R*T)
    /P) - (a/(T^{1/2}*P))*V - (a*b)/(T^{1/2}*P) = 0$ 
19
20 // At 320 C and 70 bar pressure
21 P_1 = 70; // [bar]
22 P_1 = P_1*10^(5); // [Pa]
23
24 a = (0.42748*(R^(2))*(Tc^(2.5)))/Pc; // [Pa*m^(6)*K
    ^{1/2}/mol]
25 b = (0.08664*R*Tc)/Pc; // [m^(3)/mol]
26 // Solving the cubic equation
27 def f(y)=f1(V)', 'y=V^3 - ((R*T)/P_1)*V^2 - ((b^2))
    + ((b*R*T)/P_1) - (a/(T^{1/2}*P_1))*V - (a*b)/(T
    ^{1/2}*P_1)');
28 V1=fsolve(1,f1);
29 V2=fsolve(10,f1);
30 V3=fsolve(100,f1);
31 // The largest root is considered because at 320 C
    and 70 bar vapour phase exists.
32 V_1 = V1; // [m^(3)/mol]
33 // Thus compressibility factor is
34 Z_1 = (P_1*V_1)/(R*T);
35
36 // For Redlich-Kwong equation of state
37 //  $\log(f/P) = Z - 1 - \log(V_1/(V_1-b)) + (a/(b*R*(T$ 

```

```

    ^ (3/2))) * log(V/(V+b))
38 f_1 = P_1 * (exp(Z_1 - 1 - log(Z_1) + log(V_1/(V_1 - b))) + (a/(b
    * R * (T^(3/2)))) * log(V_1/(V_1 + b))))); // [Pa]
39 f_1 = f_1 * 10^(-5); // [bar]
40
41 printf(" The fugacity of water vapour at 320 C and
    70 bar pressure is %f bar\n\n", f_1);
42
43 // At 320 C and 170 bar pressure , we have
44 P_2 = 170; // [bar]
45 P_2 = P_2 * 10^(5); // [Pa]
46
47 // Solving the cubic equation
48 def(' [y]=f2(V) ', 'y=V^(3) - ((R*T)/P_2)*V^(2) - ((b^(2))
    + ((b*R*T)/P_2) - (a/(T^(1/2)*P_2))) * V - (a*b)/(T
    ^ (1/2)*P_2) ');
49 V4 = fsolve(1, f2);
50 V5 = fsolve(10, f2);
51 V6 = fsolve(100, f2);
52 // The above equation has only 1 real root , other two
    roots are imaginary . Therefore ,
53 V_2 = V6; // [m^(3)/mol]
54 // Thus compressibility factor is
55 Z_2 = (P_2 * V_2) / (R * T);
56
57 // For Redlich-Kwong equation of state
58 // log(f/P) = Z - 1 - log(V_1/(V_1 - b)) + (a/(b * R * (T
    ^ (3/2)))) * log(V/(V + b))
59 f_2 = P_2 * (exp(Z_2 - 1 - log(Z_2) + log(V_2/(V_2 - b))) + (a/(b
    * R * (T^(3/2)))) * log(V_2/(V_2 + b))))); // [Pa]
60 f_2 = f_2 * 10^(-5); // [bar]
61
62 printf(" The fugacity of water vapour at 320 C and
    170 bar pressure is %f bar\n\n", f_2);

```

Scilab code Exa 10.24 Determination of change in internal energy

```
1 clear;
2 clc;
3
4 //Example - 10.24
5 //Page number - 367
6 printf("Example - 10.24 and Page number - 367\n\n");
7
8 //Given
9 Vol = 0.057; //[m^(3)] - Volume of car tyre
10 P_1 = 300; //[kPa] - Initial pressure
11 P_1 = P_1*10^(3); //[Pa]
12 T_1 = 300; //[K] - Initial temperature
13 P_2 = 330; //[kPa] - Final pressure
14 P_2 = P_2*10^(3); //[Pa]
15 R = 8.314; //[J/mol*K] - Universal gas constant
16 Cv_0 = 21; //[J/mol-K] - Heat capacity for air
17
18 // For oxygen
19 Tc_O2 = 154.6; //[K] - Critical temperature
20 Pc_O2 = 50.43; //[bar] - Critical pressure
21 Pc_O2 = Pc_O2*10^(5); //[Pa]
22 y1 = 0.21; // - Mole fraction of oxygen
23 // For nitrogen
24 Tc_N2 = 126.2; //[K] - Critical temperature
25 Pc_N2 = 34.00; //[bar] - Critical pressure
26 Pc_N2 = Pc_N2*10^(5); //[Pa]
27 y2 = 0.79; // - Mole fraction of nitrogen
28
29 // (1)
30 // Assuming ideal gas behaviour. The volume remains
    the same, therefore, we get
31 //  $P_1/T_1 = P_2/T_2$ 
32 T_2 = P_2*(T_1/P_1); //[K]
33
34 n = (P_1*Vol)/(R*T_1); //[mol] - Number of moles
35 delta_U = n*Cv_0*(T_2-T_1); //[J]
```

```

36
37 printf(" (1).The change in internal energy (for
    ideal gas behaviour) is %f J\n\n",delta_U);
38
39 //(2)
40 // For van der Walls equation of state
41 a_O2 = (27*R^(2)*Tc_O2^(2))/(64*Pc_O2);//[Pa-m^(6)/
    mol^(2)]
42 a_N2 = (27*R^(2)*Tc_N2^(2))/(64*Pc_N2);//[Pa-m^(6)/
    mol^(2)]
43 a_12 = (a_O2*a_N2)^(1/2);
44 b_O2 = (R*Tc_O2)/(8*Pc_O2);//[m^(3)/mol]
45 b_N2 = (R*Tc_N2)/(8*Pc_N2);//[m^(3)/mol]
46
47 // For the mixture
48 a = y1^(2)*a_O2 + y2^(2)*a_N2 + 2*y1*y2*a_12;//[Pa-m
    ^6)/mol^(2)]
49 b = y1*b_O2 + y2*b_N2;//[m^(3)/mol]
50
51 // From the cubic form of van der Walls equation of
    state
52 // At 300 K and 300 kPa,
53 deff(' [y]=f1(V) ', 'y=V^(3)-(b+(R*T_1)/P_1)*V^(2)+(a/
    P_1)*V-(a*b)/P_1 ');
54 V_1 = fsolve(0.1,f1);
55 V_2 = fsolve(10,f1);
56 V_3 = fsolve(100,f1);
57 // The above equation has only 1 real root, other
    two roots are imaginary
58 V = V_1;//[m^(3)/mol]
59
60 // Now at P = 330 kPa and at molar volume V
61 // The van der Walls equation of state is
62 // (P + a/V^(2))*(V - b) = R*T
63 T_2_prime = ((P_2 + a/V^(2))*(V - b))/R;//[K]
64 n_prime = Vol/V;//[mol]
65
66 // Total change in internal energy is given by

```



```

67 // delta_U_prime = n_prime*delta_U_ig + n_prime*(
    U_R_2 - U_R_1)
68 // delta_U_prime = n_prime*Cv_0*(T_2_prime - T_1) +
    n_prime*a(1/V_2 - 1/V_1);
69 // Since V_1 = V_2 = V, therefore
70 delta_U_prime = n_prime*Cv_0*(T_2_prime - T_1);
71
72 printf(" (2).The change in internal energy (for van
    der Walls equation of state) is %f J\n\n",
    delta_U_prime);

```

Scilab code Exa 10.25 Calculation of enthalpy and entropy change

```

1 clear;
2 clc;
3 funcprot(0);
4
5 //Example - 10.25
6 //Page number - 369
7 printf("Example - 10.25 and Page number - 369\n\n");
8
9 //Given
10 T_1 = 150 + 273.15; //[K] - Initial emperature
11 T_2 = T_1; // Isothermal process
12 P_1 = 100*10^(3); //[Pa] - Initial pressure
13 P_2 = 450*10^(3); //[Pa] - Final pressure
14 R = 8.314; //[J/mol*K] - Universal gas constant
15 // For water
16 Tc = 647.1; //[K] - Critical temperature
17 Pc = 220.55; //[bar] - Critical pressure
18 Pc = Pc*10^(5);
19 w = 0.345;
20 Mol_wt = 18.015; //[g/mol] - Molecular weight of
    water
21 Cp_0 = 4.18; //[J/mol-K] - Standard heat capacity of

```

```

    water
22
23 // Both phases are superheated vapour phases because
    at 150 C the vapour pressure of steam is 4.67
    bar and both operating pressures are below
    saturated pressure.
24 // In Peng–Robinson equation of state
25 m = 0.37464 + 1.54226*w - 0.26992*w^(2);
26 // At T_1 and P_1, we have
27 Tr = T_1/Tc;
28 alpha = (1 + m*(1 - Tr^(1/2)))^(2);
29 a = ((0.45724*(R*Tc)^(2))/Pc)*alpha; // [Pa*m^(6)/mol
    ^ (2)]
30 b = (0.07780*R*Tc)/Pc; // [m^(3)/mol]
31
32 // Cubic form of Peng–Robinson equation of stste is
    given by
33 // V^(3)+(b-(R*T)/P)*V^(2)-((3*b^(2))+((2*R*T*b)/P)
    -(a/P))*V+b^(3)+((R*T*(b^(2)))/P)-((a*b)/P)=0;
34 // Solving the cubic equation
35 deff ('[y]=f(V)', 'y=V^(3)+(b-(R*T_1)/P_1)*V^(2)-((3*b
    ^ (2))+((2*R*T_1*b)/P_1)-(a/P_1))*V+b^(3)+((R*T_1
    *(b^(2)))/P_1)-((a*b)/P_1)');
36 V1 = fsolve(-1,f);
37 V2 = fsolve(0,f);
38 V3 = fsolve(1,f);
39 //The largest root is for vapour phase,
40 //The largest root is only considered as the
    systemis gas
41 V_1 = V3; // [m^(3)/mol]
42 // Thus compressibility factor is
43 Z_1 = (P_1*V_1)/(R*T_1);
44
45 // At T_2 and P_2, we have
46 // Cubic form of Peng–Robinson equation of stste is
    given by
47 // V^(3)+(b-(R*T)/P)*V^(2)-((3*b^(2))+((2*R*T*b)/P)
    -(a/P))*V+b^(3)+((R*T*(b^(2)))/P)-((a*b)/P)=0;

```

```

48 // Solving the cubic equation
49 def f(' [y]=f(V) ', 'y=V^(3)+(b-(R*T_2)/P_2)*V^(2)-((3*b
      ^ (2)))+(2*R*T_2*b)/P_2)-(a/P_2)*V+b^(3)+((R*T_2
      *(b^(2)))/P_2)-((a*b)/P_2)');
50 V4 = fsolve(-1,f);
51 V5 = fsolve(0,f);
52 V6 = fsolve(1,f);
53 //The largest root is for vapour phase,
54 //The largest root is only considered as the
      systemis gas
55 V_2 = V6; // [m^(3)/mol]
56 // Thus compressibility factor is
57 Z_2 = (P_2*V_2)/(R*T_2);
58
59 // In the Peng-Robinson equation of stste
60 // da/dT = -(a*m)/((alpha*T*Tc)^(1/2))
61 // The residual enthalpy is given by
62 // H_R = R*T*(Z-1) + (((T*(da_dT))-a)/(2*2^(1/2)*b))
      *log((Z+(1+2^(1/2))*((P*b)/(R*T))))/(Z+(1-2^(1/2)
      *((P*b)/(R*T))))))
63
64 // At state 1
65 da_dT_1 = -(a*m)/((alpha*T_1*Tc)^(1/2)); // [Pa*m^(6)/
      mol^(2)]
66 H_R_1 = R*T_1*(Z_1-1) + (((T_1*(da_dT_1))-a)
      /(2*(2^(1/2))*b))*log((Z_1+(1+2^(1/2))*((P_1*b)/(
      R*T_1)))/(Z_1+(1-2^(1/2))*((P_1*b)/(R*T_1))));
67
68 // At state 2
69 da_dT_2 = -(a*m)/((alpha*T_2*Tc)^(1/2)); // [Pa*m^(6)/
      mol^(2)]
70 H_R_2 = R*T_2*(Z_2-1) + (((T_2*(da_dT_2))-a)
      /(2*2^(1/2)*b))*log((Z_2+(1+2^(1/2))*((P_2*b)/(R*
      T_1)))/(Z_2+(1-2^(1/2))*((P_2*b)/(R*T_1))));
71
72
73 // Since the temperature is the same, therefore ideal
      gas change in enthalpy is zero and thus

```

```

74 delta_H = H_R_2 - H_R_1; ; // [J/mol]
75 delta_H = delta_H/Mol_wt; // [kJ/kg]
76
77 // The residual entropy relation for a substance
    following Peng – Robinson equation of state ia
78 //  $S_R = R \cdot \log(Z - (P \cdot b)/(R \cdot T)) + (da_dT/(2 \cdot 2^{(1/2)} \cdot b)) \cdot \log((Z + (1 + 2^{(1/2)}) \cdot ((P \cdot b)/(R \cdot T))))/(Z + (1 - 2^{(1/2)}) \cdot ((P \cdot b)/(R \cdot T)))$ 
79
80 // The residual entropy at state 1 is
81  $S_{R_1} = R \cdot \log(Z_1 - (P_1 \cdot b)/(R \cdot T_1)) + (da_dT_1/(2 \cdot 2^{(1/2)} \cdot b)) \cdot \log((Z_1 + (1 + 2^{(1/2)}) \cdot ((P_1 \cdot b)/(R \cdot T_1)))/(Z_1 + (1 - 2^{(1/2)}) \cdot ((P_1 \cdot b)/(R \cdot T_1))))$ ;
82
83 // The residual entropy at state 2 is
84  $S_{R_2} = R \cdot \log(Z_2 - (P_2 \cdot b)/(R \cdot T_2)) + (da_dT_2/(2 \cdot 2^{(1/2)} \cdot b)) \cdot \log((Z_2 + (1 + 2^{(1/2)}) \cdot ((P_2 \cdot b)/(R \cdot T_2)))/(Z_2 + (1 - 2^{(1/2)}) \cdot ((P_2 \cdot b)/(R \cdot T_2))))$ ;
85
86 delta_S_R = S_R_2 - S_R_1; // [J/mol-K]
87
88 // The ideal gas change in entropy is
89  $\Delta S_{ig} = C_{p_0} \cdot \log(T_2/T_1) - R \cdot \log(P_2/P_1)$ ; // [J/mol-K]
90
91 // Therefore
92  $\Delta S = \Delta S_R + \Delta S_{ig}$ ; // [J/mol-K]
93
94 printf(" The enthalpy change is given by, delta_H =
    %f kJ/mol\n\n", delta_H);
95 printf(" The entropy change is given by, delta_S =
    %f J/mol-K\n\n", delta_S);

```

Scilab code Exa 10.26 Calculation of final temperature and pressure

```

1  clear;
2  clc;
3  funcprot(0);
4
5  //Example - 10.26
6  //Page number - 370
7  printf("Example - 10.26 and Page number - 370\n\n");
8
9  //Given
10 Vol = 0.15; //[m^(3)]
11 T_1 = 170; //[K] - Initial emperature
12 P_1 = 100; //[bar] - Initial pressure
13 P_1 = P_1*10^(5); //[Pa]
14 R = 8.314; //[J/mol*K] - Universal gas constant
15 // For nitrogen
16 Tc = 126.2; //[K] - Critical tempeature
17 Pc = 34; //[bar] - Critical pressure
18 Pc = Pc*10^(5); //[Pa]
19 w = 0.038;
20 // Cp_0 = 27.2+4.2*10^(-3)*T
21
22 //(1)
23 // For van der Walls equation of state
24 a = (27*R^(2)*Tc^(2))/(64*Pc); //[Pa-m^(6)/mol^(2)]
25 b = (R*Tc)/(8*Pc); //[m^(3)/mol]
26
27 // The cubic form of van der Walls equation of state
    is given by,
28 // V^(3) - (b + (R*T)/P)*V^(2) + (a/P)*V - (a*b)/P =
    0
29 // On simplification the equation changes to
30 // V^(3) - 1.799*10^(4)*V^(2) + 1.366*10^(-8)*V -
    5.269*10^(-13) = 0
31
32 // Solving the cubic equation
33 def(' [y]=f(V) ', 'y=V^(3) -1.799*10^(-4)*V^(2) +
    1.366*10^(-8)*V - 5.269*10^(-13) ');
34 V1 = fsolve(1,f);

```

```

35 V2 = fsolve(10,f);
36 V3 = fsolve(100,f);
37 // The above equation has only 1 real root, other
    two roots are imaginrary
38 V_1 = V1; //[m^(3)/mol]
39 // Thus total number of moles is given by
40 n_1 = Vol/V_1; //[mol]
41
42 // After 500 mol are withdrawn, the final number of
    moles is given by
43 n_2 = n_1 - 500; //[mol]
44 // Thus molar volume at final state is
45 V_2 = Vol/n_2; //[m^(3)/mol]
46
47 // The ideal entropy change is guven by
48 // delta_S_ig = integrate('27.2+4.2*10^(-3)*T', 'T',
    T_1, T_2) - R*log(P_2/P_1);
49 // The residual entropy change is given by
50 // delta_S_R = R*log((P_2*(V_2-b))/(R*T_2)) - R*log
    ((P_1*(V_1-b))/(R*T_1))
51 // delta_S = delta_S_ig = delta_S_R
52 // delta_S = integrate('27.2+4.2*10^(-3)*T', 'T', T_1,
    T_2) + R*log((V_2-b)/(V_1-b));
53 // During discharging delta_S = 0, thus on
    simplification we get
54 // 18.886*log(T_2) + 4.2*10^(-3)*T_2 - 92.937 = 0
55 // Solving the above equation we get
56 deff(' [y]=f1(T_2)', 'y=18.886*log(T_2) + 4.2*10^(-3)*
    T_2 - 92.937 ');
57 T_2 = fsolve(1,f1);
58
59 // Thus at T_2,
60 P_2 = (R*T_2)/(V_2-b) - a/V_2^(2); //[N/m^(2)]
61 P_2 = P_2*10^(-5); //[bar]
62
63 printf(" (1).The final temperature is %f K\n",T_2);
64 printf("      The final pressure is %f bar\n\n",P_2);
65

```

```

66 // (2)
67 // In Peng–Robinson equation of state
68 m = 0.37464 + 1.54226*w - 0.26992*w^(2);
69 // At T_1 and P_1, we have
70 Tr = T_1/Tc;
71 alpha = (1 + m*(1 - Tr^(1/2)))^(2);
72 a_2 = ((0.45724*(R*Tc)^(2))/Pc)*alpha; // [Pa*m^(6)/
      mol^(2)]
73 b_2 = (0.07780*R*Tc)/Pc; // [m^(3)/mol]
74
75 // Cubic form of Peng–Robinson equation of state is
      given by
76 // V^(3)+(b-(R*T)/P)*V^(2)-((3*b^(2))+((2*R*T*b)/P)
      -(a/P))*V+b^(3)+((R*T*(b^(2)))/P)-((a*b)/P)=0;
77 // Solving the cubic equation
78 def f2(V) = V^3 + (b_2 - (R*T_1)/P_1)*V^2 -
      ((3*b_2^2) + ((2*R*T_1*b_2)/P_1) - (a_2/P_1))*V +
      b_2^3 + ((R*T_1*(b_2^2))/P_1) - ((a_2*b_2)/P_1);
79 V4 = fsolve(-1, f2);
80 V5 = fsolve(0, f2);
81 V6 = fsolve(0.006, f2);
82 // The above equation has only 1 real root, the other
      two roots are imaginary
83 V_1_2 = V6; // [m^(3)/mol]
84
85 // The number of moles in the initial state is given
      by
86 n_1_2 = Vol/V_1_2; // [mol]
87 // After 500 mol are withdrawn, the final number of
      moles is given by
88 n_2_2 = n_1_2 - 500; // [mol]
89 // Thus molar volume at final state is
90 V_2_2 = Vol/n_2_2; // [m^(3)/mol]
91
92 // At the final state the relation between pressure
      and temperature is
93 // P_2_2 = (R*T_2_2)/(V_2_2-b_2) - a_2/V_2_2^(2)

```

```

94 // P_2_2 = 7.23*10^(4)*T_2 - 3.93*10^(7)*a_2
95
96 // Now let us calculate the residual entropy at
    initial state
97 Z_1 = (P_1*V_1_2)/(R*T_1);
98 da_dT_1 = -(a*m)/((alpha*T_1*Tc)^(1/2)); // [Pa*m^(6)/
    mol^(2)] - da/dT
99
100 // The residual entropy change for Peng–Robinson
    equatiob of state is given by
101 // S_R = R*log(Z-(P*b)/(R*T)) + (da_dT/(2*2^(1/2)*b)
    )*log((V+(1+2^(1/2))*b))/((V+(1-2^(1/2))*b)));
102 S_R_1 = R*(log(Z_1-(P_1*b_2)/(R*T_1))) + (da_dT_1
    /(2*2^(1/2)*b_2))*(log((V_1_2+(1+2^(1/2))*b_2)/(
    V_1_2+(1-2^(1/2))*b_2)));
103
104 // The total entropy change is given by
105 // delta_S = delta_S_ig + delta_S_R
106 // where, delta_S_ig = integrate('27.2+4.2*10^(-3)*T
    ', 'T', T_1, T_2_2) - R*log(P_2_2/P_1);
107 // and, P_2_2 = (R*T_2_2)/(V_2_2-b_2) - a_2/V_2_2
    ^2)
108 // On simplification we get
109 // delta_S = 27.2*log(T_2_2-T_1) + 4.2*10^(-3)*
    (T_2_2-T_1) - R*log(P_2_2/P_1) + R*log(Z_2-(P_2_2*
    b)/(R*T_2_2)) + 6226*(da_dT_2) + 9.22
110
111 // Now we have the determine the value of T_2_2 such
    that delta_S = 0
112 // Starting with a temperature of 150 K
113 T_prime = 100; // [K]
114 error = 10;
115 while(error>0.1)
116     Tr_prime = T_prime/Tc;
117     alpha_prime = (1 + m*(1 - Tr_prime^(1/2)))^2);
118     a_prime = ((0.45724*(R*Tc)^(2))/Pc)*alpha_prime;
119     P_prime = 7.23*10^(4)*T_prime - 3.93*10^(7)*
        a_prime;

```



```

120     Z_prime = (P_prime*V_2_2)/(R*T_prime);
121     da_dT_prime = -(a_prime*m)/((alpha_prime*T_prime
        *Tc)^(1/2));
122     delta_S = 27.2*log(T_prime/T_1) + 4.2*10^(-3)*
        (T_prime-T_1) - R*log(P_prime/P_1) + R*log(
        Z_prime-((P_prime*b_2)/(R*T_prime))) + 6226*(
        da_dT_prime) + 9.22;
123     error=abs(delta_S);
124     T_prime = T_prime + 0.3;
125 end
126
127 T_2_2 = T_prime; //[K] - Final temperature
128 P_2_2 = P_prime*10^(-5); //[bar] - Final pressure
129
130 printf(" (2).The final temperature is %f K\n",T_2_2)
    ;
131 printf("      The final pressure is %f bar\n",P_2_2);

```

Scilab code Exa 10.27 Calculation of vapour pressure

```

1 clear;
2 clc;
3 funcprot(0);
4
5 //Example - 10.27
6 //Page number - 374
7 printf("Example - 10.27 and Page number - 374\n\n");
8
9 //Given
10 T = 373.15; //[K]
11 Tc = 562.16; //[K]
12 Pc = 48.98; //[bar]
13 Pc = Pc*10^(5); //[Pa]
14 R = 8.314; //[J/mol-K] - Universal gas constant
15

```

```

16 // The cubic form of Redlich Kwong equation of state
    is given by,
17 //  $V^3 - ((R*T)/P)*V^2 - ((b_1^2)) + ((b_1*R*T)
    /P) - (a/(T^{1/2}*P))*V - (a*b)/(T^{1/2}*P) = 0$ 
18
19 a = (0.42748*(R^(2))*(Tc^(2.5)))/Pc; // [Pa*m^(6)*K
    ^{1/2}/mol]
20 b = (0.08664*R*Tc)/Pc; // [m^3/mol]
21
22 // At 373.15 K, let us assume the pressure to be 2.5
    bar and under these conditions
23 P_1 = 2.5; // [bar]
24 P_1 = P_1*10^(5); // [bar]
25
26 // Putting the values in Redlich Kwong equation of
    state, the equation becomes
27 //  $V^3 - 0.0124*V^2 + 8.326*10^{(-6)}*V -
    7.74*10^{(-10)} = 0$ 
28 // Solving the cubic equation
29
30 deff(' [y]=f(V) ', 'y=V^3 - 0.0124*V^2 +
    8.326*10^{(-6)}*V - 7.74*10^{(-10)} ');
31 V1=fsolve(-9,f);
32 V2=fsolve(10,f);
33 V3=fsolve(0.1,f);
34 // The largest root and the smallest root is
    considered for liquid phase and vapour phase
    respectively.
35 V_liq = V1; // [m^3/mol] - Molar volume in liquid
    phase
36 V_vap = V3; // [m^3/mol] - Molar volume in vapour
    phase
37
38 // Let us calculate the fugacity of vapour phase
39 //  $\log(f_{\text{vap}}/P) = b/(V-b) + \log((R*T)/(P*(V-b))) - (
    a/(R*T^{1.5}))* (1/(V+b) - (1/b)*\log(V/(V+b)))$ 
40 f_vap = P_1*exp(b/(V_vap-b) + log((R*T)/(P_1*(V_vap-
    b))) - (a/(R*T^(1.5)))*(1/(V_vap+b) - (1/b)*log(

```

```

    V_vap/(V_vap+b)))));//[Pa]
41
42 // Let us calculate the fugacity of the liquid phase
43 f_liq = P_1*exp(b/(V_liq-b) + log((R*T)/(P_1*(V_liq-
    b)))) - (a/(R*T^(1.5)))*(1/(V_liq+b) - (1/b)*log(
    V_liq/(V_liq+b)))));
44
45
46 // The two fugacities are not same; therefore
    another pressure is to be assumed. The new
    pressure is
47 P_new = P_1*(f_liq/f_vap);//[Pa]
48
49 // At P_new
50 deff(' [y]=f1(V) ', 'y=V^(3) - ((R*T)/P_new)*V^(2) - (b
    ^2) + ((b*R*T)/P_new) - a/(T^(1/2)*P_new))*V - (
    a*b)/(T^(1/2)*P_new) ');
51 V4=fsolve(-9,f1);
52 V5=fsolve(10,f1);
53 V6=fsolve(0.1,f1);
54 // The largest root and the smallest root is
    considered for liquid phase and vapour phase
    respectively.
55 V_liq_2 = V4;//[m^3/mol] - Molar volume in liquid
    phase
56 V_vap_2 = V6;//[m^3/mol] - Molar volume in vapour
    phase
57
58 f_vap_prime = P_new*exp(b/(V_vap_2-b) + log((R*T)/(
    P_new*(V_vap_2-b)))) - (a/(R*T^(1.5)))*(1/(V_vap_2
    +b) - (1/b)*log(V_vap_2/(V_vap_2+b)))));//[Pa]
59 f_liq_prime = P_new*exp(b/(V_liq_2-b) + log((R*T)/(
    P_new*(V_liq_2-b)))) - (a/(R*T^(1.5)))*(1/(V_liq_2
    +b) - (1/b)*log(V_liq_2/(V_liq_2+b)))));
60
61 // Since the fugacities of liquid and vapour
    phases are almost same the assumed pressure may be
    taken as vapour pressure at 373.15 K

```

```

62 P_new = P_new*10^(-5);//[bar]
63
64 printf(" The vapour pressure of benzene using
    Redlich Kwong equation of state is %f bar\n",
    P_new);

```

Scilab code Exa 10.28 Determination of vapour pressure

```

1 clear;
2 clc;
3 funcprot(0);
4
5 //Example - 10.28
6 //Page number - 374
7 printf("Example - 10.28 and Page number - 375\n\n");
8
9 //Given
10 T = 150 + 273.15;//[K]
11 Tc = 647.1;//[K]
12 Pc = 220.55;//[bar]
13 Pc = Pc*10^(5);//[Pa]
14 w = 0.345;
15 R = 8.314;//[J/mol-K] - Universal gas constant
16
17 // Let us assume a pressure of 100 kPa.
18 P_1 = 100*10^(3);//[Pa]
19
20 // At 100 kPa and 423.15 K, from Peng-Robinson
    equation of stste
21 m = 0.37464 + 1.54226*w - 0.26992*w^(2);
22 Tr = T/Tc;
23 alpha = (1 + m*(1 - Tr^(1/2)))^(2);
24 a = ((0.45724*(R*Tc)^(2))/Pc)*alpha;//[Pa*m^(6)/mol
    ^ (2)]
25 b = (0.07780*R*Tc)/Pc;//[m^(3)/mol]

```

```

26 // Cubic form of Peng–Robinson equation of state is
    given by
27 //  $V^3 + (b - (R*T)/P) * V^2 - ((3*b^2) + ((2*R*T*b)/P) - (a/P)) * V + b^3 + ((R*T*(b^2))/P) - ((a*b)/P) = 0;$ 
28 // Solving the cubic equation
29 def f(' [y]=f(V) ', 'y= $V^3 + (b - (R*T)/P_1) * V^2 - ((3*b^2) + ((2*R*T*b)/P_1) - (a/P_1)) * V + b^3 + ((R*T*(b^2))/P_1) - ((a*b)/P_1)$  ');
30 V1 = fsolve(-1,f);
31 V2 = fsolve(0,f);
32 V3 = fsolve(1,f);
33 // The largest root and the smallest root is
    considered for liquid phase and vapour phase
    respectively .
34 V_liq = V1; // [m^3/mol] – Molar volume in liquid
    phase
35 V_vap = V3; // [m^3/mol] – Molar volume in vapour
    phase
36
37 // The compressibility factor is given by
38 Z_vap = (P_1 * V_vap) / (R * T); // For liquid phase
39 Z_liq = (P_1 * V_liq) / (R * T); // For vapour phase
40
41 // The expression for fugacity of Peng Robinson
    equation is
42 //  $\log(f/P) = (Z-1) - \log(Z - ((P*b)/(R*T))) - (a / (2*2^{1/2} * b * R * T)) * \log((Z + (1 + 2^{1/2})) * ((P*b)/(R*T))) / ((Z + (1 - 2^{1/2})) * ((P*b)/(R*T)))$ 
43 // For vapour phase
44 f_P_vap = exp((Z_vap-1) - log(Z_vap - ((P_1*b)/(R*T))) - (a/(2*2^(1/2)*b*R*T))*log((Z_vap+(1+2^(1/2))*((P_1*b)/(R*T)))/(Z_vap+(1-2^(1/2))*((P_1*b)/(R*T)))));
45 // For liquid phase
46 f_P_liq = exp((Z_liq-1) - log(Z_liq - ((P_1*b)/(R*T))) - (a/(2*2^(1/2)*b*R*T))*log((Z_liq+(1+2^(1/2))*((P_1*b)/(R*T)))/(Z_liq+(1-2^(1/2))*((P_1*b)/(R*T)))));

```

```

47
48 // Therefore f_liq/f_vap can be calculated as
49 fL_fV = (f_P_liq/f_P_vap);
50
51 // The two values (f/P)_vap and (f/P)_vap are not
    same [ (f_P_liq/f_P_vap) >1 ]; therefore another
    pressure is to be assumed. The new pressure be
52 P_new = P_1*(f_P_liq/f_P_vap);//[Pa]
53
54 // At P_new and 423.15 K, from Peng-Robinson
    equation of stste
55
56 //  $V^3 + (b - (R*T)/P) * V^2 - ((3*b^2) + ((2*R*T*b)/P) - (a/P)) * V + b^3 + ((R*T*(b^2))/P) - ((a*b)/P) = 0$ ;
57 // Solving the cubic equation
58 def f ('[y]=f(V)', 'y=V^3+(b-(R*T)/P_new)*V^2-((3*b
    ^2))+((2*R*T*b)/P_new)-(a/P_new))*V+b^3+((R*T
    *(b^2))/P_new)-((a*b)/P_new)');
59 V4 = fsolve(-1,f);
60 V5 = fsolve(0,f);
61 V6 = fsolve(1,f);
62 // The largest root and the smallest root is
    considered for liquid phase and vapour phase
    respectively.
63 V_liq_2 = V4;//[m^3/mol] - Molar volume in liquid
    phase
64 V_vap_2 = V6;//[m^3/mol] - Molar volume in vapour
    phase
65
66 // The compressibility factor is given by
67 Z_vap_2 = (P_new*V_vap_2)/(R*T);// For liquid phase
68 Z_liq_2 = (P_new*V_liq_2)/(R*T);// For vapour phase
69
70 // For vapour phase
71 f_P_vap_2 = exp((Z_vap_2-1) - log(Z_vap_2-((P_new*b)
    /(R*T)))) - (a/(2*2^(1/2)*b*R*T))*log((Z_vap_2
    +(1+2^(1/2))*((P_new*b)/(R*T)))/(Z_vap_2
    +(1-2^(1/2))*((P_new*b)/(R*T))));

```

```

72 // For liquid phase
73 f_P_liq_2 = exp((Z_liq_2-1) - log(Z_liq_2-((P_new*b)
      /(R*T))) - (a/(2*2^(1/2)*b*R*T))*log((Z_liq_2
      +(1+2^(1/2))*((P_new*b)/(R*T)))/(Z_liq_2
      +(1-2^(1/2))*((P_new*b)/(R*T))));
74
75 // Therefore f_liq/f_vap can be calculated as
76 fL_fV_2 = (f_P_liq_2/f_P_vap_2);
77
78 // And new pressure is given by
79 P_new_prime = P_new*(f_P_liq_2/f_P_vap_2);//[Pa]
80 P_new_prime = P_new_prime*10^(-5);
81
82 // Since the change in pressure is small, so we can
      take this to be the vapour pressure at 150 C
83
84 printf(" The vapour pressure of water using Peng-
      Robinson equation of stste is %f bar\n",
      P_new_prime);

```

Chapter 11

Properties of a Component in a Mixture

Scilab code Exa 11.1 Determination of volumes of ethanol and water

```
1 clear;
2 clc;
3
4 //Example - 11.1
5 //Page number - 385
6 printf("Example - 11.1 and Page number - 385\n\n");
7
8 //Given
9 Vol_total = 3; //[m^(3)] - Total volume of solution
10 x_ethanol = 0.6; //Mole fraction of ethanol
11 x_water = 0.4; //Mole fraction of water
12
13 //The partial molar volumes of the components in the
    mixture are
14 V_ethanol_bar = 57.5*10^(-6); //[m^(3)/mol]
15 V_water_bar = 16*10^(-6); //[m^(3)/mol]
16
17 //The molar volumes of the pure components are
18 V_ethanol = 57.9*10^(-6); //[m^(3)/mol]
```



```

19 V_water = 18*10(-6);//[m(3)/mol]
20
21 //The molar volume of the solution is
22 V_sol = x_ethanol*V_ethanol_bar + x_water*
    V_water_bar;//[m(3)/mol]
23 //Total number of moles can be calculated as
24 n_total = Vol_total/V_sol;//[mol]
25
26 //Moles of the components are
27 n_ethanol = n_total*x_ethanol;//[mol]
28 n_water = n_total*x_water;//[mol]
29
30 //Finally the volume of the pure components required
    can be calculated as
31 Vol_ethanol = V_ethanol*n_ethanol;
32 Vol_water = V_water*n_water;
33
34 printf("Required volume of ethanol is %f cubic metre
    \n\n",Vol_ethanol);
35 printf("Required volume of water is %f cubic metre",
    Vol_water);

```

Scilab code Exa 11.2 Developing an expression

```

1 clear;
2 clc;
3
4 //Example - 11.2
5 //Page number - 385
6 printf("Example - 11.2 and Page number - 385\n\n");
7
8 //Given
9 T = 25+273.15;//[K] - Temperature
10 P = 1;//[atm]
11 //Component 1 = water

```

```

12 //component 2 = methanol
13 a = -3.2;//[cm^(3)/mol] - A constant
14 V2 = 40.7;//[cm^(3)/mol] - Molar volume of pure
    component 2 (methanol)
15 //V1_bar = 18.1 + a*x_2^(2)
16
17 //From Gibbs–Duhem equation at constant temperature
    and pressure we have
18 //x_1*dV1_bar + x_2*dV2_bar = 0
19 //dV2_bar = -(x_1/x_2)*dV1_bar = -(x_1/x_2)*a*2*x_2*
    dx_2 = -2*a*x_1*dx_2 = 2*a*x_1*dx_1
20
21 //At x_1 = 0: x_2 = 1 and thus V2_bar = V2
22 //Integrating the above equation from x_1 = 0 to x_1
    in the RHS, and from V2_bar = V2 to V2 in the
    LHS, we get
23 //V2_bar = V2 + a*x_1^(2) - Molar volume of
    component 2(methanol) in the mixture
24
25 printf("The expression for the partial molar volume
    of methanol(2) is\nV2_bar = V2 + a*x_1^(2) [cm
    ^ (3)/mol]\n\n");
26
27 //At infinite dilution , x_2 approach 0 and thus x_1
    approach 1, therefore
28 x_1 = 1; // Mole fraction of component 1(water) at
    infinite dilution
29 V2_bar_infinite = V2 + a*(x_1^(2));//[cm^(3)/mol]
30
31 printf("The partial molar volume of methanol at
    infinite dilution is %f cm^(3)/mol",
    V2_bar_infinite);

```

Scilab code Exa 11.3 Determination of partial molar volume

```

1 clear;
2 clc;
3
4 //Example - 11.3
5 //Page number - 386
6 printf("Example - 11.3 and Page number - 386\n\n");
7
8 //This problem involves proving a relation in which
   no mathematics and no calculations are involved.
9 //For prove refer to this example 11.3 on page
   number 386 of the book.
10 printf(" This problem involves proving a relation in
   which no mathematics and no calculations are
   involved.\n\n");
11 printf(" For prove refer to this example 11.3 on
   page number 386 of the book.")

```

Scilab code Exa 11.4 Calculation of enthalpies

```

1 clear;
2 clc;
3
4 //Example - 11.4
5 //Page number - 387
6 printf("Example - 11.4 and Page number - 387\n\n");
7
8 //Given
9 //H = a*x_1 + b*x_2 +c*x_1*x_2
10
11 //The values of the constants are
12 a = 15000; //[J/mol]
13 b = 20000; //[J/mol]
14 c = -2000; //[J/mol]
15
16 //(1)

```

```

17 //Enthalpy of pure component 1 = H1 is obtained at
    x_2 = 0, thus
18 x_2 = 0;
19 x_1 = 1;
20 H1 = a*x_1 + b*x_2 +c*x_1*x_2;//[J/mol]
21 printf("(a).The enthalpy of pure component 1 is %f J
    /mol\n",H1);
22
23 //Similarly for component 2,
24 //Enthalpy of pure component 2 = H2 is obtained at
    x_1 = 0, thus
25 x_1_prime = 0;
26 x_2_prime = 1;
27 H2 = a*x_1_prime + b*x_2_prime +c*x_1_prime*
    x_2_prime;//[J/mol]
28 printf("    The enthalpy of pure component 2 is %f J
    /mol\n\n",H2);
29
30 //(b)
31 //This part involves proving a relation in which no
    mathematics and no calculations are involved.
32 //For prove refer to this example 11.4 on page
    number 387 of the book.
33
34 //(c)
35 //From part (b), we have the relation
36 //H1_bar = a + c*(x_2^(2))
37 //H2_bar = b + c*(x_1^(2))
38
39 //For enthalpy of component 1 at infinite dilution ,
    x_1 approach 0 and thus x_2 approach 1, therefore
40 x_1_c = 0;
41 x_2_c = 1;
42 H1_infinite = a + c*(x_2_c^(2));//[cm^(3)/mol]
43 printf("(C).The enthalpy of componenet 1 at infinite
    dilution (at x_1 = 0) is %f J/mol\n",H1_infinite
    );
44

```

```

45 //At x_1 = 0.2
46 x_1_c1 = 0.2;
47 x_2_c1 = 0.8;
48 H1_bar_c1 = a + c*(x_2_c1^(2));//[J/mol]
49 printf("    The enthalpy of componenet 1 at (at x_1
    = 0.2) is %f J/mol\n",H1_bar_c1);
50
51 //At x_1 = 0.8
52 x_1_c2 = 0.8;
53 x_2_c2 = 0.2;
54 H1_bar_c2 = a + c*(x_2_c2^(2));//[J/mol]
55 printf("    The enthalpy of componenet 1 at (at x_1
    = 0.8) is %f J/mol",H1_bar_c2);
56
57 //As x_1 increases , 'H1_bar' approaches the value of
    'H1'
58
59 //(d)
60 //This part involves proving a relation in which no
    mathematics and no calculations are involved.
61 //For prove refer to this example 11.4 on page
    number 387 of the book.

```

Scilab code Exa 11.5 Developing an expression and calculation for enthalpy change of mixture

