Scilab Textbook Companion for Mass - Transfer Operations by R. E. Treybal[1]

Created by
Kumar Saurabh
B.Tech
Chemical Engineering
IT BHU
College Teacher
Prakash Kotecha
Cross-Checked by
Mukul R. Kulkarni

August 10, 2013

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

Book Description

Title: Mass - Transfer Operations

Author: R. E. Treybal

Publisher: McGraw - Hill Book Company, Malaysia

Edition: 3

Year: 1980

ISBN: 0-07-065176-0

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

Exa Example (Solved example)

Eqn Equation (Particular equation of the above book)

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

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

Contents

List of Scilab Codes

Chapter 1

The Mass Transfer Operations

Scilab code Exa 1.1 Conversion of Units

```
1  clear;
2  clc;
3
4  // Illustration 1.1
5  // Page: 17
6
7  printf('Illustration 1.1 - Page: 17\n\n');
8
9  // solution
10
11 // Taking conversion factor from table 1.5 (Pg 15)
12 // viscosity: [(lb/ft.h)]*4.134*10^(-4) [kg/m.s] (Pg
       15)
13 // time: [h] = 3600 [s]
14 // Density: [lb/cubic feet]*16.09 = [kg/cubic m] (Pg
       15)
15 // Length: [ft]*0.3048 = [m]
16 N = (2.778*10^(-4))*(30600/(1/(0.3048^(3/2))))
     *((1/(4.134*(10^(-4))*16.019))^0.111)
     *(((1/16.019)/(1/16.019))^0.26);
17 printf('The coeffecient for S.I. Unit is %f',N);
```

Chapter 2

Molecular Diffusion In Fluids

Scilab code Exa 2.1 Steady State equimolal counterdiffusion

```
1  clear;
2  clc;
3
4  // Illustration 2.1
5  // Page: 30
6
7  printf('Illustration 2.1 - Page 30\n\n');
8
9  // solution
10
11 //***Data***//
12 // a = O2 & b = CO
13 Dab = 1.87*10^(-5);//[square m/s]
14 Pt = 10^5;//[N/square m]
15 z = 0.002;//[m]
16 R = 8314;//[Nm/kmol]
17 T = 273;//[K]
18 Pa1 = 13*10^(3);//[N/square m]
19 Pb1 = 10^(5)-13*10^(3);//[N/square m]
20 Pa2 = 6500;//[N/square m]
21 Pb2 = 10^(5)-6500;//[N/square m]
```

```
22 //********//
23
24 // Calculation from Eqn. 2.30
25 Pbm = (Pb1-Pb2)/log(Pb1/Pb2);//[N/square m]
26 Na = Dab*Pt*(Pa1-Pa2)/(R*T*z*Pbm);//[kmol/square m.s
   ]
27 printf('Rate of diffusion of oxygen is %e kmol/
   square m.sec',Na);
```

Scilab code Exa 2.2 Steady state diffusion in multicomponent mixtures

```
1 clear;
2 clc;
3
4 // Illustration2.2
5 // Page: 30
6
7 printf('Illustration 2.2 - Page: 30\n\n');
8
9 // solution
10
11 //***Data***//
12 Pt = 10^5;//[N/square m]
13 z = 0.002;//[m]
14 R = 8314;//[Nm/kmol]
15 T = 273;//[K]
16 //a = O2 b = CH4 c = H2
17 Pa1 = 13*10^(3);//[N/square m]
18 Pb1 = 10^(5)-13*10^(3);//[N/square m]
19 Pa2 = 6500;//[N/square m]
20 Pb2 = 10^(5)-6500;//[N/square m]
21 Dac = 6.99*10^(-5);//[N/square m]
22 Dab = 1.86*10^(-5);//[N/square m]
23 //*******//
24
```

```
25 // Calculation from Eqn. 2.30
26 Pbm = (Pb1-Pb2)/log(Pb1/Pb2);//[N/square m]
27 Yb_prime = 2/(2+1);
28 Yc_prime = 1-Yb_prime;
29 Dam = 1/((Yb_prime/Dab)+(Yc_prime/Dac));//[square m.
   s]
30 Na = Dam*(Pa1-Pa2)*Pt/(R*T*z*Pbm);//[kmol/square m.s
   ]
31 printf('Rate of diffusion is %e kmol/square m.sec',
   Na);
```

Scilab code Exa 2.3 Diffusivity of gases

```
1 clear;
2 clc;
3
4 // Illustration2.3
5 // Page: 32
6
7 printf('Illustration 2.3 - Page: 32\n\n');
8
9 // solution
10
11 //***Data***//
12 // a = C2H5OH b = air
13 Pt = 101.3*10^(3);//[N/square m]
14 T = 273;//[K]
15 //********//
16
17 Ma = 46.07;// [kg/kmol]
18 Mb = 29;// [kg/kmol]
19 //For air from Table 2.2 (Pg 33)
20 Eb_by_k = 78.6;// [K]
21 rb = 0.3711; // [nm]
22 // For C2H5OH using Eqn. 2.38 & 2.39
```

```
23 // From Table 2.3
24 Va = (2*0.0148)+(6*0.0037)+(0.0074);// [cubic m/kmol
     ]
25 Tba = 351.4;// [K]
26 ra = 1.18*(Va^(1/3));//[nm]
27 Ea_by_k = 1.21*Tba;// [K]
28 rab = (ra+rb)/2;// [nm]
29 Eab_by_k = sqrt(Ea_by_k*Eb_by_k);// [K]
30 Collision_value = T/Eab_by_k;
31 //From Fig. 2.5 (Page: 32) f(collision value)
32 Collision_func = 0.595;
33 Dab = (10^(-4)*(1.084-(0.249*sqrt((1/Ma)+(1/Mb))))*T
     ^(3/2)*sqrt((1/Ma)+(1/Mb)))/(Pt*(rab^2)*
     Collision_func);//[square m/s]
34 printf('The diffusivity of ethanol through air at 1
     atm. & 0C is %e square m/s\n',Dab);
35 printf('The observed value (Table 2.1) is
     1.02*10^(-5) square m/s')
```