```

1 clear;
2 clc;
3
4 //Example - 11.5
5 //Page number - 389
6 printf("Example - 11.5 and Page number - 389\n\n");
7
8 //This problem involves proving a relation in which
    no mathematical components are involved.

```

```
9 //For prove refer to this example 11.5 on page
   number 389 of the book.
10 printf(" This problem involves proving a relation in
   which no mathematical components are involved.\n
   \n");
11 printf(" For prove refer to this example 11.5 on
   page number 389 of the book.")
```

Scilab code Exa 11.6 Proving a mathematical relation

```
1 clear;
2 clc;
3
4 //Example - 11.6
5 //Page number - 390
6 printf("Example - 11.6 and Page number - 390\n\n");
7
8 //This problem involves proving a relation in which
   no mathematics and no calculations are involved.
9 //For prove refer to this example 11.6 on page
   number 390 of the book.
10 printf(" This problem involves proving a relation in
   which no mathematics and no calculations are
   involved.\n\n");
11 printf(" For prove refer to this example 11.6 on
   page number 390 of the book.")
```

Scilab code Exa 11.7 Proving a mathematical relation

```
1 clear;
2 clc;
3
4 //Example - 11.7
```

```

5 //Page number – 393
6 printf("Example – 11.7 and Page number – 393\n\n");
7
8 //This problem involves proving a relation in which
  no mathematics and no calculations are involved.
9 //For prove refer to this example 11.7 on page
  number 393 of the book.
10 printf(" This problem involves proving a relation in
  which no mathematics and no calculations are
  involved.\n\n");
11 printf(" For prove refer to this example 11.7 on
  page number 393 of the book.")

```

Scilab code Exa 11.8 Proving a mathematical relation

```

1 clear;
2 clc;
3
4 //Example – 11.8
5 //Page number – 394
6 printf("Example – 11.8 and Page number – 394\n\n");
7
8 //This problem involves proving a relation in which
  no mathematics and no calculations are involved.
9 //For prove refer to this example 11.8 on page
  number 394 of the book.
10 printf(" This problem involves proving a relation in
  which no mathematics and no calculations are
  involved.\n\n");
11 printf(" For prove refer to this example 11.8 on
  page number 394 of the book.")

```

Scilab code Exa 11.9 Calculation of minimum work required

```

1 clear;
2 clc;
3
4 //Example - 11.9
5 //Page number - 395
6 printf("Example - 11.9 and Page number - 395\n\n");
7
8 //Given
9 n = 1*10^(3);//[mol] - No of moles
10 P = 0.1;//[MPa] - Pressure of the surrounding
11 T = 300;//[K] - Temperature of the surrounding
12 x_1 = 0.79; //Mole fraction of N2 in the air
13 x_2 = 0.21; //Mole fraction of O2 in the air
14 R=8.314; //[J/mol*K]
15
16 //Change in availability when x_1 moles of component
17 //x_1*(si_1 - si_2) = x_1*[H1 - H1_bar - T_0*(S1 -
18 //S1_bar)]
19 //Similarly change in availability of x_2 moles of
20 //component 2 is
21 //x_2*(si_1 - si_2) = x_2*[H2 - H2_bar - T_0*(S2 -
22 //S2_bar)]
23 //and thus total availability change when 1 mol of
24 //mixture is formed from x_1 mol of component 1 and
25 //x_2 mol of component 2 is equal to reversible
26 //work
27 //W_rev = x_1*[H1 - H1_bar - T_0*(S1 - S1_bar)] +
28 //x_2*[H2 - H2_bar - T_0*(S2 - S2_bar)]
29 //W_rev = -[x_1*(H1_bar - H1) + x_2*(H2_bar - H2)] +
30 //T_0*[x_1*(S1_bar - S1) + x_2*(S2_bar - S2)]
31 //W_rev = -[delta_H_mix] +T_0*[delta_S_mix]
32
33 //If T = T_0 that is ,temperature of mixing is same
34 //as that of surroundings , W_rev = -delta_G_mix.
35 //W_rev = -delta_G_mix = R*T*(x_1*log(x_1) + x_2*log
36 //(x_2))

```



```

28 W_rev = R*T*(x_1*log(x_1) + x_2*log(x_2));//[J/mol]
29
30 //Therefore total work transfer is given by
31 W_min = (n*W_rev)/1000;//[kJ]
32
33 printf("The minimum work required is %f kJ",W_min);

```

Scilab code Exa 11.10 Calculation of fugacity Of the mixture

```

1 clear;
2 clc;
3
4 //Example - 11.10
5 //Page number - 400
6 printf("Example - 11.10 and Page number - 400\n\n");
7
8 //Given
9 x_A = 0.20; // Mole fraction of A
10 x_B = 0.35; // Mole fraction of B
11 x_C = 0.45; // Mole fraction of C
12
13 phi_A = 0.7; // Fugacity coefficient of A
14 phi_B = 0.6; // Fugacity coefficient of B
15 phi_C = 0.9; // Fugacity coefficient of C
16
17 P = 6.08; // [MPa] - Pressure
18 T = 384; // [K] - Temperature
19
20 //We know that
21 //log(phi) = x_1*log(phi_1) + x_2*log(phi_2) + x_3*
    log(phi_3)
22 log_phi = x_A*log(phi_A) + x_B*log(phi_B) + x_C*log(
    phi_C); // Fugacity coefficient
23 phi = exp(log_phi);
24

```

```
25 //Thus fugacity is given by,
26 f_mixture = phi*P;// [MPa]
27
28 printf("The fugacity of the mixture is %f MPa",
        f_mixture);
```

Scilab code Exa 11.11 Proving a mathematical relation

```
1 clear;
2 clc;
3
4 //Example - 11.11
5 //Page number - 400
6 printf("Example - 11.11 and Page number - 400\n\n");
7
8 //This problem involves proving a relation in which
9 //no mathematics and no calculations are involved.
10 //For prove refer to this example 11.11 on page
11 //number 400 of the book.
12 printf(" This problem involves proving a relation in
13 //which no mathematics and no calculations are
14 //involved.\n\n");
15 printf(" For prove refer to this example 11.11 on
16 //page number 400 of the book.")
```

Scilab code Exa 11.12 Proving a mathematical relation

```
1 clear;
2 clc;
3
4 //Example - 11.12
5 //Page number - 401
6 printf("Example - 11.12 and Page number - 401\n\n");
```

```
7
8 //This problem involves proving a relation in which
   no mathematics and no calculations are involved.
9 //For prove refer to this example 11.12 on page
   number 401 of the book.
10 printf(" This problem involves proving a relation in
   which no mathematics and no calculations are
   involved.\n\n");
11 printf(" For prove refer to this example 11.12 on
   page number 401 of the book.")
```

Scilab code Exa 11.13 Proving a mathematical relation

```
1 clear;
2 clc;
3
4 //Example - 11.13
5 //Page number - 405
6 printf("Example - 11.13 and Page number - 405\n\n");
7
8 //This problem involves proving a relation in which
   no mathematics and no calculations are involved.
9 //For prove refer to this example 11.13 on page
   number 405 of the book.
10 printf(" This problem involves proving a relation in
   which no mathematics and no calculations are
   involved.\n\n");
11 printf(" For prove refer to this example 11.13 on
   page number 405 of the book.")
```

Chapter 12

Partial Molar Volume and Enthalpy from Experimental Data

Scilab code Exa 12.1 Calculation of partial molar volume

```
1 clear;
2 clc;
3
4 //Example - 12.1
5 //Page number - 419
6 printf("Example - 12.1 and Page number - 419\n\n");
7
8 //Given
9 //component 1 = methanol
10 //component 2 = water
11 T = 0 + 273.15; // [K] - Temperature
12 P = 1; // [atm] - Pressure
13 x_methanol = 0.5; // Mole fraction of methanol at
    which molar volume is to be calculated
14 x_water = 0.5; // Mole fraction at which molar volume
    is to be calculated
15
```

```

16 //V = V1 at x1 = 1 and V = V2 at x1 = 0, therefore
17 V1 = 40.7; //[cm^(3)/mol] - Molar volume of pure
    component 1
18 V2 = 18.1; //[cm^(3)/mol] - Molar volume of pure
    component 2
19
20 x1=[0.114,0.197,0.249,0.495,0.692,0.785,0.892]; //
    Values of mole fraction of component 1
21 V=[20.3,21.9,23.0,28.3,32.9,35.2,37.9]; // Values of
    molar volume
22 x2=zeros(1,7); // Mole fraction of component 2
23 x_V=zeros(1,7); // x_V = x1*V_1 + x2*V_2
24 V_mix=zeros(1,7); // V_mix = V - x1*V_1 - x2*V_2
25 del_V=zeros(1,7); //del_V = V_mix/(x1*x2)
26
27 for i=1:7;
28     x2(1,i)=1-x1(i);
29     x_V(1,i)=x1(i)*V1 + x2(i)*V2;
30     V_mix(1,i)=V(i)-x1(i)*V1- x2(i)*V2;
31     del_V(1,i)=V_mix(i)/(x1(i)*x2(i));
32 end
33
34 //Now employing the concept of quadratic regression
    of the data ( x1 , del_V ) to solve the equation
    of the type
35 //y = a0 + a1*x + a2*x^(2)
36 //Here the above equation is in the form of
37 //del_V = V_mix/(x1*x2) = a0 + a1*x1 + a2*x1^(2)
38
39 //From the matrix method to solve simultaneous
    linear equations , we have
40 a=[7 sum(x1) sum(x1^2);sum(x1) sum(x1^2) sum(x1^3);
    sum(x1^2) sum(x1^3) sum(x1^4)];
41 b=[sum(del_V);sum(x1.*del_V);sum((x1^2).*del_V)];
42 soln=a\b;
43 a0=soln(1);
44 a1=soln(2);
45 a2=soln(3);

```

```

46
47 //del_V = V_mix/(x1*x2) = a0 + a1*x1 + a2*x1^(2)
48 //V_mix = (a0 + a1*x1 + a2*x1^(2))*(x1*(1 - x1))
49 // For x1 = 0.5
50 x1 = 0.5;
51 V_mix_prime = (a0+(a1*x1)+(a2*x1^2))*(x1*(1-x1));//[
    cm^(3)/mol]
52
53 //Now differentiating the above equation with
    respect to x we get
54 //d/dx(V_mix) = (-4*a2*x1^3) + (3*(a2-a1)*x1^2) +
    (2*(a1-a0)*x1)+ a0
55 //Again for x1 = 0.5
56 x1_prime = 0.5;
57 del_V_mix_prime = (-4*a2*x1_prime^3)+(3*(a2-a1)*
    x1_prime^2)+(2*(a1-a0)*x1_prime)+a0;
58
59 //Finally ,calculating the partial molar volumes
60 V1_bar = V1 + V_mix_prime + x_water*del_V_mix_prime;
    //[cm^(3)/mol]
61 V2_bar = V2 + V_mix_prime - x_methanol*
    del_V_mix_prime;//[cm^(3)/mol]
62
63 printf("The partial molar volume of methanol (
    component 1) is %f cm^(3)/mol\n\n",V1_bar);
64 printf("The partial molar volume of water (component
    2) is %f cm^(3)/mol",V2_bar);

```

Scilab code Exa 12.2 Determination of volume of the mixture

```

1 clear;
2 clc;
3
4 //Example - 12.2
5 //Page number - 421

```

```

6 printf("Example - 12.2 and Page number - 421\n\n");
7
8 //Given
9 //component 1 = water
10 //component 2 = methanol
11 T = 25 + 273.15; // [K] - Temperature
12
13 //delta_V_mix = x_1*x_2*(-3.377 - 2.945*x_1 + 3.31*
    x_1^(2))
14 V1 = 18.0684; // [cm^(3)/mol] - Molar volume of pure
    component 1
15 V2 = 40.7221; // [cm^(3)/mol] - Molar volume of pure
    component 2
16 Vol_1 = 1000; // [cm^(3)] - Volume of pure component 1
17 Vol_2 = 1000; // [cm^(3)] - Volume of pure component 2
18
19 //Moles of the componenets can be calculated as
20 n_1 = Vol_1/V1; // [mol]
21 n_2 = Vol_2/V2; // [mol]
22
23 //Mole fraction of the components
24 x_1 = n_1/(n_1 + n_2);
25 x_2 = n_2/(n_1 + n_2);
26
27 delta_V_mix = x_1*x_2*(-3.377 - 2.945*x_1 + 3.31*x_1
    ^2); // [cm^(3)/mol]
28
29 //Differentiating the above equation , we get
30 //d/dx(delta_V_mix) = (1 - 2*x_1)*(-3.377 - 2.945*
    x_1 + 3.31*x_1^(2)) + (x_1 - x_1^(2))*(-2.945 +
    6.62*x_1)
31 del_delta_V_mix = (1 - 2*x_1)*(-3.377 - 2.945*x_1 +
    3.31*x_1^(2)) + (x_1 - x_1^(2))*(-2.945 + 6.62*
    x_1); // [cm^(3)/mol]
32
33 //Now calculating the partial molar volumes
34 V1_bar = V1 + delta_V_mix + x_1*del_delta_V_mix; // [
    cm^(3)/mol]

```

```

35 V2_bar = V2 + delta_V_mix - x_2*del_delta_V_mix;//[
    cm^(3)/mol]
36
37 //Finally molar volume of the solution is given by
38 V_sol = x_1*V1_bar + x_2*V2_bar;//[cm^(3)/mol]
39
40 // Total volume of the solution is given by
41 V_total = (n_1 + n_2)*V_sol;//[cm^(3)]
42
43 printf("The molar volume of the solution is %f cm
    ^{(3)}/mol\n\n",V_sol);
44 printf("The total volume of the solution is %f cm
    ^{(3)}",V_total);

```

Scilab code Exa 12.3 Determination of volumes

```

1 clear;
2 clc;
3
4 //Example - 12.3
5 //Page number - 422
6 printf("Example - 12.3 and Page number - 422\n\n");
7
8 //Given
9 //component 1 = methanol
10 //component 2 = water
11 Vol = 20;//[cm^(3)] - Volume of the solution
12 T = 22 + 273.15;//[K] - Temperature
13 W_bottle = 11.5485;//[g] - Weight of density bottle
14 Mol_meth = 32.04;//[Molecular weight of methanol]
15 Mol_water = 18.015;//[Molecular weight of water]
16
17 //Density of pure components can be found out at 0%
    and 100% of volume percent.
18 den_meth = 0.7929;//[cm^(3)/mol] - Density of pure

```



```

    methanol
19 den_water = 0.9937; //[cm^(3)/mol] - Density of pure
    water
20
21
22 Vol_perc=[5,10,20,30,40,50,60,70,80,90,95]; //
    Volumes percent of component 1 (methanol)
23 W_total
    =[31.2706,31.1468,30.8907,30.6346,30.3396,30.0053,29.5865,29.1453
    // Weight of solution + weight of density bottle
24
25 W_sol=zeros(1,11); // Weight of 20 cm^(3) of solution
26 den=zeros(1,11); // density of the solution
27 x1=zeros(1,11); // Mole fraction of methanol
28 x2=zeros(1,11); // Mole fraction of water
29
30 for i=1:11;
31     W_sol(1,i)=W_total(i)-W_bottle;
32     den(1,i)=W_sol(i)/Vol;
33     x1(1,i)=((Vol_perc(i)*den_meth)/Mol_meth)/(((
        Vol_perc(i)*den_meth)/Mol_meth)+(((100-
        Vol_perc(i))*den_water)/Mol_water));
34     x2(1,i)=1-x1(1,i);
35 end
36
37 //Again we have ,
38 V_kg=zeros(1,11); //[cm^(3)] - Volume of 1 kg of
    solution
39 n_mol=zeros(1,11); //[mol] - Number of moles in 1 kg
    of solution
40 V_mol=zeros(1,11); //[cm^(3)/mol] - Volume of 1 mol
    of solution
41 x_V=zeros(1,11); //[cm^(3)/mol] - x_V = x1*V_meth +
    x2*V_water
42 V_mix=zeros(1,11); //[cm^(3)/mol] - V_mix = V_mol -
    x1*V_meth - x2*V_water
43 del_V=zeros(1,11); // [cm^(3)/mol] - del_V = V_mix/(
    x1*x2)

```

```

44
45 //V_mol = V_meth at x1 = 1 and V_mol = V_water at x1
    = 0, therefore
46 V_meth = 40.4114;//[cm^(3)/mol] - Molar volume of
    pure component 1 (methanol)
47 V_water = 18.1286;//[cm^(3)/mol] - Molar volume of
    pure component 2 (water)
48
49 for i=1:11;
50     V_kg(1,i)=1000/den(i);
51     n_mol(1,i)=1000/(x1(i)*Mol_meth+x2(i)*Mol_water)
        ;
52     V_mol(1,i)=V_kg(i)/n_mol(i);
53     x_V(1,i)=V_meth*x1(i)+V_water*x2(i);
54     V_mix(1,i)=V_mol(i)-x1(i)*V_meth-x2(i)*V_water;
55     del_V(1,i)=V_mix(i)/(x1(i)*x2(i));
56 end
57
58 //Now employing the concept of quadratic regression
    of the data ( x1 , del_V ) to solve the equation
    of the type
59 //y = a0 + a1*x + a2*x^(2)
60 //Here the above equation is in the form of
61 //del_V = V_mix/(x1*x2) = a0 + a1*x1 + a2*x1^(2)
62
63 //From the matrix method to solve simultaneous
    linear equations , we have
64 a=[11 sum(x1) sum(x1^2);sum(x1) sum(x1^2) sum(x1^3);
    sum(x1^2) sum(x1^3) sum(x1^4)];
65 b=[sum(del_V);sum(x1.*del_V);sum((x1^2).*del_V)];
66 soln=a\b;
67 a0=soln(1);
68 a1=soln(2);
69 a2=soln(3);
70
71 //del_V = V_mix/(x1*x2) = a0 + a1*x1 + a2*x1^(2)
72 //V_mix = (a0 + a1*x1 + a2*x1^(2))*(x1*(1 - x1))
73 //Solving the above equation for x1,

```

```

74 def f(' [y]=f(x1)', 'y=(a0+(a1*x1)+(a2*x1^2))*(x1*(1-x1
    )')');
75
76 //Now differentiating the above equation with
    respect to x we get
77 //d/dx(V_mix) = (-4*a2*x1^3) + (3*(a2-a1)*x1^2) +
    (2*(a1-a0)*x1)+ a0
78 //Again solving it for x1
79 def f1(' [y]=f1(x1)', 'y=(-4*a2*x1^3)+(3*(a2-a1)*x1^2)
    +(2*(a1-a0)*x1)+a0');
80
81 //Now
82
83 x1_prime=[0,0.25,0.50,0.75,1.0];
84 V_mix_prime=zeros(1,5); // [cm^(3)/mol] - V_mix = V -
    x1*V_meth - x2*V_water
85 del_V_prime=zeros(1,5); // [cm^(3)/mol] - del_V =
    V_mix/(x1*x2)
86 V1_bar=zeros(1,5); // [cm^(3)/mol] - Partial molar
    volume of component 1
87 V2_bar=zeros(1,5); // [cm^(3)/mol] - Partial molar
    volume of component 1
88
89 for j=1:5;
90     V_mix_prime(j)=f(x1_prime(j));
91     del_V_prime(j)=f1(x1_prime(j));
92     V1_bar(j)=V_meth+V_mix_prime(j)+(1-x1_prime(j))*
        del_V_prime(j);
93     V2_bar(j)=V_water+V_mix_prime(j)-x1_prime(j)*
        del_V_prime(j);
94     printf("For x1 = %f\n",x1_prime(j));
95     printf("The partial molar volume of methanol (
        component 1) is %f cm^(3)/mol\n",V1_bar(j));
96     printf("The partial molar volume of water (
        component 2) is %f cm^(3)/mol\n\n",V2_bar(j))
        ;
97 end

```

Scilab code Exa 12.4 Determination of partial molar volumes

```
1 clear;
2 clc;
3
4 //Example - 12.4
5 //Page number - 424
6 printf("Example - 12.4 and Page number - 424\n\n");
7
8 //Given
9 //component 1 = formic acid
10 //component 2 = water
11 T = 20 + 273.15; // [K] - Temperature
12 Mol_form = 46.027; //Molecular weight of formic acid
13 Mol_water = 18.015; // Molecular weight of water
14
15 Wt_perc=[10,18,30,50,72,78]; //Weight percent of
    formic acid
16 den=[1.0246,1.0441,1.0729,1.1207,1.1702,1.1818]; // [g
    /cm^(3)] - Density of solution
17
18 V_g=zeros(1,6); // [cm^(3)/g] - Volume of 1 g of
    solution
19 x1=zeros(1,6); // Mole fraction of component 1
20 x2=zeros(1,6); // Mole fraction of component 2
21 n=zeros(1,6); // Number of moles in 1 g
22 V_mol=zeros(1,6); // [cm^(3)/mol] - Volume of 1 mol of
    solution
23 x_V=zeros(1,6); // [cm^(3)/mol] -  $x_V = x1*V_{form} + x2$ 
    * $V_{water}$ 
24 V_mix=zeros(1,6); // [cm^(3)/mol] -  $V_{mix} = V - x1*$ 
     $V_{form} - x2*V_{water}$ 
25 del_V=zeros(1,6); // [cm^(3)/mol] -  $del_V = V_{mix}/(x1$ 
    * $x2)$ 
```

```

26
27 //V_mol = V_form at x1 = 1 and V_mol = V_water at x1
    = 0, therefore
28 V_form = 37.737;//[cm^(3)/mol] - Molar volume of
    pure formic acid (component 1)
29 V_water = 18.050;//[cm^(3)/mol] - Molar volume of
    pure water (component 2)
30
31 for i=1:6;
32     V_g(i)=1/den(i);
33     x1(1,i)=(Wt_perc(i)/Mol_form)/((Wt_perc(i)/
        Mol_form)+((100-Wt_perc(i))/Mol_water));
34     x2(1,i)=1-x1(i);
35     n(1,i)=((Wt_perc(i)/100)/Mol_form)+(((100-
        Wt_perc(i))/100)/Mol_water);
36     V_mol(1,i)=V_g(i)/n(i);
37     x_V(1,i)=V_form*x1(i)+V_water*x2(i);
38     V_mix(1,i)=V_mol(i)-x1(i)*V_form-x2(i)*V_water;
39     del_V(1,i)=V_mix(i)/(x1(i)*x2(i));
40 end
41
42 //Now employing the concept of quadratic regression
    of the data ( x1 , del_V ) to solve the equation
    of the type
43 //y = a0 + a1*x + a2*x^(2)
44 //Here the above equation is in the form of
45 //del_V = V_mix/(x1*x2) = a0 + a1*x1 + a2*x1^(2)
46
47 //From the matrix method to solve simultaneous
    linear equations, we have
48 a=[11 sum(x1) sum(x1^2);sum(x1) sum(x1^2) sum(x1^3);
    sum(x1^2) sum(x1^3) sum(x1^4)];
49 b=[sum(del_V);sum(x1.*del_V);sum((x1^2).*del_V)];
50 soln=a\b;
51 a0=soln(1);
52 a1=soln(2);
53 a2=soln(3);
54

```

```

55 //del_V = V_mix/(x1*x2) = a0 + a1*x1 + a2*x1^(2)
56 //V_mix = (a0 + a1*x1 + a2*x1^(2))*(x1*(1 - x1))
57 //Solving the above equation for x1,
58 deff(' [y]=f(x1)', 'y=(a0+(a1*x1)+(a2*x1^2))*(x1*(1-x1)
    )');
59
60 //Now differentiating the above equation with
    respect to x we get
61 //d/dx(V_mix) = (-4*a2*x1^3) + (3*(a2-a1)*x1^2) +
    (2*(a1-a0)*x1)+ a0
62 //Again solving it for x1
63 deff(' [y]=f1(x1)', 'y=(-4*a2*x1^3)+(3*(a2-a1)*x1^2)
    +(2*(a1-a0)*x1)+a0');
64
65 //At 15 Wt% of formic acid, x1 is given by
66 x1_prime_1 = (15/Mol_form)/((15/Mol_form)+((100-15)/
    Mol_water));
67 //Similarly at 75 Wt% of formic acid, x1 is given by
68 x1_prime_2 = (75/Mol_form)/((75/Mol_form)+((100-75)/
    Mol_water));
69
70 Wt_perc_prime=[15,75];
71 x1_prime=[x1_prime_1,x1_prime_2];
72 V_mix_prime=zeros(1,2);//[cm^(3)/mol] - V_mix = V -
    x1*V_meth - x2*V_water
73 del_V_prime=zeros(1,2);//[cm^(3)/mol] - del_V =
    V_mix/(x1*x2)
74 V1_bar=zeros(1,2);//[cm^(3)/mol] - Partial molar
    volume of component 1
75 V2_bar=zeros(1,2);//[cm^(3)/mol] - Partial molar
    volume of component 1
76
77 for j=1:2;
78     V_mix_prime(j)=f(x1_prime(j));
79     del_V_prime(j)=f1(x1_prime(j));
80     V1_bar(j)=V_form+V_mix_prime(j)+(1-x1_prime(j))*
        del_V_prime(j);
81     V2_bar(j)=V_water+V_mix_prime(j)-x1_prime(j)*

```

```

    del_V_prime(j);
82  printf("For weight percent of formic acid = %f
        percent\n",Wt_perc_prime(j));
83  printf("The partial molar volume of formic acid
        (component 1) is %f cm^(3)/mol\n",V1_bar(j));
84  printf("The partial molar volume of water (
        component 2) is %f cm^(3)/mol\n\n",V2_bar(j))
    ;
85  end

```

Scilab code Exa 12.5 Determination of enthalpy

```

1  clear;
2  clc;
3
4  //Example - 12.5
5  //Page number - 426
6  printf("Example - 12.5 and Page number - 426\n\n");
7
8  //Given
9  T = 40 + 273.15; // [K] - Temperature
10
11  x1
    =[0.083,0.176,0.268,0.353,0.428,0.720,0.780,0.850,0.900];
    // Mole fraction of component 1
12  delta_H_mix
    =[0.250,0.488,0.670,0.790,0.863,0.775,0.669,0.510,0.362];
    // [kJ/mol] - Enthalpy of the solution
13
14  x2=zeros(1,9); // Mole fraction of component 2
15  del_H=zeros(1,9); // [kJ/mol] - del_H = delta_H_mix/(
    x1*x2)
16
17  for i=1:9;
18      x2(1,i)=1-x1(i);

```

```

19     del_H(1,i)=delta_H_mix(i)/(x1(i)*x2(i));
20 end
21
22 //Now employing the concept of quadratic regression
    of the data ( x1 , del_H ) to solve the equation
    of the type
23 //y = a0 + a1*x + a2*x^(2)
24 //Here the above equation is in the form of
25 //del_H = delta_H_mix/(x1*x2) = a0 + a1*x1 + a2*x1
    ^ (2)
26
27 //From the matrix method to solve simultaneous
    linear equations , we have
28 a=[9 sum(x1) sum(x1^2);sum(x1) sum(x1^2) sum(x1^3);
    sum(x1^2) sum(x1^3) sum(x1^4)];
29 b=[sum(del_H);sum(x1.*del_H);sum((x1^2).*del_H)];
30 soln=a\b;
31 a0=soln(1);
32 a1=soln(2);
33 a2=soln(3);
34
35 //del_H = delta_H_mix/(x1*x2) = a0 + a1*x1 + a2*x1
    ^ (2)
36 //delta_H_mix = (a0 + a1*x1 + a2*x1^(2))*(x1*(1 - x1
    ))
37 //At x1 = 0.25,
38 x_1 = 0.25;//[mol]
39 delta_H_mix = (a0+(a1*x_1)+(a2*x_1^2))*(x_1*(1-x_1))
    ;//[kJ/mol]
40
41 //Now differentiating the above equation with
    respect to x we get
42 //d/dx(delta_H_mix) = del_delta_H_mix = (-4*a2*x1^3)
    + (3*(a2-a1)*x1^2) + (2*(a1-a0)*x1)+ a0
43 //Again for x1 = 0.25
44 x_1_prime = 0.25;//[mol]
45 del_delta_H_mix = (-4*a2*x_1_prime^3)+(3*(a2-a1)*
    x_1_prime^2)+(2*(a1-a0)*x_1_prime)+a0;//[kJ/mol]

```



```

46
47 //We have the relation
48 //  $H1\_bar - H1 = \Delta H_{mix} + x2 \cdot \Delta H_{mix}$  ,
    and
49 //  $H2\_bar - H2 = \Delta H_{mix} - x1 \cdot \Delta H_{mix}$ 
50
51 //Let us suppose
52 //  $k_1 = H1\_bar - H1$  , and
53 //  $k_2 = H2\_bar - H2$ 
54
55  $k_1 = \Delta H_{mix} + (1-x_{1\_prime}) \cdot \Delta H_{mix}$ ; //
    [kJ/mol]
56  $k_2 = \Delta H_{mix} - x_{1\_prime} \cdot \Delta H_{mix}$ ; // [kJ/
    mol]
57
58
59 printf("The value of ( $H1\_bar - H1$ ) at  $x1 = 0.25$  is
    %f kJ/mol\n\n",k_1);
60 printf("The value of ( $H2\_bar - H2$ ) at  $x1 = 0.25$  is
    %f kJ/mol",k_2);

```

Chapter 13

Fugacity of a Component in a Mixture by Equations of State

Scilab code Exa 13.1 Proving a mathematical relation

```
1 clear;
2 clc;
3
4 //Example - 13.1
5 //Page number - 432
6 printf("Example - 13.1 and Page number - 432\n\n");
7
8 //This problem involves proving a relation in which
9 //no mathematics and no calculations are involved.
10 //For prove refer to this example 13.1 on page
11 //number 432 of the book.
12 printf(" This problem involves proving a relation in
13 which no mathematics and no calculations are
14 involved.\n\n");
15 printf(" For prove refer to this example 13.1 on
16 page number 432 of the book.")
```

Scilab code Exa 13.2 Calculation of fugacity coefficients

```
1 clear;
2 clc;
3
4 //Example - 13.2
5 //Page number - 433
6 printf("Example - 13.2 and Page number - 433\n\n");
7
8 //Given
9 T = 310.93; // [K] - Temperature
10 P = 2.76*10^(6); // [Pa] - Pressure
11 y1 = 0.8942; // [mol] - mole fraction of component 1
12 y2 = 1 - y1; // [mol] - mole fraction of component 2
13 R=8.314; // [J/mol*K] - Universal gas constant
14
15 //For component 1 (methane)
16 Tc_1 = 190.6; // [K] - Critical temperature
17 Pc_1 = 45.99*10^(5); // [N/m^(2)] - Critical pressure
18 Vc_1 = 98.6; // [cm^(3)/mol] - Critical molar volume
19 Zc_1 = 0.286; // - Critical compressibility factor
20 w_1 = 0.012; // - Critical acentric factor
21 //Similarly for component 2 (n-Butane)
22 Tc_2 = 425.1; // [K]
23 Pc_2 = 37.96*10^(5); // [N/m^(2)]
24 Vc_2 = 255; // [cm^(3)/mol]
25 Zc_2=0.274;
26 w_2=0.2;
27
28 //For component 1
29 Tr_1 = T/Tc_1; //Reduced temperature
30 //At reduced temperature
31 B1_0 = 0.083-(0.422/(Tr_1)^(1.6));
32 B1_1 = 0.139-(0.172/(Tr_1)^(4.2));
33 //We know, (B*Pc)/(R*Tc) = B_0+(w*B_1)
34 B_11 = ((B1_0+(w_1*B1_1))*(R*Tc_1))/Pc_1; // [m^(3)/
    mol]
35
```

```

36 //Similarly for component 2
37 Tr_2 = T/Tc_2;//Reduced temperature
38 //At reduced temperature Tr_2,
39 B2_0 = 0.083-(0.422/(Tr_2)^(1.6));
40 B2_1 = 0.139-(0.172/(Tr_2)^(4.2));
41 B_22 = ((B2_0+(w_2*B2_1))*(R*Tc_2))/Pc_2;//[m^(3)/
      mol]
42
43 //For cross coefficient
44 Tc_12 = (Tc_1*Tc_2)^(1/2);//[K]
45 w_12 = (w_1+w_2)/2;
46 Zc_12 = (Zc_1+Zc_2)/2;
47 Vc_12 = (((Vc_1)^(1/3)+(Vc_2)^(1/3))/2)^(3);//[cm
      ^3)/mol]
48 Vc_12 = Vc_12*10^(-6);//[m^(3)/mol]
49 Pc_12 = (Zc_12*R*Tc_12)/Vc_12;//[N/m^(2)]
50
51 //Given, Z = 1 + (B*P)/(R*T)
52 //Now we have, (B_12*Pc_12)/(R*Tc_12) = B_0 + (w_12*
      B_1)
53 //where B_0 and B_1 are to be evaluated at Tr_12
54 Tr_12 = T/Tc_12;
55 //At reduced temperature Tr_12
56 B_0 = 0.083-(0.422/(Tr_12)^(1.6));
57 B_1 = 0.139-(0.172/(Tr_12)^(4.2));
58 B_12 = ((B_0+(w_12*B_1))*R*Tc_12)/Pc_12;//[m^(3)/mol
      ]
59 //For the mixture
60 B = y1^(2)*B_11+2*y1*y2*B_12+y2^(2)*B_22;//[m^(3)/
      mol]
61
62 //Now del_12 can be calculated as,
63 del_12 = 2*B_12 - B_11 - B_22;//[m^(3)/mol]
64
65 //We have the relation, log(phi_1) = (P/(R*T))*(B_11
      + y2^(2)*del_12), therefore
66 phi_1 = exp((P/(R*T))*(B_11 + y2^(2)*del_12));
67 //Similarly for component 2

```

```

68 phi_2 = exp((P/(R*T))*(B_22 + y1^(2)*del_12));
69
70 printf(" The value of fugacity coefficient of
       component 1 (phi_1) is %f\n\n",phi_1);
71 printf(" The value of fugacity coefficient of
       component 2 (phi_2) is %f\n\n",phi_2);
72
73 //Finally fugacity coefficient of the mixture is
       given by
74 //log(phi) = y1*log(phi_1) + y2*log(phi_2);
75 phi = exp(y1*log(phi_1) + y2*log(phi_2));
76
77 printf(" The value of fugacity coefficient of the
       mixture (phi) is %f ",phi);
78 //The fugacity coefficient of the mixture can also
       be obtained using
79 //log(phi) = (B*P)/(R*T)

```

Scilab code Exa 13.3 Proving a mathematical relation

```

1 clear;
2 clc;
3
4 //Example - 13.3
5 //Page number - 435
6 printf("Example - 13.3 and Page number - 435\n\n");
7
8 //This problem involves proving a relation in which
       no mathematics and no calculations are involved.
9 //For prove refer to this example 13.3 on page
       number 435 of the book.
10 printf(" This problem involves proving a relation in
       which no mathematics and no calculations are
       involved.\n\n");
11 printf(" For prove refer to this example 13.3 on

```

page number 435 of the book.”)

Scilab code Exa 13.4 Proving a mathematical relation

```
1 clear;
2 clc;
3
4 //Example - 13.4
5 //Page number - 436
6 printf("Example - 13.4 and Page number - 436\n\n");
7
8 //This problem involves proving a relation in which
9 //no mathematics and no calculations are involved.
10 //For prove refer to this example 13.4 on page
11 //number 436 of the book.
12 printf(" This problem involves proving a relation in
13 //which no mathematics and no calculations are
14 //involved.\n\n");
15 printf(" For prove refer to this example 13.4 on
16 //page number 436 of the book.")
```

Scilab code Exa 13.5 Proving a mathematical relation

```
1 clear;
2 clc;
3
4 //Example - 13.5
5 //Page number - 442
6 printf("Example - 13.5 and Page number - 442\n\n");
7
8 //This problem involves proving a relation in which
9 //no mathematics and no calculations are involved.
```

```

9 //For prove refer to this example 13.5 on page
   number 442 of the book.
10 printf(" This problem involves proving a relation in
   which no mathematics and no calculations are
   involved.\n\n");
11 printf(" For prove refer to this example 13.5 on
   page number 442 of the book.")

```

Scilab code Exa 13.6 Proving a mathematical relation

```

1 clear;
2 clc;
3
4 //Example - 13.6
5 //Page number - 446
6 printf("Example - 13.6 and Page number - 446\n\n");
7
8 //This problem involves proving a relation in which
   no mathematics and no calculations are involved.
9 //For prove refer to this example 13.6 on page
   number 446 of the book.
10 printf(" This problem involves proving a relation in
   which no mathematics and no calculations are
   involved.\n\n");
11 printf(" For prove refer to this example 13.6 on
   page number 446 of the book.")

```

Scilab code Exa 13.7 Calculation of fugacity coefficients

```

1 clear;
2 clc;
3
4 // Example - 13.7

```

```

5 // Page number - 447
6 printf("Example - 13.7 and Page number - 447\n\n");
7
8 // Given
9 T = 460; // [K] - Temperature
10 P = 40*10^(5); // [Pa] - Pressure
11 R=8.314; // [J/mol*K] - Universal gas constant
12 // component 1 = nitrogen
13 // component 2 = n-Butane
14 y1 = 0.4974; // Mole percent of nitrogen
15 y2 = 0.5026; // Mole percent of n-Butane
16 Tc_nit = 126.2; // [K]
17 Pc_nit = 34.00*10^(5); // [Pa]
18 Tc_but = 425.1; // [K]
19 Pc_but = 37.96*10^(5); // [Pa]
20
21 // (1). van der Walls equation of state
22
23 // The fugacity coefficient of component 1 in a
    binary mixture following van der Walls equation
    of state is given by,
24 //  $\log(\phi_1) = b_1/(V-b) - \log(Z-B) - 2*(y1*a_{11} +$ 
     $y2*a_{12})/(R*T*V)$ 
25 // and for component 2 is given by,
26 //  $\log(\phi_2) = b_2/(V-b) - \log(Z-B) - 2*(y1*a_{12} +$ 
     $y2*a_{22})/(R*T*V)$ 
27 // Where  $B = (P*b)/(R*T)$ 
28
29 // For componenet 1 (nitrogen)
30 a_1 = (27*R^(2)*Tc_nit^(2))/(64*Pc_nit); // [Pa-m^(6)/
    mol^(2)]
31 b_1 = (R*Tc_nit)/(8*Pc_nit); // [m^(3)/mol]
32
33 // Similarly for componenet 2 (n-Butane)
34 a_2 = (27*R^(2)*Tc_but^(2))/(64*Pc_but); // [Pa-m^(6)/
    mol^(2)]
35 b_2 = (R*Tc_but)/(8*Pc_but); // [m^(3)/mol]
36

```



```

37 // Here
38 a_11 = a_1;
39 a_22 = a_2;
40 // For cross coefficient
41 a_12 = (a_1*a_2)^(1/2);//[Pa-m^(6)/mol^(2)]
42
43 // For the mixture
44 a = y1^(2)*a_11 + y2^(2)*a_22 + 2*y1*y2*a_12;//[Pa-m
      ^ (6)/mol^(2)]
45 b = y1*b_1 + y2*b_2;//[m^(3)/mol]
46
47 // The cubic form of the van der Walls equation of
      state is given by,
48 // V^(3) - (b+(R*T)/P)*V^(2) + (a/P)*V - (a*b)/P = 0
49 // Substituting the value and solving for V, we get
50 // Solving the cubic equation
51 def f(' [y]=f(V) ', 'y=V^(3)-(b+(R*T)/P)*V^(2)+(a/P)*V-(
      a*b)/P');
52 V_1=fsolve(-1,f);
53 V_2=fsolve(0,f);
54 V_3=fsolve(1,f);
55 // The molar volume V = V_3, the other two roots are
      imaginary
56 V = V_3;//[m^(3)/mol]
57
58 // The comprssibility factor of the mixture is
59 Z = (P*V)/(R*T);
60 // And B can also be calculated as
61 B = (P*b)/(R*T);
62
63 // The fugacity coefficient of component 1 in the
      mixture is
64 phi_1 = exp(b_1/(V-b) - log(Z-B) -2*(y1*a_11 + y2*
      a_12)/(R*T*V));
65 // Similarly fugacity coefficient of component 2 in
      the mixture is
66 phi_2 = exp(b_2/(V-b) - log(Z-B) -2*(y1*a_12 + y2*
      a_22)/(R*T*V));

```