Scilab code Exa 2.4 Molecular Diffusion in Liquids

```
1 clear;
2 clc;
3
4 // Illustration 2.4
5 // Page: 34
6
7 printf('Illustration 2.4 - Page: 34\n\n');
8
9 // solution
10
11 //***Data****//
12 // a = acetic acid b = H2O
13 z = 0.001;// [m]
14 Dab = 0.95*10^(-9);//[square m/s]
```

```
15 //************//
16
17 Ma = 60.03;// [kg/kmol]
18 Mb = 18.02;// [kg/kmol]
19 //At 17 C & 9% solution
20 density1 = 1012; //[kg/cubic m]
21 Xa1 = (0.09/Ma)/((0.09/Ma)+(0.91/Mb));
22 Xb1 = 1-Xa1;
23 M1 = 1/((0.09/Ma)+(0.91/Mb));// [kg/kmol]
24 //At 17 C & 3% solution
25 density2 = 1003.2; //[kg/cubic m]
26 Xa2 = (0.03/Ma)/((0.03/Ma)+(0.97/Mb));
27 Xb2 = 1-Xa2;
28 M2 = 1/((0.03/Ma)+(0.97/Mb));// [kg/kmol]
29 avg_density_by_M = ((density1/M1)+(density2/M2))/2;
     //[kmol/cubic m]
30 // From Eqn. 2.42
31 Xbm = (Xb2-Xb1)/log(Xb2/Xb1);
32 // From Eqn. 2.41
33 Na = Dab*(avg_density_by_M)*(Xa1-Xa2)/(Xbm*z); //[
     square m/s]
34 printf('The rate of diffusion is %e square m/s',Na);
```

Scilab code Exa 2.5 Diffusivity of Liquids

```
1 clear;
2 clc;
3
4 // Illustration 2.5
5 // Page: 37
6
7 printf('Illustration 2.5 - Page: 37\n\n');
8
9 // solution
10
```

```
11 //***Data****//
12 // a = mannitol b = H2O
13 T = 293; // [K]
14 //*****//
15
16 Mb = 18.02;// [kg/kmol]
17 // From Table 2.3 (Pg 33)
18 Va = (0.0148*6)+(0.0037*14)+(0.0074*6); // [cubic m/
   kmol]
19 viscosity = 0.001005; // [kg/m.s]
20 association_factor = 2.26; // [water as a solvent]
21 Dab = (117.3*10^(-18))*((association_factor*Mb)^0.5)
   *T/(viscosity*Va^0.6); // [square m/s]
22 printf('Diffusivity of mannitol is %e square m/s\n',
   Dab);
23 printf('Observed value is 0.56*10^(-9) square m/s');
```

Scilab code Exa 2.6 Diffusivity of Liquids

```
1 clear;
2 clc;
3
4 // Illustration 2.6
5 // Page: 37
6
7 printf('Illustration 2.6 - Page 37\n\n');
8
9 // solution
10
11 //****Data****//
12 T2 = 70+273;// [K]
13 //**********//
14
15 // a = mannitol b = H2O
16 // From Illustration 2.5 at 20 C
```

```
17 viscosity1 = 1.005*10^(-3); // [kg/m.s]
18 Dab1 = 0.56*10^(-9); //[m^2/s]
19 T1 = 273+20;// [K]
20 // At 70 C
21 viscosity2 = 0.4061*10^(-3); // kg/m.s
22 // Eqn. 2.44 indicates Dab*viscocity/T = constnt
23 Dab2 = Dab1*(T2)*(viscosity1)/(T1*viscosity2);// [
   square m/s]
24 printf('Diffusivity of mannitol at 70 OC is %e
   square/s\n',Dab2);
25 printf('Observed value at 70 OC is 1.56*10^(-9)
   square m/s');
```

Chapter 3

Mass Transfer Coeffecients

Scilab code Exa 3.1 Mass Transfer Coeffecient in Laminar Flow

```
clear;
clc;

// Illustration 3.1
// Page: 53

printf('Illustration 3.1 - Page: 53\n\n');

// solution

//****Data*****//
// a = CO2 b = H2O
Ca0 = 0;//[kmol/cubic m]
Cai = 0.0336;//[kmol/cubic m]
Dab = 1.96*10^(-9);// [square m/s]
//*******//

density = 998;// [kg/cubic m]
viscosity = 8.94*10^(-4);//[kg/m.s]
rate = 0.05;//[kg/m.s] mass flow rate of liquid
L = 1;//[m]
```

```
22 g = 9.81;//[m/square s]
23 // From Eqn. 3.10
24 del = ((3*viscosity*rate)/((density^2)*g))^(1/3);//
      [m]
25 Re = 4*rate/viscosity;
26 // Flow comes out to be laminar
27 // From Eqn. 3.19
28 Kl_avg = ((6*Dab*rate)/(3.141*density*del*L))^(1/2);
      //[kmol/square m.s.(kmol/cubic m)]
29 bulk_avg_velocity = rate/(density*del);//[m/s]
30 // At the top: Cai-Ca = Cai_Ca0 = Cai
31 //At the bottom: Cai-Cal
32 // From Eqn. 3.21 & 3.22
33 Cal = Cai*(1-(1/(exp(Kl_avg/(bulk_avg_velocity*del))
      )));// [kmol/cubic m]
34 rate_absorption = bulk_avg_velocity*del*(Cal-Ca0);//
       [kmol/s].(m of width)
35 printf('The rate of absorption is %e',
      rate_absorption);
36 // The actual value may be substantially larger.
```