```

67
68 // The fugacity coefficient of the mixture is given
    by,
69 //  $\log(\phi) = y_1 \log(\phi_1) + y_2 \log(\phi_2)$ 
70  $\phi = \exp(y_1 \log(\phi_1) + y_2 \log(\phi_2))$ ;
71
72 // Also the fugacity coefficient of the mixture
    following van der Walls equation of state is
    given by,
73 //  $\log(\phi) = b/(V-b) - \log(Z-B) - 2*a/(R*T*V)$ 
74  $\phi\_dash = \exp(b/(V-b) - \log(Z-B) - 2*a/(R*T*V))$ ;
75 // The result is same as obtained above
76
77 printf(" (1)van der Walls equation of state\n");
78 printf(" The value of fugacity coefficient of
    component 1 (nitrogen) is %f\n", $\phi_1$ );
79 printf(" The value of fugacity coefficient of
    component 2 (n-butane) is %f\n", $\phi_2$ );
80 printf(" The value of fugacity coefficient of the
    mixture is %f\n", $\phi$ );
81 printf(" Also the fugacity coefficient of the
    mixture from van der Walls equation of state is
    %f (which is same as above)\n\n", $\phi\_dash$ );
82
83 // (2). Redlich-Kwong equation of state
84
85 // For component 1,
86  $a\_1\_prime = (0.42748*R^{(2)}*Tc\_nit^{(2.5)})/Pc\_nit$ ; // [
    Pa-m6/mol2]
87  $b\_1\_prime = (0.08664*R*Tc\_nit)/Pc\_nit$ ; // [m3/mol]
88
89 //similarly for component 2,
90  $a\_2\_prime = (0.42748*R^{(2)}*Tc\_but^{(2.5)})/Pc\_but$ ; // [
    Pa-m6/mol2]
91  $b\_2\_prime = (0.08664*R*Tc\_but)/Pc\_but$ ; // [m3/mol]
92
93 // For cross coefficient
94  $a\_12\_prime = (a\_1\_prime*a\_2\_prime)^{(1/2)}$ ; // [Pa-m6)

```

```

    /mol^(2) ]
95 // For the mixture
96 a_prime = y1^(2)*a_1_prime + y2^(2)*a_2_prime +2*y1*
    y2*a_12_prime;// [Pa-m^(6)/mol^(2) ]
97 b_prime = y1*b_1_prime +y2*b_2_prime;// [m^(3)/mol]
98
99
100 //The cubic form of Redlich Kwong equation of state
    is given by,
101 //  $V^3 - ((R*T)/P)*V^2 - ((b^2) + ((b*R*T)/P) - (a/(T
    ^{1/2}*P)))*V - (a*b)/(T^{1/2}*P) = 0$ 
102 // Solving the cubic equation
103 deff ('[y]=f1(V)', 'y=V^3 - ((R*T)/P)*V^2 - ((b_prime
    ^2) + ((b_prime*R*T)/P) - (a_prime/(T^{1/2}*P)))*V
    - (a_prime*b_prime)/(T^{1/2}*P)');
104 V_4=fsolve(1,f1);
105 V_5=fsolve(0,f1);
106 V_6=fsolve(-1,f1);
107 // The molar volume V = V_4, the other two roots are
    imaginary
108 V_prime = V_4;// [m^(3)/mol]
109
110 // The compressibility factor of the mixture is
111 Z_prime = (P*V_prime)/(R*T);
112 // And B can also be calculated as
113 B_prime = (P*b_prime)/(R*T);
114
115 // The fugacity coefficient of component 1 in the
    binary mixture is given by
116 //  $\log(\phi_1) = b_1/(V-b) - \log(Z-B) + ((a*b_1)/((b
    ^2)*R*T^{3/2}))*(\log((V+b)/V) - (b/(V+b))) - (2*(y1
    *a_1+y2*a_12)/(R*T^{3/2}*b))*(\log(V+b)/b)$ 
117
118 phi_1_prime = exp((b_1_prime/(V_prime-b_prime))-log(
    Z_prime-B_prime)+((a_prime*b_1_prime)/((b_prime
    ^2)*R*(T^{3/2}))))*(log((V_prime+b_prime)/
    V_prime)-(b_prime/(V_prime+b_prime)))-(2*(y1*
    a_1_prime+y2*a_12_prime)/(R*(T^{3/2})*b_prime))*

```

```

    log((V_prime+b_prime)/V_prime)));
119
120
121 // Similarly fugacity coefficient of component 2 in
    the mixture is
122 phi_2_prime = exp((b_2_prime/(V_prime-b_prime))-log(
    Z_prime-B_prime)+((a_prime*b_2_prime)/((b_prime
    ^2)*R*(T^(3/2))))*(log((V_prime+b_prime)/
    V_prime)-(b_prime/(V_prime+b_prime))-(2*(y1*
    a_12_prime+y2*a_2_prime)/(R*(T^(3/2))*b_prime))*(
    log((V_prime+b_prime)/V_prime)));
123
124 // The fugacity coefficient of the mixture is given
    by,
125 // log(phi) = y1*log(phi_1) + y2*log(phi_2)
126 phi_prime = exp(y1*log(phi_1_prime) + y2*log(
    phi_2_prime));
127
128 // Also the fugacity coefficient for the mixture
    following Redlich-Kwong equation of state is also
    given by
129 // log(phi) = b/(V-b) - log(Z-B) - (a/(R*T^(3/2)))
    *(1/(V+b)+(1/b)*log((V+b)/V))
130 phi_prime_dash = exp(b_prime/(V_prime-b_prime) - log
    (Z_prime-B_prime) - (a_prime/(R*T^(3/2)))*(1/(
    V_prime+b_prime)+(1/b_prime)*log((V_prime+b_prime
    )/
    V_prime)));
131
132 printf(" (2) Redlich-Kwong equation of state\n");
133 printf(" The value of fugacity coefficient of
    component 1 (nitrogen) is %f\n",phi_1_prime);
134 printf(" The value of fugacity coefficient of
    component 2 (n-butane) is %f\n",phi_2_prime);
135 printf(" The value of fugacity coefficient of the
    mixture is %f\n",phi_prime);
136 printf(" Also the fugacity coefficient for the
    mixture from Redlich-Kwong equation of state is
    %f (which is same as above)\n\n",phi_prime_dash);

```


Chapter 14

Activity Coefficients Models for Liquid Mixtures

Scilab code Exa 14.1 Determination of expression for activity coefficients

```
1 clear;
2 clc;
3
4 //Example - 14.1
5 //Page number - 455
6 printf("Example - 14.1 and Page number - 455\n\n");
7
8 //This problem involves proving a relation in which
9 //no mathematics and no calculations are involved.
10 //For prove refer to this example 14.1 on page
11 //number 455 of the book.
12 printf(" This problem involves proving a relation in
13 which no mathematics and no calculations are
14 involved.\n\n");
15 printf(" For prove refer to this example 14.1 on
16 page number 455 of the book.")
```

Scilab code Exa 14.2 Proving a mathematical relation

```
1 clear;
2 clc;
3
4 //Example - 14.2
5 //Page number - 456
6 printf("Example - 14.2 and Page number - 456\n\n");
7
8 //This problem involves proving a relation in which
9 //no mathematics and no calculations are involved.
10 //For prove refer to this example 14.2 on page
11 //number 456 of the book.
12 printf(" This problem involves proving a relation in
13 //which no mathematics and no calculations are
14 //involved.\n\n");
15 printf(" For prove refer to this example 14.2 on
16 //page number 456 of the book.")
```

Scilab code Exa 14.3 Proving a mathematical relation

```
1 clear;
2 clc;
3
4 //Example - 14.3
5 //Page number - 458
6 printf("Example - 14.3 and Page number - 458\n\n");
7
8 //This problem involves proving a relation in which
9 //no mathematics and no calculations are involved.
10 //For prove refer to this example 14.3 on page
11 //number 458 of the book.
12 printf(" This problem involves proving a relation in
13 //which no mathematics and no calculations are
14 //involved.\n\n");
```

```
11 printf(" For prove refer to this example 14.3 on
    page number 458 of the book.")
```

Scilab code Exa 14.4 Determination of value of Gibbs free energy change and enthalpy change

```
1 clear;
2 clc;
3
4 //Example - 14.4
5 //Page number - 461
6 printf("Example - 14.4 and Page number - 461\n\n");
7
8 //Given ,
9 T = 300; //[K] - Temperature
10 b = 100; //[ cal/mol]
11 R = 1.987; //[ cal/mol*K] - Universal gas constant
12 // R*T*log(Y_1) = b*x_2^(2)
13 // R*T*log(Y_2) = b*x_1^(2)
14
15 //For equimolar mixture
16 x_1 = 0.5; //Mole fraction of component 1
17 x_2 = 0.5; //Mole fraction of component 2
18
19 //The excess Gibbs free energy is given by
20 // G_excess = R*T*(x_1*log(Y_1) + x_2*log(Y_2)) = b*
    x_1*x_2^(2) + b*x_2*x_1^(2) = b*x_1*(x_1 + x_2) =
    b*x_1*x_2
21 G_excess = b*x_1*x_2; //[ cal/mol]
22
23 //The ideal Gibbs free energy change of mixing is
    given by,
24 delta_G_id_mix = R*T*(x_1*log(x_1)+x_2*log(x_2)); //[
    cal/mol]
25
```



```

26 //The Gibbs free energy of mixing is given by
27 delta_G_mix = delta_G_id_mix + G_excess;//[cal/mol]
28
29 //It is given that entropy change of mixing is that
    of ideal mixture ,therefore
30 // delta_S_mix = delta_S_id_mix = - R*sum(x_i*log(
    x_i))
31
32 //delta_G_mix = delta_H_mix - T*delta_S_mix =
    delta_H_mix + R*T*(x_1*log(x_1)+x_2*log(x_2))
33 delta_H_mix = b*x_1*x_2;//[cal/mol]
34
35 printf("The value of Gibbs free energy change for
    equimolar mixture formation is %f cal/mol\n\n",
    delta_G_mix);
36 printf("The value of enthalpy change for equimolar
    mixture formation is %f cal/mol\n\n",delta_H_mix)
    ;
37
38 //Work required for separation of mixture into pure
    components is
39 W = delta_G_mix;
40 printf("The least amount of work required for
    separation at 300 K is %f cal/mol\n\n",W);

```

Scilab code Exa 14.5 Proving a mathematical relation

```

1 clear;
2 clc;
3
4 //Example - 14.5
5 //Page number - 461
6 printf("Example - 14.5 and Page number - 461\n\n");
7
8 //This problem involves proving a relation in which

```

```

    no mathematics and no calculations are involved.
9 //For prove refer to this example 14.5 on page
    number 461 of the book.
10 printf(" This problem involves proving a relation in
    which no mathematics and no calculations are
    involved.\n\n");
11 printf(" For prove refer to this example 14.5 on
    page number 461 of the book.")

```

Scilab code Exa 14.6 Proving a mathematical relation

```

1 clear;
2 clc;
3
4 //Example - 14.6
5 //Page number - 463
6 printf("Example - 14.6 and Page number - 463\n\n");
7
8 //This problem involves proving a relation in which
    no mathematics and no calculations are involved.
9 //For prove refer to this example 14.6 on page
    number 463 of the book.
10 printf(" This problem involves proving a relation in
    which no mathematics and no calculations are
    involved.\n\n");
11 printf(" For prove refer to this example 14.6 on
    page number 463 of the book.")

```

Scilab code Exa 14.7 Proving a mathematical relation

```

1 clear;
2 clc;
3

```

```

4 //Example - 14.7
5 //Page number - 464
6 printf("Example - 14.7 and Page number - 464\n\n");
7
8 //This problem involves proving a relation in which
   no mathematics and no calculations are involved.
9 //For prove refer to this example 14.7 on page
   number 464 of the book.
10 printf(" This problem involves proving a relation in
   which no mathematics and no calculations are
   involved.\n\n");
11 printf(" For prove refer to this example 14.7 on
   page number 464 of the book.")

```

Scilab code Exa 14.8 Proving a mathematical relation

```

1 clear;
2 clc;
3
4 //Example - 14.8
5 //Page number - 465
6 printf("Example - 14.8 and Page number - 465\n\n");
7
8 //This problem involves proving a relation in which
   no mathematics and no calculations are involved.
9 //For prove refer to this example 14.8 on page
   number 465 of the book.
10 printf(" This problem involves proving a relation in
   which no mathematics and no calculations are
   involved.\n\n");
11 printf(" For prove refer to this example 14.8 on
   page number 465 of the book.")

```

Scilab code Exa 14.9 Proving a mathematical relation

```
1 clear;
2 clc;
3
4 //Example - 14.9
5 //Page number - 465
6 printf("Example - 14.9 and Page number - 465\n\n");
7
8 //This problem involves proving a relation in which
9 //no mathematics and no calculations are involved.
10 //For prove refer to this example 14.9 on page
11 //number 465 of the book.
12 printf(" This problem involves proving a relation in
13 //which no mathematics and no calculations are
14 //involved.\n\n");
15 printf(" For prove refer to this example 14.9 on
16 //page number 465 of the book.")
```

Scilab code Exa 14.10 Comparison of Margules and van Laar equations

```
1 clear;
2 clc;
3
4 //Example - 14.10
5 //Page number - 466
6 printf("Example - 14.10 and Page number - 466\n\n");
7
8 //Given ,
9 T = 25 +173.15; // [K] - Temperature
10 x_1
11     = [0.0115, 0.0160, 0.0250, 0.0300, 0.0575, 0.1125, 0.1775, 0.2330, 0.4235,
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

```

12 y_2
    =[1.0037,1.0099,1.0102,1.0047,1.0203,1.0399,1.1051,1.1695,1.4462,
13
14 x_2 = zeros(1,15); // x_2 = (1 - x_1)
15 G_RT = zeros(1,15); // G_RT = G_excess/(R*T)
16 x1x2_G_RT = zeros(1,15); // x1x2_G_RT = (x_1*x_2/(
    G_excess/(R*T))
17 G_RTx1x2 = zeros(1,15); // G_RTx1x1 = G_excess/(R*T*
    x_1*x_2)
18
19 for i=1:15;
20     x_2(1,i)=(1-x_1(i));
21     G_RT(1,i)=(x_1(i)*log(y_1(i)))+(x_2(i)*log(y_2(i)
    )));
22     x1x2_G_RT(1,i)=(x_1(i)*x_2(i))/G_RT(i);
23     G_RTx1x2(1,i)=1/x1x2_G_RT(i);
24
25 end
26
27 //From Van Laar equation
28 // G_RTx1x2 = (x_1*x_2/(G_excess/(R*T))) = 1/A + (1/
    B -1/A)*x_1
29 //slope = (1/B -1/A) and intercept = 1/A
30
31 //Now employing the concept of linear regression of
    the data ( x_1 , x1x2_G_RT ) to find the value of
    intercept and slope of the above equation
32 //Let slope = slop and intercept = intr
33
34 [slop,intr,sig]=reglin(x_1,x1x2_G_RT);
35
36 A = 1/intr;
37 B = 1/(slop+(1/A));
38 printf(" The value of van Laar parameters are\n A =
    %f and B = %f\n\n",A,B);
39

```

```

40 // Now from Margules equation
41 //  $G_{RTx_1x_2} = G_{\text{excess}}/(R \cdot T \cdot x_1 \cdot x_2) = B_1 \cdot x_1 + A_1 \cdot x_1^2$ 
42 //slope = (B1 - A1) and intercept = A1
43
44 // Again employing the concept of linear regression
45 // of the data ( x_1 , G_RT_x1x2 ) to find the value
46 // of intercept and slope of the above equation
47 //Let slope = slop1 and intercept = intr1
48
49 [slop1 ,intr1 ,sig1]=reglin(x_1 ,G_RT_x1x2);
50
51 A1 = intr1;
52 B1 = slop1 + A1;
53 printf(" The value of Margules parameters are\n A =
54 %f and B = %f\n\n" ,A1 ,B1);
55
56 printf(" Because of the higher value of correlation
57 factor for Van Laar model, it fits the data
58 better.");

```

Scilab code Exa 14.11 Calculation of activity coefficients

```

1 clear;
2 clc;
3
4 //Example - 14.11
5 //Page number - 470
6 printf("Example - 14.11 and Page number - 470\n\n")
7
8 //Given ,
9 T = 60 + 273.15; // [K] - Temperature
10 R = 1.987; // [cal/mol*K] - Universal gas constant
11 //component 1 = acetone
12 //component 2 = water

```

```

13 x_1 = 0.3; // Mole fraction of component 1
14 x_2 = 1 - x_1; // Mole fraction of component 2
15 V_mol_1 = 74.05; // [cm3/mol] - Molar volume of
    pure component 1
16 V_mol_2 = 18.07; // [cm3/mol] - Molar volume of
    pure component 2
17
18 //for Wilson equation
19 a_12 = 291.27; // [cal/mol]
20 a_21 = 1448.01; // [cal/mol]
21
22 //For NRTL
23 b_12 = 631.05; // [cal/mol]
24 b_21 = 1197.41; // [cal/mol]
25 alpha = 0.5343;
26
27 //From wilson equation
28 A_12=(V_mol_2/V_mol_1)*(exp(-a_12/(R*T)));
29 A_21 = (V_mol_1/V_mol_2)*(exp(-a_21/(R*T)));
30 Beta = A_12/(x_1+x_2*A_12) - A_21/(x_2+x_1*A_21);
31 //log(Y1) = -log(x_1 + x_2*A_12) + x_2*Beta;
32 Y1 = exp(-log(x_1+x_2*A_12)+x_2*Beta);
33 //similarly for Y2
34 Y2 = exp(-log(x_2+x_1*A_21)-x_1*Beta);
35 printf("The value of activity coefficients for
    Wilson equation are\n Y1 = %f \t and \t Y2 = %f\n
    \n", Y1, Y2);
36
37 //From NRTL equation ,
38 t_12 = b_12/(R*T);
39 t_21 = b_21/(R*T);
40 G_12 = exp(-alpha*t_12);
41 G_21 = exp(-alpha*t_21);
42
43 //log(Y1) = x_12*[t_12*(G_12/(x_1+x_2*G_12))2
    + (G_12*t_12)/((G_12/(x_1+x_2*G_12))2)]
44 Y1_prime = exp(x_22*(t_21*(G_21/(x_1+x_2*G_21))
    2+(G_12*t_12)/((x_2+x_1*G_12))2));

```

```

45 //Similarly for Y2
46 Y2_prime = exp(x_1^(2)*(t_12*(G_12/(x_2+x_1*G_12))
      ^ (2)+(G_21*t_21)/(((x_1+x_2*G_21))^(2)))));
47
48 printf("The value of activity coefficients for NRTL
      equation are\n Y1 = %f \t and \t Y2 = %f\n\n",
      Y1_prime, Y2_prime);

```

Scilab code Exa 14.12 Calculation of the value of activity coefficients

```

1 clear;
2 clc;
3
4 //Example - 14.12
5 //Page number - 474
6 printf("Example - 14.12 and Page number - 474\n\n");
7
8 //Given
9 T = 307; // [K]
10 x_1 = 0.047;
11 x_2 = 1 - x_1;
12
13 // The subgroups in the two components are
14 // Acetone (1) : 1 CH3, 1 CH3CO
15 // n-pentane (2) : 2 CH3, 3 CH2
16
17 //Group volume (Rk) and surface area (Qk) parameters
      of the subgroup are
18 k_CH3 = 1;
19 k_CH2 = 2;
20 k_CH3CO = 19;
21 Rk_CH3 = 0.9011;
22 Rk_CH2 = 0.6744;
23 Rk_CH3CO = 1.6724;
24 Qk_CH3 = 0.848;

```



```

25 Qk_CH2 = 0.540;
26 Qk_CH3CO = 1.4880;
27
28 // Interaction parameters of the subgroups in kelvin
   (K) are
29 a_1_1 = 0;
30 a_1_2 = 0;
31 a_1_19 = 476.40;
32 a_2_1 = 0;
33 a_2_2 = 0;
34 a_2_19 = 476.40;
35 a_19_1 = 26.76;
36 a_19_2 = 26.76;
37 a_19_19 = 0;
38
39 r_1 = 1*Rk_CH3 + 1*Rk_CH3CO;
40 r_2 = 2*Rk_CH3 + 3*Rk_CH2;
41 q_1 = 1*Qk_CH3 + 1*Qk_CH3CO;
42 q_2 = 2*Qk_CH3 + 3*Qk_CH2;
43
44 J_1 = r_1/(r_1*x_1+r_2*x_2);
45 J_2 = r_2/(r_1*x_1+r_2*x_2);
46 L_1 = q_1/(q_1*x_1+q_2*x_2);
47 L_2 = q_2/(q_1*x_1+q_2*x_2);
48 t_1_1 = exp(-a_1_1/T);
49 t_1_2 = exp(-a_1_2/T);
50 t_1_19 = exp(-a_1_19/T);
51 t_2_1 = exp(-a_2_1/T);
52 t_2_2 = exp(-a_2_2/T);
53 t_2_19 = exp(-a_2_19/T);
54 t_19_1 = exp(-a_19_1/T);
55 t_19_2 = exp(-a_19_2/T);
56 t_19_19 = exp(-a_19_19/T);
57
58 e_1_1 = 1*Qk_CH3/q_1;
59 e_2_1 = 0;
60 e_19_1 = (1*Qk_CH3CO/q_1);
61 e_1_2 = 2*Qk_CH3/q_2;

```

```

62 e_2_2 = 3*Qk_CH2/q_2;
63 e_19_2 = 0;
64
65 B_1_1 = e_1_1*t_1_1 + e_2_1*t_2_1 + e_19_1*t_19_1;
66 B_1_2 = e_1_1*t_1_2 + e_2_1*t_2_2 + e_19_1*t_19_2;
67 B_1_19 = e_1_1*t_1_19 + e_2_1*t_2_19 + e_19_1*
    t_19_19;
68 B_2_1 = e_1_2*t_1_1 + e_2_2*t_2_1 + e_19_2*t_19_1;
69 B_2_2 = e_1_2*t_1_2 + e_2_2*t_2_2 + e_19_2*t_19_2;
70 B_2_19 = e_1_2*t_1_19 + e_2_2*t_2_19 + e_19_2*
    t_19_19;
71
72 theta_1 = (x_1*q_1*e_1_1 + x_2*q_2*e_1_2)/(x_1*q_1 +
    x_2*q_2);
73 theta_2 = (x_1*q_1*e_2_1 + x_2*q_2*e_2_2)/(x_1*q_1 +
    x_2*q_2);
74 theta_19 = (x_1*q_1*e_19_1 + x_2*q_2*e_19_2)/(x_1*
    q_1 + x_2*q_2);
75
76 s_1 = theta_1*t_1_1 + theta_2*t_2_1 + theta_19*
    t_19_1;
77 s_2 = theta_1*t_1_2 + theta_2*t_2_2 + theta_19*
    t_19_2;
78 s_19 = theta_1*t_1_19 + theta_2*t_2_19 + theta_19*
    t_19_19;
79
80 // log(Y1_C) = 1 - J_1 + log(J_1) - 5*q_1*(1- (J_1/
    L_1) + log(J_1/L_1))
81 // log(Y2_C) = 1 - J_2 + log(J_2) - 5*q_2*(1- (J_2/
    L_2) + log(J_2/L_2))
82 Y1_C = exp(1 - J_1 + log(J_1) - 5*q_1*(1- (J_1/L_1)
    + log(J_1/L_1)));
83 Y2_C = exp(1 - J_2 + log(J_2) - 5*q_2*(1- (J_2/L_2)
    + log(J_2/L_2)));
84
85 // For species 1
86 summation_theta_k_1 = theta_1*(B_1_1/s_1) + theta_2
    *(B_1_2/s_2) + theta_19*(B_1_19/s_19);

```

```

87 summation_e_ki_1 = e_1_1*log(B_1_1/s_1) + e_2_1*log(
      B_1_2/s_2) + e_19_1*log(B_1_19/s_19);
88
89 // For species 2
90 summation_theta_k_2 = theta_1*(B_2_1/s_1) + theta_2
      *(B_2_2/s_2) + theta_19*(B_2_19/s_19);
91 summation_e_ki_2 = e_1_2*log(B_2_1/s_1) + e_2_2*log(
      B_2_2/s_2) + e_19_2*log(B_2_19/s_19);
92
93 // log(Y1_R) = q_1(1 - summation_theta_k_1 +
      summation_e_ki_1)
94 // log(Y2_R) = q_2(1 - summation_theta_k_2 +
      summation_e_ki_2)
95 Y1_R = exp(q_1*(1 - summation_theta_k_1 +
      summation_e_ki_1));
96 Y2_R = exp(q_2*(1 - summation_theta_k_2 +
      summation_e_ki_2));
97
98 // log(Y1) = log(Y1_C) + log(Y1_R)
99 // log(Y2) = log(Y2_C) + log(Y2_R)
100 Y1 = exp(log(Y1_C) + log(Y1_R));
101 Y2 = exp(log(Y2_C) + log(Y2_R));
102
103 printf(" The activity coefficients are Y1 = %f and
      Y2 = %f\n", Y1, Y2);

```

Scilab code Exa 14.13 Calculation of the value of activity coefficients

```

1 clear;
2 clc;
3
4 //Example - 14.15
5 //Page number - 481
6 printf("Example - 14.15 and Page number - 481\n\n")
7

```

```

8 //Given ,
9 T = 25 + 273.15; // [K] - Temperature
10 R = 1.987; // [cal/mol*K] - Universal gas constant
11 //component 1 = chloroform
12 //component 2 = carbon tetrachloride
13 x_1 = 0.5; //Mole fraction of component 1 //Equimolar
    mixture
14 x_2 = 0.5; //Mole fraction of component 2
15 V_mol_1 = 81; // [cm^(3)/mol] - Molar volume of pure
    component 1
16 V_mol_2 = 97; // [cm^(3)/mol] - Molar volume of pure
    component 2
17 del_1 = 9.2; // [(cal/cm^(3))^(1/2)] - Mole fraction
    of component 1
18 del_2 = 8.6; // [(cal/cm^(3))^(1/2)] - Mole fraction
    of component 2
19
20 //Scatchard - Hilderbrand model
21 phi_1 = (x_1*V_mol_1)/(x_1*V_mol_1+x_2*V_mol_2);
22 phi_2 = (x_2*V_mol_2)/(x_1*V_mol_1+x_2*V_mol_2);
23
24 //log(Y1) = (V_mol_1/(R*T))*phi_1^(2)*(del_1-del_2)
    ^ (2)
25 Y1 = exp((V_mol_1/(R*T))*(phi_1^(2))*((del_1-del_2)
    ^ (2)));
26
27 //Similarly , for Y2
28 Y2 = exp((V_mol_2/(R*T))*(phi_2^(2))*((del_1-del_2)
    ^ (2)));
29
30 printf("The value of activity coefficients for
    Scatchard-Hilderbrand model are\n Y1 = %f \t and
    \t Y2 = %f\n\n",Y1,Y2);

```

Scilab code Exa 14.14 Calculation of the value of activity coefficients

```

1 clear;
2 clc;
3
4 //Example - 14.14
5 //Page number - 485
6 printf("Example - 14.14 and Page number - 485\n\n")
7
8 //Given,
9 T = 25 + 273.15; // [K] - Temperature
10 mol_HCl = 0.001; // [mol/kg] - Molality of HCl
11 A = 0.510; // [(kg/mol)^(1/2)]
12 Z_positive = 1; // Stoichiometric coefficient of 'H'
   ion
13 Z_negative = -1; // Stoichiometric coefficient of 'Cl'
   ion
14 m_H_positive = mol_HCl; //
15 m_Cl_negative = mol_HCl;
16
17 // I = 1/2*[((Z_positive)^(2))*m_H_positive + ((
   Z_negative)^(2))*m_Cl_negative]
18 I = 1/2*(((Z_positive)^(2))*m_H_positive + ((
   Z_negative)^(2))*m_Cl_negative);
19
20 //Using Debye-Huckel limiting law we get,
21 // log(Y1) = -A*(abs(Z_positive*Z_negative))*(I
   ^(1/2))
22 Y = 10^(-A*(abs(Z_positive*Z_negative))*(I^(1/2)));
23 printf("The mean activity coefficient at 25 C using
   Debye-Huckel limiting law is Y = %f\n\n",Y);
24
25 //Using Debye-Huckel extended model we get
26 //log(Y_prime) = (-A*(abs(Z_positive*Z_negative))*(I
   ^(1/2)))/(1 + (I^(1/2)));
27 Y_prime = 10^((-A*(abs(Z_positive*Z_negative))*(I
   ^(1/2)))/(1 + (I^(1/2))));
28 printf("The mean activity coefficient at 25 C using
   Debye-Huckel extended model is Y = %f\n\n",
   Y_prime);

```

Scilab code Exa 14.15 Calculation of the value of activity coefficients

```
1 clear;
2 clc;
3
4 //Example - 14.15
5 //Page number - 485
6 printf("Example - 14.15 and Page number - 485\n\n")
7
8 //Given ,
9 T = 25 + 273.15; // [K] - Temperature
10 mol_CaCl2 = 0.001; // [mol/kg] - Molality of HCl
11 A = 0.510; // [(kg/mol)^(1/2)]
12 Z_positive = 2; // Stoichiometric coefficient of 'Ca'
    ion
13 Z_negative = -1; // Stoichiometric coefficient of 'Cl'
    ion
14 m_Ca_positive = mol_CaCl2;
15 m_Cl_negative = 2*mol_CaCl2;
16
17 //  $I = \frac{1}{2} * [((Z\_positive)^2) * m\_Ca\_positive + ((Z\_negative)^2) * m\_Cl\_negative]$ 
18 I = 1/2 * (((Z_positive)^2) * m_Ca_positive + ((Z_negative)^2) * m_Cl_negative);
19
20 // Using Debye-Huckel limiting law we get ,
21 //  $\log(Y1) = -A * (abs(Z\_positive * Z\_negative)) * (I^{1/2})$ 
22 Y = 10^(-A * (abs(Z_positive * Z_negative)) * (I^(1/2)));
23 printf("The mean activity coefficient at 25 C using Debye-Huckel limiting law is Y = %f\n\n", Y);
```

Scilab code Exa 14.16 Proving a mathematical relation

```
1 clear;
2 clc;
3
4 //Example - 14.16
5 //Page number - 486
6 printf("Example - 14.16 and Page number - 486\n\n");
7
8 //This problem involves proving a relation in which
9 //no mathematics and no calculations are involved.
10 //For prove refer to this example 14.16 on page
11 //number 486 of the book.
12 printf(" This problem involves proving a relation in
13 //which no mathematics and no calculations are
14 //involved.\n\n");
15 printf(" For prove refer to this example 14.16 on
16 //page number 486 of the book.")
```

Scilab code Exa 14.17 Calculation of pressure

```
1 clear;
2 clc;
3
4 //Example - 14.17
5 //Page number - 488
6 printf("Example - 14.17 and Page number - 488\n\n");
7
8 //Given ,
9 T = 50 + 273.15; //[K] - Temperature
10 R=8.314; //[J/mol*K] - Universal gas constant
11 x_1 = 0.3; // Mole fraction of component 1
12 x_2 = (1-x_1); // Mole fraction of component 2
13 //Increment of 1% means Y2 = 1.01*Y1
14
```

```

15 //Excess volume of the mixture is given by,
16 V_excess = 4*x_1*x_2;//[cm^(3)/mol]
17 //And therefore
18 V_1_excess = 4*x_2*x_2*10^(-6);//[m^(3)/mol] - Exces
    volume of component 1
19 V_2_excess = 4*x_1*x_1*10^(-6);//[m^(3)/mol] - Exces
    volume of component 2
20
21 //We have from equation 14.89 of the book,
22 //V_i_excess/(R*T) = (del_log(Y_i)/del_P)_T,x
23
24 //Rearranging above equation
25 //d(log(Y1)) = (V1_excess/(R*T))dP
26 //Integrating the above equation at constant 'T' and
    'x' in the limit from 'Y1' to '1.01*Y1' in the
    LHS and from 'P' to 'P+delta_P' in the RHS
27 //On simplification we get
28 //log(1.01*Y1/Y1) = (V_1_excess/(R*T))*delta_P
29 delta_P = log(1.01)/(V_1_excess/(R*T));//[N/m^(2)]
30 delta_P = delta_P*10^(-6);//[MPa]
31
32 printf("The required pressure to increase the
    activity coefficient by 1%% is %f MPa",delta_P);

```

Scilab code Exa 14.18 Proving a mathematical relation

```

1 clear;
2 clc;
3
4 //Example - 14.18
5 //Page number - 488
6 printf("Example - 14.18 and Page number - 488\n\n");
7
8 //This problem involves proving a relation in which
    no mathematics and no calculations are involved.

```



```

9 //For prove refer to this example 14.18 on page
   number 488 of the book.
10 printf(" This problem involves proving a relation in
   which no mathematics and no calculations are
   involved.\n\n");
11 printf(" For prove refer to this example 14.18 on
   page number 488 of the book.")

```

Scilab code Exa 14.19 Determination of enthalpy

```

1 clear;
2 clc;
3
4 //Example - 14.19
5 //Page number - 489
6 printf("Example - 14.19 and Page number - 489\n\n");
7
8 //given
9 P = 2;//[bar] - Pressure
10 T = 310;//[K] - Temperature
11 R=8.314;//[J/mol*K] - Universal gas constant
12 A = (0.1665 + 233.74/T);// Margules parameter
13 B = (0.5908 + 197.55/T);// Margules parameter
14
15 //two parameter Margules equation is given by
16 //G_excess/(R*T*x_1*x_2) = B*x_1 + A*x_2
17 //On simplification and putting the values we get
18 //G_excess = ((0.5908 + 197.55/T)*x_1^(2)*x_2 +
   (0.1665 + 233.74/T)*x_2^(2)*x_1)
19
20
21 //H_excess/(R*T^(2)) = -[d/dT(G_excess/(R*T*x_1*x_2)
   )]_P,x
22 //On simplification and putting the values we get
23 //H_excess/(R*T^(2)) = (197.55/T^(2))*x_1^(2)*x_2 +

```

```

    (233.74/T^(2))*x_1*x_2^(2)
24
25 //We know that enthalpy change of mixing is given by
26 // delta_H_mix = H - x_1*H_1 - x_2*H_2 =
    delta_H_id_mix + H_excess
27
28 //But, delta_H_id_mix = 0 and H_excess is positive ,
    therefore enthalpy of muxture
29 // H > (x_1*H_1 + x_2*H_2)
30 // Therefore heat is required during the formation of
    mixture
31
32 printf("Since enthalpy of mixture formation (H)
    comes out to be positive , threfore steam is
    required to maintain the constant temperature.");

```

Scilab code Exa 14.20 Determination of an expression

```

1 clear;
2 clc;
3
4 //Example - 14.20
5 //Page number - 490
6 printf("Example - 14.20 and Page number - 490\n\n");
7
8 T = 40 + 273.15; // [K]
9 P = 101.3; // [kPa]
10
11 // G_E/(R*T) = A*x_1*x_2
12
13 // The parameter A at 101.3 kPa and various
    temperatures are
14 A_35 = 0.479; // A at 35 C
15 A_40 = 0.458; // A at 40 C
16 A_45 = 0.439; // A at 45 C

```

```

17
18 // At 40 C, G_E/(R*T) is given by
19 // G_E/(R*T) = A_40*x1*x2
20 // Therefore log(Y1) = A_40*x2^(2) and log(Y2) =
    A_40*x1^(2)
21
22 dA_dT = (A_45-A_35)/(45-35); // [K^(-1)] - dA/dT
23 // H_E/(R*T^(2)) = -[del(G_E/(R*T))/del(T)]_P,x = (
    dA/dT)*x1*x2
24 // H_E/(R*T) = -T*(dA/dT)*x1*x2 = 1.25*x1*x2
25
26 // S_E = (H_E - G_E)/T = (-R*T^(2)*(dA/dT)*x1*x2 - A
    *R*T*x1*x2)/T = -(R*T*(dA/dT) + A*R)*x1*x2
27 // Thus S_E/R = -(T*(dA/dT) + A)*x1*x2 = 0.795*x1*x2
28
29 printf(" The expressions are H_E/(R*T) = 1.25*x1*x2
    \n\t\t S_E/R = 0.795*x1*x2");

```

Scilab code Exa 14.21 Calculation of enthalpy entropy and Gibbs free energy

```

1 clear;
2 clc;
3
4 //Example - 14.21
5 //Page number - 490
6 printf("Example - 14.21 and Page number - 490\n\n");
7
8 //given
9 T = 293.15; // [K] - Temperature
10 R=8.314; // [J/mol*K] - Universal gas constant
11 A = 1280; // [J/mol]
12
13 // (dA/dT)_P,x = del_A (say)
14 dA_dT = -7.0; // [J/mol-K]

```

```

15
16 //For equimolar mixture ,
17 x_1 = 0.5; // Mole fraction of component 1
18 x_2 = 0.5; // Mole fraction of component 2
19
20 //log(Y1) = (A/(R*T))*x_2^(2)
21 //log(Y2) = (A/(R*T))*x_1^(2)
22 Y1 = exp((A/(R*T))*x_2^(2));
23 Y2 = exp((A/(R*T))*x_1^(2));
24
25 //G_excess/(R*T) = x_1*log(Y1) + x_2*log(Y2) = (A/(
    R*T))*x_1*x_2
26 G_excess = A*x_1*x_2; // [J/mol] - Excess Gibbs free
    energy
27
28 //H_excess/(R*T^(2)) = -[d/dT(G_excess/(R*T))]_P,x
29 //H_excess/(R*T^(2)) = -((x_1*x_2)/R)*[d/dT(A/T)]_P,
    x
30 //On simplification and putting the values we get
31 H_excess = A*x_1*x_2 - T*dA_dT*x_1*x_2; // [J/mol] -
    Excess enthalpy
32
33 //Now excess entropy is given by
34 S_excess = (H_excess - G_excess)/T; // [J/mol-K] -
    Excess entropy
35
36 printf("For equimolar mixture\n\n");
37 printf("Excess Gibbs free energy (G_excess) is %f J/
    mol\n\n", G_excess);
38 printf("Excess enthalpy (H_excess) is %f J/mol\n\n",
    H_excess);
39 printf("Excess entropy (S_excess) is %f J/mol\n\n",
    S_excess);
40 printf("The value of activity coefficient Y1 is %f\n
    \n", Y1);
41 printf("The value of activity coefficient Y2 is %f\n
    \n", Y2);

```

Scilab code Exa 14.22 Determination of Gibbs free energy and enthalpy change

```
1 clear;
2 clc;
3
4 // Example - 14.22
5 // Page number - 491
6 printf("Example - 14.22 and Page number - 491\n\n");
7
8 // Given
9 T = 60 + 273.15; // [K] - Temperature
10 R = 8.314; // [J/mol*K] - Universal gas constant
11
12 //  $\log(Y1\_inf) = \log(Y2\_inf) = 0.15 + 10/T$ 
13
14 // Since the two liquids are slightly dissimilar ,
    we assume the activity coefficients to follow
    van Laar equation
15 // From van Laaar equation
16 //  $A = \log(Y1\_inf)$  and  $B = \log(Y2\_inf)$  and since it
    is given that  $\log(Y1\_inf) = \log(Y2\_inf)$ ,
    therefore  $A = B$ 
17 //  $(x_1*x_2)/(G\_excess/R*T) = x_1/B + x_2/A = X_1/A +$ 
     $x_2/A = 1/A$ 
18 //  $G\_excess/(R*T) = A*x_1*x_2$ 
19
20 // For equimolar mixture ,
21  $x_1 = 0.5$ ; // Mole fraction of component 1
22  $x_2 = 0.5$ ; // Mole fraction of component 2
23
24 // Expression for A can be written as
25 //  $A = 0.15 + 10/T$ , where T is in C. Therefore
26  $A = 0.15 + 10/(T - 273.15)$ ;
```

```

27 // Differentiating it with respect to temprature we
    get
28 dA_dT = - 10/((T-273.15)^(2));
29
30 // The excess Gibbs free energy can be calculated as
31 G_excess = A*x_1*x_2*(R*T);//[J/mol]
32
33 // The ideal Gibbs free energy change can be
    calculated as
34 delta_G_id_mix = R*T*(x_1*log(x_1) + x_2*log(x_2));
    //[J/mol]
35
36 // Finally we have ,
37 delta_G_mix = G_excess + delta_G_id_mix;//[J/mol]
38
39 printf("The Gibbs free energy change of mixing for
    equimolar mixture is %f J/mol\n\n",delta_G_mix);
40
41
42 // Now let us determine the excess enthalpy. We know
    that
43 //  $H_{\text{excess}}/(R*T^2) = -[d/dT(G_{\text{excess}}/R*T)]_{P,x} =$ 
     $-x_1*x_2*[d/dT(A)]_{P,x}$ 
44 // Therefore at 'T' = 60 C the excess enthalpy is
    given by
45 H_excess = -R*(T^(2))*x_1*x_2*dA_dT;//[J/mol]
46
47 delta_H_id_mix = 0;//[J/mol] - Enthalpy change of
    mixing for ideal solution is zero.
48
49 //Thus enthalpy change of mixing for an equimolar
    mixture at 333.15 K is given by
50 delta_H_mix = delta_H_id_mix + H_excess;//[J/mol]
51
52
53 printf("The enthalpy change of mixing for equimolar
    mixture is %f J/mol",delta_H_mix);