Scilab code Exa 3.2 Eddy Diffusion

```
1 clear;
2 clc;
3
4 // Illustration 3.2
5 // Page: 56
6
7 printf('Illustration 3.2 - Page: 56\n\n');
8
9 // solution
10
11 //***Data****//
12 d = 0.025;// [m]
```

```
13 avg_velocity = 3;// [m/s]
14 viscosity = 8.937*10^(-4);// [kg/m.s]
15 density = 997;// [kg/m^3]
16 //*********//
17
18 kinematic_viscosity = viscosity/density;// [square m
     /s]
19 Re = d*avg_velocity*density/viscosity;
20 // Reynold's number comes out to be 83670
21 // At this Reynold's number fanning factor = 0.0047
22 f = 0.0047;
23 L = 1;// [m]
24 press_drop = 2*density*f*L*(avg_velocity^2)/(d);// [
     N/square m]
25 P = 3.141*(d^2)*avg_velocity*press_drop/4;// [N.m/s]
      for 1m pipe
26 m = 3.141*(d^2)*L*density/4;
27 // From Eqn. 3.24
28 Ld = ((kinematic_viscosity^3)*m/P)^(1/4);// [m]
29 // From Eqn. 3.25
30 Ud = (kinematic_viscosity*P/m)^(1/4);// [m/s]
31 printf('Velocity of small eddies is %f m/s\n',Ud);
32 printf('Length scale of small eddies is %e m',Ld);
```

Scilab code Exa 3.3 Mass Heat And Momentum Transfer Analogies

```
1 clear;
2 clc;
3
4 // Illustration 3.3
5 // Page: 69
6
7 printf('Illustration 3.3 - Page: 69\n\n');
8
9 // solution
```

```
10
11 // Heat transfer analog to Eqn. 3.12
12 // The Eqn. remains the same with the dimensionless
      conc. ratio replaced by ((tl-to)/(ti-to))
13
14 // The dimensionless group:
15 // eta = 2*Dab*L/(3*del^2*velocity);
16 // eta = (2/3)*(Dab/(del*velocity))*(L/del);
17 // Ped = Peclet no. for mass transfer
18 // eta = (2/3)*(1/Ped)*(L/del);
19
20 // For heat transfer is replaced by
21 // Peh = Peclet no. for heat transfer
22 // eta = (2/3)*(1/Peh)*(L/del);
23 // eta = (2/3)*(alpha/(del*velocity))*(L/del);
24 // eta = (2*alpha*L)/(3*del^2*velocity);
25 printf('Heat transfer analog to Eqn. 3.21 is eta =
      (2*alpha*L)/(3*del^2*velocity)');
```

Scilab code Exa 3.4 Mass Heat And Momentum Transfer Analogies

```
1 clear;
2 clc;
3
4 // Illustration 3.4
5 // Page: 69
6
7 printf('Illustration 3.4 - Page: 69\n\n');
8
9 // solution
10
11 //***Data****//
12 // a = UF6 b = air
13 // The average heat transfer coefficient: Nu_avg =
      0.43+0.532(Re^0.5)(Pr^0.31)
```

```
14 // The analogus expression for mass transfer
      coefficient: Sh_avg = 0.43+0.532(Re^0.5)(Sc^0.31)
15 d = 0.006;// [m]
16 velocity = 3;// [m/s]
17 surf_temp = 43;// [C]
18 bulk_temp = 60;// [C]
19 avg_temp = (surf_temp+bulk_temp)/2; //[C]
20 density = 4.10;// [kg/cubic m]
21 viscosity = 2.7*10^(-5);// [kg/m.s]
22 Dab = 9.04*10^(-6);// [square m/s]
23 press = 53.32;// [kN/square m]
24 tot_press = 101.33;// [kN/square m]
25 //******//
26
27 avg_press = press/2; // [kN/square m]
28 Xa = avg_press/tot_press;
29 Xb = 1-Xa;
30 Re = d*velocity*density/viscosity;
31 Sc = viscosity/(density*Dab);
32 Sh_avg = 0.43+(0.532*(2733^0.5)*(0.728^0.5));
33 c = 273.2/(22.41*(273.2+avg_temp));// [kmol/cubic m]
34 F_avg = Sh_avg*c*Dab/d;//[kmol/cubic m]
35 Nb = 0;
36 Ca1_by_C = press/tot_press;
37 Ca2_by_C = 0;
38 Flux_a = 1;
39 // Using Eqn. 3.1
40 Na = Flux_a*F_avg*log((Flux_a-Ca2_by_C)/(Flux_a-
      Ca1_by_C));//[kmol UF6/square m.s]
41 printf('Rate of sublimation is %e kmol UF6/square m.
      s',Na);
```