```

Chapter 15

Vapour Liquid Equilibria

Scilab code Exa 15.1 Calculation of number of moles in liquid and vapour phase

```
1 clear;
2 clc;
3
4 //Example - 15.1
5 //Page number - 503
6 printf("Example - 15.1 and Page number - 503\n\n");
7
8 //Given
9 T = 90+ 273.15; // [K] - Temperature
10 P = 1; // [atm] - Pressure
11 x_1 = 0.5748; // Equilibrium composition of liquid
    phase
12 y_1 = 0.7725; // Equilibrium composition of vapour
    phase
13
14 // We start with 1 mol of mixture of composition z_1
    = 0.6, from material balance we get
15 // (L + V)*z_1 = L*x_1 + V*y_1
16 // Since total number of moles is 1, we get
17 // z_1 = L*x_1 + (1-L)*y_1
```

```

18
19 z_1_1 = 0.6; // - Mole fraction of benzene
20 L_1 = (z_1_1 - y_1)/(x_1 - y_1);
21 V_1 = 1 - L_1;
22
23 printf(" For z_1 = 0.6\n");
24 printf(" The moles in the liquid phase is %f mol\n",
        L_1);
25 printf(" The moles in the vapour phase is %f mol\n\n
        ",V_1);
26
27 z_1_2 = 0.7; // - Mole fraction of benzene
28 L_2 = (z_1_2 - y_1)/(x_1 - y_1);
29 V_2 = 1 - L_2;
30
31 printf(" For z_1 = 0.7\n");
32 printf(" The moles in the liquid phase is %f mol\n",
        L_2);
33 printf(" The moles in the vapour phase is %f mol\n\n
        ",V_2);
34
35
36 // For z = 0.8
37 // The feed remains vapour and the liquid is not
        formed at all as it is outside the two-phase
        region (x_1 = 0.5748 and y_1 = 0.7725).
38 printf(" For z_1 = 0.8\n");
39 printf(" The feed remains vapour and the liquid is
        not formed at all as it is outside the two-phase
        region (x_1 = 0.5748 and y_1 = 0.7725)")

```

Scilab code Exa 15.2 Calculation of pressure temperature and composition

```
1 clear;
```



```

2  clc;
3  funcprot(0);
4
5  //Example - 15.2
6  //Page number - 515
7  printf("Example - 15.2 and Page number - 515\n\n");
8
9  //Given
10 // log(P_1_sat) = 14.39155 - 2795.817/(t + 230.002)
11 // log(P_2_sat) = 16.26205 - 3799.887/(t + 226.346)
12
13 //(a)
14 x_1_a = 0.43; // Equilibrium composition of liquid
    phase
15 t_a = 76; // [C] - Temperature
16 x_2_a = 1 - x_1_a;
17
18 // Since liquid phase composition is given we use
    the relation
19 // P = x_1*P_1_sat + x_2*P_2_sat
20 // At t = 76 C
21 P_1_sat_a = exp(14.39155 - 2795.817/(t_a + 230.002))
    ;
22 P_2_sat_a = exp(16.26205 - 3799.887/(t_a + 226.346))
    ;
23 // Therefore total pressure is
24 P_a = x_1_a*P_1_sat_a + x_2_a*P_2_sat_a; // [kPa]
25 y_1_a = (x_1_a*P_1_sat_a)/(P_a);
26 y_2_a = (x_2_a*P_2_sat_a)/(P_a);
27
28 printf("(a).The system pressure is , P = %f kPa\n",
    P_a);
29 printf("    The vapour phase composition is , y_1 =
    %f\n\n", y_1_a);
30
31 //(b)
32 y_1_b = 0.43; // Equilibrium composition of vapour
    phase

```

```

33 y_2_b = 1 - y_1_b;
34 t_b = 76; //[C] - Temperature
35
36 P_1_sat_b = exp(14.39155 - 2795.817/(t_b + 230.002))
    ;
37 P_2_sat_b = exp(16.26205 - 3799.887/(t_b + 226.346))
    ;
38
39 // Since vapour phase composition is given ,the
    system pressure is given by
40 // 1/P = y_1/P_1_sat + y_2/P_2_sat
41 P_b = 1/(y_1_b/P_1_sat_b + y_2_b/P_2_sat_b);
42
43 x_1_b = (y_1_b*P_b)/P_1_sat_b;
44 x_2_b = (y_2_b*P_b)/P_2_sat_b;
45
46 printf("(b).The system pressure is , P = %f kPa\n",
    P_b);
47 printf("    The liquid phase composition is , x_1 =
    %f\n\n",x_1_b);
48
49 //(c)
50 x_1_c = 0.32; // Equilibrium composition of liquid
    phase
51 x_2_c = 1 - x_1_c;
52 P_c = 101.33; //[kPa] - Pressure of the system
53
54 // We have , P = x_1*P_1_sat + x_2*P_2_sat
55 t_1_sat = 2795.817/(14.39155 - log(P_c)) - 230.002;
56 t_2_sat = 3799.887/(16.26205 - log(P_c)) - 226.346;
57 t = x_1_c*t_1_sat + x_2_c*t_2_sat;
58
59 error = 10;
60 while(error>0.1)
61     P_1_sat = exp(14.39155 - 2795.817/(t + 230.002))
        ;
62     P_2_sat = exp(16.26205 - 3799.887/(t + 226.346))
        ;

```

```

63     P = x_1_c*P_1_sat + x_2_c*P_2_sat;
64     error=abs(P - P_c);
65     t = t - 0.1;
66 end
67
68 P_1_sat_c = exp(14.39155 - 2795.817/(t + 230.002));
69 P_2_sat_c = exp(16.26205 - 3799.887/(t + 226.346));
70
71 y_1_c = (x_1_c*P_1_sat_c)/(P_c);
72 y_2_c = 1 - y_1_c;
73
74 printf("(c).The system temperature is , t = %f C\n",t
    );
75 printf("    The vapour phase composition is , y_1 =
    %f\n\n",y_1_c);
76
77 //(d)
78 y_1_d = 0.57;// Vapour phase composition
79 y_2_d = 1 - y_1_d;
80 P_d = 101.33;//[kPa] - Pressure of the system
81
82 // Since vapour phase composition is given , we can
    use the relation
83 // 1/P = y_1/P_1_sat + y_2/P_2_sat
84 t_1_sat_d = 2795.817/(14.39155 - log(P_d)) -
    230.002;
85 t_2_sat_d = 3799.887/(16.26205 - log(P_d)) -
    226.346;
86 t_d = y_1_d*t_1_sat_d + y_2_d*t_2_sat_d;
87
88 fault = 10;
89 while(fault>0.1)
90     P_1_sat_prime = exp(14.39155 - 2795.817/(t_d +
        230.002));
91     P_2_sat_prime = exp(16.26205 - 3799.887/(t_d +
        226.346));
92     P_prime = 1/(y_1_d/P_1_sat_prime + y_2_d/
        P_2_sat_prime);

```

```

93     fault=abs(P_prime - P_d);
94     t_d = t_d + 0.01;
95 end
96
97 P_1_sat_d = exp(14.39155 - 2795.817/(t_d + 230.002))
    ;
98 P_2_sat_d = exp(16.26205 - 3799.887/(t_d + 226.346))
    ;
99
100 x_1_d = (y_1_d*P_d)/(P_1_sat_d);
101 x_2_d = 1 - x_1_d;
102
103 printf("(d).The system temperature is , t = %f C\n",
    t_d);
104 printf("    The liquid phase composition is , x_1 =
    %f\n\n", x_1_d);

```

Scilab code Exa 15.3 Calculation of pressure temperature and composition

```

1 clear;
2 clc;
3 funcprot(0);
4
5 //Example - 15.3
6 //Page number - 516
7 printf("Example - 15.3 and Page number - 516\n\n");
8
9 //Given
10 // log(P_1_sat) = 14.3916 - 2795.82/(t + 230.00)
11 // log(P_2_sat) = 14.2724 - 2945.47/(t + 224.00)
12 // log(P_3_sat) = 14.2043 - 2972.64/(t + 209.00)
13
14 //(a)
15 x_1_a = 0.25; // Equilibrium composition of liquid

```

```

    phase
16 x_2_a = 0.35;
17 x_3_a = 1 - x_1_a - x_2_a;
18 t_a = 80; //[C] - Temperature
19
20 // At t = 80 C
21 P_1_sat_a = exp(14.3916 - 2795.82/(t_a + 230.00));
22 P_2_sat_a = exp(14.2724 - 2945.47/(t_a + 224.00));
23 P_3_sat_a = exp(14.2043 - 2972.64/(t_a + 209.00));
24
25 // Since liquid phase composition is given we use
    the relation
26 P_a = x_1_a*P_1_sat_a + x_2_a*P_2_sat_a + x_3_a*
    P_3_sat_a; //[kPa]
27
28 y_1_a = (x_1_a*P_1_sat_a)/(P_a);
29 y_2_a = (x_2_a*P_2_sat_a)/(P_a);
30 y_3_a = (x_3_a*P_3_sat_a)/(P_a);
31
32 printf("(a).The system pressure is , P = %f kPa\n",
    P_a);
33 printf("    The vapour phase composition is given by
    , y_1 = %f, y_2 = %f and y_3 = %f \n\n",y_1_a,
    y_2_a,y_3_a);
34
35 //(2)
36 y_1_b = 0.50; // Equilibrium composition of liquid
    phase
37 y_2_b = 0.30;
38 y_3_b = 1 - y_1_a - y_2_a;
39 t_b = 85; //[C] - Temperature
40
41 // At t = 80 C
42 P_1_sat_b = exp(14.3916 - 2795.82/(t_b + 230.00));
43 P_2_sat_b = exp(14.2724 - 2945.47/(t_b + 224.00));
44 P_3_sat_b = exp(14.2043 - 2972.64/(t_b + 209.00));
45
46 // Since vapour phase composition is given we use

```

```

    the relation
47 P_b = 1/(y_1_b/P_1_sat_b + y_2_b/P_2_sat_b + y_3_b/
    P_3_sat_b); // [kPa]
48
49 // Therefore we have
50 x_1_b = (y_1_b*P_b)/P_1_sat_b;
51 x_2_b = (y_2_b*P_b)/P_2_sat_b;
52 x_3_b = (y_3_b*P_b)/P_3_sat_b;
53
54 printf("(b).The system pressure is , P = %f kPa\n",
    P_b);
55 printf("    The liquid phase composition is given by
    , x_1 = %f, x_2 = %f and x_3 = %f \n\n",x_1_b,
    x_2_b,x_3_b);
56
57 //(c)
58 x_1_c = 0.30; // Equilibrium composition of liquid
    phase
59 x_2_c = 0.45;
60 x_3_c = 1 - x_1_c - x_2_c;
61 P_c = 80; // [kPa] - Pressure of the system
62
63 // We have , P = x_1*P_1_sat + x_2*P_2_sat + x_3*
    P_3_sat
64 t_1_sat = 2795.82/(14.3916 - log(P_c)) - 230.00; // [C
    ]
65 t_2_sat = 2945.47/(14.2724 - log(P_c)) - 224.00; // [C
    ]
66 t_3_sat = 2972.64/(14.2043 - log(P_c)) - 209.00; // [C
    ]
67 t = x_1_c*t_1_sat + x_2_c*t_2_sat + x_3_c*t_3_sat;
68
69 error = 10;
70 while(error>0.5)
71     P_1_sat = exp(14.3916 - 2795.82/(t + 230.00));
72     P_2_sat = exp(14.2724 - 2945.47/(t + 224.00));
73     P_3_sat = exp(14.2043 - 2972.64/(t + 209.00));
74     P = x_1_c*P_1_sat + x_2_c*P_2_sat + x_3_c*

```

```

        P_3_sat;
75     error=abs(P - P_c);
76     t = t - 0.2;
77 end
78
79 P_1_sat_c = exp(14.3916 - 2795.82/(t + 230.00));
80 P_2_sat_c = exp(14.2724 - 2945.47/(t + 224.00));
81 P_3_sat_c = exp(14.2043 - 2972.64/(t + 209.00));
82
83 y_1_c = (x_1_c*P_1_sat_c)/(P_c);
84 y_2_c = (x_2_c*P_2_sat_c)/(P_c);
85 y_3_c = 1 - y_1_c - y_2_c;
86
87 printf("(c).The system temperature is , t = %f C\n",t
    );
88 printf("    The vapour phase composition is , y_1 =
    %f, y_2 %f and y_3 = %f\n\n",y_1_c,y_2_c,y_3_c);
89
90 //(d)
91 y_1_d = 0.6; // Vapour phase composition
92 y_2_d = 0.2;
93 y_3_d = 1 - y_1_d - y_2_d;
94 P_d = 90; // [kPa] - Pressure of the system
95
96 // Since vapour phase composition is given , we can
    use the relation
97 //  $1/P = y_1/P_{1\_sat} + y_2/P_{2\_sat} + y_3/P_{3\_sat}$ 
98 t_1_sat_d = 2795.82/(14.3916 - log(P_d)) - 230.00;
99 t_2_sat_d = 2945.47/(14.2724 - log(P_d)) - 224.00;
100 t_3_sat_d = 2972.64/(14.2043 - log(P_d)) - 209.00;
101 t_d = y_1_d*t_1_sat_d + y_2_d*t_2_sat_d + y_3_d*
    t_3_sat_d;
102
103 fault = 10;
104 while(fault>0.5)
105     P_1_sat_prime = exp(14.3916 - 2795.82/(t_d +
        230.00));
106     P_2_sat_prime = exp(14.2724 - 2945.47/(t_d +

```

```

        224.00));
107     P_3_sat_prime = exp(14.2043 - 2972.64/(t_d +
        209.00));
108     P_prime = 1/(y_1_d/P_1_sat_prime + y_2_d/
        P_2_sat_prime + y_3_d/P_3_sat_prime);
109     fault=abs(P_prime - P_d);
110     t_d = t_d + 0.2;
111 end
112
113 P_1_sat_d = exp(14.3916 - 2795.82/(t_d + 230.00));
114 P_2_sat_d = exp(14.2724 - 2945.47/(t_d + 224.00));
115 P_3_sat_d = exp(14.2043 - 2972.64/(t_d + 209.00));
116
117 x_1_d = (y_1_d*P_d)/(P_1_sat_d);
118 x_2_d = (y_2_d*P_d)/(P_2_sat_d);
119 x_3_d = 1 - x_1_d - x_2_d;
120
121 printf("(d).The system temperature is , t = %f C\n",
        t_d);
122 printf("    The liquid phase composition is , x_1 =
        %f, x_2 = %f and x_3 = %f\n\n",x_1_d,x_2_d,x_3_d)
        ;

```

Scilab code Exa 15.4 Calculation of pressure and composition

```

1 clear;
2 clc;
3
4 //Example - 15.4
5 //Page number - 519
6 printf("Example - 15.4 and Page number - 519\n\n");
7
8 //Given
9 T = 120; //[C] - Temperature
10 P_1 = 1; //[atm] - Initial pressure

```



```

11 P_1 = P_1*101.325; //[kPa]
12 R = 8.314; //[J/mol*K] - Universal gas constant
13
14 y_1 = 0.3; // Mole fraction of propane
15 y_2 = 0.5; // Mole fraction of butane
16 y_3 = 0.2; // Mole fraction of hexane
17
18 // log(P_1_sat) = 13.7713 - 1892.47/(t + 248.82)
19 // log(P_2_sat) = 13.7224 - 2151.63/(t + 236.91)
20 // log(P_3_sat) = 13.8216 - 2697.55/(t + 224.37)
21
22 //(a)
23 P_1_sat = exp(13.7713 - 1892.47/(T + 248.82));
24 P_2_sat = exp(13.7224 - 2151.63/(T + 236.91));
25 P_3_sat = exp(13.8216 - 2697.55/(T + 224.37));
26
27 // Since vapour phase composition is given we can
    use the relation ,
28 P_2 = 1/(y_1/P_1_sat + y_2/P_2_sat + y_3/P_3_sat); //
    [kPa]
29
30 printf(" (a).The pressure of the mixture when first
    drop condenses is given by, P = %f kPa\n\n",P_2);
31
32 //(b)
33 x_1 = (y_1*P_2)/P_1_sat;
34 x_2 = (y_2*P_2)/P_2_sat;
35 x_3 = (y_3*P_2)/P_3_sat;
36
37 printf(" (b).The liquid phase composition is given
    by, x_1 (propane) = %f, x_2 (butane) = %f and x_3
    (hexane) = %f \n\n",x_1,x_2,x_3);
38
39 //(c)
40 // Work transfer per mol is given by
41 //  $W = \int P*dV = \int (R*T/V)*dV = R*T*$ 
     $\log(V_2/V_1) = R*T*\log(P_1/P_2)$ 
42 w = R*(T+273.15)*log(P_1/P_2); //[J/mol]

```

```

43 // For 100 mol
44 W = w*100; //[J]
45 W = W*10^(-3); //[kJ]
46 printf(" (c).The work required for 100 mol of
      mixture handeled is %f kJ",W);

```

Scilab code Exa 15.5 Calculation of pressure temperature and composition

```

1 clear;
2 clc;
3 funcprot(0);
4
5 //Example - 15.5
6 //Page number - 520
7 printf("Example - 15.5 and Page number - 520\n\n");
8
9 //Given
10 T = 27; //[C] - Temperature
11
12 // log(P_1_sat) = 13.8216 - 2697.55/(t + 224.37)
13 // log(P_2_sat) = 13.8587 - 2911.32/(t + 216.64)
14
15 //(a)
16 x_1_a = 0.2;
17 x_2_a = 1 - x_1_a;
18
19 // At t = 27 C
20 P_1_sat = exp(13.8216 - 2697.55/(T + 224.37)); //[kPa
    ]
21 P_2_sat = exp(13.8587 - 2911.32/(T + 216.64)); //[kPa
    ]
22 P_a = x_1_a*P_1_sat + x_2_a*P_2_sat; //[kPa]
23
24 y_1_a = x_1_a*P_1_sat/P_a;

```

```

25 y_2_a = x_2_a*P_2_sat/P_a;
26
27 printf("(a).The total pressure is , P = %f kPa\n",P_a
    );
28 printf("    The vapour phase composition is given by
    , y_1 = %f and y_2 = %f\n\n",y_1_a,y_2_a);
29
30 //(b)
31 y_1_b = 0.2;
32 y_2_b = 1 - y_1_b;
33 // Since vapour phase composition is given we can
    use the relation
34 P_b = 1/(y_1_b/P_1_sat + y_2_b/P_2_sat);//[kPa]
35
36 // Therefore we have
37 x_1_b = (y_1_b*P_b)/P_1_sat;
38 x_2_b = (y_2_b*P_b)/P_2_sat;
39
40 printf("(b).The total pressure is , P = %f kPa\n",P_b
    );
41 printf("    The liquid phase composition is given by
    , x_1 = %f and x_2 = %f\n\n",x_1_b,x_2_b);
42
43 //(c)
44 P_c = 30;//[kPa] - Total pressure
45 x_1_c = 0.2;
46 x_2_c = 1 - x_1_c;
47
48 // We have , P = x_1*P_1_sat + x_2*P_2_sat
49 t_1_sat = 2697.55/(13.8216 - log(P_c)) - 224.37;
50 t_2_sat = 2911.32/(13.8587 - log(P_c)) - 216.64;
51 t = x_1_c*t_1_sat + x_2_c*t_2_sat;
52
53 fault = 10;
54 while(fault>0.3)
55     P_1_sat = exp(13.8216 - 2697.55/(t + 224.37));
56     P_2_sat = exp(13.8587 - 2911.32/(t + 216.64));
57     P = x_1_c*P_1_sat + x_2_c*P_2_sat;

```

```

58     fault = abs(P - P_c);
59     t = t - 0.1;
60 end
61
62 P_1_sat_c = exp(13.8216 - 2697.55/(t + 224.37));
63 P_2_sat_c = exp(13.8587 - 2911.32/(t + 216.64));
64
65 y_1_c = (x_1_c*P_1_sat_c)/(P_c);
66 y_2_c = 1 - y_1_c;
67
68 printf("(c).The system temperature is , t = %f C\n",t
);
69 printf("    The vapour phase composition is , y_1 =
%f and y_2 = %f \n\n",y_1_c,y_2_c);

```

Scilab code Exa 15.6 Determinatin of DPT and BPT

```

1 clear;
2 clc;
3 funcprot(0);
4
5 //Example - 15.6
6 //Page number - 521
7 printf("Example - 15.6 and Page number - 521\n\n");
8
9 //Given
10 P = 90;//[kPa] - Pressure
11 R = 8.314;//[J/mol*K] - Universal gas constant
12
13 // log(P_sat) = A - B/(t + C)
14
15 // For benzene
16 A_1 = 13.8594;
17 B_1 = 2773.78;
18 C_1 = 220.07;

```

```

19 // For ethyl benzene
20 A_2 = 14.0045;
21 B_2 = 3279.47;
22 C_2 = 213.20;
23
24 x_1 = 0.5; // Equimolar mixture
25 x_2 = 0.5;
26
27 // The bubble point temperature equation is
28 //  $P = x_1 P_{1\_sat} + x_2 P_{2\_sat}$ 
29
30 t_1_sat = B_1 / (A_1 - log(P)) - C_1;
31 t_2_sat = B_2 / (A_2 - log(P)) - C_2;
32 t = x_1 * t_1_sat + x_2 * t_2_sat;
33
34 fault = 10;
35 while (fault > 0.3)
36     P_1_sat = exp(A_1 - B_1 / (t + C_1));
37     P_2_sat = exp(A_2 - B_2 / (t + C_2));
38     P_net = x_1 * P_1_sat + x_2 * P_2_sat;
39     fault = abs(P_net - P);
40     t = t - 0.1;
41 end
42
43 printf(" The bubble point temperature is %f C\n\n", t
44 );
45 // Now let us determine dew point temperature for
46     y_1 = 0.5, and P = 90 kPa
47 y_1 = 0.5; // Equimolar mixture
48 y_2 = 0.5;
49
50 //  $1/P = y_1/P_{1\_sat} + y_2/P_{2\_sat}$ 
51 // Let us start with t = 104.07
52 t_old = 104.07;
53 error = 10;
54 while (error > 0.3)
55     P_1_sat_prime = exp(A_1 - B_1 / (t_old + C_1));

```

```

55     P_2_sat_prime = exp(A_2 - B_2/(t_old + C_2));
56     P_net_prime = 1/(y_1/P_1_sat_prime + y_2/
        P_2_sat_prime);
57     error=abs(P_net_prime - P);
58     t_old = t_old + 0.1;
59 end
60
61 printf(" The dew point temperature is %f C",t_old);

```

Scilab code Exa 15.7 Determination of range of temperature for which two phase exists

```

1  clear;
2  clc;
3  funcprot(0);
4
5  //Example - 15.7
6  //Page number - 522
7  printf("Example - 15.7 and Page number - 522\n\n");
8
9  //Given
10 P = 1; //[bar] - Pressure
11 P = P*10^(2); //[kPa]
12
13 // log(P_1_sat) = 13.8594 - 2773.78/(t + 220.07)
14 // log(P_2_sat) = 14.0098 - 3103.01/(t + 219.79)
15
16 // The bubble point equation is
17 // P = x_1*P_1_sat + x_2*P_2_sat;
18
19 t_1_sat = 2773.78/(13.8594 - log(P)) - 220.07;
20 t_2_sat = 3103.01/(14.0098 - log(P)) - 219.79;
21
22 // For x_1 = 0.1
23 // t = x_1_1*t_1_sat + x_2_1*t_2_sat;

```

```

24 x_1 = [0.1,0.5,0.9];
25
26 for i=1:3
27 x_2(i) = 1 - x_1(i);
28 t = x_1(i)*t_1_sat + x_2(i)*t_2_sat;
29 fault = 10;
30 while(fault>0.3)
31     P_1_sat = exp(13.8594 - 2773.78/(t + 220.07));
32     P_2_sat = exp(14.0098 - 3103.01/(t + 219.79));
33     P_net = x_1(i)*P_1_sat + x_2(i)*P_2_sat;
34     fault=abs(P_net - P);
35     t = t - 0.1;
36 end
37
38 printf(" The bubble point temperature (for x_1 = %f)
        is %f C\n",x_1(i),t);
39
40 end
41
42 printf("\n\n");
43
44 // Now let us determine dew point temperature
45 // 1/P = y_1/P_1_sat + y_2/P_2_sat
46
47 y_1 = [0.1,0.5,0.9];
48
49 for j=1:3
50 y_2(j) = 1 - y_1(j);
51 t_prime = y_1(j)*t_1_sat + y_2(j)*t_2_sat;
52
53 error = 10;
54 while(error>0.3)
55     P_1_sat = exp(13.8594 - 2773.78/(t_prime +
        220.07));
56     P_2_sat = exp(14.0098 - 3103.01/(t_prime +
        219.79));
57     P_net = 1/(y_1(j)/P_1_sat + y_2(j)/P_2_sat);
58     error=abs(P_net - P);

```

```

59     t_prime = t_prime + 0.1;
60 end
61
62 printf(" The dew point temperature (for y_1 = %f) is
        %f C\n", y_1(j), t_prime);
63
64 end
65
66 printf("\n\n");
67
68 //Therefore
69 printf(" The temperature range for (z_1 = 0.1) which
        two phase exist is 105.61 C to 108.11 C\n");
70 printf(" The temperature range for (z_1 = 0.5) which
        two phase exist is 91.61 C to 98.40 C\n");
71 printf(" The temperature range for (z_1 = 0.9) which
        two phase exist is 81.71 C to 84.51 C\n");

```

Scilab code Exa 15.8 Calculation of DPT and BPT

```

1 clear;
2 clc;
3 funcprot(0);
4
5 //Example - 15.8
6 //Page number - 524
7 printf("Example - 15.8 and Page number - 524\n\n");
8
9 //Given
10 x_1 = 0.20;
11 x_2 = 0.45;
12 x_3 = 0.35;
13 P = 10; // [atm]
14 P = P*101325*10^(-3); // [kPa]
15

```



```

16 // log(P_1_sat) = 13.7713 - 1892.47/(t + 248.82)
17 // log(P_2_sat) = 13.7224 - 2151.63/(t + 236.91)
18 // log(P_3_sat) = 13.8183 - 2477.07/(t + 233.21)
19
20 //(a)
21 // The bubble point equation is
22 // P = x_1*P_1_sat + x_2*P_2_sat + x_3*P_3_sat
23
24 t_1_sat = 1892.47/(13.7713 - log(P)) - 248.82;
25 t_2_sat = 2151.63/(13.7224 - log(P)) - 236.91;
26 t_3_sat = 2477.07/(13.8183 - log(P)) - 233.21;
27 t = x_1*t_1_sat + x_2*t_2_sat + x_3*t_3_sat;
28
29 fault = 10;
30 while(fault>0.1)
31     P_1_sat = exp(13.7713 - 1892.47/(t + 248.82));
32     P_2_sat = exp(13.7224 - 2151.63/(t + 236.91));
33     P_3_sat = exp(13.8183 - 2477.07/(t + 233.21));
34     P_net = x_1*P_1_sat + x_2*P_2_sat + x_3*P_3_sat;
35     fault=abs(P_net - P);
36     t = t - 0.003;
37 end
38
39 BPT = t;
40 printf(" (a).The bubble point temperature is %f C\n\n",BPT);
41
42 //(b)
43 // Now let us determine dew point temperature for
44 // y_1 = 0.5, and P = 90 kPa
44 y_1 = 0.20;
45 y_2 = 0.45;
46 y_3 = 0.35;
47
48 // 1/P = y_1/P_1_sat + y_2/P_2_sat + y_3/P_3_sat
49
50 t_old = 90; //[C]
51 error = 10;

```

```

52 while(error>0.1)
53     P_1_sat_prime = exp(13.7713 - 1892.47/(t_old +
        248.82));
54     P_2_sat_prime = exp(13.7224 - 2151.63/(t_old +
        236.91));
55     P_3_sat_prime = exp(13.8183 - 2477.07/(t_old +
        233.21));
56     P_net_prime = 1/(y_1/P_1_sat_prime + y_2/
        P_2_sat_prime + y_3/P_3_sat_prime);
57     error=abs(P_net_prime - P);
58     t_old = t_old + 0.003;
59 end
60
61 DPT = t_old;
62 printf(" (b).The dew point temperature is %f C\n\n",
        DPT);
63
64 // (c)
65 // For the given composition and pressure the two
        phase region exists in the temperature range of
        DPT and BPT
66 // Therefore at 82 C two phase exists
67 // At 82 C and P = 1013.25 kPa pressure
68 T_c = 82;//[C]
69 P_c = 1013.25;//[kPa]
70 z_1 = 0.20;
71 z_2 = 0.45;
72 z_3 = 0.35;
73
74 P_1_sat_c = exp(13.7713 - 1892.47/(T_c + 248.82));
75 P_2_sat_c = exp(13.7224 - 2151.63/(T_c + 236.91));
76 P_3_sat_c = exp(13.8183 - 2477.07/(T_c + 233.21));
77
78 K_1 = P_1_sat_c/P_c;
79 K_2 = P_2_sat_c/P_c;
80 K_3 = P_3_sat_c/P_c;
81
82 // We have to find such a V that the following

```

```

equation is satisfied.
83 // summation(y_i) = K_i*z_i/(1-V+V*K_i) = 1
84 // K_1*z_1/(1-V+V*K_1) + K_2*z_2/(1-V+V*K_2) + K_3*
    z_3/(1-V+V*K_3) = 1
85
86 def f(' [y]=f1(V) ', 'y= K_1*z_1/(1-V+V*K_1) + K_2*z_2
    /(1-V+V*K_2) + K_3*z_3/(1-V+V*K_3)-1 ');
87 V = fsolve(0.4, f1);
88
89 // Therefore now we have
90 y_1_c = K_1*z_1/(1-V+V*K_1);
91 y_2_c = K_2*z_2/(1-V+V*K_2);
92 y_3_c = K_3*z_3/(1-V+V*K_3);
93 x_1_c = y_1_c/K_1;
94 x_2_c = y_2_c/K_2;
95 x_3_c = y_3_c/K_3;
96
97 printf(" (c).The proportion of vapour is given by, V
    = %f\n\n", V);
98 printf("      The composition of vapour foemed is
    given by, y_1 = %f, y_2 = %f and y_3 = %f \n\n",
    y_1_c, y_2_c, y_3_c);
99 printf("      The composition of liquid formed is
    given by, x_1 = %f, x_2 = %f and x_3 = %f \n\n",
    x_1_c, x_2_c, x_3_c);

```

Scilab code Exa 15.9 Calculation of range of pressure for which two phase exists

```

1 clear;
2 clc;
3
4 //Example - 15.9
5 //Page number - 526
6 printf("Example - 15.9 and Page number - 526\n\n");

```

```

7
8 //Given
9 T = 27; //[C] - Temperature
10 z_1 = 0.4;
11 z_2 = 0.3;
12 z_3 = 0.3;
13
14 // log(P_sat) = A - B/(t + C)
15
16 // For propane
17 A_1 = 13.7713;
18 B_1 = 1892.47;
19 C_1 = 248.82;
20 // For i-butane
21 A_2 = 13.4331;
22 B_2 = 1989.35;
23 C_2 = 236.84;
24 // For n-butane
25 A_3 = 13.7224;
26 B_3 = 2151.63;
27 C_3 = 236.91;
28
29 //(a)
30 // The pressure range for the existence of two phase
    region lies between dew point and bubble point
    pressures.
31 // At the dew point the whole feed lies in the
    vapour phase and a drop of liquid is formed,
    therefore
32 y_1 = z_1;
33 y_2 = z_2;
34 y_3 = z_3;
35
36 // At 27 C,
37 P_1_sat = exp(A_1 - B_1/(T + C_1));
38 P_2_sat = exp(A_2 - B_2/(T + C_2));
39 P_3_sat = exp(A_3 - B_3/(T + C_3));
40

```

```

41 // The dew point pressure is given by
42 P_1 = 1/(y_1/P_1_sat + y_2/P_2_sat + y_3/P_3_sat);
43
44 // At the bubble point the whole feed lies in the
    liquid phase and an infinitesimal amount of
    vapour is formed, therefore
45 x_1 = z_1;
46 x_2 = z_2;
47 x_3 = z_3;
48
49 // The bubble point pressure is given by
50 P_2 = x_1*P_1_sat + x_2*P_2_sat + x_3*P_3_sat;
51
52 printf(" (a).The two phase region exists between %f
    and %f kPa\n\n",P_1,P_2);
53
54 //(b)
55 // The mean of the two-phase pressure range is given
    by
56 P_mean = (P_1 + P_2)/2;
57
58 // Now let us calculate the K values of the
    components
59 K_1 = P_1_sat/P_mean;
60 K_2 = P_2_sat/P_mean;
61 K_3 = P_3_sat/P_mean;
62
63 // summation of y_i = 1, gives
64 // (K_1*z_1)/(1-V-K_1*V) + (K_2*z_2)/(1-V-K_2*V) + (
    K_3*z_3)/(1-V-K_3*V) = 1
65 // Solving we get
66 deff(' [y]=f(V) ', 'y=(K_1*z_1)/(1-V+K_1*V) + (K_2*z_2)
    /(1-V+K_2*V) + (K_3*z_3)/(1-V+K_3*V)-1');
67 V = fsolve(0.1,f);
68
69 y_1_prime = (z_1*K_1)/(1-V+K_1*V);
70
71 printf(" (b).The mole fraction of propane in vapour

```

```
phase is %f whereas in the feed is %f and
fraction of vapour in the system is %f",y_1_prime
,z_1,V);
```

Scilab code Exa 15.10 Determination of vapour and liquid phase composition

```
1 clear;
2 clc;
3
4 //Example - 15.10
5 //Page number - 527
6 printf("Example - 15.10 and Page number - 527\n\n");
7
8 //Given
9 T = 50; //[C] - Temperature
10 P = 64; //[kPa] - Pressure
11 z_1 = 0.7;
12 z_2 = 0.3;
13
14 // log(P_sat) = A - B/(t + C)
15
16 // For acetone
17 A_1 = 14.37824;
18 B_1 = 2787.498;
19 C_1 = 229.664;
20 // For acetonitrile
21 A_2 = 14.88567;
22 B_2 = 3413.099;
23 C_2 = 250.523;
24
25 // At 50 C,
26 P_1_sat = exp(A_1 - B_1/(T + C_1)); //[kPa]
27 P_2_sat = exp(A_2 - B_2/(T + C_2)); //[kPa]
28
```

```

29 // Now let us calculate the K values of the
    components
30 K_1 = P_1_sat/P;
31 K_2 = P_2_sat/P;
32
33 // summation of y_i = 1, gives
34 // (K_1*z_1)/(1-V-K_1*V) + (K_2*z_2)/(1-V-K_2*V) = 1
35 // Solving we get
36 def(' [y]=f(V) ', 'y=(K_1*z_1)/(1-V+K_1*V) + (K_2*z_2)
    /(1-V+K_2*V) -1 ');
37 V = fsolve(0.1,f);
38 L = 1 - V;
39 // Therefore
40 y_1 = (K_1*z_1)/(1-V+K_1*V);
41 y_2 = (K_2*z_2)/(1-V+K_2*V);
42
43 x_1 = y_1/K_1;
44 x_2 = y_2/K_2;
45
46 printf(" The value of V = %f\n",V);
47 printf(" The value of L = %f\n\n",L);
48 printf(" The liquid phase composition is , x_1 = %f
    and x_2 = %f\n",x_1,x_2);
49 printf(" The vapour phase composition is , y_1 = %f
    and y_2 = %f",y_1,y_2);

```

Scilab code Exa 15.11 Calculation of temperature

```

1 clear;
2 clc;
3 funcprot(0);
4
5 //Example - 15.11
6 //Page number - 528
7 printf("Example - 15.11 and Page number - 528\n\n");

```

```

8
9 // Given
10 P = 12.25*101325*10^(-3); // [kPa]
11 z_1 = 0.8;
12 z_2 = 1 - z_1;
13 V = 0.4;
14 // log(P_1_sat) = 13.7713 - 2892.47/(T + 248.82)
15 // log(P_2_sat) = 13.7224 - 2151.63/(T + 236.91)
16
17 // P_1_sat = exp(13.7713 - 21892.47/(T + 248.82));
18 // P_2_sat = exp(13.7224 - 2151.63/(T + 236.91));
19
20 // Let the total mixture be 1 mol
21 // We have to assume a temperature such that,
22 // y_1 + y_2 = (K_1*z_1)/(1-V-K_1*V) + (K_2*z_2)/(1-
    V-K_2*V) = 1
23
24 // To assume a temperature we have to determine the
    BPT and DPT and take a temperature in between the
    range BPT to DPT
25
26 // At the bubble point the whole feed lies in the
    liquid phase and an infinitesimal amount of
    vapour is formed, therefore
27 x_1 = z_1;
28 x_2 = z_2;
29
30 // The bubble point pressure is given by
31 // P = x_1*(exp(13.7713 - 21892.47/(T + 248.82))) +
    x_2*(exp(13.7224 - 2151.63/(T + 236.91)));
32
33 def f(' [y]=f(T) ', 'y=x_1*(exp(13.7713 - 1892.47/(T +
    248.82))) + x_2*(exp(13.7224 - 2151.63/(T +
    236.91))) - P');
34 T_1 = fsolve(0.1, f);
35 BPT = T_1;
36
37 // At the dew point the whole feed lies in the

```



```

        vapour phase and a drop of liquid is formed,
        therefore
38 y_1 = z_1;
39 y_2 = z_2;
40
41 // The dew point equation is given by
42 //  $1/P = y_1/P_{1\_sat} + y_2/P_{2\_sat}$ 
43 deff(' [y]=f1(T) ', 'y=1/(y_1/(exp(13.7713 - 1892.47/(T
        + 248.82))) + y_2/(exp(13.7224 - 2151.63/(T +
        236.91)))) - P');
44 T_2 = fsolve(0.1,f1);
45 DPT = T_2;
46
47 // Now the assumed temperature should be in the
        range of BPT and DPT
48 // Let the assumed temperature be 47 C
49 T = 47; // [C]
50 error = 10;
51 while(error>0.001)
52     P_1_sat = exp(13.7713 - 1892.47/(T + 248.82));
53     P_2_sat = exp(13.7224 - 2151.63/(T + 236.91));
54     K_1 = P_1_sat/P;
55     K_2 = P_2_sat/P;
56     y1 = (K_1*z_1)/(1-V+K_1*V);
57     y2 = (K_2*z_2)/(1-V+K_2*V);
58     y = y1 + y2;
59     error=abs(y - 1);
60     T = T - 0.0001;
61 end
62
63 printf(" The temperature when 40 mol %% of mixture
        is in the vapour is %f C",T);

```

Scilab code Exa 15.12 Determination of number of moles in liquid and vapour phase

```

1  clear;
2  clc;
3
4  //Example - 15.12
5  //Page number - 529
6  printf("Example - 15.12 and Page number - 529\n\n");
7
8  //Given
9  T = 105; //[C]
10 P = 1.5; //[atm]
11 P = P*101325*10^(-3); //[kPa]
12 z = [0.4,0.3667,0.2333]; // Feed composition
13 x = [0.3,0.3,0.4]; // Equilibrium liquid phase
    composition
14 y = [0.45,0.40,0.15]; // Equilibrium vapour phase
    composition
15
16 // From the material balance equation of component
    1, we get
17 // (L + V)*z_1 = L*x_1 + V*y_1
18
19 // Since total moles are one, therefore L + V = 1
    and thus
20 // z_1 = L*x_1 + (1-L)*y_1
21
22 for i=1:3;
23     L = (z(i) - y(i))/(x(i) - y(i));
24     V = 1 - L;
25     printf(" The number of moles in liquid phase (z
        = %f) is given by ,L = %f\n",z(i),L);
26     printf(" The number of moles in vapour phase (z
        = %f) is given by ,V = %f\n\n",z(i),V);
27 end