Scilab code Exa 3.5 Flux Variation with Concentration

```
1 clear;
```

```
clc;

// Illustration 3.5
// Page: 73

printf('Illustration 3.5 - Page: 73\n\n');

// solution

//****Data****//
velocity = 15;// [m/s]
G = 21.3;// [kg/square m.s]
//******//

// Since the experimental data do not include the
   effects of changing Prandtl number.

// Jh = (h/(Cp*density*viscosity)) = (h/Cp*G)*(Pr
   ^(2/3)) = Shi(Re);

// Shi(Re) must be compatible with 21.3*(G^0.6);
// Let Shi(Re) = b*(Re^n);
// Re = (l*G)/viscosity;

// h = (Cp*G/(Pr^(2/3)))*b*(Re^n);
// h = (Cp*G/(Pr^(2/3)))*b*((l*b/viscosity)^n) =
   21.3*(G^0.6);

n = 0.6-1;
// b = 21.3*((Pr^(2/3))/Cp)*((l/viscosity)^(-n));

// Using data for air at 38 C & 1 std atm.
Cp1 = 1002;// [kJ/kg.K]
viscosity1 = 1.85*10^(-5);//[kg/m.s]
k1 = 0.0273;//[W/m.K]
Pr1 = (Cp1*viscosity1)/k1;
b_prime = 21.3*(Pr1^(2/3)/Cp1)*((1/viscosity1)^0.4);
// b = b_prime*l^(0.4);
```

```
// Jh = (h/(Cp*G))*Pr^(2/3) = b_prime*((1/Re)^(0.4))
    = Shi(Re);

// The heat mass transfer analogy will be used to
   estimate the mass transfer coefficient. (Jd = Jh)

// Jd = (KG*Pbm*Mav*Sc^(2/3))/(density*viscosity) =
   Shi(Re) = b_prime*((1/Re)^0.4);

// KG*Pbm = F = (b_prime*density*viscosity)/(Re^0.4*
   Mav*Sc^(2/3)) = (b_prime*(density*velocity)^0.6*(
   viscosity^0.4))/(Mav*Sc^(2/3));

// For H2-H20, 38 C, 1std atm
viscosity2 = 9*10^(-6);// [kg/m.s]
density2 = 0.0794;// [kg/cubic m]
Dab = 7.75*10^(-5);// [square m/s]
Sc = viscosity2/(density2*Dab);

// Assuming desity, Molecular weight and viscosity
   of the gas are essentially those of H2

Mav = 2.02;// [kg/kmol]
F = (b_prime*(density2*velocity)^0.6*(viscosity2
   ^0.4))/(Mav*Sc^(2/3));// [kmol/square m.s]
printf('The required mass transfer: %f kmol/square m
   .s',F);
```

Scilab code Exa 3.6 Calculation of Bed depth

```
clear;
clc;

// Illustration 3.6
// Page: 77
```

```
6
7  printf('Illustration 3.6 - Page: 77\n\n');
8
9  // solution
10
11 //***Data***//
12 Dp = 0.0125;// [m]
13 viscosity = 2.4*10^(-5);// [kg/m.s]
14 Sc = 2;
15 E = 0.3;
16 Go = (2*10^(-3))/0.1;// molar superficial mass
      velocity [kmol/square m.s]
17 //********//
18
19 // a = CO b = Ni(CO)4
20 // Nb = -(Na/4);
21 Flux_a = 4/3;
22 Ca2_by_C = 0;// At the metal interface
23 // Ca1_by_C = Ya //mole fraction of CO in the bulk
24
25 // Eqn. 3.1 becomes: Na = (4/3)*F*log((4/3)/((4/3)-
      Ya));
26
27 // Let G = kmol gas/(square m bed cross section).s
28 // a = specific metal surface
29 // z = depth
30 // Therefore, Na = -(diff(Ya*G))/(a*diff(z));// [
      kmol/((square m metal surface).s)];
31 // For each kmol of CO consumed, (1/4)kmol Ni(CO)4
      forms, representing a loss of (3/4) kmol per kmol
       of CO consumed.
32 // The CO consumed through bed depth dz is therefore
       (Go-G)(4/3) kmol;
33 // Ya = (Go-(Go-G)*(4/3))/G;
34 // G = Go/(4-(3*Ya));
35 // diff(YaG) = ((4*Go)/(4-3*Ya)^2)*diff(Ya);
36
37 // Substituting in Eqn. 3.64
```

```
38 // -(4*Go/((4-3*Ya)^2*a))*(diff(Ya)/diff(z)) = (4/3)
   *F*log(4/(4-3*Ya));
39
40 // At depth z:
41 // Mass velocity of CO = (Go-(Go-G)/(4/3))*28;
42 // Mass velocity of Ni(CO)4 = ((Go-G)*(1/3))*170.7;
43 // G_prime = 47.6*Go-19.6G; // total mass velocity [
   kg/square m.s]
44 // Substituting G leads to:
45 // G_prime = Go*(47.6-19.6*(4-3*Ya));// [kg/m.s]
46 // Re = (Dp*G')/viscosity
47
48 // With Go = 0.002 kmol/square m.s & Ya in the range
    1-0.005, the range of Re is 292-444;
49 // From table 3.3:
50 // Jd = (F/G)*(Sc^(2/3)) = (2.06/E)*Re^(-0.575);
51 // F = (2.06/E*(Sc)^(2/3))*(Go/(4-3*Ya))*Re^(-0.575)
   ;
52
53 a = 6*(1-E)/Dp;
54
55 // Result after arrangement:
56 Z = integrate('-((4*Go)/((4-(3*Ya))^2*a))*(3/4)*(E*(
   Sc^(2/3))*(4-(3*Ya))/(2.06*Go)*(1/log(4/(4-(3*Ya)
   ))))*(((Dp/viscosity)*(Go*(47.6-(19.6/(4-(3*Ya)))
   )))^0.575)','Ya',1,0.005);// [m]
57 printf('The bed depth required to reduce the CO
   content to 0.005 is %f m', Z);
```