```

Scilab code Exa 15.13 Determination of vapour and liquid phase composition

```
1 clear;
2 clc;
3
4 //Example - 15.13
5 //Page number - 530
6 printf("Example - 15.13 and Page number - 530\n\n");
7
8 //Given
9 T = 90; // [C]
10 P = 1; // [atm]
11 P = P*101325*10^(-3); // [kPa]
12 z_1 = [0.1, 0.5, 0.8];
13
14 // log(P_1_sat) = 13.8594 - 2773.78/(t + 220.07)
15 // log(P_2_sat) = 14.0098 - 3103.01/(t + 219.79)
16
17 //At T = 90 C
18 P_1_sat = exp(13.8594 - 2773.78/(T + 220.07));
19 P_2_sat = exp(14.0098 - 3103.01/(T + 219.79));
20 K_1 = P_1_sat/P;
21 K_2 = P_2_sat/P;
22
23 // For z_1 = 0.1
24 // y1 = (K_1*z_1(i))/(1-V+K_1*V);
25 // y2 = (K_2*z_2)/(1-V+K_2*V);
26 // We do not get a value between 0 and 1 such that,
    y = y1 + y2 = 1;
27 // This means that at z_1 = 0.1 two phases do not
    exist.
28 // At given temperature and pressure, let us
    determine the equilibrium liquid and vapour
    phase compositions
29
30 x_1 = (P - P_2_sat)/(P_1_sat - P_2_sat);
31 y_1 = (x_1*P_1_sat)/(P);
```

```

32
33 // For two phase region to exist at 90 C and 101.325
    kPa, z_1 sholud lie between x_1 and y_1
34
35 printf(" For two phase region to exist at 90 C and
    101.325 kPa, z_1 sholud lie between %f and %f\n\n
    ",x_1,y_1);
36 printf(" For z_1 = 0.1 and z_1 = 0.5, only liquid
    phase exists (V = 0).\n\n");
37 printf(" For z_1 = 0.8, only vapour exists (V = 1).\n
    n")

```

Scilab code Exa 15.14 Preparation of table having composition and pressure data

```

1 clear;
2 clc;
3
4 //Example - 15.14
5 //Page number - 531
6 printf("Example - 15.14 and Page number - 531\n\n");
7
8 //Given
9 T = 90; //[C]
10 P = 1; //[atm]
11 P = P*101325*10^(-3); //[kPa]
12 z_1 = [0.1,0.5,0.8];
13
14 // log(P_1_sat) = 13.8594 - 2773.78/(t + 220.07)
15 // log(P_2_sat) = 14.0098 - 3103.01/(t + 219.79)
16
17 //(a)
18 //At T = 90 C
19 P_1_sat = exp(13.8594 - 2773.78/(T + 220.07));
20 P_2_sat = exp(14.0098 - 3103.01/(T + 219.79));

```

```

21 K_1 = P_1_sat/P;
22
23 x_1 = [0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0];
24 P_prime = zeros(1,11);
25 x_2 = zeros(11);
26 y_1 = zeros(11);
27
28 printf(" (a).\n\n");
29 printf(" x_1 \t\t P \t\t y_1 \n\n");
30
31 for i=1:11;
32     x_2(i) = 1 - x_1(i);
33     P_prime(i) = x_1(i)*P_1_sat + x_2(i)*P_2_sat;
34     y_1(i) = (x_1(i)*P_1_sat)/P_prime(i);
35     printf(" %f \t %f \t %f \n",x_1(i),P_prime(
        i),y_1(i));
36 end
37
38 //(b)
39 T_1_sat = 2773.78/(13.8594-log(P)) - 220.07; //[C]
40 T_2_sat = 3103.01/(14.0098-log(P)) - 219.79; //[C]
41
42 T_prime =
    [110.62,107,104,101,98,95,92,89,86,83,80.09];
43
44 P1_sat = zeros(11);
45 P2_sat = zeros(11);
46 x_1 = zeros(11);
47 y_1 = zeros(11);
48
49 printf(" \n\n (b).\n\n");
50 printf(" T(C) \t\t P_1_sat (kPa) \t\t P_2_sat (
    kPa) \t\t x_1 \t\t y_1 \n\n");
51
52 for j=1:11;
53     P1_sat(j) = exp(13.8594 - 2773.78/(T_prime(j) +
        220.07));
54     P2_sat(j) = exp(14.0098 - 3103.01/(T_prime(j) +

```

```

        219.79));
55     x_1(j) = (P-P2_sat(j))/(P1_sat(j)-P2_sat(j));
56     y_1(j) = (x_1(j)*P1_sat(j))/P;
57     printf(" %f \t      %f      \t      %f      \t      %f \t
           %f \n", T_prime(j), P1_sat(j), P2_sat(j)
           ,x_1(j), y_1(j));
58 end

```

Scilab code Exa 15.15 Calculation of DPT and BPT

```

1  clear;
2  clc;
3  funcprot(0);
4
5  //Example - 15.15
6  //Page number - 533
7  printf("Example - 15.15 and Page number - 533\n\n");
8
9  //Given
10 // log(Y1) = 0.95*x_2^(2)
11 // log(Y2) = 0.95*x_1^(2)
12 P_1_sat = 79.80; // [kPa]
13 P_2_sat = 40.45; // [kPa]
14
15 //(1)
16 T = 373.15; // [K]
17 x_1 = 0.05;
18 x_2 = 1 - x_1;
19 Y1 = exp(0.95*x_2^(2));
20 Y2 = exp(0.95*x_1^(2));
21
22 // The total pressure of the system is given by
23 P = x_1*Y1*P_1_sat + x_2*Y2*P_2_sat; // [kPa]
24 y_1 = x_1*Y1*P_1_sat/P;
25 y_2 = x_2*Y2*P_2_sat/P;

```

```

26
27 printf(" (1).The first bubble is formed at %f kPa
    and the composition , y_1 = %f\n\n",P,y_1);
28
29 //(2)
30 T = 373.15;//[K]
31 y_1_prime = 0.05;
32 y_2_prime = 1 - y_1_prime;
33
34 // Let us assume a value of x_1 ,
35 x_1_prime = 0.0001;
36
37 error = 10;
38 while(error>0.001)
39     x_2_prime = 1 - x_1_prime;
40     Y1_prime = exp(0.95*x_2_prime^(2));
41     Y2_prime = exp(0.95*x_1_prime^(2));
42     P_prime = x_1_prime*Y1_prime*P_1_sat + x_2_prime
        *Y2_prime*P_2_sat;
43     x_1 = (y_1_prime*P_prime)/(Y1_prime*P_1_sat);
44     error=abs(x_1_prime - x_1);
45     x_1_prime = x_1_prime + 0.00001;
46 end
47
48 P_2 = x_1_prime*Y1_prime*P_1_sat + x_2_prime*
    Y2_prime*P_2_sat;
49
50 printf(" (2).The first drop is formed at %f kPa and
    has the composition , x_1 = %f",P_2,x_1_prime);

```

Scilab code Exa 15.16 Calculation of pressure

```

1 clear;
2 clc;
3 funcprot(0);

```

```

4
5 //Example - 15.16
6 //Page number - 534
7 printf("Example - 15.16 and Page number - 534\n\n");
8
9 //Given
10 T = 78.15; // [C]
11 P_1_sat = 755; // [mm Hg]
12 P_2_sat = 329; // [mm Hg]
13
14 z_1 = 0.3;
15 V = 0.5;
16
17 // log(Y1) = 0.845/(1 + 0.845*(x_1/x_2))^(2)
18 // log(Y2) = 1/(1 + 1.183*(x_2/x_1))^(2)
19
20 // A value of x_1 is to determined for which V =
    0.5
21 // Let us assume a value of x_1, say x_1 = 0.150
22 x_1 = 0.150;
23
24 error = 10;
25 while(error>0.001)
26     x_2 = 1 - x_1;
27     Y1 = exp(0.845/(1 + 0.845*(x_1/x_2))^(2));
28     Y2 = exp(1/(1 + 1.183*(x_2/x_1))^(2));
29     P = x_1*Y1*P_1_sat + x_2*Y2*P_2_sat;
30     y_1 = (x_1*Y1*P_1_sat)/P;
31     V_prime = (z_1 - x_1)/(y_1 - x_1);
32     error=abs(V_prime - V);
33     x_1 = x_1 + 0.00001;
34 end
35
36 P_prime = x_1*Y1*P_1_sat + x_2*Y2*P_2_sat; // [mm hg]
37
38 // At x_1 , V = 0.5 ,
39 // Therefore when the mixture is 50 % vaporized at
    78.15 C the mole fraction of component 1 in the

```



```

    liquid phase is x_1 and the system pressure is
    P_prime
40
41 printf(" The required pressure is %f mm Hg\n\n",
    P_prime);
42 printf(" and the mole fraction of component 1 in the
    liquid phase for this pressure is x_1 = %f\n\n",
    x_1);

```

Scilab code Exa 15.17 Calculation of van Laar activity coefficient parameters

```

1 clear;
2 clc;
3
4 //Example - 15.17
5 //Page number - 536
6 printf("Example - 15.17 and Page number - 536\n\n");
7
8 //Given
9 T = 25; // [C] - Temperature
10 P =
    [118.05,124.95,137.90,145.00,172.90,207.70,227.70,237.85,253.90,2
    // [mm Hg]
11 x_1 =
    [0.0115,0.0160,0.0250,0.0300,0.0575,0.1125,0.1775,0.2330,0.4235,0
12 y_1 =
    [0.1810,0.2250,0.3040,0.3450,0.4580,0.5670,0.6110,0.6325,0.6800,0
13
14 // Pressure value for which x_1 = y_1 = 0,
    corresponds to P_2_sat, therefore
15 P_2_sat = 97.45; // [mm Hg]
16 // Pressure value for which x_1 = y_1 = 1,

```

```

    corresponds to P_1_sat , therefore
17 P_1_sat = 230.40; // [mm Hg]
18
19 x_2 = zeros(1,15);
20 y_2 = zeros(1,15);
21 Y1 = zeros(1,15);
22 Y2 = zeros(1,15);
23 GE_RT = zeros(1,15);
24 x1x2_GE_RT = zeros(1,15);
25 for i=1:15;
26     x_2(1,i) = 1 - x_1(i);
27     y_2(1,i) = 1 - y_1(i);
28     Y1(1,i) = (y_1(i)*P(i))/(x_1(i)*P_1_sat);
29     Y2(1,i) = (y_2(i)*P(i))/(x_2(i)*P_2_sat);
30     GE_RT(1,i) = x_1(i)*log(Y1(i)) + x_2(i)*log(Y2(i)
        )); // G.E/(R*T)
31     x1x2_GE_RT(1,i) = (x_1(i)*x_2(i))/GE_RT(i);
32 end
33
34 [M,N,sig]=reglin(x_1,x1x2_GE_RT);
35
36 // Linear regression between x_1 and x_1*x_2/(G.E/R*
    T) gives intercept = N and slope = M
37
38 // van Laar equation is x_1*x_2/(G.E/R*T) = 1/A +
    (1/B - 1/A)
39 // 1/A = N
40 A = 1/N;
41 B = 1/(M + 1/A);
42
43 printf(" The value of Van Laar coefficient A = %f\n
    \n",A);
44 printf(" The value of Van Laar coefficient B = %f\n
    ",B);

```

Scilab code Exa 15.18 Prediction of azeotrope formation

```
1 clear;
2 clc;
3
4 //Example - 15.18
5 //Page number - 541
6 printf("Example - 15.18 and Page number - 541\n\n");
7
8 //Given
9 T = 343.15; //[K] - Temperature
10 // At 343.15 K
11 // log(Y1) = 0.95*x_2^(2)
12 // log(Y2) = 0.95*x_1^(2)
13 P_1_sat = 79.80; //[kPa]
14 P_2_sat = 40.50; //[kPa]
15
16 // At x_1 = 0,
17 Y1_infinity = exp(0.95);
18 alpha_12_x0 = (Y1_infinity*P_1_sat)/(P_2_sat);
19 // At x_1 = 1,
20 Y2_infinity = exp(0.95);
21 alpha_12_x1 = (P_1_sat)/(Y2_infinity*P_2_sat);
22
23 // Within the range alpha_12_x0 and alpha_12_x1, the
    relative volatility continuously decrease and
    thus a value of 1.0 is obtained and thus
    azeotrope is formed.
24 // At azeotrope, Y1*P1_sat = Y2*P2_sat
25 // Y2/Y1 = P_1_sat/P_2_sat
26 // Taking logarithm of both sides we get
27 // log(Y2) - log(Y1) = log(P_1_sat/P_2_sat)
28 // 0.95*x_1^(2) - 0.95*x_2^(2) = log(P_1_sat/P_2_sat
    )
29 // Solving the above equation
30 def f(' [y]=f(x_1)', 'y=0.95*x_1^(2) - 0.95*(1-x_1)^(2)
    - log(P_1_sat/P_2_sat) ');
31 x_1 = fsolve(0.1,f);
```

```

32
33 // At x_1
34 x_2 = 1 - x_1;
35 Y1 = exp(0.95*x_2^(2));
36 Y2 = exp(0.95*x_1^(2));
37 P = x_1*Y1*P_1_sat + x_2*Y2*P_2_sat; //[kPa] -
    Azeotrope pressure
38 y_1 = (x_1*Y1*P_1_sat)/P;
39
40 // Since x_1 = y_1, (almost equal) ,the above
    condition is of azeotrope formation
41
42 // Since alpha_12 is a continuous curve and in
    between a value of alpha_12 = 1, shall come and
    at this composition the azeotrope shall get
    formed.
43
44 printf(" Since (alpha_12_x=0) = %f and (alpha_12_x
    =1) = %f \n",alpha_12_x0,alpha_12_x1);
45 printf(" and since alpha_12 is a continuous curve
    and in between a value of alpha_12 = 1, shall
    come and at this composition the azeotrope shall
    get formed.\n\n")
46 printf(" The azeotrope composition is x_1 = y_1 = %f
    \n\n",x_1);
47 printf(" The azeotrope presssure is %f kPa\n",P);

```

Scilab code Exa 15.19 Tabulation of activity coefficients relative volatility and compositions

```

1 clear;
2 clc;
3
4 //Example - 15.19
5 //Page number - 541

```

```

6 printf("Example - 15.19 and Page number - 541\n\n");
7
8 //Given
9 T = 45; //[C] - Temperature
10
11 x_1 =
    [0.0455,0.0940,0.1829,0.2909,0.3980,0.5069,0.5458,0.5946,0.7206,0
12 y_1 =
    [0.1056,0.1818,0.2783,0.3607,0.4274,0.4885,0.5098,0.5375,0.6157,0
13 P =
    [31.957,33.553,35.285,36.457,36.996,37.068,36.978,36.778,35.792,3
14
15 // Pressure value for which x_1 = y_1 = 0,
    corresponds to P_2_sat, therefore
16 P_2_sat = 29.819; //[kPa]
17 // Pressure value for which x_1 = y_1 = 1,
    corresponds to P_1_sat, therefore
18 P_1_sat = 27.778; //[kPa]
19
20 x_2 = zeros(1,12);
21 y_2 = zeros(1,12);
22 Y1 = zeros(1,12);
23 Y2 = zeros(1,12);
24 alpha_12 = zeros(1,12);
25 GE_RT = zeros(1,12);
26 x1x2_GE_RT = zeros(1,12);
27
28 printf(" x_1 \t\t y_1 \t\t P \t\t Y1 \t\t
    tY2 \t\t alpha_12 \t\t G.E/RT \t\t x1*x2/(G.E/
    RT)\n\n");
29
30 for i=1:12;
31     x_2(1,i) = 1 - x_1(i);
32     y_2(1,i) = 1 - y_1(i);
33     Y1(1,i) = (y_1(i)*P(i))/(x_1(i)*P_1_sat);

```

```

34     Y2(1,i) = (y_2(i)*P(i))/(x_2(i)*P_2_sat);
35     alpha_12(1,i) = (y_1(i)/x_1(i))/(y_2(i)/x_2(i));
36     GE_RT(1,i) = x_1(i)*log(Y1(i)) + x_2(i)*log(Y2(i)
    )); // G.E/(R*T)
37     x1x2_GE_RT(1,i) = (x_1(i)*x_2(i))/GE_RT(i);
38     printf(" %f\t %f\t %f \t %f \t %f \t
    %f\t %f \t%f\n\n",x_1(i),y_1(i),P(i),
    Y1(i),Y2(i),alpha_12(i),GE_RT(i),x1x2_GE_RT(i)
    ));
39 end
40
41 [M,N,sig]=reglin(x_1,x1x2_GE_RT);
42
43 // Linear regression between x_1 and x_1*x_2/(G.E/R*
    T) gives intercept = N and slope = M
44
45 // Now let us assume the system to follow van Laar
    activity coefficient model.
46 //  $x_1*x_2/(G.E/(R*T)) = x_1/B + x_2/A = x_1/B + (1
    - x_1)/A = 1/A + (1/B - 1/A)*x_1 = N + M*x_1$ 
47
48 //  $1/A = N$ 
49 A = 1/N;
50 //  $(1/B - 1/A) = M$ 
51 B = 1/(M + 1/A);
52
53 printf("\n\n")
54 printf(" The value of van Laar parameters are , A =
    %f and B = %f \n\n",A,B);
55
56 Y1_infinity = exp(A);
57 Y2_infinity = exp(B);
58
59
60 // Azeotrope is formed when maxima ( or mainina) in
    pressure is observed and relative volatility
    becomes 1.
61 // This is the case for x_1 between 0.2980 and

```

```

0.5458.
62 // The ezeotropr os maximum pressure (and thus
    minimum boiling) because at azeotrope the system
    pressure is greater than vapour pressure of pure
    components.
63
64 // Now let us calculate the azeotrope composition.
65 // At azeotrope , Y1*P1_sat = Y2*P2_sat
66 // log(Y1/Y2) = log(P_2_sat/P_1_sat)
67 // From van Laar model we get
68 // log(P_2_sat/P_1_sat) = (A*B^(2)*2*x_2^(2))/(A*x_1
    + B*x_2)^(2) + (B*A^(2)*2*x_1^(2))/(A*x_1 + B*
    x_2)^(2)
69 // Solving the above equation
70 deff('[y]=f(x_1)', 'y= log(P_2_sat/P_1_sat) - (A*B
    ^2*(1-x_1)^(2))/(A*x_1 + B*(1-x_1))^(2) + (B*A
    ^2*x_1^(2))/(A*x_1 + B*(1-x_1))^(2)');
71 x_1 = fsolve(0.1,f);
72
73 printf(" The azeotrope composition is given by x_1 =
    y_1 = %f\n",x_1);

```

Scilab code Exa 15.20 Tabulation of partial pressure and total pressure data of components

```

1 clear;
2 clc;
3
4 //Example - 15.20
5 //Page number - 541
6 printf("Example - 15.20 and Page number - 543\n\n");
7
8 //Given
9 T = 25; //[C] - Temperature
10 P_1_sat = 230.4; //[mm Hg]

```

```

11 P_2_sat = 97.45; // [mm Hg]
12 Y1_infinity = 8.6;
13 Y2_infinity = 6.6;
14
15 // Assuming ideal vapour behaviour means that phi = 1
    and since system pressure is low, therefore
16 // f_i = P_i_sat
17 // Assuming the activity coefficients to follow van
    Laar model we get
18 A = log(Y1_infinity);
19 B = log(Y2_infinity);
20
21 // log(Y1) = A/(1+ (A*x_1)/(B*x_2))^(2)
22 // log(Y2) = B/(1+ (B*x_2)/(A*x_1))^(2)
23
24 x_1 = [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9];
25
26 x_2 = zeros(9);
27 Y1 = zeros(9);
28 Y2 = zeros(9);
29 y1_P = zeros(9);
30 y2_P = zeros(9);
31 P = zeros(9);
32 y_1 = zeros(9);
33
34 printf(" (a).\n\n");
35 printf(" x_1      \t\t\t Y1      \t\t\t Y2      \t\t\t
    y1*P      \t\t\t y2*P      \t\t\t P      \t\t\t y_1
\n\n");
36
37 for i=1:9;
38     x_2(i) = 1 - x_1(i);
39     Y1(i) = exp(A/(1+ (A*x_1(i))/(B*x_2(i))))^(2));
40     Y2(i) = exp(B/(1+ (B*x_2(i))/(A*x_1(i))))^(2));
41     y1_P(i) = x_1(i)*Y1(i)*P_1_sat;
42     y2_P(i) = x_2(i)*Y2(i)*P_2_sat;
43     P(i) = x_1(i)*Y1(i)*P_1_sat + x_2(i)*Y2(i)*
        P_2_sat;

```



```

44     y_1(i) = (x_1(i)*Y1(i)*P_1_sat)/P(i);
45     printf(" %f\t\t %f\t\t %f \t\t %f \t\t %f \t\t
           %f \t\t %f\n\n",x_1(i),Y1(i),Y2(i),y1_P(i),
           y2_P(i),P(i),y_1(i));
46
47 end
48
49 //(b)
50 // The total system pressure versus x_1 shows a
    maxima and thus azeotrope is formed by the VLE
    system
51 // The maxima occurs in the range of x_1 = 0.6 to
    0.8, so an azeotrope is formed in this
    composition range
52
53 // At the azeotrope point, Y1*P1_sat = Y2*P2_sat
54 // log(Y1) - log(Y2) = log(P_2_sat/P_1_sat)
55 // On putting the values and then solving the above
    equation we get
56 deff(' [y]=f(x_1)', 'y= A/(1+1.14*x_1/(1-x_1))^(2)- B
        /(1+0.877*(1-x_1)/x_1)^(2) - log(P_2_sat/P_1_sat)
        ');
57 x_1_prime = fsolve(0.1,f);
58
59 // At x_1
60 x_2_prime = 1 - x_1_prime;
61 Y1_prime = exp(A/(1+ (A*x_1_prime)/(B*x_2_prime))
        ^2);
62 Y2_prime = exp(B/(1+ (B*x_2_prime)/(A*x_1_prime))
        ^2);
63 P_prime = x_1_prime*Y1_prime*P_1_sat + x_2_prime*
        Y2_prime*P_2_sat; // [kPa] - Azeotrope pressure
64 y_1_prime = (x_1_prime*Y1_prime*P_1_sat)/P_prime;
65
66 // Since x_1 = y_1, azeotrope formation will take
    place
67 printf(" (b)\n\n");
68 printf(" The total system pressure versus x_1 shows

```

```

    a maxima and thus azeotrope is formed by the VLE
    system\n\n");
69 printf(" The azeotrope composition is x_1 = y_1 = %f
    \n\n",x_1_prime);
70 printf(" The azeotrope pressure is %f mm Hg\n",
    P_prime);

```

Scilab code Exa 15.21 Determination of azeotrope formation

```

1 clear;
2 clc;
3
4 //Example - 15.21
5 //Page number - 544
6 printf("Example - 15.21 and Page number - 544\n\n");
7
8 //Given
9 T = 50;//[C]
10 // At 50 C
11 P_1_sat = 0.67;//[atm]
12 P_2_sat = 0.18;//[atm]
13 Y1_infinity = 2.5;
14 Y2_infinity = 7.2;
15
16 //(1)
17 //  $\alpha_{12} = (y_1/x_1)/(y_2/x_2) = (Y1*P_1\_sat)/((Y2*P_2\_sat))$ 
18 // At x_1 tending to zero ,
19  $\alpha_{12\_x0} = (Y1\_infinity*P_1\_sat)/(P_2\_sat);$ 
20 // At x_1 tending to 1,
21  $\alpha_{12\_x1} = (P_1\_sat)/((Y2\_infinity*P_2\_sat));$ 
22
23 // Since alpha_12 is a continuous curve and in
    between a value of alpha_12 = 1, shall come and
    at this composition the azeotrope shall get

```

```

formed.
24 printf(" (1). Since (alpha_12_x=0) = %f and (
    alpha_12_x=1) = %f \n", alpha_12_x0, alpha_12_x1);
25 printf("      and since alpha_12 is a continuous
    curve and in between a value of alpha_12 = 1,
    shall come and at this composition azeotrope
    shall get formed.\n\n")
26
27 //(b)
28 // Since the activity coefficient values are greater
    than 1 ,therefore the deviations from Roults
    law is positive
29 // and the azeotrope is maximum pressure (or minimum
    boiling)
30 printf(" (2). Since the activity coefficient values
    are greater than 1 ,therefore the deviations from
    Roults law is positive\n");
31 printf("      and the azeotrope is maximum pressure (
    or minimum boiling)\n\n");
32
33 //(3)
34 // Let us assume the system to follow van Laar
    activity coefficient model
35 A = log(Y1_infinity);
36 B = log(Y2_infinity);
37
38 // log(Y1) = A/(1+ (A*x_1)/(B*x_2))^(2)
39 // log(Y2) = B/(1+ (B*x_2)/(A*x_1))^(2)
40
41 // At the azeotrope point , Y1*P1_sat = Y2*P2_sat
42 // log(Y1) - log(Y2) = log(P_2_sat/P_2_sat)
43 // On putting the values and then solving the above
    equation
44 def f(' [y]=f(x_1)', 'y= A/(1+ (A*x_1)/(B*(1-x_1)))^(2)
    - B/(1+ (B*(1-x_1))/(A*x_1))^(2) - log(P_2_sat/
    P_1_sat)');
45 x_1 = fsolve(0.1, f);
46

```

```

47 // At x_1
48 x_2 = 1 - x_1;
49 Y1 = exp(A/(1+ (A*x_1)/(B*x_2))^(2));
50 Y2 = exp(B/(1+ (B*x_2)/(A*x_1))^(2));
51 P = x_1*Y1*P_1_sat + x_2*Y2*P_2_sat; // [kPa] -
    Azeotrope pressure
52 y_1 = (x_1*Y1*P_1_sat)/P;
53
54 // Since x_1 = y_1, the azeotrope formation will take
    place
55
56 printf(" (3). The azeotrope composition is x_1 = y_1
    = %f\n", x_1);
57 printf(" The azeotrope pressure is %f atm\n", P)
    ;

```

Scilab code Exa 15.22 Tabulation of pressure and composition data

```

1 clear;
2 clc;
3
4 //Example - 15.22
5 //Page number - 545
6 printf("Example - 15.22 and Page number - 545\n\n");
7
8 //Given
9 T = 25; // [C]
10 // At 50 C
11 P_1_sat = 7.866; // [kPa]
12 P_2_sat = 3.140; // [kPa]
13
14 //  $G_E/(R*T) = 1.4938*x_1*x_2/(1.54*x_1 + 0.97*x_2)$ 
15
16 // The excess Gibbs free energy expression can be
    written as

```

```

17 //  $x_1 x_2 / (G E / (R T)) = 1.54 x_1 / 1.4938 + 0.97 x_2 / 1.4938 = x_1 / 0.97 + x_2 / 1.54$ 
18
19 // Comparing with the van Laar expression
20 //  $x_1 x_2 / (G E / (R T)) = x_1 / B + x_2 / A$ , we get
21 A = 1.54;
22 B = 0.97;
23
24 // The activity coefficients are thus given by
25 //  $\log(Y1) = A / (1 + (A x_1) / (B x_2))^{(2)}$ 
26 //  $\log(Y2) = B / (1 + (B x_2) / (A x_1))^{(2)}$ 
27
28 x_1 = [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,0.95];
29
30 x_2 = zeros(1,10);
31 Y1 = zeros(1,10);
32 Y2 = zeros(1,10);
33 P = zeros(1,10);
34 y_1 = zeros(1,10);
35
36 printf(" x_1      \t\t Y1      \t\t Y2      \t\t P
      (kPa)      \t\t y_1  \n\n");
37
38 for i=1:10;
39     x_2(1,i) = 1 - x_1(i);
40     Y1(1,i) = exp(A/(1+ (A*x_1(i))/(B*x_2(i))))^(2));
41     Y2(1,i) = exp(B/(1+ (B*x_2(i))/(A*x_1(i))))^(2));
42     P(1,i) = x_1(i)*Y1(i)*P_1_sat + x_2(i)*Y2(i)*
        P_2_sat;
43     y_1(1,i) = (x_1(i)*Y1(i)*P_1_sat)/P(i);
44     printf(" %f\t\t %f\t\t %f \t\t %f \t\t %f \n\n"
        ,x_1(i),Y1(i),Y2(i),P(i),y_1(i));
45
46 end
47
48 // The azeotrope is formed near x_1 = 0.95 as in
    this region a maxima in pressure is obtained.
49

```

```

50 // At the azeotrope point , Y1*P1_sat = Y2*P2_sat
51 // log(Y1) - log(Y2) = log(P_2_sat/P_2_sat)
52 // On putting the values and then solving the above
    equation
53 def f(' [y]=f(x_1)', 'y= A/(1+ (A*x_1)/(B*(1-x_1)))^(2)
    - B/(1+ (B*(1-x_1))/(A*x_1))^(2) - log(P_2_sat/
    P_1_sat)');
54 x_1_prime = fsolve(0.1,f);
55
56 // At x_1
57 x_2_prime = 1 - x_1_prime;
58 Y1_prime = exp(A/(1+ (A*x_1_prime)/(B*x_2_prime))
    ^2);
59 Y2_prime = exp(B/(1+ (B*x_2_prime)/(A*x_1_prime))
    ^2);
60 P_prime = x_1_prime*Y1_prime*P_1_sat + x_2_prime*
    Y2_prime*P_2_sat; // [kPa] - Azeotrope pressure
61 y_1_prime = (x_1_prime*Y1_prime*P_1_sat)/P_prime;
62
63 // Since x_1_prime = y_1_prime ,the azeotrope
    formation will take place
64
65 printf(" \n\nPart II \n\n");
66 printf(" The azeotrope composition is x_1 = y_1 = %f
    \n",x_1_prime);
67 printf(" The azeotrope presssure is %f kPa \n",
    P_prime);

```

Scilab code Exa 15.23 Determination of van Laar parameters

```

1 clear;
2 clc;
3
4 //Example - 15.23
5 //Page number - 547

```

```

6 printf("Example - 15.23 and Page number - 546\n\n");
7
8 //Given
9 T = 58.7; //[C]
10 P = 1; //[atm]
11 P = P*101325*10^(-3); //[kPa]
12
13 // log(P_sat) = 16.6758 - 3674.49/(t + 226.45) - For
    ethyl alcohol
14 // log(P_sat) = 13.8216 - 2697.55/(t + 224.37) - For
    hexane
15
16 // Let us take hexane as (1) and ethanol as (2)
17 // At 58.7 C
18 P_1_sat = exp(13.8216 - 2697.55/(T + 224.37)); //[kPa
    ]
19 P_2_sat = exp(16.6758 - 3674.49/(T + 226.45)); //[kPa
    ]
20
21 Y1 = P/P_1_sat;
22 Y2 = P/P_2_sat;
23
24 x_2 = 0.332; // Mol % of ethanol (given)
25 x_1 = 1 - x_2; // Mol % of hehane
26
27 // The van Laar parameters are given by
28 A = ((1 + (x_2*log(Y2))/(x_1*log(Y1)))^(2))*log(Y1);
29 B = ((1 + (x_1*log(Y1))/(x_2*log(Y2)))^(2))*log(Y2);
30
31 printf(" The value of van Laar parameters are , A =
    %f and B = %f \n\n",A,B);
32
33 // Now let us calvulate the distribution coefficient
    K
34 x_1_prime = 0.5; //[given]
35 x_2_prime = 1 - x_1_prime;
36
37 // The activity coefficients are thus given by

```

```

38 // log(Y1) = A/(1+ (A*x_1)/(B*x_2))^(2)
39 // log(Y2) = B/(1+ (B*x_2)/(A*x_1))^(2)
40
41 Y1_prime = exp(A/(1+ (A*x_1_prime)/(B*x_2_prime))
    ^ (2));
42 Y2_prime = exp(B/(1+ (B*x_2_prime)/(A*x_1_prime))
    ^ (2));
43 P_prime = x_1_prime*Y1_prime*P_1_sat + x_2_prime*
    Y2_prime*P_2_sat;
44
45 // We have, K_1 = y_1/x_1 = Y1*P_1_sat/P
46 K_1 = Y1_prime*P_1_sat/P_prime;
47
48 printf(" The distribution coefficient is given by
    K_1 = %f\n",K_1)

```

Chapter 16

Other Phase Equilibria

Scilab code Exa 16.1 Determination of solubility

```
1 clear;
2 clc;
3 funcprot(0);
4
5 // Example - 16.1
6 // Page number - 564
7 printf("Example - 16.1 and Page number - 564\n\n");
8
9 // Given
10 T = 0 + 273.15; // [K] - Temperature
11 P = 20*10^(5); // [Pa] - Pressure
12 R = 8.314; // [J/mol*K] - Universal gas constant
13
14 //component 1 : methane (1)
15 //component 2 : methanol (2)
16
17 H_constant = 1022; // [bar] - Henry's law constant
18 H_constant = H_constant*10^(5); // [Pa]
19
20 // The second virial coefficients are
21 B_11 = -53.9; // [cm^(3)/mol]
```

```

22 B_11 = B_11*10(-6); // [m(3)/mol]
23 B_12 = -166; // [cm(3)/mol]
24 B_12 = B_12*10(-6); // [m(3)/mol]
25 B_22 = -4068; // [cm(3)/mol]
26 B_22 = B_22*10(-6); // [m(3)/mol]
27
28 den_meth = 0.8102; // [g/cm(3)] - Density of methanol
    at 0 C
29 Mol_wt_meth = 32.04; // Molecular weight of methanol
30 P_2_sat = 0.0401; // [bar] - Vapour pressure of
    methanol at 0 C
31
32 //The molar volume of methanol can be calculated as
33 V_2_liq = (1/(den_meth/Mol_wt_meth))*10(-6); // [m
    ^(3)/mol]
34
35 //The phase equilibrium equation of the components
    at high pressure
36 //y1*phi_1*P = x_1*H_1
37 //y2*phi_2*P = x_2*H_2
38
39 //Since methane follows Henry's law therefore
    methanol follows the lewis-Rnadall rule
40 //f_2 is the fugacity of the compressed liquid which
    is calculated using
41 //f_2 = f_2_sat*exp[V_2_liq*(P - P_sat_2)/(R*T)]
42 //where f_2_sat can be calculated using virial
    equation
43 // log(phi_2_sat) = log(f_2_sat/P_2_sat) = (B_22*
    P_2_sat)/(R*T)
44
45 f_2_sat = P_2_sat*exp((B_22*P_2_sat*10(5))/(R*T));
    // [bar]
46
47 //Putting the value of 'f_2_sat' in the expression
    of f_2 , we get
48 f_2 = f_2_sat*exp(V_2_liq*(P - P_2_sat*10(5))/(R*T)
    ); // [bar]

```

```

49
50 //Now let us calculate the fugacity coefficients of
    the species in the vapour mixture
51 del_12 = 2*B_12 - B_11 - B_22;//[m^(3)/mol]
52
53 //log(phi_1) = (P/(R*T))*(B_11 + y2^(2)*del_12)
54 //log(phi_2) = (P/(R*T))*(B_22 + y1^(2)*del_12)
55
56
57 //The calculation procedure is to assume a value of
    y1, calculate 'phi_1' and 'phi_2' and calculate '
    x_1' and 'x_2' from the phase equilibrium
    equations and see whether x_1 + x_2 = 1,if not
    then another value of y1 is assumed.
58
59 y2 = 0.1;
60 error=10;
61
62 while(error>0.001)
63     y1=1-y2;
64     phi_1 = exp((P/(R*T))*((B_11 + y2^(2)*del_12)));
65     phi_2 = exp((P/(R*T))*((B_22 + y1^(2)*del_12)));
66     x_1 = (y1*phi_1*P)/H_constant;
67     x_2 = (y2*phi_2*P)/(f_2*10^(5));
68     x = x_1 + x_2;
69     error=abs(1-x);
70     y2=y2 - 0.000001;
71 end
72
73 printf(" The solubility of methane in methanol is
    given by x1 = %f\n",x_1);

```

Scilab code Exa 16.2 Determination of solubility

```
1 clear;
```

```

2  clc;
3
4  // Example - 16.2
5  // Page number - 566
6  printf("Example - 16.2 and Page number - 566\n\n");
7
8  // Given
9  x_C2H6_1 = 0.33*10^(-4); // Solubility of ethane in
    water at 25 C and 1 bar
10
11 //component 1 : ethane (1)
12 //component 2 : water (2)
13
14 // Z = 1 - 7.63*10^(3)*P - 7.22*10^(-5)*P^(2)
15
16 //The phase equilibrium equation of ethane is
17 //f_1_V = x_1*H_1
18 //since vapour is pure gas, f_1_V = x_1*H_1 or,
    phi_1*P = x_1*H_1, where 'phi_1' is fugacity
    coefficient of pure ethane
19 // log(phi) = integral('Z-1)/P) from limit '0' to 'P
    ,
20
21 P1 = 0;
22 P2 = 1;
23 P3 = 35;
24 intgral = integrate('(1-7.63*10^(-3)*P-7.22*10^(-5)*
    P^(2)-1)/P', 'P', P1, P2);
25 phi_1_1 = exp(intgral); // - Fugacity coefficient of
    ethane at 1 bar
26 f_1_1 = phi_1_1*P2; // [bar] - Fugacity of ethane at 1
    bar
27
28 // Similarly
29 intgral_1 = integrate('(1-7.63*10^(-3)*P
    -7.22*10^(-5)*P^(2)-1)/P', 'P', P1, P3);
30 phi_1_35 = exp(intgral_1); // Fugacity coefficient of
    ethane at 35 bar

```

```

31 f_1_35 = phi_1_35*P3;//[bar] - Fugacity of ethane at
    35 bar
32
33 // At ethane pressure of 1 bar , x_C2H6_1*H_1 =
    phi_1_1
34 H_1 = phi_1_1/x_C2H6_1;//[bar] - Henry's constant
35
36 // At ethane pressure of 35 bar , x_C2H6_35*H_1 =
    phi_1_35
37 x_C2H6_35 = f_1_35/H_1;// Solubility of ethane at 35
    bar pressure
38
39 printf("The solubility of ethane at 35 bar is given
    by x_C2H6 = %e",x_C2H6_35);

```

Scilab code Exa 16.3 Proving a mathematical relation

```

1 clear;
2 clc;
3
4 //Example - 16.3
5 //Page number - 567
6 printf("Example - 16.3 and Page number - 567\n\n");
7
8 //This problem involves proving a relation in which
    no mathematics and no calculations are involved.
9 //For prove refer to this example 16.3 on page
    number 567 of the book.
10 printf(" This problem involves proving a relation in
    which no mathematics and no calculations are
    involved.\n\n");
11 printf(" For prove refer to this example 16.3 on
    page number 567 of the book.")

```

Scilab code Exa 16.4 Determination of composition

```
1 clear;
2 clc;
3
4 //Example - 16.4
5 //Page number - 571
6 printf("Example - 16.4 and Page number - 571\n\n");
7
8 //Given
9 T = 200; // [K]
10 R = 8.314; // [J/mol*K] - universal gas constant
11 // G_E = A*x_1*x_2
12 A = 4000; // [J/mol]
13 x_1 = 0.6; // Mle fraction of feed composition
14
15 // Since A is given to be independent of temperature
16 UCST = A/(2*R); // [K] - Upper critical solution
    temperature
17 printf(" The UCST of the system is %f K\n\n", UCST);
18
19 // Since the given temperature is less than UCST
    therefore two phase can get formed at the given
    temperature.
20
21 // x1_alpha = 1 - x1_beta
22 // We know that, x1_alpha*Y_1_alpha = x2_alpha*
    Y_2_alpha
23 // x1_alpha*exp[(A/(R*T))*(x2_alpha)^(2)] = (1 -
    x1_alpha)*exp[(A/(R*T))*(x1_alpha)^(2)]
24 // where use has been made of the fact that
    x1_alpha = 1 - x1_beta and x2_beta = 1 - x1_beta
    = x1_alpha . Taking logarithm of both side we get
25 // log(x1_alpha) + (A/(R*T))*(1 - x1_alpha)^(2) =
```

```

26     log(1 - x1_alpha) + (A/(R*T))*x1_alpha^(2)
// log(x1_alpha/(1-x1_alpha)) = (A/(R*T))*(2*
    x1_alpha - 1)
27
28 deff(' [y]=f(x1_alpha)', 'y= log(x1_alpha/(1-x1_alpha)
    ) - (A/(R*T))*(2*x1_alpha - 1)');
29 x1_alpha = fsolve(0.1,f);
30 x1_beta = fsolve(0.9,f);
31 // Because of symmetry 1 - x1_beta = x1_alpha
32
33 // It can be seen that the equation, log(x1/(1-x1))
    = (A/(R*T))*(2*x1 - 1) has two roots.
34 // The two roots acn be determined by taking
    different values
35 // Starting with x1 = 0.1, we get x1 = 0.169 as the
    solution and starting with x1 = 0.9,we get x1 =
    0.831 as the solution.
36 // Thus x1 = 0.169 is the composition of phase alpha
    and x1 = 0.831 is of phase beta
37 printf(" The composition of two liquid phases in
    equilibrium is given by, x1_alpha = %f and
    x1_beta = %f\n\n",x1_alpha,x1_beta);
38
39 // From the equilibrium data it is seen that if the
    feed has composition x1 less than 0.169 or more
    than 0.831 the liquid mixture is of single phase
40 // whereas if the overall (feed) composition is
    between 0.169 and 0.831 two phases shall be
    formed.
41 // The amounts of phases can also be calculated. The
    feed composition is given to be z1 = 0.6
42 z1 = 0.6;
43 // z1 = x1_alpha*alpha + x1_beta*beta
44 // beta = 1 - alpha
45 alpha = (z1-x1_beta)/(x1_alpha-x1_beta);//[mol]
46 Beta = 1 - alpha;//[mol]
47 printf(" The relative amount of phases is given by,
    alpha = %f mol and beta = %f mol\n\n\n",alpha,

```

```

    Beta);
48
49 // the relative amounts of the phases changes with
    the feed composition
50
51 //log(x1/(1-x1)) = (A/(R*T))*(2*x1 - 1)
52 // If the above equation has two real roots of x1 (
    one for phase alpha and the other for phase beta)
    then two liquid phases get formed
53 // and if it has no real roots then a homogeneous
    liquid mixtures is obtained.
54
55 printf(" log(x1/(1-x1)) = (A/(R*T))*(2*x1 - 1)\n");
56 printf(" If the above equation has two real roots of
    x1 (one for phase alpha and the other for phase
    beta) then two liquid phases get formed\n");
57 printf(" and if it has no real roots then a
    homogeneous liquid mixture is obtained\n");

```

Scilab code Exa 16.5 Determination of equilibrium composition

```

1 clear;
2 clc;
3
4 //Example - 16.5
5 //Page number - 573
6 printf("Example - 16.5 and Page number - 573\n\n");
7
8 //Given
9 T = 300; // [K]
10 R = 8.314; // [J/mol*K] - universal gas constant
11 A = 7000; // [J/mol]
12
13 // log(x_1/(1-x_1)) = (A/(R*T))*(2*x_1-1)
14

```



```

15 deff(' [y]=f(x_1)', 'y=log(x_1/(1-x_1)) -((A/(R*T)) *(2*
    x_1-1)) ');
16
17 x1_alpha=fsolve(0.1,f);
18
19 x1_beta=1-x1_alpha;
20
21 printf("The equilibrium compositin of the two liquid
    phase system is given by\n x1_alpha \t = %f \n
    x1_beta \t = %f",x1_alpha,x1_beta);

```

Scilab code Exa 16.6 Proving a mathematical relation

```

1 clear;
2 clc;
3
4 //Example - 16.6
5 //Page number - 577
6 printf("Example - 16.6 and Page number - 577\n\n");
7
8 //This problem involves proving a relation in which
    no mathematics and no calculations are involved.
9 //For prove refer to this example 16.6 on page
    number 577 of the book.
10 printf(" This problem involves proving a relation in
    which no mathematics and no calculations are
    involved.\n\n");
11 printf(" For prove refer to this example 16.6 on
    page number 577 of the book.")

```

Scilab code Exa 16.7 Determination of freezing point depression

```

1 clear;

```

```

2  clc;
3
4  //Example - 16.7
5  //Page number - 579
6  printf("Example - 16.7 and Page number - 579\n\n");
7
8  //Given
9  R = 8.314;//[J/mol*K] - Universal gas constant
10 M_wt_meth = 32;// Molecular weight of methanol
11 M_wt_water = 18;// Molecular weight of water
12 m_meth = 0.01;//[g] - Mass of methanol added per cm
    ^ (3) of solution
13
14 //Since the concentration of methanol is very small
    therefore we can assume that the density of
    solution = pure water
15 den_sol = 1;//[g/cm^ (3)]
16
17 //The mole fraction of solute is given by
18 //x_2 = (moles of solute in cm^ (3) of solution)/(
    moles of solute + moles of water) in 1 cm^ (3) of
    solution
19 x_2 = (m_meth/M_wt_meth)/((m_meth/M_wt_meth)+((1-
    m_meth)/M_wt_water));
20
21 //We know that heat of fusion of water is
22 H_fus = -80;//[cal/g] - Enthalpy change of fusion at
    0 C
23 H_fus = H_fus*4.186*M_wt_water;//[J/mol]
24
25 //Therefore freezing point depression is given by
26 // T - T_m = (R*(T^ (2))*x_2)/H_fus
27 T_f = 273.15;//[K] - Freezing point of water
28 delta_T_f = (R*(T_f^ (2))*x_2)/H_fus;//[K]
29
30 printf("The depression in freezing point is given by
    \n delta_T = %f K",delta_T_f);

```

Scilab code Exa 16.8 Determination of freezing point

```
1 clear;
2 clc;
3
4 //Example - 16.8
5 //Page number - 580
6 printf("Example - 16.8 and Page number - 580\n\n");
7
8 //Given
9 R = 8.314; //[J/mol*K] - universal gas constant
10 T_f = 273.15; //[K] - Freezing point of water
11 m_water = 100; //[g] - Mass of water
12 m_NaCl = 3.5; //[g] - Mass of NaCl
13 M_wt_water = 18.015; // Molecular weight of water
14 M_wt_NaCl = 58.5; // Molecular weight of NaCl
15 mol_water = m_water/M_wt_water; //[mol] - Moles of
    water
16 mol_NaCl = m_NaCl/M_wt_NaCl; //[mol] - Moles of NaCl
17
18 H_fus = -80; //[cal/g] - Enthalpy change of fusion at
    0 C
19 H_fus = H_fus*4.186*M_wt_water; //[J/mol]
20
21 //Mole fraction of the solute (NaCl) is given by
22 x_2 = mol_NaCl/(mol_NaCl+mol_water);
23
24 //But NaCl is completely ionized and thus each ion
    acts independently to lower the water mole
    fraction.
25 x_2_act = 2*x_2; // Actual mole fraction
26
27 //Now depression in freezing point is given by
28 //  $T - T_m = (R*(T^2)*x_{2\_act})/H_{fus}$ 
```

```

29 delta_T_f = (R*(T_f^(2))*x_2_act)/H_fus; //[C]
30
31 //Thus freezing point of seawater = depression in
    freezing point
32
33 printf("The freezing point of seawater is %f C",
    delta_T_f);

```

Scilab code Exa 16.9 Proving a mathematical relation

```

1 clear;
2 clc;
3
4 //Example - 16.9
5 //Page number - 580
6 printf("Example - 16.9 and Page number - 580\n\n");
7
8 //This problem involves proving a relation in which
    no mathematics and no calculations are involved.
9 //For prove refer to this example 16.9 on page
    number 580 of the book.
10 printf(" This problem involves proving a relation in
    which no mathematics and no calculations are
    involved.\n\n");
11 printf(" For prove refer to this example 16.9 on
    page number 580 of the book.")

```

Scilab code Exa 16.10 Determination of boiling point elevation

```

1 clear;
2 clc;
3
4 //Example - 16.10

```

```

5 //Page number - 583
6 printf("Example - 16.10 and Page number - 583\n\n");
7
8 //Given
9 R = 8.314; //[J/mol*K] - universal gas constant
10 T_b = 373.15; //[K] - Boiling point of water
11 m_water = 100; //[g] - Mass of water
12 m_C12H22 = 5; //[g] - Mass of glucise (C12H22)
13 M_wt_water = 18.015; // Molecular weight of water
14 M_wt_C12H22 = 342.30; // Molecular weight of C12H22
15 mol_water = m_water/M_wt_water; //[mol] - Moles of
    water
16 mol_C12H22 = m_C12H22/M_wt_C12H22; //[mol] - Moles of
    C12H22
17
18 H_vap = 540; //[cal/g] - Enthalpy change of
    vaporisation
19 H_vap = H_vap*4.186*M_wt_water; //[J/mol]
20
21 //Mole fraction of the solute (C12H22) is given by
22 x_2 = mol_C12H22/(mol_C12H22+mol_water);
23
24 //The boiling point elevation is given by
25 //  $T - T_b = (R*T_b^2*x_2)/H_{vap}$ 
26
27 delta_T_b = (R*T_b^2*x_2)/(H_vap);
28
29 printf("The elevation in boiling point is given by \
    n delta_T = %f C", delta_T_b);

```

Scilab code Exa 16.11 Determination of osmotic pressure

```

1 clear;
2 clc;
3

```

```

4 //Example - 16.11
5 //Page number - 584
6 printf("Example - 16.11 and Page number - 584\n\n");
7
8 //Given
9 R = 8.314; //[J/mol*K] - Universal gas constant
10 T = 25 + 273.15; //[K] - Surrounding temperature
11 den_water = 1000; //[kg/m^(3)] - Density of water
12 m_water = 100; //[g] - Mass of water
13 m_C12H22 = 5; //[g] - Mass of glucise (C12H22)
14 M_wt_water = 18.015; // Molecular weight of water
15 M_wt_C12H22 = 342.30; // Molecular weight of C12H22
16 mol_water = m_water/M_wt_water; //[mol] - Moles of
   water
17 mol_C12H22 = m_C12H22/M_wt_C12H22; //[mol] - Moles of
   C12H22
18
19 //Mole fraction of the water is given by
20 x_1 = mol_water/(mol_C12H22+mol_water);
21
22 //Molar volume of water can be calculated as
23 V_l_water = (1/den_water)*M_wt_water*10^(-3); //[m
   ^ (3)/mol]
24
25 //The osmotic pressure is given by
26 pi = -(R*T*log(x_1))/V_l_water; //[N/m^(2)]
27 pi = pi*10^(-5); //[bar]
28
29 printf("The osmotic pressure of the mixture is %f
   bar",pi);

```

Scilab code Exa 16.12 Determination of pressure

```

1 clear;
2 clc;

```

```

3
4 //Example - 16.12
5 //Page number - 585
6 printf("Example - 16.12 and Page number - 585\n\n");
7
8 //Given
9 R = 8.314; //[J/mol*K] - universal gas constant
10 T = 25 + 273.15; //[K] - Surrounding temperature
11 den_water = 1000; //[kg/m^(3)] - Density of water
12 m_water = 100; //[g] - Mass of water
13 m_NaCl = 3.5; //[g] - Mass of NaCl
14 M_wt_water = 18.015; // Molecular weight of water
15 M_wt_NaCl = 58.5; // Molecular weight of NaCl
16 mol_water = m_water/M_wt_water; //[mol] - Moles of
    water
17 mol_NaCl = m_NaCl/M_wt_NaCl; //[mol] - Moles of NaCl
18
19 H_fus = -80; //[cal/g] - Enthalpy change of fusion at
    0 C
20 H_fus = H_fus*4.186*M_wt_water; //[J/mol]
21
22 //Mole fraction of the solute (NaCl) is given by
23 x_2 = mol_NaCl/(mol_NaCl+mol_water);
24
25 //But NaCl is completely ionized and thus each ion
    acts independently to lower the water mole
    fraction.
26 x_2_act = 2*x_2; // Actual mole fraction
27
28 x_1 = 1 - x_2_act;
29
30 //Molar volume of water can be calculated as
31 V_l_water = (1/den_water)*M_wt_water*10^(-3); //[m
    ^ (3)/mol]
32
33 //The osmotic pressure is given by
34 pi = -(R*T*log(x_1))/V_l_water; //[N/m^(2)]
35 pi = pi*10^(-5); //[bar]

```

```

36 //The minimum pressure to desalinate sea water is
    nothing but the osmotic pressure
37
38 printf("The minimum pressure to desalinate sea water
    is %f bar",pi);

```

Scilab code Exa 16.13 Determination of amount of precipitate

```

1  clear;
2  clc;
3
4  //Example - 16.13
5  //Page number - 586
6  printf("Example - 16.13 and Page number - 586\n\n");
7
8  //Given
9  R = 8.314; //[J/mol*K] - universal gas constant
10 T = 173.15; //[K] - Surrounding temperature
11 P = 60; //[bar]
12 P = P*10^(5); //[Pa]
13
14 //componenet 1 : CO2 (1)
15 //componenet 2 : H2 (2)
16 P_1_sat = 0.1392; //[bar] - Vapour pressre of pure
    solid CO2
17 P_1_sat = P_1_sat*10^(5); //[bar]
18 V_s_1 = 27.6; //[cm^(3)/mol] - Molar volume of solid
    CO2
19 V_s_1 = V_s_1*10^(-6); //[m^(3)/mol]
20 n_1 = 0.01; //[mol] - Initial number of moles of CO2
21 n_2 = 0.99; //[mol] - Initial number of moles of H2
22
23 //Let us determine the fugacity of solid CO2 (1) at
    60 C and 173.15 K
24 // f_1 = f_1_sat*exp(V_s_1*(P-P_1_sat)/(R*T))

```



```

25
26 //Since vapour pressure of pure solid CO2 is very
    small, therefore
27 f_1_sat = P_1_sat;
28 f_1 = f_1_sat*exp(V_s_1*(P-P_1_sat)/(R*T));
29
30 //Since gas phase is ideal therefore
31 // y1*P = f_1
32 y1 = f_1/P;
33
34 //Number of moles of H2 in vapour phase at
    equilibrium remains the same as initial number of
    moles.
35 //Number of moles of CO2 in vapour phase at
    equilibrium can be calculated as
36 //y1 = (n_1_eq/(n_1_eq + n_2)). Therefore
37 n_1_eq = n_2*y1/(1-y1);
38
39 //Therefore moles of CO2 precipitated is
40 n_ppt = n_1 - n_1_eq; //[mol]
41
42 printf("The moles of CO2 precipitated is %f mol",
    n_ppt);

```

Scilab code Exa 16.14 Calculation of pressure

```

1 clear;
2 clc;
3
4 //Example - 16.14
5 //Page number - 586
6 printf("Example - 16.14 and Page number - 586\n\n");
7
8 //Given
9 R = 8.314; //[J/mol*K] - universal gas constant

```

```

10 T = 350; //[K] - Surrounding temperature
11
12 //componenet 1 : organic solid (1)
13 //componenet 2 : CO2 (2)
14
15 P_1_sat = 133.3; //[N/m^(2)] - Vapour pressre of
    organic solid
16 V_s_1 = 200; //[cm^(3)/mol] - Molar volume of organic
    solid
17 V_s_1 = V_s_1*10^(-6); //[m^(3)/mol]
18
19 ///At 350 K, the values of the coefficients
20 B_11 = -500; //[cm^(3)/mol]
21 B_22 = -85; //[cm^(3)/mol]
22 B_12 = -430; //[cm^(3)/mol]
23
24 //From phase equilibrium equation of component 1, we
    get
25 // y1*P*phi_1 = f_1
26 // f_1 = f_1_sat*exp(V_s_1*(P-P_1_sat)/(R*T))
27
28 //Since vapour pressure of organic solid is very
    small, therefore
29 f_1_sat = P_1_sat;
30
31 // Now let us determine the fugacity coefficient of
    organic solid in the vapour mixture.
32 // log(phi_1) = (P/(R*T))*(B_11 + y2^(2)*del_12)
33 del_12 = (2*B_12 - B_11 - B_22)*10^(-6); //[m^(3)/mol
    ]
34
35 //It is given that the partial pressure of component
    1 in the vapour mixture is 1333 N/m^(2)
36 // y1*P = 1333 N/m^(2) or, y1 = 1333/P
37 // y2 = 1- 1333/P
38 // log(phi_1) = (P/(R*T))*(B_11 + (1- 1333/P)^(2)*
    del_12)
39

```

```

40 //The phase equilibrium equation becomes
41 //  $y_1 P \phi_1 = f_{1\_sat} \exp(V_{s\_1} (P - P_{1\_sat}) / (R T))$ 
42 // Taking log on both side we have
43 //  $\log(y_1 P) + \log(\phi_1) = \log(f_{1\_sat}) + (V_{s\_1} (P - P_{1\_sat}) / (R T))$ 
44 //  $(V_{s\_1} (P - P_{1\_sat}) / (R T)) - \log(\phi_1) = \log(1333/133.3) = \log(10)$ 
45
46 // substituting for  $\log(\phi_1)$  from previous into the
   above equation we get
47 //  $(V_{s\_1} (P - P_{1\_sat}) / (R T)) - (P / (R T)) * (B_{11} + (1 - 1333/P)^2 * \Delta_{12}) - \log(10) = 0$ 
48 // On simplification we get ,
49 //  $975 * P^2 - 6.7 * 10^9 * P + 4.89 * 10^8 = 0$ 
50 // Solving the above quadratic equation using
   shreedharcharya rule
51
52 P3 = (6.7*10^(9) + ((-6.7*10^(9))^2
   -4*975*4.98*10^(8))^(1/2))/(2*975);//[Pa]
53 P4 = (6.7*10^(9) - ((-6.7*10^(9))^2
   -4*975*4.98*10^(8))^(1/2))/(2*975);//[Pa]
54 // The second value is not possible , therefore
   pressure of the system is P3
55 P3 = P3*10^(-5);//[bar]
56
57 printf(" The total pressure of the system is %f bar"
   ,P3);

```

Chapter 17

Chemical Reactions Equilibria

Scilab code Exa 17.1 Proving a mathematical relation

```
1 clear;
2 clc;
3
4 //Example - 17.1
5 //Page number - 595
6 printf("Example - 17.1 and Page number - 595\n\n");
7
8 //This problem involves proving a relation in which
9 //no mathematical components are involved.
10 //For prove refer to this example 17.1 on page
11 //number 595 of the book.
12 printf("This problem involves proving a relation in
13 which no mathematical components are involved.\n\n");
14 printf("For prove refer to this example 17.1 on page
15 number 595 of the book.")
```

Scilab code Exa 17.2 Determination of number of moles

```

1 clear;
2 clc;
3
4 //Example - 17.2
5 //Page number - 598
6 printf("Example - 17.2 and Page number - 598\n\n");
7
8 // Given
9 P = 1; //[atm] - Reactor pressure
10 T = 749; //[K] - Reactor temperature
11 K = 74; // Equilibrium constant
12
13 // SO2 + (1/2)*O2 - SO3
14
15 Kp = P^(1);
16 Ky = K/Kp;
17
18 //(1)
19 // Initial number of moles of the components are
20 n_SO2_1_in = 12;
21 n_O2_1_in = 9;
22 n_SO3_1_in = 0;
23
24 // Let the reaction coordinate at equilibrium for
    the reaction be X
25 // At equilibrium, the moles of the components be
26 // n_SO2_1_eq = 12 - X
27 // n_O2_1_eq = 9 - 0.5*X
28 // n_SO3_1_eq = X
29 // Total moles = 21 - 0.5*X
30
31 // The mole fractions of the components at
    equilibrium are
32 // y_SO3 = X/(21-0.5*X)
33 // y_SO2 = (12-X)/(21-0.5*X)
34 // y_O2 = (9-0.5*X)/(21-0.5*X)
35
36 // Ky = y_SO3/(y_SO2*y_O2^(2))

```

```

37 // Ky = (X*(21-0.5*X)^(1/2))/((12-X)*(9-0.5*X)^(1/2)
    )
38 deff(' [y]=f(X)', 'y= Ky-(X*(21-0.5*X)^(1/2))/((12-X)
    *(9-0.5*X)^(1/2))');
39 X_1 = fsolve(11,f);
40
41 y_SO3_1 = X_1/(21-0.5*X_1);
42 y_SO2_1 = (12-X_1)/(21-0.5*X_1);
43 y_O2_1 = (9-0.5*X_1)/(21-0.5*X_1);
44
45 printf(" (1).The moles of SO3 formed = %f mol\n",X_1
    );
46 printf("          The mole fractions at equilibrium are
    y_SO3 = %f, y_SO2 = %f and y_O2 = %f\n\n",y_SO3_1
    ,y_SO2_1,y_O2_1);
47
48 //(2)
49 // Initial number of moles of the components are
50 n_SO2_2_in = 24;
51 n_O2_2_in = 18;
52 n_SO3_2_in = 0;
53
54 // At equilibrium, the moles of the components be
55 // n_SO2_1_eq = 24 - X
56 // n_O2_1_eq = 18 - 0.5*X
57 // n_SO3_1_eq = X
58 // Total moles = 42 - 0.5*X
59
60 // The mole fractions of the components at
    equilibrium are
61 // y_SO3 = X/(42-0.5*X)
62 // y_SO2 = (24-X)/(42-0.5*X)
63 // y_O2 = (18-0.5*X)/(42-0.5*X)
64
65 // Ky = y_SO3/(y_SO2*y_O2^(2))
66 // Ky = (X*(42-0.5*X)^(1/2))/((24-X)*(18-0.5*X)
    ^ (1/2))
67 deff(' [y]=f1(X)', 'y= Ky-(X*(42-0.5*X)^(1/2))/((24-X)

```

```

        *(18-0.5*X)^(1/2))');
68 X_2 = fsolve(22,f1);
69
70 y_SO3_2 = X_2/(42-0.5*X_2);
71 y_SO2_2 = (24-X_2)/(42-0.5*X_2);
72 y_O2_2 = (18-0.5*X_2)/(42-0.5*X_2);
73 printf(" (2).The moles of SO3 formed = %f mol\n",X_2
    );
74 printf("      The mole fractions at equilibrium are
    y_SO3 = %f, y_SO2 = %f and y_O2 = %f\n\n",y_SO3_2
    ,y_SO2_2,y_O2_2);
75
76 //(3)
77 // Initial number of moles of the components are
78 n_SO2_3_in = 12;
79 n_O2_3_in = 9;
80 n_SO3_3_in = 0;
81 n_N2 = 79;
82
83 // At equilibrium, the moles of the components be
84 // n_SO2_1_eq = 12 - X
85 // n_O2_1_eq = 9 - 0.5*X
86 // n_SO3_1_eq = X
87 // Total moles = 100 - 0.5*X
88
89 // The mole fractions of the components at
    equilibrium are
90 // y_SO3 = X/(100-0.5*X)
91 // y_SO2 = (12-X)/(100-0.5*X)
92 // y_O2 = (9-0.5*X)/(100-0.5*X)
93
94 // Ky = y_SO3/(y_SO2*y_O2^(2))
95 // Ky = (X*(100-0.5*X)^(1/2))/((12-X)*(9-0.5*X)
    ^ (1/2))
96 deff(' [y]=f2(X)', 'y= Ky-(X*(100-0.5*X)^(1/2))/((12-X
    )*(9-0.5*X)^(1/2))');
97 X_3 = fsolve(10,f2);
98

```

```

99 y_SO3_3 = X_3/(100-0.5*X_3);
100 y_SO2_3 = (12-X_3)/(100-0.5*X_3);
101 y_O2_3 = (9-0.5*X_3)/(100-0.5*X_3);
102
103 printf(" (3).The moles of SO3 formed = %f mol\n",X_3
);
104 printf("      The mole fractions at equilibrium are
      y_SO3 = %f, y_SO2 = %f and y_O2 = %f\n\n",y_SO3_3
      ,y_SO2_3 ,y_O2_3);

```

Scilab code Exa 17.3 Determination of equilibrium composition

```

1 clear;
2 clc;
3
4 //Example - 17.3
5 //Page number - 599
6 printf("Example - 17.3 and Page number - 599\n\n");
7
8 // Given
9 T = 600;//[K] - Reactor temperature
10 P = 300;//[atm] - Reactor pressure
11 K = 0.91*10^(-4);// Equilibrium constant
12
13 // The fugacity coefficients of the components are
14 phi_CO = 1.0;
15 phi_H2 = 1.2;
16 phi_CH3OH = 0.47;
17
18 // CO + 2*H2 - CH3OH
19
20 // For gas phase reactions the standard state is
      pure ideal gas and thus fi_0 = 1 atm and thus
21 // ai_cap = fi_cap/fi_0 = yi*P*phi_i_cap/1
22 // Thus K = Ky*Kp*K_phi

```



```

23 Kp = P^(1-3);
24 K_phi = phi_CH3OH/(phi_CO*phi_H2^(2));
25 Ky = K/(Kp*K_phi);
26
27 // Let the reaction coordinate at equilibrium for
    the reaction be X
28 // At equilibrium ,the moles of the components be
29 // n_CO = 1 - X
30 // n_H2 = 3 - 2*X
31 // n_CH3OH = X
32 // Total moles = 4 - 2*X
33
34 // The mole fractions of the components at
    equilibrium are
35 // y_CO = (1-X)/(4-2*X)
36 // y_H2 = (3-2*X)/(4-2*X)
37 // y_CH3OH = (X)/(4-2*X)
38
39 // Ky = y_CH3OH/(y_CO*y_H2^(2)) = (X/(4-2*X))/(((1-X)
    )/(4-2*X))*((3-2*X)/(4-2*X))^(2))
40 deff(' [y]=f(X) ', 'y=Ky-(X/(4-2*X))/(((1-X)/(4-2*X))
    *((3-2*X)/(4-2*X))^(2)) ');
41 X = fsolve(0.1, f);
42
43 // Therefore at equilibrium
44 y_CO = (1-X)/(4-2*X);
45 y_H2 = (3-2*X)/(4-2*X);
46 y_CH3OH = (X)/(4-2*X);
47
48 printf(" The mole fractions at equilibrium are y_CO
    = %f, y_H2 = %f and y_CH3OH = %f", y_CO, y_H2,
    y_CH3OH);

```

Scilab code Exa 17.4 Determination of the value of equilibrium constant

```

1 clear;
2 clc;
3
4 //Example - 17.4
5 //Page number - 600
6 printf("Example - 17.4 and Page number - 600\n\n");
7
8 // Given
9 T = 600; // [K] - Reactor temperature
10 P = 4; // [atm] - Reactor pressure
11 K = 1.175; // Equilibrium constant
12
13 // (1/2)*N2 + (3/2)*H2 - NH3
14
15 // Initial number of moles of the components are
16 n_N2 = 1;
17 n_H2 = 3;
18 n_NH3 = 0;
19
20 // Let the reaction coordinate at equilibrium for
    the reaction be X.
21 // At equilibrium ,the moles of the components be
22 // n_N2 = 1 - 0.5*X
23 // n_H2 = 3 - 1.5*X
24 // n_NH3 = X
25 // Total moles = 4 - X
26
27 // We have , K = Ky*Kp
28 Kp = P^(1-2); // [atm^(-1)]
29 Ky = K/(Kp);
30
31 // Ky = y_NH3/(y_N2^(1/2)*y_H2^(3/2)) = (X/(4-X))
    /(((1-0.5*X)/(4-X))^(1/2)*((3-1.5*X)/(4-X))^(3/2)
    )
32 // Solving the above equation we get
33 def f(' [y]=f(X) ', 'y=Ky - (X/(4-X))/(((1-0.5*X)/(4-X))
    ^ (1/2)*((3-1.5*X)/(4-X))^(3/2)) ');
34 X = fsolve(0.1,f);

```

```

35
36 y_NH3 = X/(4-X); // Mole fraction of NH3 at
    equilibrium
37
38 printf(" The value of Kp = %f and Ky = %f \n",Kp,Ky)
    ;
39 printf(" The mole fractions of NH3 at equilibrium is
    %f\n\n",y_NH3);
40
41 // If reaction carried out at constant temperature
    and volume
42
43 // We know that for ideal gas, P*V = n*R*T and thus
    P is directly proportional to n at constant V and
    T.
44 // Let P = k*n
45 // Initially P = 4 atm and n = 4 moles, thus K = 1
    and we get p = n, where P is in atm.
46 // Thus at equilibrium P = 4 - X
47
48 // Ky = K/Kp = 1.175*P = 1.175*(4 - X)
49 // (X/(4-X))/(((1-0.5*X)/(4-X))^(1/2)*((3-1.5*X)/(4-
    X))^(3/2)) = 1.175*(4 - X)
50 // Solving the above equation we get
51 deff(' [y]=f1(X)', 'y=(X/(4-X))/(((1-0.5*X)/(4-X))
    ^ (1/2)*((3-1.5*X)/(4-X))^(3/2))-1.175*(4-X)');
52 X_prime = fsolve(1,f1);
53
54 // Therefore at equilibrium
55 P_prime = 4 - X_prime;
56 y_NH3_prime = X_prime/(4-X_prime);
57
58 printf(" If reaction is carried out at constant
    temperature and volume, then\n");
59 printf(" The equilibrium pressure is %f atm\n",
    P_prime);
60 printf(" The equilibrium mole fractions of NH3 in
    the reactor is %f\n\n",y_NH3_prime);

```

Scilab code Exa 17.5 Determination of mole fraction

```
1 clear;
2 clc;
3
4 //Example - 17.5
5 //Page number - 601
6 printf("Example - 17.5 and Page number - 601\n\n");
7
8 // Given
9 T = 400;//[K] - Reactor temperature
10 P = 1;//[atm] - Reactor pressure
11 K = 1.52;// Equilibrium constant
12 y_H2 = 0.4;// Equilibrium mole fraction of hydrogen
13
14 // CO(g) + 2*H_2(g) - CH3OH(g)
15
16 // K = y_CH3OH/(y_CO*y_H2^(2)*P^(2))
17 // Let total number of moles at equilibrium be 1
18 // y_CH3OH = 0.6 - y_CO;
19 // (0.6 - y_CO)/y_CO = K*P^(2)*y_H2^(2)
20
21 y_CO = 0.6/(1 + K*P^(2)*y_H2^(2));
22 y_CH3OH = 0.6 - y_CO;
23
24 printf(" The mole fractions are , y_CO = %f and
        y_CH3OH = %f \n", y_CO, y_CH3OH);
```

Scilab code Exa 17.6 Determination of number of moles

```
1 clear;
```

```

2  clc;
3
4  //Example - 17.6
5  //Page number - 602
6  printf("Example - 17.6 and Page number - 602\n\n");
7
8  // Given
9  T = 749; //[K] - Reactor temperature
10 P = 1; //[atm] - Reactor pressure
11 K = 74;
12
13 Kp = P^(-1/2); //[atm^(-1/2)]
14 Ky = K/Kp;
15
16 // SO2 + (1/2)*O2 - SO3
17
18 // Initial number of moles of the components are
19 n_SO2_1_in = 10;
20 n_O2_1_in = 8;
21 n_SO3_1_in = 0;
22
23 // Let the reaction coordinate at equilibrium for
    the reaction be X
24 // At equilibrium, the moles of the components be
25 // n_SO2_1_eq = 10 - X
26 // n_O2_1_eq = 8 - 0.5*X
27 // n_SO3_1_eq = X
28 // Total moles = 18 - 0.5*X
29
30 // The mole fractions of the components at
    equilibrium are
31 // y_SO3 = X/(18-0.5*X)
32 // y_SO2 = (10-X)/(18-0.5*X)
33 // y_O2 = (8-0.5*X)/(18-0.5*X)
34
35 // Ky = y_SO3/(y_SO2*y_O2^(2))
36 // Ky = (X*(18-0.5*X)^(1/2))/((10-X)*(8-0.5*X)^(1/2))
    )

```

```

37 deff(' [y]=f(X) ', 'y= Ky-(X*(18-0.5*X)^(1/2))/((10-X)
      *(8-0.5*X)^(1/2)) ');
38 X_1 = fsolve(11,f);
39
40 n_SO3 = X_1;
41 n_SO2 = 10 - X_1;
42 n_O2 = 8 - 0.5*X_1;
43
44 printf(" (1).The moles of the components at
      equilibrium are, n_SO3 = %f mol, n_SO2 = %f mol
      and n_O2 = %f mol\n\n",n_SO3,n_SO2,n_O2);
45
46 // Now for the reaction
47 // 2*SO2 + O2 - 2*SO3
48
49 // The equilibrium constant for this reaction is KP
      ^ (2)
50 Ky_prime = Ky^(2);
51
52 // At equilibrium, the moles of the components be
53 // n_SO2_1_eq = 10 - 2*X
54 // n_O2_1_eq = 8 - X
55 // SO3_1_eq = 2*X
56 // Total moles = 18 - X
57
58 // The mole fractions of the components at
      equilibrium are
59 // y_SO3 = 2*X/(18-X)
60 // y_SO2 = (10-2*X)/(18-X)
61 // y_O2 = (8- X)/(18-X)
62
63 // Ky_prime = y_SO3^(2)/(y_SO2^(2)*y_O2)
64 // Ky_prime = ((2*X)^(2)*(18-X))/((10-2*X)^(2)*(8-X)
      )
65 deff(' [y]=f1(X) ', 'y= Ky_prime -((2*X)^(2)*(18-X))
      /(((10-2*X)^(2))*(8-X)) ');
66 X_2 = fsolve(6,f1);
67

```

```

68 n_SO3_prime = 2*X_2;
69 n_SO2_prime = 10 - 2*X_2;
70 n_O2_prime = 8 - X_2;
71
72 printf(" (2).The moles of the components at
        equilibrium are, n_SO3 = %f mol, n_SO2 = %f mol
        and n_O2 = %f mol\n\n",n_SO3_prime,n_SO2_prime,
        n_O2_prime);
73 printf("      Thus the number of moles remains the
        same irrespective of the stoichoimetry of the
        reaction")

```

Scilab code Exa 17.7 Calculation of mole fraction

```

1 clear;
2 clc;
3
4 //Example - 17.7
5 //Page number - 603
6 printf("Example - 17.7 and Page number - 603\n\n");
7
8 // Given
9 T = 500;//[K]
10 // For the reaction , 0.5*A2 + 0.5*B2 - AB
11 delta_G = -4200;//[J/mol]
12 R = 8.314;//[J/mol*K] - Universal gas constant
13
14 //(1)
15 // A2 + B2 - 2*AB
16
17 // We know delta_G_rkn_0 = -R*T*log(K)
18 delta_G_1 = 2*delta_G;
19 K_1 = exp(-delta_G_1/(R*T));// Equilibrium constant
        at 500 K for the above reaction
20 // As can be seen the reaction is not affected by

```

```

        pressure and therefore  $K = K_y$  as  $K_p = 1$ 
21  $K_y = K_1$ ;
22
23 // Initial number of moles of the components are
24  $n_{A2\_1\_in} = 0.5$ ;
25  $n_{B2\_1\_in} = 0.5$ ;
26  $n_{AB\_1\_in} = 0$ ;
27
28 // Let the reaction coordinate at equilibrium for
    the reaction be X
29 // At equilibrium, the moles of the components be
30 //  $n_{A2\_1\_eq} = 0.5 - X$ 
31 //  $n_{B2\_1\_eq} = 0.5 - X$ 
32 //  $n_{AB\_1\_eq} = 2*X$ 
33 // Total moles = 1
34
35 //  $K_y = (2*X)^2 / (0.5 - X)^2$ 
36 deff('[y]=f(X)', 'y=  $K_y - (2*X)^2 / (0.5 - X)^2$  ');
37  $X_1 = \text{fsolve}(0.2, f)$ ;
38
39 // The mole fractions of the components at
    equilibrium are
40  $y_{A2\_1} = 0.5 - X_1$ ;
41  $y_{B2\_1} = 0.5 - X_1$ ;
42  $y_{AB\_1} = 2*X_1$ ;
43
44 printf(" (1).The mole fractions at equilibrium are
     $y_{A2} = \%f$ ,  $y_{B2} = \%f$  and  $y_{AB} = \%f$ \n\n",  $y_{A2\_1}$ ,
     $y_{B2\_1}$ ,  $y_{AB\_1}$ );
45
46 //(2)
47 //  $0.5*A2 + 0.5*B2 - AB$ 
48
49 // We know  $\Delta G_{rkn,0} = -R*T*\log(K)$ 
50  $\Delta G_2 = \Delta G$ ;
51  $K_2 = \exp(-\Delta G_2 / (R*T))$ ; // Equilibrium constant
    at 500 K for the above reaction
52

```



```

53 // As can be seen the reaction is not affected by
    pressure and therefore  $K = K_y$  as  $K_p = 1$ 
54  $K_y_2 = K_2$ ;
55
56 // Initial number of moles of the components are
57  $n_{A2\_2\_in} = 0.5$ ;
58  $n_{B2\_2\_in} = 0.5$ ;
59  $n_{AB\_2\_in} = 0$ ;
60
61 // Let the reaction coordinate at equilibrium for
    the reaction be  $X$ 
62 // At equilibrium, the moles of the components be
63 //  $n_{A2\_2\_eq} = 0.5 - 0.5*X$ 
64 //  $n_{B2\_2\_eq} = 0.5 - 0.5*X$ 
65 //  $n_{AB\_2\_eq} = X$ 
66 // Total moles = 1
67
68 //  $K_y = y_{AB} / (y_{A2}^{(1/2)} * y_{B2}^{(1/2)})$ 
69 //  $K_y = X / (0.5 - 0.5*X)$ 
70  $X_2 = 0.5 * K_y_2 / (1 + 0.5 * K_y_2)$ ;
71
72 // The mole fractions of the components at
    equilibrium are
73  $y_{A2\_2} = 0.5 - 0.5 * X_2$ ;
74  $y_{B2\_2} = 0.5 - 0.5 * X_2$ ;
75  $y_{AB\_2} = X_2$ ;
76
77 printf(" (2). The mole fractions at equilibrium are
     $y_{A2} = %f$ ,  $y_{B2} = %f$  and  $y_{AB} = %f$ \n\n",  $y_{A2\_2}$ ,
     $y_{B2\_2}$ ,  $y_{AB\_2}$ );
78
79 // (3)
80 //  $2*AB \rightleftharpoons A2 + B2$ 
81
82  $K_3 = 1/K_1$ ; // Equilibrium constant at 500 K for the
    above reaction
83 // As can be seen the reaction is not affected by
    pressure and therefore  $K = K_y$  as  $K_p = 1$ 

```

```

84 Ky_3 = K_3;
85
86 // Initial number of moles of the components are
87 n_AB_3_in = 1;
88 n_A2_3_in = 0;
89 n_B2_3_in = 0;
90
91 // Let the reaction coordinate at equilibrium for
    the reaction be X
92 // At equilibrium, the moles of the components be
93 // n_AB_3_eq = 1 - X
94 // n_A2_3_eq = X/2
95 // n_B2_3_eq = X/2
96 // Total moles = 1
97
98 // Ky = (X/2)^(2)/(1-X)^(2)
99 def f1(X) = Ky_3 - (X/2)^(2)/(1-X)^(2);
100 X_3 = fsolve(0.4, f1);
101
102 // The mole fractions of the components at
    equilibrium are
103 y_A2_3 = X_3/2;
104 y_B2_3 = X_3/2;
105 y_AB_3 = 1-X_3;
106
107 printf(" (3).The mole fractions at equilibrium are
    y_A2 = %f, y_B2 = %f and y_AB = %f\n\n", y_A2_3,
    y_B2_3, y_AB_3);

```

Scilab code Exa 17.8 Calculation of heat exchange

```

1 clear;
2 clc;
3
4 //Example - 17.8

```

```

5 //Page number - 606
6 printf("Example - 17.8 and Page number - 606\n\n");
7
8 // Given
9 // P*P +q*Q - r*R + s*S
10 // Let Cp_P = p, Cp_Q = Q, Cp_R = R and Cp_S = S
11
12 //(1)
13 // When reactants are heated from 10 to 25 C,
    reaction takes place at 25 C and products are
    raised from 25 C to 1500 K the heat exchange is
    given by
14 T_1 = 10 + 273.15; // [K]
15 T_2 = 25 + 273.15; // [K]
16 T_3 = 1500; // [K]
17 // Q = integrate('(p*Cp_P + q*Cp-q)*dT', 'T', T_1, T_2)
    + delta_H_rkn_298 + integrate('(r*Cp_R + s*Cp_S)
    *dT', 'T' T_2, T_3);
18 printf(" (1).The expression for the heat exchange
    with the surrounding by the first path is given
    below\n");
19 printf("      Q = integrate((p*Cp_P + q*Cp-q)*dT,T,
    T_1,T_2) + delta_H_rkn_298 + integrate((r*Cp_R +
    s*Cp_S)*dT,TT_2,T_3);\n\n")
20
21 //(2)
22 // When reactants are heated from 10 C to 1500 K,
    reaction take place at 1500 K the heat exchange
    is given by
23 // Q = integrate('(p*Cp_P + q*Cp-q)*dT', 'T', T_1, T_3)
    + delta_H_rkn_1500
24 // where, delta_H_rkn_1500 = delta_H_rkn_298 +
    integrate('(r*Cp_R + s*Cp_S - p*Cp_P - q*Cp-q)*dT
    ', 'T' T_2, T_3);
25 // Therefore
26 // Q = integrate('(p*Cp_P + q*Cp-q)*dT', 'T', T_1, T_3)
    + delta_H_rkn_298 + integrate('(r*Cp_R + s*Cp_S
    - p*Cp_P - q*Cp-q)*dT', 'T' T_2, T_3);