Scilab code Exa 3.7 Local rate of condensation

```
1 clear;
2 clc;
3
4 // Illustration 3.7
```

```
5  // Page: 80
6
7  printf('Illustration 3.7 - Page: 80\n\n');
8
9  // solution
10
11 //****Data*****//
12 // a = water b = air
13 out_dia = 0.0254;// [m]
14 wall_thick = 0.00165;// [m]
15 avg_velocity = 4.6;// [m/s]
16 T1 = 66;// [C]
17 P = 1;// [atm]
18 Pa1 = 0.24;// [atm]
19 k1 = 11400;// [W/(square m.K)]
20 T2 = 24;// [C]
21 k2 = 570;// [W/square m.K]
22 k_Cu = 381;// [w/square m.K]
23 //******//
24
25 // For the metal tube
26 int_dia = out_dia-(2*wall_thick);// [m]
27 avg_dia = (out_dia+int_dia)/2;// [mm]
28 Nb = 0;
29 Flux_a = 1;
30 Ya1 = 0.24;
31 Yb1 = 1-Ya1;
32 Mav = (Ya1*18.02)+(Yb1*29);// [kg/kmol]
33 density = (Mav/22.41)*(273/(273+T1));// [kg/cubic m]
34 viscosity = 1.75*10^(-5);// [kg/m.s]
35 Cpa = 1880;// [J/kg.K]
36 Cpmix = 1145;// [J/kg.K]
37 Sc = 0.6;
38 Pr = 0.75;
39 G_prime = avg_velocity*density;// [kg/square m.s]
40 G = G_prime/Mav;// [kmol/square m.s]
41 Re = avg_dia*G_prime/viscosity;
42 // From Table 3.3:
```

```
// Jd = Std*Sc^(2/3) = (F/G)*Sc^(2/3) = 0.023*Re^(-0.17);
Jd = 0.023*Re^(-0.17);
F = (0.023*G)*(Re^(-0.17)/Sc^(2/3));

// The heat transfer coeffecient in the absence of mass transfer will be estimated through Jd = Jh
// Jh = Sth*Pr^(2/3) = (h/Cp*G_prime)*(Pr^(2/3)) = Jd
h = Jd*Cpmix*G_prime/(Pr^(2/3));

U = 1/((1/k1)+((wall_thick/k_Cu)*(int_dia/avg_dia))+((1/k2)*(int_dia/out_dia)));// W/square m.K

// Using Eqn. 3.70 & 3.71 with Nb = 0
// Qt = (Na*18.02*Cpa/1-exp(-(Na*18.02*Cpa/h)))*(T1-Ti)+(Lambda_a*Na);
// Qt = 618*(Ti-T2);
// Using Eqn. 3.67, with Nb = 0, Cai/C = pai, Ca1/C = Ya1 = 0.24;
// Na = F*log(((Flux_a)-(pai))/((Flux_a)-(Ya1));

// Solving above three Eqn. simultaneously:
Ti = 42.2;// [C]
pai = 0.0806;// [atm]
Lambda_a = 43.4*10^6;// [J/kmol]
Na = F*log(((Flux_a)-(pai))/((Flux_a)-(Ya1)));// [kmol/square m.s]
Qt1 = 618*(Ti-T2);// [W/square m]
Qt2 = ((Na*18.02*Cpa/(1-exp(-(Na*18.02*Cpa/h))))*(T1-Ti))+(Lambda_a*Na);// [W/square m]

// since the value of Qt1 & Qt2 are relatively close
printf('The local rate of condensation of water is %e kmol/square m.s',Na);
```

Scilab code Exa 3.8 Simultaneous Heat and Mass Transfer

```
clear;
clc;

// Illustration 3.8
// Page: 81
printf('Illustration 3.8 - Page: 81\n\n');
printf('Illustration 3.8 (a)\n\n');

// Solution (a)

//***Data****//
// a = water b = air
Nb = 0;
h = 1100;// [W/square m]
//*****//

Ma = 18.02;// [kg/kmol]
Cpa = 2090;// [J/kg.K]
T1 = 600;// [C]
Ti = 260;// [C]
// The positive dirn. is taken to be from the bulk
   gas to the surface.
Has = 2.684*(10^6);// enthapy of saturated steam at
   1.2 std atm, rel. to the liquid at 0 C in [J/kg]
Hai = 2.994*(10^6);// enthalpy of steam at 1 std atm
   , 260 C in [J/kg]

// Radiation contributions to the heat transfer from
     the gas to the surface are negligible. Eqn.
   3.70 reduces to
Na = -((h/(Ma*Cpa))*log(1-((Cpa*(T1-Ti))/(Has-Hai)))
   );// [kmol/square m.s]
```

```
27 printf('The rate of steam flow reqd. is %f kmol/
     square m.s\n\n',Na);
28 // negative sign indicates that the mass flux is
     into the gas
29
30 printf('Illustration 3.8 (b)\n\n');
31
32 // Solution (b)
33
34 //***Data****//
35 // a = water b = air
36 h = 572;// [W/square m]
37 T1 = 25;// [C]
38 //******//
39
40 Ti = 260;// [C]
41 // The positive dirn. is taken to be from the bulk
     gas to the surface.
42 Has = 1.047*10^(5);// enthapy of saturated steam
     at 1.2 std atm, rel. to the liquid at 0 C in [J/
     kg]
43 Hai = 2.994*(10^6);// enthalpy of steam at 1 std
     atm, 260 C in [J/kg]
44
45 // Radiation contributions to the heat transfer from
      the gas to the surface are negligible. Eqn.
     3.70 reduces to
46 Na = -((h/(Ma*Cpa))*log(1-((Cpa*(T1-Ti))/(Has-Hai)
     )));// [kmol/square m.s]
47 printf('The rate of steam flow reqd. is %f kmol/
     square m.s',Na);
48 // negative sign indicates that the mass flux is
     into
```

Chapter 4

Diffusion In Solids

Scilab code Exa 4.1 Ficks Law Diffusion

```
1 clear;
2 clc;
3
4 // Illustration 4.1
5 // Page: 89
6
7 printf('Illustration 4.1 - Page: 89\n\n');
8
9 // solution
10
11 //***Data****//
12 P = 2;// [atm]
13 a1 = 0.025;// [m]
14 a2 = 0.050;// [m]
15 solub = 0.053*P;// [cubic m H2 (STP)/(cubic m rubber
     )]
16 Ca1 = solub/22.41;// inner surface of the pipe
17 Ca2 = 0;// resistance to difusion of H2 away from
     the surface is negligible.
18 Da = 1.8*10^(-10);// [square m/s]
19 l = 1;// [m]
```

```
20 //********//
21
22 z = (a2-a1)/2;// [m]
23 // Using Eqn. 4.4
24 Sav = (2*(%pi)*l*(a2-a1))/(2*log(a2/a1));// [square
   m]
25 // Using Eqn. 4.3
26 w = (Da*Sav*(Ca1-Ca2))/z;// [kmol H2/s for 1m length
   ]
27 w = w*2.02*10^3*3600;// [g H2/m.h]
28 printf('The rate of loss of H2 by diffusion per m of
    pipe length: %e g H2/m.h',w);
```

Scilab code Exa 4.2 Unsteady State Diffusion

```
1 clear;
2 clc;
3
4 // Illustration 4.2
5 // Page: 92
6
7 printf('Illustration 4.2 - Page: 92\n\n');
8 printf('Illustration 4.2 (a)\n\n');
9
10 // solution (a)
11
12 // Given
13 a = 3/2;// [cm]
14 thetha = 68*3600;// [s]
15 // Ca can e calculated in terms of g/100 cubic cm
16 Cao = 5;// [g/100 cubic cm]
17 Ca_thetha = 3;// [g/100 cubic cm]
18 Ca_Inf = 0;// [g/100 cubic cm]
19 //**********//
20
```

```
21 E = (Ca_thetha-Ca_Inf)/(Cao-Ca_Inf);
22 // E = 0.6;
23 // From Fig. 4.2 (Pg 91): For diffusion from only
      one exposed surface D*thetha/(4*a^2) = 0.128
24 D = 0.128*4*(a^2)/thetha;// [square cm/s]
25 D = D*10^(-4);// [square m/s]
26 printf('Diffusivity of urea in gel from only one
      exposed durface: %e square m/s\n\n',D);
27
28 printf('Illustration 4.2 (b)\n\n');
29
30 // Solution (b)
31
32 //****Data****//
33 // Ca can e calculated in terms of g/100 cubic cm
34 Cao = 5;// [g/100 cubic cm]
35 Ca_thetha = 1;// [g/100 cubic cm]
36 Ca_Inf = 0;// [g/100 cubic cm]
37 //*********//
38
39 E = (Ca_thetha-Ca_Inf)/(Cao-Ca_Inf);
40 // E = 0.2;
41 // From Fig. 4.2 (Pg 91): For diffusion from only
      one exposed surface D*thetha/(4*a^2) = 0.568
42 D = 4.70*10^(-6);// From Illusration 4.2(a) [square
      cm/s]
43 thetha = 0.568*4*a^2/D;// [s]
44 thetha = thetha/3600;// [h]
45 printf('The time taken for the avg. conc. to fall to
       1g/100 cubic cm is:%f h\n\n',thetha);
46
47 printf('Illustration 4.2 (c)\n\n');
48
49 // solution (c)
50
51 //****Data*****//
52 Cao = 5;// [g/100 cubic cm]
53 Ca_thetha = 1;// [g/100 cubic cm]
```

```
54 Ca_Inf = 0;// [g/100 cubic cm]
55 //*******//
56
57 E = (Ca_thetha-Ca_Inf)/(Cao-Ca_Inf);
58 // E = 0.2;
59 // From Fig. 4.2: For diffusion from two opposite
      exposed surface D*thetha/(a^2) = 0.568
60 D = 4.70*10^(-6);// From Illusration 4.2(a) [square
      cm/s]
61 thetha = 0.568*(a^2)/D;// [s]
62 thetha = thetha/3600;// [h]
63 printf('The time taken for the avg. conc. to fall to
       1g/100 cubic cm is:%f h',thetha);
```