```

```

27 // Q = integrate('(p*Cp_P + q*Cp_q)*dT', 'T', T_1, T_2)
    + delta_H_rkn_298 + integrate('(r*Cp_R + s*Cp_S)
    *dT', 'T' T_2, T_3);
28
29 // which is same as in method (1). The total
    enthalpy change between two fixed points is a
    function only of state and not the path taken.
30
31 printf(" (2).The expression for the heat exchange
    with the surrounding by the second path is given
    below\n");
32 printf("      Q = integrate((p*Cp_P + q*Cp_q)*dT, T,
    T_1, T_2) + delta_H_rkn_298 + integrate((r*Cp_R +
    s*Cp_S)*dT, T T_2, T_3);\n\n")

```

Scilab code Exa 17.9 Determination of heat of reaction

```

1  clear;
2  clc;
3
4  //Example - 17.9
5  //Page number - 606
6  printf("Example - 17.9 and Page number - 606\n\n");
7
8  // Given
9  // SO2 + (1/2)*O2 - SO3
10 R = 1.987; //[cal/mol-K]
11
12 delta_H_S02_298 = -70.96; //[kcal/mol] - Enthalpy of
    formation of S02 at 298.15 K
13 delta_H_S03_298 = -94.45; //[kcal/mol] - Enthalpy of
    formation of S03 at 298.15 K
14 delta_G_S02_298 = -71.79; //[kcal/mol] - Gibbs free
    energy change for formation of SO2 at 298.15 K
15 delta_G_S03_298 = -88.52; //[kcal/mol] - Gibbs free

```

```

        energy change for formation of SO3 at 298.15 K
16
17 // Cp_0 = a + b*T + c*T^(2) + d*T^(3)
18
19 a_S02 = 6.157;
20 a_S03 = 3.918;
21 a_O2 = 6.085;
22 b_S02 = 1.384*10^(-2);
23 b_S03 = 3.483*10^(-2);
24 b_O2 = 0.3631*10^(-2);
25 c_S02 = -0.9103*10^(-5);
26 c_S03 = -2.675*10^(-5);
27 c_O2 = -0.1709*10^(-5);
28 d_S02 = 2.057*10^(-9);
29 d_S03 = 7.744*10^(-9);
30 d_O2 = 0.3133*10^(-9);
31
32 //(1)
33 T_1 = 298.15; // [K]
34
35 delta_H_rkn_298 = delta_H_S03_298 - delta_H_S02_298;
    // [kcal]
36 delta_H_rkn_298 = delta_H_rkn_298*10^(3); // [cal]
37 delta_G_rkn_298 = delta_G_S03_298 - delta_G_S02_298;
    // [kcal]
38 delta_G_rkn_298 = delta_G_rkn_298*10^(3); // [cal]
39
40 delta_a = a_S03 - a_S02 - (a_O2/2);
41 delta_b = b_S03 - b_S02 - (b_O2/2);
42 delta_c = c_S03 - c_S02 - (c_O2/2);
43 delta_d = d_S03 - d_S02 - (d_O2/2);
44
45 // delta_H_rkn_T = delta_H_rkn_298 + integrate('
    delta_a+(delta_b*T)+(delta_c*T^(2))+(delta_d*T
    ^ (3))', 'T', T_1, T);
46 // On simplification we get
47 // delta_H_rkn_T = -22630.14 - 5.2815*T +
    0.9587*10^(-2)*T^(2) - 0.5598*10^(-5)*T^(3) +

```

```

1.3826*10(-9)*T(4)
48
49 printf(" (1).The expression for delta_H_rkn_T as a
    function of T is given by\n");
50 printf("      delta_H_rkn_T = -22630.14 - 5.2815*T +
    0.9587*10(-2)*T(2) - 0.5598*10(-5)*T(3) +
    1.3826*10(-9)*T(4)\n\n");
51
52 //(2)
53 // R*log(K_T/K_298) = integrate('delta_H_rkn_T/T(2)
    ',T,T_1,T)
54 // First let us calculate K_298.
55 // delta_G_rkn_T = - R*T*log(K)
56 K_298 = exp(-delta_G_rkn_298/(R*T_1));
57
58 // On substituting the values and simplifying we get
    the expression
59 // log(K) = 3.87 + 11380.10/T - 2.6580*log(T) +
    0.4825*10(-2)*T - 0.1409*10(-5)*T(2) +
    0.2320*10(-9)*T(3)
60
61 printf(" (2).The expression for log(K) as a function
    of T is given by\n");
62 printf("      log(K) = 3.87 + 11380.10/T - 2.6580*log
    (T) + 0.4825*10(-2)*T - 0.1409*10(-5)*T(2) +
    0.2320*10(-9)*T(3)\n\n");
63
64 //(3)
65 P = 1; //[atm]
66 T = 880; //[K]
67 K = exp(3.87 + 11380.10/T - 2.6580*log(T) +
    0.4825*10(-2)*T - 0.1409*10(-5)*T(2) +
    0.2320*10(-9)*T(3));
68 Kp = P(-1/2); //[atm(-1/2)]
69 Ky = K/Kp;
70
71 // Let the reaction coordinate at equilibrium for
    the reaction be X

```

```

72 // At equilibrium , the moles of the components be
73 // n_SO2_eq = 1 - X
74 // n_O2_eq = 0.5 - 0.5*X
75 // n_SO3_1_eq = X
76 // Total moles = 1.5 - 0.5*X
77
78 // Ky = (X*(1.5 - 0.5*X)^(1/2))/((1-X)*(0.5 - 0.5*X)
      ^ (1/2))
79 def f(' [y]=f(X) ', 'y= Ky - (X*(1.5 - 0.5*X)^(1/2))/((1-X
      )*(0.5 - 0.5*X)^(1/2)) ');
80 X = fsolve(0.8, f);
81
82 // The mole fraction of SO3 at equilibrium is given
      by
83 y_SO3 = X/(1.5 - 0.5*X);
84
85 printf(" (3). The mole fraction of SO3 at equilibrium
      is given by, y_SO3 = %f\n", y_SO3);

```

Scilab code Exa 17.10 Tabulation of equilibrium constant values

```

1 clear;
2 clc;
3
4 //Example - 17.10
5 //Page number - 609
6 printf("Example - 17.10 and Page number - 606\n\n");
7
8 // Given
9 // (1/2)*N2 + (1/2)*O2 - NO
10
11 R = 1.987; // [cal/mol-K]
12
13 delta_H_NO_298 = 21.600; // [kcal/mol] - Enthalpy of
      formation of SO2 at 298.15 K

```

```

14 delta_G_NO_298 = 20.719;//[kcal/mol] - Gibbs free
    energy change for formation of SO2 at 298.15 K
15
16 // Cp_0 = a + b*T + c*T^(2) + d*T^(3)
17
18 a_N2 = 6.157;
19 a_O2 = 6.085;
20 a_NO = 6.461;
21 b_N2 = -0.03753*10^(-2);
22 b_O2 = 0.3631*10^(-2);
23 b_NO = 0.2358*10^(-2);
24 c_N2 = 0.1930*10^(-5);
25 c_O2 = -0.1709*10^(-5);
26 c_NO = -0.07705*10^(-5);
27 d_N2 = -0.6861*10^(-9);
28 d_O2 = 0.3133*10^(-9);
29 d_NO = 0.08729*10^(-9);
30
31 //(1)
32 T_1 = 298.15;//[K]
33
34 delta_H_rkn_298 = delta_H_NO_298;//[kcal]
35 delta_H_rkn_298 = delta_H_rkn_298*10^(3);//[cal]
36 delta_G_rkn_298 = delta_G_NO_298;//[kcal]
37 delta_G_rkn_298 = delta_G_rkn_298*10^(3);//[cal]
38
39 delta_a = a_NO - (a_N2/2) - (a_O2/2);
40 delta_b = b_NO - (b_N2/2) - (b_O2/2);
41 delta_c = c_NO - (c_N2/2) - (c_O2/2);
42 delta_d = d_NO - (d_N2/2) - (d_O2/2);
43
44 // delta_H_rkn_T = delta_H_rkn_298 + integrate('
    delta_a+(delta_b*T)+(delta_c*T^(2))+(delta_d*T
    ^3)', 'T', T_1, T);
45 // On simplification we get
46 // delta_H_rkn_T = 21584.63 - 0.033*T +
    0.0365*10^(-2)*T^(2) - 0.0293*10^(-5)*T^(3) +
    0.0685*10^(-9)*T^(4)

```



```

47
48 printf(" The expression for delta_H_rkn_T as a
    function of T is given by\n");
49 printf(" delta_H_rkn_T = 21584.63 - 0.033*T +
    0.0365*10^(-2)*T^(2) - 0.0293*10^(-5)*T^(3) +
    0.0685*10^(-9)*T^(4)\n\n");
50
51 // Now let us calculate K_298 (at 298 K)
52 // We know delta_G_rkn_298 = -R*T*log(K_298)
53 K_298 = exp(-delta_G_rkn_298/(R*T_1)); // Equilibrium
    constant at 298.15 K
54
55 // log(K_2/K_1) = integrate('delta_H_rkn_298/(R*T
    ^2) ','T',T_1,T)
56 // On substituting the values and simplifying we get
    the expression
57 // log(K) = 1.5103 - 10862.92/T - 0.0166*log(T) +
    1.84*10^(-4)*T - 7.35*10^(-8)*T^(2) +
    1.15*10^(-11)*T^(3)
58
59 printf(" The expression for log(K) as a function of
    T is given by\n");
60 printf(" log(K) = 1.5103 - 10862.92/T - 0.0166*log(T
    ) + 1.84*10^(-4)*T - 7.35*10^(-8)*T^(2) +
    1.15*10^(-11)*T^(3)\n\n\n\n");
61
62 T = [500,1000,1500,2000,2500];
63 K = zeros(5);
64
65 printf(" T (K) \t\t\t K \n\n");
66
67 for i=1:5;
68     K(i) = exp(1.5103-10862.92/T(i) - 0.0166*log(T(i)
        )) + 1.84*10^(-4)*T(i) - 7.35*10^(-8)*T(i)
        ^2 + 1.15*10^(-11)*T(i)^3);
69
70     printf(" %f \t\t %e \n\n",T(i),K(i));
71 end

```

```

72
73 printf("\n\n");
74
75 // It can be seen that for endothermic reactions
    equilibrium constant increases with temperature.
76 printf(" It can be seen that for endothermic
    reactions equilibrium constant increases with
    temperature\n");

```

Scilab code Exa 17.11 Determination of mean standard enthalpy of reaction

```

1 clear;
2 clc;
3
4 //Example - 17.11
5 //Page number - 611
6 printf("Example - 17.11 and Page number - 611\n\n");
7
8 // Given
9 // SO2 + (1/2)*O2 - SO3
10 R = 8.314; // [J/mol-K]
11
12 K_800 = 0.0319; // Equilibrium constant at 800 K
13 K_900 = 0.153; // Equilibrium constant at 900 K
14 T_1 = 800; // [K]
15 T_2 = 900; // [K]
16
17 // We have the relation
18 // log(K_2/K_1) = -(delta_H_rkn/R)*(1/T_2 - 1/T_1)
19 // log(K_900/K_800) = -(delta_H_rkn_850/R)*(1/T_2 -
    1/T_1)
20 delta_H_rkn_850 = -R*log(K_900/K_800)/(1/T_2 - 1/T_1
    ); // [J]
21 delta_H_rkn_850 = delta_H_rkn_850*10^(-3); // [kJ]

```

```

22
23 printf(" The mean standard enthalpy change of
    reaction in the region 800 to 900 is given by
    delta_H_rkn_850 = %f \n",delta_H_rkn_850);

```

Scilab code Exa 17.12 Derivation of expression for enthalpy of reaction

```

1 clear;
2 clc;
3
4 //Example - 17.12
5 //Page number - 611
6 printf("Example - 17.12 and Page number - 611\n\n");
7
8 // Given
9 // log(K) = -2.61 + 23672.62/T - 7.17*log(T) +
    1.24*10^(-2)*T - 0.148*10^(-5)*T^(2) +
    0.039*10^(-9)*T^(3)
10
11 // We know that
12 // dlog(K)/dT = delta_H_rkn/(R*T^(2))
13
14 // Differentiating the above expression of log(K),we
    get
15 // dlog(K)/dT = -23672.62/T^(2) - 7.17/T +
    1.24*10^(-2) - 2*0.148*10^(-5)*T +
    3*0.039*10^(-9)*T^(2)
16
17 // On further simplification we get
18 // delta_H_rkn = -47037.5 - 14.24*T + 2.46*10^(-2)*T
    ^2 - 0.59*10^(-5)*T^(3) + 0.23*10^(-9)*T^(4)
19 printf(" delta_H_rkn = -47037.5 - 14.24*T +
    2.46*10^(-2)*T^(2) - 0.59*10^(-5)*T^(3) +
    0.23*10^(-9)*T^(4)\n");

```

Scilab code Exa 17.13 Determination of equilibrium composition

```
1 clear;
2 clc;
3
4 //Example - 17.13
5 //Page number - 611
6 printf("Example - 17.13 and Page number - 611\n\n");
7
8 // Given
9 T_1 = 298.15; // [K]
10 T = 2600; // [K]
11 R = 1.987; // [cal/mol-K] - Universal gas constant
12
13 // Cp_0 = a + b*T + c*T^(2) + d*T^(3)
14 delta_H_CO_298 = -26.416; // [kcal/mol] - Enthalpy of
    formation of CO at 298.15 K
15 delta_G_CO_298 = -32.808; // [kcal/mol] - Gibbs free
    energy change for formation of CO at 298.15 K
16 delta_H_CO2_298 = -94.052; // [kcal/mol] - Enthalpy of
    formation of CO2 at 298.15 K
17 delta_G_CO2_298 = -94.260; // [kcal/mol] - Gibbs free
    energy change for formation of CO2 at 298.15 K
18
19 // CO + (1/2)*O2 - CO2
20
21 a_CO = 6.726;
22 a_O2 = 6.0685;
23 a_CO2 = 5.316;
24 b_CO = 0.04001*10^(-2);
25 b_O2 = 0.3631*10^(-2);
26 b_CO2 = 1.4285*10^(-2);
27 c_CO = 0.1283*10^(-5);
28 c_O2 = -0.1709*10^(-5);
```

```

29 c_CO2 = -0.8362*10^(-5);
30 d_CO = -0.5307*10^(-9);
31 d_O2 = 0.3133*10^(-9);
32 d_CO2 = 1.784*10^(-9);
33
34
35 delta_H_rkn_298 = delta_H_CO2_298 - delta_H_CO_298;
    // [kcal]
36 delta_H_rkn_298 = delta_H_rkn_298*10^(3); // [cal]
37 delta_G_rkn_298 = delta_G_CO2_298 - delta_G_CO_298;
    // [kcal]
38 delta_G_rkn_298 = delta_G_rkn_298*10^(3); // [cal]
39
40 delta_a = a_CO2 - (a_CO) - (a_O2/2);
41 delta_b = b_CO2 - (b_CO) - (b_O2/2);
42 delta_c = c_CO2 - (c_CO) - (c_O2/2);
43 delta_d = d_CO2 - (d_CO) - (d_O2/2);
44
45 // delta_H_rkn_T = delta_H_rkn_298 + integrate('
    delta_a+(delta_b*T)+(delta_c*T^(2))+(delta_d*T
    ^3)', 'T', T_1, T);
46 // On simplification we get
47 delta_H_rkn_T = -66773.56 - 4.45*T + 0.605*10^(-2)*T
    ^2 - 0.29*10^(-5)*T^3 + 0.54*10^(-9)*T^4;
48
49 // log(K/K_298) = integrate('delta_H_rkn_T/(R*T^(2))
    ', 'T', T_1, T)
50
51 // We know that delta_G_rkn_T = -R*T*log(K)
52 // At 298.15 K
53 K_298 = exp(-delta_G_rkn_298/(R*T_1));
54
55 // Therefore on simplification we get
56 // log(K) = 2.94 + 33605.2/T - 2.24*log(T) +
    0.304*10^(-2)*T - 0.073*10^(-5)*T^2 +
    0.09*10^(-9)*T^3
57 K = exp(2.94 + 33605.2/T - 2.24*log(T) +
    0.304*10^(-2)*T - 0.073*10^(-5)*T^2 +

```

```

    0.09*10(-9)*T(3));
58
59 printf(" The value of equilibrium constant at 2600
    K is given by, K298 = %f\n\n",K);
60
61
62 //(a)
63 P1 = 1; // [atm]
64 Kp1 = P1(-1/2);
65 Ky1 = K/Kp1;
66
67 // Let the reaction coordinate at equilibrium for
    the reaction be X
68 // At equilibrium, the moles of the components be
69 // nCO1eq = 1 - X
70 // nO21eq = 0.5 - 0.5X
71 // nCO21eq = X
72 // Total moles = 1.5 - 0.5*X
73
74 // Ky = yCO2/(yCO(1/2)*yCO)
75 //ky = (X*(1.5 - 0.5*X)(1/2))/((1-X)*(0.5 - 0.5*X)
    (1/2))
76
77 def ('[y]=f(X)', 'y= Ky1-(X*(1.5 - 0.5*X)(1/2))/((1-X)
    )*(0.5 - 0.5*X)(1/2) ');
78 X1 = fsolve(0.9,f);
79
80 yCO21 = X1/(1.5-0.5*X1);
81 yCO1 = (1-X1)/(1.5-0.5*X1);
82 yO21 = (0.5-0.5*X1)/(1.5-0.5*X1);
83
84 printf(" (a).The equilibrium composition (at 1 atm)
    is given by, yCO2 = %f, yCO = %f and yO2 = %f\n
    n\n",yCO21,yCO1,yO21);
85
86 //(b)
87 P2 = 10; // [atm]
88 Kp2 = P2(-1/2);

```

```

89 Ky_2 = K/Kp_2;
90
91 // Ky = y_CO2/(y_CO^(1/2)*y_CO)
92 //ky = (X*(1.5-0.5*X)^(1/2))/((1-X)*(0.5-0.5*X)
    ^ (1/2))
93
94 deff(' [y]=f1(X)', 'y= Ky_2-(X*(1.5-0.5*X)^(1/2))/((1-
    X)*(0.5-0.5*X)^(1/2))');
95 X_2 = fsolve(0.9, f1);
96
97 y_CO2_2 = X_2/(1.5-0.5*X_2);
98 y_CO_2 = (1-X_2)/(1.5-0.5*X_2);
99 y_O2_2 = (0.5-0.5*X_2)/(1.5-0.5*X_2);
100
101 printf(" (b).The equilibrium composition (at 10 atm)
    is given by, y_CO2 = %f, y_CO = %f and y_O2 = %f
    \n\n", y_CO2_2, y_CO_2, y_O2_2);
102
103 // (c)
104 P_3 = 1; // [atm]
105 Kp_3 = P_3^(-1/2);
106 Ky_3 = K/Kp_3;
107
108 // Ky = y_CO2/(y_CO^(1/2)*y_CO)
109 //ky = (X*(1.5-0.5*X)^(1/2))/((1-X)*(0.5-0.5*X)
    ^ (1/2))
110
111 // At equilibrium, the moles of the components be
112 // n_CO_3_eq = 1 - X
113 // n_O2_3_eq = 0.5 - 0.5X
114 // n_CO2_3_eq = X
115 // n_N2_eq = 1;
116 // Total moles = 2.5 - 0.5*X
117
118 deff(' [y]=f2(X)', 'y= Ky_3-(X*(2.5-0.5*X)^(1/2))/((1-
    X)*(0.5-0.5*X)^(1/2))');
119 X_3 = fsolve(0.9, f2);
120

```

```

121 y_CO2_3 = X_3/(2.5-0.5*X_3);
122 y_CO_3 = (1-X_3)/(2.5-0.5*X_3);
123 y_O2_3 = (0.5-0.5*X_3)/(2.5-0.5*X_3);
124 y_N2 = 1/(2.5-0.5*X_3);
125
126 printf(" (c).The equilibrium composition (at 1 atm
      and 1 mol N2) is given by, y_CO2 = %f, y_CO = %f
      , y_O2 = %f and y_N2 = %f\n\n",y_CO2_3,y_CO_3,
      y_O2_3,y_N2);

```

Scilab code Exa 17.14 Determination of equilibrium composition

```

1 clear;
2 clc;
3
4 //Example - 17.14
5 //Page number - 614
6 printf("Example - 17.14 and Page number - 614\n\n");
7
8 // Given
9 T = 25 + 298.15;//[K] - Temperature
10 R = 8.314;//[J/mol-K]
11 delta_G_298 = -1000;//[J] - Gibbs free energy change
    at 298 K
12
13 //  $G_E/(R*T) = x_1*x_2$ 
14
15 // We know that  $\Delta G_{rkn} = -R*T*\log(K)$ ,
    therefore
16 K = exp(-delta_G_298/(R*T));
17
18 //(1)
19 // Let x_1 is the mole fraction of A and x_2 be the
    mole fraction of B
20 // If A and B form an ideal mixture then,

```



```

21 Ky = 1;
22 // and K = Kx = x_2/x_1
23 // and at equilibrium x_2/x_1 = K
24 // (1-x_1)/x_1 = K
25 x_1 = 1/(1+K);
26
27 printf(" (1).The equilibrium composition (for ideal
    behaviour) is given by x_1 = %f\n\n",x_1);
28
29 //(2)
30 // The activity coefficients are given by
31 // log(Y1) = [del(n*G_E/(R*T))/del(n_1)]_T,P,n_2 =
    x_2^(2)
32 // similarly , log(Y2) = x_1^(2)
33
34 // The equilibrium constant for the liquid phase
    reaction is given by
35 // K = Kx*Ky = (x_2*Y2)/(x_1*Y1) = (x_2*exp(x_1^(2))
    )/(x_1*exp(x_2^(2)))
36 // Solving the above equation we get
37
38 def(' [y]=f2(x_1)', 'y= K -((1-x_1)*exp(x_1^(2)))/(
    x_1*exp((1-x_1)^(2)))');
39 x_1_prime = fsolve(0.9,f2);
40
41 printf(" (2).The equilibrium composition (for non-
    ideal behaviour) is given by x_1 = %f\n\n",
    x_1_prime);

```

Scilab code Exa 17.15 Determination of the

```

1 clear;
2 clc;
3
4 //Example - 17.15

```

```

5 //Page number - 615
6 printf("Example - 17.15 and Page number - 615\n\n");
7
8 // Given
9 T_1 = 298.15; // [K] - Standard reaction temperature
10 T_2 = 373; // [K] - Reaction temperature
11 P = 1; // [atm]
12 R = 1.987; // [cal/mol-K] - Universal gas constant
13
14 // CH3COOH (1) + C2H5OH (1) - CH3COOC2H5 (1) + H2O (
15     1)
16 delta_H_CH3COOH_298 = -116.2*10^(3); // [cal/mol]
17 delta_H_C2H5OH_298 = -66.35*10^(3); // [cal/mol]
18 delta_H_CH3COOC2H5_298 = -110.72*10^(3); // [cal/mol]
19 delta_H_H2O_298 = -68.3174*10^(3); // [cal/mol]
20
21 delta_G_CH3COOH_298 = -93.56*10^(3); // [cal/mol]
22 delta_G_C2H5OH_298 = -41.76*10^(3); // [cal/mol]
23 delta_G_CH3COOC2H5_298 = -76.11*10^(3); // [cal/mol]
24 delta_G_H2O_298 = -56.6899*10^(3); // [cal/mol]
25
26 delta_H_rkn_298 = delta_H_CH3COOC2H5_298 +
27     delta_H_H2O_298 - delta_H_CH3COOH_298 -
28     delta_H_C2H5OH_298; // [cal/mol]
29 delta_G_rkn_298 = delta_G_CH3COOC2H5_298 +
30     delta_G_H2O_298 - delta_G_CH3COOH_298 -
31     delta_G_C2H5OH_298; // [cal/mol]
32
33 // We know that delta_G_rkn_T = -R*T*log(K)
34 // At 298.15 K
35 K_298 = exp(-delta_G_rkn_298/(R*T_1) );
36
37 // We know that dlog(K)/dT = delta_H_rkn/(R*T^(2))
38 // If delta_H_rkn is assumed constant we get
39 // log(K_2/K_1) = (-delta_H_rkn/R)*(1/T_2 - 1/T_1)
40 // log(K_373/K_298) = (-delta_H_rkn_298/R)*(1/T_2 -
41     1/T_1)

```

```

37
38 K_373 = exp(log(K_298) + (-delta_H_rkn_298/R)*(1/T_2
    - 1/T_1));
39 // Note that the equilibrium constant value rises
    because the reaction is endothermic
40
41 printf(" The value of equilibrium constant at 373 K
    is , K_373 = %f\n\n",K_373);
42
43 // Let the reaction coordinate at equilibrium for
    the reaction be X
44 // At equilibrium , the moles of the components be
45 // n_CH3COOH = 1 - X
46 // n_C2H5OH = 1 - X
47 // n_CH3COOC2H5 = X
48 // n_H2O = X
49 // Total moles = 2
50
51 // Kx = (x_CH3COOH*x_C2H5OH)/(x_CH3COOC2H5*x_H2O)
52 // Assuming the liquid mixture to be ideal ,that is
    Ky = 1, therefore K_x = K
53 K_x = K_373;
54 // X^(2)/(1-X)^(2) = K_x
55 X = (K_x)^(1/2)/(1+(K_x)^(1/2));
56
57 // The mole fraction of ethyl acetate is given by
58 x_CH3COOC2H5 = X/2;
59
60 printf(" The mole fraction of ethyl acetate in the
    equilibrium reaction mixture is given by,
    x_CH3COOC2H5 = %f\n",x_CH3COOC2H5);

```

Scilab code Exa 17.16 Calculation of the value of Gibbs free energy

```
1 clear;
```

```

2  clc;
3
4  //Example - 17.16
5  //Page number - 617
6  printf("Example - 17.16 and Page number - 617\n\n");
7
8  // Given
9  // CaCO3 (s1) - CaO (s2) + CO2 (g)
10 T_1 = 898 + 273.15; // [K]
11 T_2 = 700 + 273.15; // [K]
12 R = 8.314; // [J/mol-K] - Universal gas constant
13
14 P_CO2_T_1 = 1; // [atm] - Decomposition pressure of
    CO2 over CaCO3 at 898 C
15 P_CO2_T_2 = 0.0333; // [atm] - Decomposition pressure
    of CO2 over CaCO3 at 700 C
16
17 // The equilibrium constant of the reaction is given
    by
18 //  $K = (a_{CO2} \cdot a_{CaO}) / a_{CaCO3}$ 
19
20 // Since the pressure is small therefore carbon
    dioxide can be assumed to behave as ideal gas and
    thus
21 //  $a_{CO2} = y_{CO2} \cdot P / 1 = P_{CO2}$ 
22
23 // The activity of CaO is (CaO is pure)
24 //  $a_{CaO} = f_{CaO} / f_{0,CaO} = \exp[V_{CaO} \cdot (P - P_0) / (R \cdot T)]$ 
    = 1 (since pressure is low)
25
26 // The activity of CaCO3 is (CaCO3 is pure)
27 //  $a_{CaCO3} = f_{CaCO3} / f_{0,CaCO3} = \exp[V_{CaCO3} \cdot (P -$ 
    P_0) / (R \cdot T)] = 1 (since pressure is low)
28
29 // Since pressures are around 1 atm, therefore
    Poynting factors are also near 1, and thus
    activity of CaO and CaCO3 is unity and
    equilibrium constant is given by

```

```

30 //K = P_CO2 , therefore
31
32 // At 898 C
33 K_T_1 = P_CO2_T_1;
34 delta_G_T_1 = -R*T_1*log(K_T_1);
35
36 // At 700 C
37 K_T_2 = P_CO2_T_2;
38 delta_G_T_2 = -R*T_2*log(K_T_2);
39
40 printf(" The value of delta_G_rkn at 700 C is %f J\n
    \n",delta_G_T_1);
41 printf(" The value of delta_G_rkn at 898 C is %f J\n
    \n",delta_G_T_2);

```

Scilab code Exa 17.17 Calculation of nu

```

1 clear;
2 clc;
3
4 //Example - 17.17
5 //Page number - 618
6 printf("Example - 17.17 and Page number - 618\n\n");
7
8 // Given
9 T = 700 + 273.15; // [K]
10 K = 7.403; // Equilibrium constant for the reaction
    at 700 C
11
12 // CH4 - C (s) + 2*H2
13
14 // The equilibrium constant of the reaction is given
    by
15 // K = (a_C*a_H2^(2))/a_CH4
16

```

```

17 // Since carbon is pure therefore its activity is
    given by
18 //  $a_C = f/f_0 = 1$  as pressure is 1 atm.
19 // Since the pressure is low, therefore the gas phase
    can be taken to be ideal, therefore
20 //  $K = (y_{H_2}^2 * P^2) / (y_{CH_4} * P) = y_{H_2}^2 / y_{CH_4}$ 
    (as  $P = 1$  atm)
21  $K_y = K$ ; // ( $K_p = 1$  atm)
22
23 // Initial moles of the species are
24  $n_{CH_4} = 1$ ;
25  $n_{H_2} = 0$ ;
26  $n_C = 0$ ;
27
28 // Let the reaction coordinate at equilibrium for
    the reaction be X
29 // Moles at equilibrium be
30 //  $n_{CH_4,eq} = 1 - X$ 
31 //  $n_{H_2,eq} = 2 * x$ 
32 //  $n_{C,eq} = X$ 
33
34 // Moles present in gas phase
35 //  $n_{CH_4,gas} = 1 - X$ 
36 //  $n_{H_2,gas} = 2 * x$ 
37 // Total moles =  $1 + X$ 
38
39 // gas phase mole fraction
40 //  $y_{CH_4,gas} = (1 - X) / (1 + X)$ 
41 //  $y_{H_2,gas} = 2 * x / (1 + X)$ 
42
43 //  $K_y = y_{H_2,gas}^2 / y_{CH_4,gas}$ 
44  $X = (K / (4 + K))^{(1/2)}$ ;
45
46 printf(" The number of moles of carbon black formed
    from one mole of methane is %f mol\n\n", X);
47
48 // Therefore mole fraction of components in the gas
    phase at equilibrium is

```

```

49 y_CH4 = (1-X)/(1+X);
50 y_H2 = 2*X/(1+X);
51
52 printf(" The mole fraction of components in the gas
    phase at equilibrium is given by y_CH4 = %f and
    y_H2 = %f \n",y_CH4,y_H2);

```

Scilab code Exa 17.18 Determination of value of the equilibrium constant

```

1 clear;
2 clc;
3
4 //Example - 17.18
5 //Page number - 619
6 printf("Example - 17.18 and Page number - 619\n\n");
7
8 // Given
9 T_1 = 298.15;//[K] - Standard reaction temperature
10 T_2 = 1042;//[K] - Reaction temperature
11 R = 1.987;//[cal/mol-K] - Universal gas constant
12
13 // CaCO3 (s1) - CaO (s2) + CO2 (g)
14
15 delta_H_CaCO3_298 = -289.5;//[kcal/mol] - Enthalpy
    of formation of CaCO3 at 298.15 K
16 delta_H_CaO_298 = -151.7;//[kcal/mol] - Enthalpy of
    formation of CaO at 298.15 K
17 delta_H_CO2_298 = -94.052;//[kcal/mol] - Enthalpy of
    formation of CO2 at 298.15 K
18 delta_G_CaCO3_298 = -270.8;//[kcal/mol] - Gibbs free
    energy change for formation of CaCO3 at 298.15 K
19 delta_G_CaO_298 = -144.3;//[kcal/mol] - Gibbs free
    energy change for formation of CaO at 298.15 K
20 delta_G_CO2_298 = -94.260;//[kcal/mol] - Gibbs free
    energy change for formation of CO2 at 298.15 K

```

```

21
22 // The standard heat capacity data as a function of
    temperature are given below
23 // Cp_CO2 = 5.316 + 1.4285*10^(2)*T - 0.8362*10^(-5)
    *T^(2) + 1.784*10^(-9)*T^(3)
24 // Cp_CaO = 12.129 + 0.88*10^(-3)*T - 2.08*10^(5)*T
    ^(-2)
25 // Cp_CaCO3 = 24.98 + 5.240*10^(-3)*T - 6.199*10^(5)
    *T^(-2)
26 // Therefore Cp_0 is given by
27 // Cp_0 = Cp_CO2 + Cp_CaO - Cp_CaCO3
28 // Cp_0 = -7.535 + 9.925*10^(-3)*T - 0.8362*10^(-5)*
    T^(2) + 1.784*10^(-9)*T^(3) + 4.119*10^(5)*T^(-2)
29
30 // The standard enthalpy change of the reaction at
    298.15 K is given by
31 delta_H_rkn_298 = delta_H_CO2_298 + delta_H_CaO_298
    - delta_H_CaCO3_298;//[kcal]
32 delta_H_rkn_298 = delta_H_rkn_298*10^(3);//[cal]
33 delta_G_rkn_298 = delta_G_CO2_298 + delta_G_CaO_298
    - delta_G_CaCO3_298;//[kcal]
34 delta_G_rkn_298 = delta_G_rkn_298*10^(3);//[cal]
35
36 // The standard enthalpy change of the reaction at
    temperature T is given by
37 // delta_H_rkn_T = delta_H_rkn_298 + integrate
    ('-7.535 + 9.925*10^(-3)*T - 0.8362*10^(-5)*T^(2)
    + 1.784*10^(-9)*T^(3) + 4.119*10^(5)*T^(-2)', 'T
    ', T_1, T);
38 // On simplification we get
39 // delta_H_rkn_T = 47005.3 - 7.535*T +
    4.9625*10^(-3)*T^(2) - 0.2787*10^(-5)*T^(3) +
    0.446*10^(-9)*T^(4) - 4.119*10^(5)*T^(-1)
40
41
42 // log(K_2/K_1) = integrate('delta_H_rkn_T/(R*T^(2))
    ', 'T', T_1, T)
43 log_K2_K1 = integrate('(47005.3 - 7.535*T

```



```

+4.9625*10^(-3)*T^(2) - 0.2787*10^(-5)*T^(3)
+0.446*10^(-9)*T^(4) - 4.119*10^(5)*T^(-1))/T^(2)
', 'T', T_1, T_2); // log(K_2/K_1)
44
45 // We know that delta_G_rkn_T = -R*T*log(K)
46 // At 298.15 K
47 K_298 = exp(-delta_G_rkn_298/(R*T_1) );
48
49 // Putting the values in the above expression we get
50 // log(K_1042/K_298) = log_K2_K1/R
51 K_1042 = K_298*exp(log_K2_K1/R);
52
53 printf(" The value of equilibrium constant at 1042 K
is , K_1042 = %f\n\n", K_1042);
54
55 // Since for the given reaction K = P_CO2, where P is
in atm, therefore ,
56 P_CO2 = K_1042;
57 // and thus decomposition takes place till the
partial pressure of CO2 reaches 0.095 atm
58 // After that the decomposition in the closed vessel
stops as equilibrium is achieved.
59
60 printf(" The equilibrium pressure of CO2 is , P_CO2 =
%f atm \n", P_CO2);

```

Scilab code Exa 17.19 Determination of the value of equilibrium constant

```

1 clear;
2 clc;
3
4 //Example - 17.19
5 //Page number - 620
6 printf("Example - 17.19 and Page number - 620\n\n");
7

```

```

8 // Given
9 T_1 = 298.15; //[k] - Standard reaction temperature
10 T_2 = 1200; //[K] - Reaction temperature
11 R = 1.987; //[cal/mol-K] - Universal gas constant
12
13 // C (s) + CO2 (g) - 2*CO2 (g) // Reaction 1
14 // CO2 + H2 - CO + H2O // Reaction 2
15
16 K_1 = 63; // Equilibrium constant for the first
    reaction
17 K_2 = 1.4; // Equilibrium constant for the second
    reaction
18
19 delta_G_H2O_298 = -54640; //[cal/mol] - Standard
    Gibbs free energy of formation of water vapour
20 delta_H_H2O_298 = -57800; //[cal/mol] - Standard
    enthalpy change of formation of water vapour
21 delta_G_rkn_298 = delta_G_H2O_298;
22
23 // The standard heat capacity data of the components
    in cal/mol-K are given below
24 // Cp_H2 = 6.947 - 0.2*10^(-3)*T + 4.8*10^(-7)*T^(2)
25 // Cp_O2 = 6.148 + 3.1*10^(-3)*T - 9.2*10^(-7)*T^(2)
26 // Cp_H2O = 7.256 + 2.3*10^(-3)*T + 2.8*10^(-7)*T
    ^ (2)
27
28 // Let us consider two more reactions
29 // C (s) + (1/2)*O2 - CO // Reaction 3
30 // H2 + (1/2)*O2 - H2O // Reaction 4
31
32 // Considering the above 4 reactions, it can be
    shown that reaction (3) = reaction (1) + reaction
    (4) - reaction (2)
33 // Thus, delta_G_rkn_3 = delta_G_rkn_1 +
    delta_G_rkn_4 - delta_G_rkn_2
34 // or, -R*T*log(K_3) = -R*T*log(K_1) + -R*T*log(K_4)
    - -R*T*log(K_2)
35 // K_3 = (K_1*K_4/K_2)

```

```

36
37 // Now we have to determine K_4 at 1200 K.
38 // The standard enthalpy change of reaction (4) as a
   // fuction of temperature is given by
39 //  $\Delta H_{\text{rkn}_T} = -57020 - 2.765 \cdot T + 0.475 \cdot 10^{-3} \cdot T^2 + 8.67 \cdot 10^{-8} \cdot T^3$ ;
40
41 //  $\log(K_{4_2}/K_{4_1}) = \int_{T_1}^{T_2} \frac{\Delta H_{\text{rkn}_T}}{R \cdot T^2} dT$ 
42 log_K2_K1_4 = integrate('(-57020-2.765*T
   // +0.475*10^(-3)*T^(2)+8.67*10^(-8)*T^(3))/T^(2)',
   // T',T_1,T_2);
43
44 // We know that  $\Delta G_{\text{rkn}_T} = -R \cdot T \cdot \log(K)$ 
45 // At 298.15 K
46 K_4_298 = exp(-delta_G_rkn_298/(R*T_1) );
47
48 // Putting the values in the above expression we get
49 //  $\log(K_{4_1200}/K_{4_298}) = \log_K2_K1_R/R$ 
50 K_4_1200 = K_4_298*exp(log_K2_K1_4/R);
51 K_4 = K_4_1200;
52
53 // Therefore the equilibrium constant for reaction
   // (3) at 1200 K is given by
54 K_3 = (K_1*K_4)/K_2;
55
56 printf(" The equilibrium constant at 1200 K for the
   // given reaction is , K = %e\n",K_3);

```

Scilab code Exa 17.20 Calculation of standard equilibrium cell voltage

```

1 clear;
2 clc;
3
4 //Example - 17.20

```

```

5 //Page number - 622
6 printf("Example - 17.20 and Page number - 622\n\n");
7
8 // Given
9 delta_G_H2O_298 = -237.034; //[kJ/mol] - Standard
    Gibbs free energy of formation of H2O (l) at 298
    K
10 F = 96485; //[C/mol] - Faraday constant
11
12 //(1)
13 // For the reaction
14 // H2 (g) + (1/2)*O2 (g) - H2O (l)
15 n = 2; // Number of electrons transferred in the
    reaction
16
17 // The standard Gibbs free energy change of the
    reaction is given by
18 // delta_G_rkn = delta_G_for_H2O(l) - delta_G_for_H2
    (g) - (1/2)*delta_G_for_O2(g)
19 // Since delta_G_for_H2 = 0 and delta_G_for_O2 = 0
    (pure components)
20 delta_G_rkn = delta_G_H2O_298; //[kJ]
21 delta_G_rkn = delta_G_rkn*10^(3); //[J]
22
23 // delta_G_rkn = -n*F*E_0
24 // Thus standard equilibrium cell voltage is given
    by
25 E_0 = - delta_G_rkn/(n*F); //[V]
26
27 printf(" (1).The standard equilibrium cell voltage
    is %f V\n\n",E_0);
28
29 //(2)
30 // For the reaction
31 // 2*H2 (g) + O2 (g) - 2*H2O (l)
32 n_prime = 4; // Number of electrons transferred in
    the reaction
33