Scilab code Exa 4.3 Diffusion through Polymers

```
1 clear;
2 clc;
3
4 // Illustration 4.3
5 // Page: 94
6
7 printf('Illustration 4.3 - Page: 94\n\n');
8
9 // solution
10
11 //****Data****//
12 z = 0.1;// [cm]
13 pa1 = 1;// [cmHg]
14 pa2 = 0;// [cmHg]
15 Da = 1.1*10^(-10)*10^4;// [square cm/s]
16 //***********//
17
18 // Solubility coeffecient in terms of Hg
19 Sa = 0.90/76;// [cubic cm gas (STP)/cubic cm.cmHg]
```

```
20 // Using Eqn. 4.15
21 Va = (Da*Sa*(pa1-pa2))/z;// [cubic cm(STP)/square cm
      .s]
22 // Using Eqn. 4.16
23 P = Da*Sa;// [cubic cm gas (STP)/square cm.s.(cmHg/
      cm)]
24 printf('The rate of diffusion of CO is:%e cubic cm(
      STP)/square cm.s\n',Va);
25 printf('The permeability of the membrane is %e cubic
       cm gas (STP)/square cm.s.(cmHg/cm)',P)
```

Scilab code Exa 4.4 Diffusion in Porous Solids

```
1  clear;
2  clc;
3
4  // Illustration 4.4
5  // Page: 96
6
7  printf('Illustration 4.4 - Page: 96\n\n');
8
9  // solution
10
11 //****Data****//
12 a = 0.005;// [m]
13 // For the KCl diffusion
14 Dab1 = 1.84*10^(-9);// [square m/s]
15 thetha = 4.75*3600;// [s]
16 Ca_Inf = 0;
17 // For K2CrO4 diffusion
18 Cao = 0.28;// [g/cubic cm]
19 Ca_Inf = 0.002;// [g/cubic cm]
20 Dab2 = 1.14*10^(-9);// [square m/s]
21 //*******//
22
```

```
E = 0.1;// For 90% removal of KCl
// From Fig. 4.2 (Pg 91): Deff*thetha/a^2 = 0.18
Deff = 0.18*a^2/thetha;// [square m/s]
Dab_by_Deff = Dab1/Deff;
Ca_thetha = 0.1*0.28;// [g/cubic cm]
Es = (Ca_thetha-Ca_Inf)/(Cao-Ca_Inf);
// From Fig. 4.2 (Pg 91): Deff*thetha/a^2 = 0.30
Deff = Dab2/Dab_by_Deff;// [square m/s]
thetha = 0.3*a^2/Deff;// [s]
thetha = thetha/3600;// [h]
printf('The time reqd. is:%f h',thetha);
```

Scilab code Exa 4.5 Diffusion in Porous Solids

```
clear;
clc;

// Illustration 4.5
// Page: 98
printf('Illustration 4.5 - Page: 98\n\n');
printf('Illustration 4.5 (a)\n\n');

// solution (a)

//****Data****//
// a = H2 b = N2
Dab_eff = 5.3*10^(-6);// [square m/s]
Dkb_eff = 1.17*10^(-5);// [square m/s]
Dab = 7.63*10^(-5);// [square m/s]
//*******//

R = 8314;//[Nm/kmol]
Mb = 2.02;// [kg/kmol]
T = 293;// [K]
Dtrue_by_Deff = Dab/Dab_eff;
```

```
// Since the ratio is strictly a matter of the
   geometry of the solid.
Dkb = Dkb_eff*Dtrue_by_Deff;// [square m/s]
// From Eqn. 4.20
d = 3*Dkb*((%pi*Mb)/(8*R*T))^0.5;// [m]
printf('The equivalent pore diameter is: %e m\n\n',d
   );

printf('Illustration 4.5 (b)\n\n');

// Solution (b)

//****Data*****//
// a = O2 b = N2 c = H2
Ya1 = 0.8;
Ya2 = 0.2;
Pt = 10133;// [N/square m]
z = 0.002;// [m]
T = 293;// [K]
//*******//

// From Table 2.1 (Pg 31):
Dab = 1.81*10^(-5);// [square m/s] at STP
Dkc = 1.684*10^(-4);// [square m/s] From
   Illustration 4.5(a)
Mc = 2.02;// [kg/kmol]
Ma = 32;// [kg/kmol]
Mb = 28.02;// [kg/kmol]
Dab = Dab*(1/0.1)*((293/273)^1.5);// [square m/s] at
    0.1 atm & 20 C
DabEff = Dab/14.4;// [square m/s] From Illustration
   4.5(a)
Dka = Dkc*((Mc/Ma)^0.5);// [square m/s]
DkaEff = Dka/14.4;// [square m/s]
Nb_by_Na = -(Ma/Mb)^0.5;
// Na/(Na+Nb) = 1/(1+(Nb/Na))
Na_by_NaSumNb = 1/(1+(Nb_by_Na));
DabEff_by_DkaEff = DabEff/DkaEff;
```

```
55 // By Eqn. 4.23
56 Na = (Na_by_NaSumNb)*(DabEff*Pt/(R*T*z))*log((((
      Na_by_NaSumNb)*(1+DabEff_by_DkaEff))-Ya2)/(((
      Na_by_NaSumNb)*(1+DabEff_by_DkaEff))-Ya1));// [
      kmol/square m.s]
57 Nb = Na*(Nb_by_Na);// [kmol/square m.s]
58 printf("Diffusion flux of O2 is %e kmol/square m.s\n
      ",Na);
59 printf("Diffusion flux of N2 is %e kmol/square m.s\n
      ",Nb);
```

Scilab code Exa 4.6 Hydrodynamic flow of gases

```
1  clear;
2  clc;
3
4  // Illustration 4.6
5  // Page: 100
6
7  printf('Illustration 4.6 - Page: 100\n\n');
8
9  // solution
10
11 //***Data***//
12 // a = N2
13 // For N2 at 300K
14 viscosity1 = 1.8*10^(-5);// [kg/m.s]
15 Pt1 = 10133;// [N/square m.sec]
16 T = 300;// [K]
17 z = 0.0254;// [m]
18 T2 = 393;// [K]
19 //***********//
20
21 Ma = 28.02;// [kg/kmol]
22 R = 8314;// [J/K.kgmol]
```

```
23 //From Eqn 4.22
24 lambda = (3.2*viscosity1/Pt1)*(R*T/(2*(%pi)*Ma))
      ^0.5;
25 d = 10^(-4);// [m]
26 d_by_lambda = d/lambda;
27 // Kundsen flow will not occur
28 // N2 flow corresponding to 9 cubic ft/square ft.min
       at 300K & 1 std atm = 0.0457 cubic m/square m.
      min
29 Na1 = 0.0457*(273/T)*(1/22.41);// [kmol/square m.s]
30 Pt1_diff_Pt2 = 2*3386/13.6;// [N/square m]
31 Ptav = Pt1+(Pt1_diff_Pt2/2);// [N/square m]
32 // From Eqn. 4.26
33 k1 = Na1*R*T*z/(Ptav*(Pt1_diff_Pt2));// [m^4/N.s]
34
35 //For N2 at 393K
36 viscosity2 = 2.2*10^(-5);// [kg/m.s]
37 k2 = (k1*viscosity1)/(viscosity2);// [m^4/N.s]
38 // From Eqn 4.26
39 Na = (k2*Ptav*Pt1_diff_Pt2)/(R*T2*z);// [kmol/square
       m.s]
40 printf("Flow rate to be expected is %e kmol/square m
      .s",Na);
```

Chapter 5

Interphase Mass Transfer

Scilab code Exa 5.1 Local overall mass transfer coeffecient

```
1  clear;
2  clc;
3
4  // Illustration 5.1
5  // Page: 114
6
7  printf('Illustration 5.1 - Page: 114\n\n');
8
9  // solution
10
11 //***Data***//
12 // a = NH3, b = H2O
13 d = 2.54*10^(-2);// [m]
14 Yag = 0.80;
15 Xal = 0.05;
16 T = 273+26.7;// [K]
17 Kl = 2.87*10^(-5);// [kmol/square m.s.(kmol/cubic m)
       ]
18 Sh = 40;
19 Da = 2.297*10^(-5);// [square m.s]
20 P = 1.0133*10^(5);// [N/square m]
```

```
21 Xbm = 1.0;
22 //*********//
23
24 Ma = 18;// [kg/kmol]
25 // Liquid:
26 // Because of large conc. of ammonia in gas F's
      rather than k's are used.
27 // Molecular weight of water and ammonia are nearly
      same.
28 // The density of the solution is practically that
      of water.
29 MolarDensity1 = 1000/Ma;// [kmol/cubic m]
30 // Kl is determined for dilute soln. where Xbm is
      practically 1.0
31 Fl = Kl*Xbm*MolarDensity1;// [kmol/square m.s]
32 Ma = 18;// [kg-/kmol]
33 // Gas:
34 MolarDensity2 = (1/22.41)*(273/(273+26.7));// [kmol/
      cubic m]
35 Fg = Sh*MolarDensity2*Da/d;// [kmol/square m.s]
36
37 // Mass Transfer Flux
38 // Th eqb. distribuion data for NH3 from "The
      Chemical Engineers Handbook" 5th Edt. p3-68:
39 // Data = [Xa,pa]
40 // Xa = NH3 mole fraction in gas phas
41 // pa = NH3 partial pressure in N/square m
42 Data = [0 0;0.05 7171;0.10 13652;0.25 59917;0.30
      93220];
43 // Ya_star = mole fraction of NH3 in gas phase at
      eqb.
44 Ya_star = zeros(5);
45 for i = 1:5
46     Ya_star(i) = (Data(i,2)/P);
47 end
48 // For transfer of only one component
49 Na_by_SummationN = 1.0;
50 Ya = zeros(5);
```

```
for i = 1:5
    Ya(i) = 1-((1-Yag)*(1-Xal)/(1-Data(i)));
end
scf(0);
plot(Data(:,1),Ya_star,Data(:,1),Ya);
xgrid();
xlabel('Xa = mole fraction of NH3 in liquid phase');
ylabel('Ya = mole fraction of NH3 in gas phase');
legend('equilibrium line','operating line');
title('Ya Vs Xa');

// From intersection of operating line & Eqb. line
Xai = 0.274;
Yai = 0.732;

// From Eqn.5.20
Na = Na_by_SummationN*Fg*log((Na_by_SummationN-Yai)
   /(Na_by_SummationN-Yag));// [kmol NH3 absorbed/
   square m.s]
printf("Local mass transfer flux for ammonia is %e
   kmol/square m.s",Na);
```

Scilab code Exa 5.2 Stages and Mass Transfer Rates

```
clear;
clc;

// Illustration 5.2
// Page: 130

printf('Illustration 5.2 - Page: 130\n\n');

// solution

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

```
// Eqb. data
// Data = [