```

```

34 // The standard Gibbs free energy change of the
    reaction is given by
35 //  $\Delta G_{\text{rkn}} = 2\Delta G_{\text{for\_H2O}(l)} - 2\Delta G_{\text{for\_H2}(g)} - \Delta G_{\text{for\_O2}(g)}$ 
36 // Since  $\Delta G_{\text{for\_H2}} = 0$  and  $\Delta G_{\text{for\_O2}} = 0$ 
    (pure components)
37  $\Delta G_{\text{rkn\_prime}} = 2\Delta G_{\text{H2O\_298}}$ ;//[kJ]
38  $\Delta G_{\text{rkn\_prime}} = \Delta G_{\text{rkn\_prime}} \times 10^3$ ;//[J]
39
40 //  $\Delta G_{\text{rkn}} = -nF E_0$ 
41 // Thus standard equilibrium cell voltage is given
    by
42  $E_0_{\text{prime}} = -\Delta G_{\text{rkn\_prime}} / (n_{\text{prime}} F)$ ;//[V]
43
44 printf(" (2).The standard equilibrium cell voltage
    is %f V\n\n",E_0_prime);
45
46 // Thus the result obtained is same for both the
    reactions

```

Scilab code Exa 17.21 Calculation of number of chemical reactions

```

1 clear;
2 clc;
3
4 //Example - 17.21
5 //Page number - 624
6 printf("Example - 17.21 and Page number - 624\n\n");
7
8 // Given
9 P = 2; // Number of phases
10 C = 5; // Number of components
11
12 // First we write chemical equations for formation
    of each component from basic elements

```

```

13 // C + 2*H2 = CH4 // (reaction 1)
14 // H2 + (1/2)*O2 = H2O // (reaction 2)
15 // C + (1/2)*O2 = CO // (reaction 3)
16 // C + O2 = CO2 // (reaction 4)
17
18 // We do not have C in the equilibrium reaction
    mixture, therefore we have to eliminate it.
19 // Substituting for C from reaction (1) into
    reactions (3) and (4) we get the following set of
    reactions
20 // H2 + (1/2)*O2 = H2O
21 // CH4 - 2*H2 + (1/2)*O2 = CO
22 // CH4 - 2*H2 + O2 = CO2
23
24 // or ,
25 // H2 + (1/2)*O2 = H2O
26 // CH4 + (1/2)*O2 = CO + 2*H2
27 // CH4 + O2 = CO2 + 2*H2
28
29 // We do not have O2 in the equilibrium reaction
    mixture, therefore we have to eliminate it
30 // Substituting for O2 from the first reaction of
    the above set into second and third reactions of
    the above set we get the following set of
    reactions.
31 // CH4 + H2O - H2 = CO + 2*H2
32 // CH4 + 2*H2O - 2*H2 = CO2 + 2*H2
33
34 // Therefore one set of independent reactions is
35 // CH4 + H2O = CO + 3*H2
36 // CH4 + 2*H2O = CO2 + 4*H2
37
38 // Another set of independent reactions can be
    obtained by substituting for CH4 from the first
    reaction into second and we get
39 // CH4 + H2O = CO + 3*H2
40 // CO + 3*H2 - H2O + 2*H2O = CO2 + 4*H2
41

```

```

42 // Or,
43 // CH4 + H2O = CO + 3*H2
44 // CO + H2O = CO2 + H2
45 // This is another set of independent reactions.
    Thus whatever be the set of independent reactions
    , the number of independent reactions are two
46 // If different set of reactions are considered ,
    then we get different equilibrium constants ,
    different reaction coordinates but the final
    composition will be same
47 // Thus only 2 independent reactions occur ,therefore
48 r = 2;
49
50 // and the number of degree of freedom becomes ,
51 F = r - P + C;
52
53 printf(" The number of independent chemical
    reactions are %f \n\n",r);
54
55 printf(" The first set of independent reactions are
    given below\n");
56 printf(" CH4 + H2O = CO + 3*H2\n");
57 printf(" CH4 + 2*H2O = CO2 + 4*H2\n\n");
58
59 printf(" The second set of independent reactions are
    given below\n");
60 printf(" CH4 + H2O = CO + 3*H2\n");
61 printf(" CO + H2O = CO2 + H2");

```

Scilab code Exa 17.22 Calculation of number of chemical reactions

```

1 clear;
2 clc;
3
4 //Example - 17.22

```

```

5 //Page number - 626
6 printf("Example - 17.22 and Page number - 626\n\n");
7
8 // Given
9 T = 400; //[K] - Temperature
10 P = 1; //[atm] - Pressure
11 R = 1.987; //[cal/mol-K] - Universal gas constannt
12
13 delta_G_n_pentane_400 = 9600; //[cal/mol] - Standard
    Gibbs free energy of formation of n-pentane at
    400 K
14 delta_G_iso_pentane_400 = 8200; //[cal/mol] -
    Standard Gibbs free energy of formation of iso-
    pentane at 400 K
15 delta_G_neo_pentane_400 = 9000; //[cal/mol] -
    Standard Gibbs free energy of formation of neo-
    pentane at 400 K
16
17 // The three reactions for the formation of these
    isomers can be written as follows
18 // 5*C + 6*H2 = n-pentane
19 // 5*C + 6*H2 = iso-pentane
20 // 5*C + 6*H2 = neo-pentane
21
22 // We can eliminate elemental carbon and hydrogen as
    they are not present in equilibrium reaction
    mixture and get the following two sets of
    independent reactions
23 // n-pentane = iso-pentane
24 // n-pentane = neo-pentane
25
26 // or ,
27 // iso-pentane = n-pentane
28 // iso-pentane = neo-pentane
29
30 // or ,
31 // noe-pentane = iso-pentane
32 // neo-pentane = n-pentane

```



```

33
34 // Let us take the following set of independent
    reactions
35 // iso-pentane = n-pentane // (reaction 1)
36 delta_G_rkn_1 = delta_G_n_pentane_400 -
    delta_G_iso_pentane_400;//[cal]
37 K_1 = exp(-delta_G_rkn_1/(R*T));
38 // iso-pentane = neo-pentane // (reaction 2)
39 delta_G_rkn_2 = delta_G_neo_pentane_400 -
    delta_G_iso_pentane_400;//[cal]
40 K_2 = exp(-delta_G_rkn_2/(R*T));
41
42 // Let the initial number of moles be
43 // n_iso_pentane = 1
44 // n_n_pentane = 0
45 // n_neo_pentane = 0
46
47 // Let the reaction coordinate at equilibrium for
    the two reaction be X_1 and X_2 respectively
48 // At equilibrium, the moles of the components be
49 // n_iso_pentane_eq = 1 - X_1 - X_2
50 // n_n_pentane_eq = X_1
51 // n_neo_pentane_eq = X_2
52 // Total moles = 1
53
54 // Pressure has no effect on these reactions ( P = 1
    atm) and therefore
55 Ky_1 = K_1;
56 Ky_2 = K_2;
57
58 // From reaction (1), we get
59 // Ky_1 = X_1/(1-X_1-X_2)
60
61 // From reaction (2), we get
62 // Ky_2 = X_2/(1-X_1-X_2)
63
64 // Putting the value of X_1 from first equation into
    the second we get

```

```

65 // X_1 = (Ky_1*(1-X_2))/(1+Ky_1)
66 // Now putting it into the second equation we grt
67 // Ky_2 = X_2/(1-((Ky_1*(1-X_2))/(1+Ky_1))-X_2)
68 // Now solving for X_2
69 def f(' [y]=f(X_2)', 'y= Ky_2 - X_2/(1-((Ky_1*(1-X_2))
    /(1+Ky_1))-X_2) ');
70 X_2 = fsolve(0.8,f);
71
72 // Therefore X_1 can be calculated as
73 X_1 = (Ky_1*(1-X_2))/(1+Ky_1);
74
75 // Finally the mole fractions of the components at
    equilibrium
76 y_n_pentane = X_1;
77 y_neo_pentane = X_2;
78 y_iso_pentane = 1 -X_1 - X_2;
79
80 printf(" The equilibrium composition is given by,
    y_n_pentane = %f, y_neo_pentane = %f and
    y_iso_pentane = %f\n\n",y_n_pentane ,
    y_neo_pentane ,y_iso_pentane);
81
82 // Let us consider another set of independent
    reactions
83
84 // n-pentane = iso-pentane // (reaction 3)
85 delta_G_rkn_3 = delta_G_iso_pentane_400 -
    delta_G_n_pentane_400;//[cal]
86 K_3 = exp(-delta_G_rkn_3/(R*T));
87 // n-pentane = neo-pentane // (reaction 4)
88 delta_G_rkn_4 = delta_G_neo_pentane_400 -
    delta_G_n_pentane_400;//[cal]
89 K_4 = exp(-delta_G_rkn_4/(R*T));
90
91 // Let the initial number of moles be
92 // n_n_pentane = 1
93 // n_iso_pentane = 0
94 // n_neo_pentane = 0

```

```

95
96 // Let the reaction coordinate at equilibrium for
    the two reaction be X_3 and X_4 respectively
97 // At equilibrium , the moles of the components be
98 // n_n_pentane_eq = 1 - X_3 - X_4
99 // n_iso_pentane_eq = X_4
100 // n_neo_pentane_eq = X_4
101 // Total moles = 1
102
103 // Pressure has no effect on these reactions (P = 1
    atm) and therefore
104 Ky_3 = K_3;
105 Ky_4 = K_4;
106
107 // From reaction (3), we get
108 // Ky_3 = X_3/(1-X_3-X_4)
109
110 // From reaction (4), we get
111 // Ky_4 = X_4/(1-X_3-X_4)
112
113 // Putting the value of X_3 from first equation into
    the second we get
114 // X_3 = (Ky_3*(1-X_4))/(1+Ky_3)
115 // Now putting it into the second equation we grt
116 // Ky_4 = X_4/(1-((Ky_1*(1-X_4))/(1+Ky_3))-X_4)
117 // Now solving for X_4
118 def f(' [y]=f1 (X_4) ', 'y= Ky_4 - X_4/(1-((Ky_3*(1-X_4))
    /(1+Ky_3))-X_4) ');
119 X_4 = fsolve(0.8, f1);
120
121 // Therefore X_3 can be calculated as
122 X_3 = (Ky_3*(1-X_4))/(1+Ky_3);
123
124 // Finally the mole fractions of the components at
    equilibrium
125 y_n_pentane1 = 1 - X_3 - X_4;
126 y_neo_pentane1 = X_4;
127 y_iso_pentane1 = X_3;

```

```

128
129 // The final composition does not depend on the set
    of reactions considered.
130
131 printf(" For another set of independent reactions
    considered\n");
132 printf(" The equilibrium composition is given by,
    y_n_pentane = %f, y_neo_pentane = %f and
    y_iso_pentane = %f\n\n\n",y_n_pentane1,
    y_neo_pentane1,y_iso_pentane1);
133 printf(" Thus the final composition does not depend
    on the set of reactions considered\n\n\n");
134 printf(" The number of independent reactions taking
    place is two");

```

Scilab code Exa 17.23 Calculation of equilibrium composition

```

1 clear;
2 clc;
3
4 //Example - 17.23
5 //Page number - 628
6 printf("Example - 17.23 and Page number - 628\n\n");
7
8 // Given
9 T = 600; //[K] - Temperature
10 P = 1; //[atm] - Pressure
11 R = 1.987; //[cal/mol-K] - Universal gas constannt
12
13 // CH4 + H2O = CO + 3*H2 // (Reaction 1)
14 // CO + H2O = CO2 + H2 // (Reaction 2)
15
16 K_1 = 0.574; // Equilibrium constant of first
    reaction
17 K_2 = 2.21; // Equilibrium constant of second

```

```

        reaction
18
19 // Initial number of moles of the components are
20 // n_CH4 = 1
21 // n_H2O = 5
22 // n_CO = 0
23 // n_H2 = 0
24 // n_CO2 = 0
25
26 // Let the reaction coordinate at equilibrium for
    the two reaction be X_1 and X_2 respectively
27 // At equilibrium, the moles of the components be
28 // n_CH4_eq = 1 - X_1
29 // n_H2O_eq = 5 - X_1 - X_2
30 // n_CO_eq = X_1 - X_2
31 // n_H2_eq = 3*X_1 + X_2
32 // n_CO2_eq = X_2
33 // Total moles = 6 + 2*X_1
34
35 // Since the pressure is 1 atm, K = Kp, Ky = K
36 Ky_1 = K_1;
37 Ky_2 = K_2;
38
39 // From reaction (1), we get
40 // Ky_1 = ((X_1-X_2)*(3*X_1+X_2)^(3))/((1-X_1)*(5-
    X_1-X_2)*(6+2*X_1)^(2))
41
42 // From reaction (2), we get
43 // Ky_2 = (X_2*(3*X_1+X_2))/((X_1-X_2)*(5-X_1-X_2))
44
45 // Let us assume a value of X_1
46 X_1 = 0.1;
47
48 fault = 10;
49 while(fault>0.05)
50     deff(' [y]=f(X_2)', 'y= Ky_1-((X_1-X_2)*(3*X_1+X_2
        )^(3))/((1-X_1)*(5-X_1-X_2)*(6+2*X_1)^(2)) ');
51     X_2 = fsolve(0.8,f);

```

```

52     X_2_prime = X_2;
53     def('y)=f1(X_1_prime)', 'y= Ky_2-(X_2_prime*(3*
        X_1_prime+X_2_prime))/((X_1_prime-X_2_prime)
        *(5-X_1_prime-X_2_prime))');
54     X_1_prime = fsolve(0.8,f1);
55     fault=abs(X_1 - X_1_prime);
56     X_1 = X_1 + 0.001;
57 end
58
59 n_CH4 = 1 - X_1;
60 n_H2O = 5 - X_1 - X_2;
61 n_CO = X_1 - X_2;
62 n_H2 = 3*X_1 + X_2;
63 n_CO2 = X_2;
64 Total_moles = 6 + 2*X_1;
65
66 printf(" The equilibrium composition of the
        resulting mixture is given by\n");
67 printf(" n_CH4 = % f mol\n n_H2O = %f mol\n n_CO =
        %f mol\n n_H2 = %f mol and\n n_CO2 = %f mol\n\n",
        n_CH4 ,n_H2O ,n_CO ,n_H2 ,n_CO2);
68 printf(" The total number of moles is %f mol\n",
        Total_moles);

```

Scilab code Exa 17.24 Determination of number of moles

```

1 clear;
2 clc;
3
4 //Example - 17.24
5 //Page number - 631
6 printf("Example - 17.24 and Page number - 631\n\n");
7
8 // Given
9 T = 600 + 273.15; // [K] - Reaction temperature

```

```

10 P = 1; //[atm] - Reaction pressure
11
12 // The Gibbs free energy of formation of various
    species at 873.15 K are
13 delta_G_CH4_RT = -2.82; // delta_G_CH4/(R*T)
14 delta_G_H2O_RT = -29.73; // delta_G_CH4/(R*T)
15 delta_G_CO_RT = -27.51; // delta_G_CH4/(R*T)
16 delta_G_H2_RT = -1.46; // delta_G_CH4/(R*T)
17 delta_G_CO2_RT = -56.68; // delta_G_CH4/(R*T)
18
19 // Initial number of moles of the components are
20 // n_CH4 = 1
21 // n_H2O = 5
22 // n_CO = 0
23 // n_H2 = 0
24 // n_CO2 = 0
25
26 // The del(F)/del(n_i) = 0 equations for CH4 (1),
    H2O (2), CO (3), H2 (4) and CO2 (5) becomes
27 // delta_G_1_T + R*T*log((n_1*P)/n) + lambda_C + 4*
    lambda_H = 0
28 // delta_G_2_T + R*T*log((n_2*P)/n) + 2*lambda_C +
    lambda_O = 0
29 // delta_G_3_T + R*T*log((n_3*P)/n) + lambda_c +
    lambda_O = 0
30 // delta_G_4_T + R*T*log((n_4*P)/n) + 2*lambda_H = 0
31 // delta_G_5_T + R*T*log((n_5*P)/n) + lambda_C + 2*
    lambda_O = 0
32
33 // Where n is the total number of moles in the
    equilibrium reaction mixture. Dividing the above
    equations by R*T we get
34 // delta_G_1_T/(R*T) + log((n_1*P)/n) + lambda_C/(R*
    T) + 4*lambda_H/(R*T) = 0
35 // delta_G_2_T/(R*T) + log((n_2*P)/n) + 2*lambda_C/(
    R*T) + lambda_O/(R*T) = 0
36 // delta_G_3_T/(R*T) + log((n_3*P)/n) + lambda_c/(R*
    T) + lambda_O/(R*T) = 0

```

```

37 // delta_G_4_T/(R*T) + log((n_4*P)/n) + 2*lambda_H/(
    R*T) = 0
38 // delta_G_5_T/(R*T) + log((n_5*P)/n) + lambda_C/(R*
    T) + 2*lambda_O/(R*T) = 0
39
40 // Substituting the values of delta_G_i_T/(R*T) in
    the above equations ,the full set of equations (
    including elemental balance) becomes
41 // -2.82 + log(n_1/n) + lambda_C/(R*T) + 4*lambda_H
    /(R*T) = 0
42 // -29.73 + log(n_2/n) + 2*lambda_H/(R*T) + lambda_O
    /(R*T) = 0
43 // -27.51 + log(n_3/n) + lambda_C/(R*T) + lambda_O/(
    R*T) = 0
44 // -1.46 + log(n_4/n) + 2*lambda_H/(R*T) = 0
45 // -56.68 + log(n_5/n) + lambda_C/(R*T) + 2*lambda_O
    /(R*T) = 0
46
47 // The constraint equations are
48 // n_1 + n_3 + n_5 = 1 // (moles of C in the
    reaction mixture = 1)
49 // 4*n_1 + 2*n_2 + 2*n_4 = 14 // (moles of H in the
    reaction mixture = 14)
50 // n_2 + n_3 + 2*n_5 = 5 // (moles of O in the
    reaction mixture = 5)
51
52 // The total moles are given by
53 // n = n_1 + n_2 + n_3 + n_4 + n_5
54
55 function [f]=solution(x)
56 f(1) = -2.82 + log(x(1)/x(6)) + x(7) + 4*x(8);
57 f(2) = -29.73 + log(x(2)/x(6)) + 2*x(8) + x(9);
58 f(3) = -27.51 + log(x(3)/x(6)) + x(7) + x(9);
59 f(4) = -1.46 + log(x(4)/x(6)) + 2*x(8);
60 f(5) = -56.68 + log(x(5)/x(6)) + x(7) + 2*x(9);
61 f(6) = x(1) + x(3) +x(5) - 1;
62 f(7) = 4*x(1) + 2*x(2) + 2*x(4) - 14;
63 f(8) = x(2) + x(3) +2*x(5) - 5;

```



```

64 f(9) = x(1)+ x(2) + x(3) + x(4) + x(5) - x(6);
65
66 funcprot(0);
67 endfunction
68 x = [0.01,3.5,0.2,3.0,0.5,5.0,2.0,1.0,25.0];
69 y = fsolve(x,solution);
70
71 printf(" n_1 = %f mol\n",y(1));
72 printf(" n_2 = %f mol\n",y(2));
73 printf(" n_3 = %f mol\n",y(3));
74 printf(" n_4 = %f mol\n",y(4));
75 printf(" n_5 = %f mol\n",y(5));
76 printf(" n = %f mol\n",y(6));
77 printf(" lambda_C/RT = %f\n",y(7));
78 printf(" lambda_H/RT = %f\n",y(8));
79 printf(" lambda_O/RT = %f\n\n",y(9));
80
81 printf(" The Lagrange multiplier values do not have
any physical significance\n");

```

Chapter 18

Adiabatic Reaction Temperature

Scilab code Exa 18.1 Calculation of heat transfer

```
1 clear;
2 clc;
3
4 //Example - 18.1
5 //Page number - 650
6 printf("Example - 18.1 and Page number - 648\n\n");
7
8 // Given
9
10 T_1 = 298.15; // [K] - Standard reaction temperature
11 T_2 = 500; // [K] - Reaction temperature
12 P = 1; // [atm] - Pressure
13
14 a_CO2 = 5.316;
15 a_O2 = 6.085;
16 a_N2 = 6.903;
17 a_H2O = 7.700;
18 a_C3H8 = -0.966;
19 b_CO2 = 1.4285*10^(-2);
```

```

20 b_O2 = 0.3631*10^(-2);
21 b_N2 = -0.03753*10^(-2);
22 b_H2O = 0.04594*10^(-2);
23 b_C3H8 = 7.279*10^(-2);
24 c_CO2 = -0.8362*10^(-5);
25 c_O2 = -0.1709*10^(-5);
26 c_N2 = 0.1930*10^(-5);
27 c_H2O = 0.2521*10^(-5);
28 c_C3H8 = -3.755*10^(-5);
29 d_CO2 = 1.784*10^(-9);
30 d_O2 = 0.3133*10^(-9);
31 d_N2 = -0.6861*10^(-9);
32 d_H2O = -0.8587*10^(-9);
33 d_C3H8 = 7.580*10^(-9);
34
35 // The standard enthalpy of formation at 298.15 K is
    given by
36 delta_H_for_CO2 = -94.052;//[kcal/mol]
37 delta_H_for_C3H8 = -24.820;//[kcal/mol]
38 delta_H_for_H2O = -57.7979;//[kcal/mol]
39
40 // The reaction with stoichiometric amount of air is
41 // C3H8 + 5(O2 + 3.7N2) - 3CO2 + 4H2O + 18.8N2
42
43 // The reaction with 100% excess air is
44 // C3H8 + 10(O2 + 3.7N2) - 3CO2 + 4H2O + 5O2 + 37.6
    N2
45
46 // The standard enthalpy change of reaction at
    298.15 K
47 delta_H_rkn_298 = 3*delta_H_for_CO2 + 4*
    delta_H_for_H2O - delta_H_for_C3H8;
48
49 // For exit stream
50 sum_ai_ni = 3*a_CO2 + 4*a_H2O + 5*a_O2 + 37.6*a_N2;
51 sum_bi_ni = 3*b_CO2 + 4*b_H2O + 5*b_O2 + 37.6*b_N2;
52 sum_ci_ni = 3*c_CO2 + 4*c_H2O + 5*c_O2 + 37.6*c_N2;
53 sum_di_ni = 3*d_CO2 + 4*d_H2O + 5*d_O2 + 37.6*d_N2;

```

```

54
55
56 // To raise the exit species from 298.15 to 500 K
    the enthalpy change is
57 delta_H_rkn = integrate('sum_ai_ni+sum_bi_ni*T+
    sum_ci_ni*T^(2)+sum_di_ni*T^(3)', 'T', T_1, T_2); //[
    cal]
58 delta_H_rkn = delta_H_rkn*10^(-3); //[kcal]
59
60 // Therefore per mole of fuel the heat exchange is
61 // Q = Heat exchange in step 1 + Heat exchange in
    step 2
62 Q = delta_H_rkn_298 + delta_H_rkn;
63
64 printf("    The heat transfer from the combustion
    chamber per mole of fuel is %f kcal (per mol of
    C3H8)", Q);

```

Scilab code Exa 18.2 Calculation of adiabatic flame temperature

```

1 clear;
2 clc;
3
4 //Example - 18.2
5 //Page number - 650
6 printf("Example - 18.2 and Page number - 650\n\n");
7
8 // Given
9
10 T_1 = 298.15; //[K] - Standard reaction temperature
11
12 a_CO2 = 5.316;
13 a_H2O = 7.700;
14 a_O2 = 6.085;
15 a_C2H6 = 1.648;

```

```

16 b_CO2 = 1.4285*10^(-2);
17 b_H2O = 0.04595*10^(-2);
18 b_O2 = 0.3631*10^(-2);
19 b_C2H6 = 4.124*10^(-2);
20 c_CO2 = -0.8362*10^(-5);
21 c_H2O = 0.2521*10^(-5);
22 c_O2 = -0.1709*10^(-5);
23 c_C2H6 = -1.530*10^(-5);
24 d_CO2 = 1.784*10^(-9);
25 d_H2O = -0.8587*10^(-9);
26 d_O2 = 0.3133*10^(-9);
27 d_C2H6 = 1.740*10^(-9);
28
29 // The standard enthalpy of formation at 298.15 K is
    given by
30 delta_H_for_CO2 = -94.052; //[kcal/mol]
31 delta_H_for_C2H6 = -20.236; //[kcal/mol]
32 delta_H_for_H2O = -57.7979; //[kcal/mol]
33
34 // The reaction with stoichiometric amount of air is
35 // C2H6 + (7/2)O2 - 2CO2 + 3H2O
36
37 // The reaction with 4 mol of O2 and 10 mol CO2 is
38 // C2H6 + 4O2 + 10CO2 - 12H2O + 3H2O + 0.5O2
39 // The product consists of 12 mol of CO2, 3 mol of
    water vapour and 0.5 mol of oxygen
40 delta_H_rkn_298 = 2*delta_H_for_CO2 + 3*
    delta_H_for_H2O - delta_H_for_C2H6; //[kcal]
41 delta_H_rkn_298 = delta_H_rkn_298*10^(3); //[cal]
42
43 // For exit stream
44 sum_ai_ni = 12*a_CO2 + 3*a_H2O + 0.5*a_O2;
45 sum_bi_ni = 12*b_CO2 + 3*b_H2O + 0.5*b_O2;
46 sum_ci_ni = 12*c_CO2 + 3*c_H2O + 0.5*c_O2;
47 sum_di_ni = 12*d_CO2 + 3*d_H2O + 0.5*d_O2;
48
49 // From energy balance equation we get
50 // delta_H_rkn_298 + sum_ai_ni*(T_2 - T_1) + (

```

```

    sum_bi_ni/2)*(T_2^(2) - T_1^(2)) + (sum_ci_ni/3)
    *(T_2^(3) - T_1^(3)) + (sum_di_ni/4)*(T_2^(4) -
    T_1^(4))
51 // Solving above equation for T_2
52 deff(' [y]=f(T_2)', 'y=delta_H_rkn_298 +sum_ai_ni*(T_2
    -T_1)+(sum_bi_ni/2)*(T_2^(2)-T_1^(2))+(sum_ci_ni
    /3)*(T_2^(3)-T_1^(3))+(sum_di_ni/4)*(T_2^(4)-T_1
    ^ (4)) ');
53 T_2 = fsolve(-1,f);
54
55 printf(" The adiabatic flame temperature is %f K",
    T_2);

```

Scilab code Exa 18.3 Calculation of mole fraction and average heat capacity

```

1 clear;
2 clc;
3
4 //Example - 18.3
5 //Page number - 651
6 printf("Example - 18.3 and Page number - 651\n\n");
7
8 // Given
9 T_1 = 298.15; // [K] - Standard reaction temperature
10
11 // The reaction with theoretical air is
12 // CH4 + 2(O2 + 3.76N2) - CO2 + 2H2O + 7.52N2
13
14 //(1)
15 n_product = (1 + 2 + 7.52); // Total number of moles
    of product
16 // The mole fraction of water vapour is
17 y_H2O = 2/(n_product);
18 printf(" (1).The mole fraction of water vapour is %f

```

```

    \n\n",y_H20);
19
20 //(2)
21 delta_H_rkn_298 = -730*10^(3);//[J/mol]
22 C = 40;//[J/mol-K] - Average molar heat capacity
23
24 // From energy balance we have
25 // delta_H_rkn_298 + n_product*C(T_2 - T_1) = 0
26 T_2 = - delta_H_rkn_298/(n_product*C) + T_1;//[K]
27 T_max = T_2 - T_1;
28
29 printf(" (2).The maximum temperature rise of the
    exhaust gases is %f K\n",T_max);

```

Scilab code Exa 18.4 Determination of adiabatic flame temperature

```

1 clear;
2 clc;
3
4 //Example - 18.4
5 //Page number - 651
6 printf("Example - 18.4 and Page number - 651\n\n");
7
8 // Given
9 T_1 = 298.15;//[K] - Standard reaction temperature
10
11 // The standard enthalpy of formation at 298.15 K is
    given by
12 delta_H_for_CO2 = -94.052;//[kcal/mol]
13 delta_H_for_C8H18 = -59.780;//[kcal/mol]
14 delta_H_for_H2O = -57.7979;//[kcal/mol]
15
16 a_CO2 = 5.316;
17 a_H2O = 7.700;
18 a_N2 = 6.903;

```

```

19 b_CO2 = 1.4285*10^(-2);
20 b_H2O = 0.04595*10^(-2);
21 b_N2 = -0.03753*10^(-2);
22 c_CO2 = -0.8362*10^(-5);
23 c_H2O = 0.2521*10^(-5);
24 c_N2 = 0.1930*10^(-5);
25 d_CO2 = 1.784*10^(-9);
26 d_H2O = -0.8587*10^(-9);
27 d_N2 = -0.6861*10^(-9);
28
29 //(a)
30 // The reaction with stoichiometric amount of air is
31 // C3H18 + 12.5(O2 + 3.76N2) - 8CO2 + 9H2O + 47N2
32
33 // The standard enthalpy change of reaction at
    298.15 K is
34 delta_H_rkn_298 = 8*delta_H_for_CO2 + 9*
    delta_H_for_H2O - delta_H_for_C8H18;//[kcal]
35 delta_H_rkn_298 = delta_H_rkn_298*10^(3);//[cal]
36
37 // For exit stream
38 sum_ai_ni = 8*a_CO2 + 9*a_H2O + 47*a_N2;
39 sum_bi_ni = 8*b_CO2 + 9*b_H2O + 47*b_N2;
40 sum_ci_ni = 8*c_CO2 + 9*c_H2O + 47*c_N2;
41 sum_di_ni = 8*d_CO2 + 9*d_H2O + 47*d_N2;
42
43 // From energy balance equation we get
44 // delta_H_rkn_298 + sum_ai_ni*(T_2 - T_1) + (
    sum_bi_ni/2)*(T_2^(2) - T_1^(2)) + (sum_ci_ni/3)
    *(T_2^(3) - T_1^(3)) + (sum_di_ni/4)*(T_2^(4) -
    T_1^(4))
45 // Solving above equation for T_2
46 def f('y)=f(T_2)', 'y=delta_H_rkn_298 +sum_ai_ni*(T_2
    -T_1)+(sum_bi_ni/2)*(T_2^(2)-T_1^(2))+(sum_ci_ni
    /3)*(T_2^(3)-T_1^(3))+(sum_di_ni/4)*(T_2^(4)-T_1
    ^ (4))');
47 T_2 = fsolve(-1,f);
48

```



```

49 printf(" (1).The adiabatic flame temperature is %f K
      \n\n",T_2);
50
51 //(2)
52 // The mean standard heat capacity of various
      components over the temperature range from 25 to
      3000 C is
53 Cp_CO2 = 13.91;//[cal/mol-K]
54 Cp_H2O = 10.16;//[cal/mol-K]
55 Cp_O2 = 7.88;//[cal/mol-K]
56 Cp_N2 = 7.45;//[cal/mol-K]
57
58 // From energy balance equation we get
59 // delta_H_rkn_298 + (8*Cp_CO2 + 9*Cp_H2O + 47*Cp_N2
      )*(T_2_prime)
60 T_2_prime = - delta_H_rkn_298/(8*Cp_CO2 + 9*Cp_H2O +
      47*Cp_N2);//[K]
61
62 printf(" (2).The adiabatic flame temperature is %f K
      ",T_2_prime);

```

Scilab code Exa 18.5 Calculation of conversion

```

1 clear;
2 clc;
3
4 //Example - 18.5
5 //Page number - 652
6 printf("Example - 18.5 and Page number - 652\n\n");
7
8 // Given
9 // N2 + 3H2 - 2NH3
10 T_1 = 700;//[K] - Reaction temperature
11 Max_adia_rise = 100;///[K] - Maximum adiabatic rise
      in temperature

```

```

12 T_2 = T_1 + Max_adia_rise; //[K] -
13
14 delta_H_rkn_700 = -94.2; //[kJ] - Standard enthalpy
    of reaction at 700 K
15 delta_H_rkn_700 = delta_H_rkn_700*10^(3); //[J]
16
17 // The mean standard heat capacity of various
    components over the temperature range from 700 to
    800 K is
18 Cp_N2 = 30.0; //[cal/mol-K]
19 Cp_H2 = 28.9; //[cal/mol-K]
20 Cp_NH3 = 49.2; //[cal/mol-K]
21
22 // The energy balance equation is
23 // X*delta_H_rkn_700 + integrate('(sum_ni-Cpi_exit)*
    dT', 'T', T_1, T_2)
24
25 //At exit, let moles of NH3 = (1-X), moles of H2 =
    (3-3X), moles of NH3 = 2X . Therefore we have,
26 // delta_H_rkn_700*X + {(1-X)*Cp_N2 + (3-3X)*Cp_H2 +
    (2X)*Cp_NH3}*(T_2 - T_1)
27 // On simplification we get, 960.3*X = 116.7
28 X = 116.7/960.3;
29
30 printf(" The maximum allowable conversion fraction
    in the reactor is given by, X = %f \n", X);

```

Scilab code Exa 18.6 Calculation of maximum pressure

```

1 clear;
2 clc;
3
4 //Example - 18.6
5 //Page number - 653
6 printf("Example - 18.6 and Page number - 653\n\n");

```

```

7
8 // Given
9 T_1 = 298.15; // [K] - Standard reaction temperature
10 V = 2.0*10^(-3); // [m^(3)] - Volume of calorimeter
11 m = 10; // [g] - Mass of liquid octane
12 Mol_wt = 114; // [g/mol] - Molecular weight of octane
13 n = m/Mol_wt; // [mol] - No of moles of octane
14 R = 8.314; // [J/mol*K] - Universal gas constant
15
16 // The standard enthalpy of formation at 298.15 K is
    given by
17 delta_H_for_CO2 = -94.052; // [kcal/mol]
18 delta_H_for_C8H18 = -59.780; // [kcal/mol]
19 delta_H_for_H2O = -57.7979; // [kcal/mol]
20
21 // The standard molar heat capacity of various
    components in high temperature range from is
    given by
22 // Cp_H2O = 6.970 + 0.3464*10^(-2)*T -
    0.04833*10^(-5)*T^(2);
23 // Cp_O2 = 6.732 + 0.1505*10^(-2)*T -
    0.01791*10^(-5)*T^(2);
24 // Cp_CO2 = 18.036 - 4.474*10^(-5)*T - 158.08/(T
    ^(1/2));
25 // Therefore we have
26 // Sum_ni_Cpi_exit = 249.09 + 0.04*T - 0.547*10^(-5)
    *T^(2) - 1264.64/(T^(1/2))
27
28 // The reaction with stoichiometric amount of oxygen
    is
29 // C8H18 + 12.5O2 - 8CO2 + 9H2O
30
31 // The reaction with 50% excess oxygen is
32 // C8H18 + 18.75O2 - 8CO2 +9H2O + 6.25O2
33
34 // The standard enthalpy change of reaction at
    298.15 K is
35 delta_H_rkn_298 = 8*delta_H_for_CO2 + 9*

```

```

    delta_H_for_H2O - delta_H_for_C8H18;//[kcal]
36 delta_H_rkn_298 = delta_H_rkn_298*10^(3);//[cal]
37
38 // From the energy balance equation we get
39 // delta_H_rkn_298 + integrate('(sum_ni_Cpi_exit)*dT
    ', 'T',T_1,T_2)
40 // delta_H_rkn_298 + 249.09*(T_2 - T_1) + (0.04/2)*
    (T_2^(2) - T_1^(2)) - ((0.547*10^(-5))/3)*(T_2^(3)
    -T_1^(3)) - (1264.64*2)*(T_2^(1/2)-T_1^(1/2))
41 // Solving above equation for T_2
42 def f(' [y]=f(T_2)', 'y=delta_H_rkn_298 + 249.09*(T_2 -
    T_1) + (0.04/2)*(T_2^(2)-T_1^(2)) -
    ((0.547*10^(-5))/3)*(T_2^(3)-T_1^(3)) -
    (1264.64*2)*(T_2^(1/2)-T_1^(1/2))');
43 T_2 = fsolve(1000,f);
44
45 // When 1 mol of octane reacts the final number of
    moles in the calorimeter is 23.25
46 // When n mol of octane reacts the final number of
    moles in the calorimeter is
47 n_total = n*23.25;//[mol]
48
49 // The maximum explosion pressure is calculated when
    no heat is dissipated to the surroundings and
    thus bomb calorimeter attains the adiabatic flame
    temperature
50 // Thus maximum explosion pressure is given by
51 P = (n_total*R*T_2)/V;//[N/m^(2)]
52 P = P*10^(-5);//[bar]
53
54 printf(" The maximum explosion pressure inside the
    bomb calorimeter is %f bar",P);

```

Scilab code Exa 18.7 Calculation of number of moles

```

1  clear;
2  clc;
3
4  //Example - 18.7
5  //Page number - 656
6  printf("Example - 18.7 and Page number - 656\n\n");
7
8  // Given
9  T_1 = 400 + 273.15; // [K]
10 // SO2(g) + 1/2*(O2) (g) - SO3 (g)
11
12 a_SO2 = 6.157;
13 a_SO3 = 3.918;
14 a_O2 = 6.085;
15 a_N2 = 6.903;
16 b_SO2 = 1.384*10^(-2);
17 b_SO3 = 3.483*10^(-2);
18 b_O2 = 0.3631*10^(-2);
19 b_N2 = -0.03753*10^(-2);
20 c_SO2 = -0.9103*10^(-5);
21 c_SO3 = -2.675*10^(-5);
22 c_O2 = -0.01709*10^(-5);
23 c_N2 = 0.1930;
24 d_SO2 = 2.057*10^(-9);
25 d_SO3 = 7.744*10^(-9);
26 d_O2 = 0.3133*10^(-9);
27 d_N2 = -0.6861*10^(-9);
28
29 // At 400 C, from the given expressions
30 delta_H_rkn_T_1 = -22630.14 - 5.2815*T_1 +
    0.9587*10^(-2)*T_1^(2) - 0.5598*10^(-5)*T_1^(3) +
    1.3826*10^(-9)*T_1^(4); // [cal]
31 // This is the standard enthalpy change of reaction
    for 1 mol of SO2 reacted. Since X moles of SO2
    are reactants therefore
32 // delta_H_rkn_T_X (for X moles of SO2 reacted) =
    delta_H_rkn_T_1*X
33

```

```

34 // Let the number of moles at equilibrium be
35 // n_O2 = 9-0.5*X
36 // n_SO2 = 12-X
37 // n_SO3 = X
38 // n_N2 = 79
39 // Total moles at equilibrium = 100-0.5X
40 // Ky = y_SO3/(y_SO2*y_O2^(1/2))
41 // Ky = (X*(100-0.5*X)^(1/2))/((12-X)*(9-0.5*X)
    ^ (1/2))
42 // We know that K = Ky*Kp. Since P = 1 atm,
    therefore Ky = K
43
44 // Now we have to account for the heat required to
    raise 9-0.5*X mol of O2, 12-X mol of SO2, X mol
    of SO3 and 79 mol of N2 from T to ART
45 // sum_ni_Cp_i = (12-X)*(a + b*T + c*T^(2) + d*T^(3))
    ) + (9-0.5*X)*(a + b*T + c*T^(2) + d*T^(3)) + X*(
    a + b*T + c*T^(2) + d*T^(3)) + 79*(a + b*T + c*T^(
    2) + d*T^(3))
46
47 // From energy balance equation we get
48 // delta_H_rkn_T_1 + integrate('sum_ni_Cp_i', 'T', T_1
    ,T)
49 // The above equation on simplification becomes
50 // (673.99-5.2815*X)*(T-T_1) + (16.91+1.9175*X)
    *(10^(-2)/2)*(T^(2)-T_1^(2)) + (2.79-1.6793*X)
    *(10^(-5)/3)*(T^(3)-T_1^(3)) + (-26.70+5.5304*X)
    *(10^(-9)/4)*(T^(4)-T_1^(4)) = delta_H_rkn_T_1
    *X
51
52 // Let us assume a temperature, say
53 T = 800; // [K]
54 fault = 10;
55
56 while(fault>0.01)
57     K = exp(3.87 + (11389.10/T) - 2.6580*log(T) +
        0.4825*10^(-2)*T - 0.1409*10^(-5)*T^(2) +
        0.2320*10^(-9)*T^(3));

```

```

58     def f(' [y]=f(X) ', 'y= K - (X*(100-0.5*X)^(1/2))
        /((12-X)*(9-0.5*X)^(1/2)) ');
59     X1 = fsolve(0.1,f); // X from equilibrium
        equation
60     def f1(' [y]=f1(X) ', 'y= (673.99-5.2815*X)*(T-T_1)
        +(16.91+1.9175*X)*(10^(-2)/2)*(T^(2)-T_1^(2))
        +(2.79-1.6793*X)*(10^(-5)/3)*(T^(3)-T_1^(3))
        +(-26.70+5.5304*X)*(10^(-9)/4)*(T^(4)-T_1^(4))
        )+delta_H_rkn_T_1*X ');
61     X2 = fsolve(1,f1); // X from energy balance
        equation
62     fault = abs(X1-X2);
63     T = T + 0.01;
64 end
65
66 printf(" The moles of SO2 reacted are %f mol\n\n",X1
        );
67 printf(" The adiabatic reaction temperature is %f K\
        n",T);

```
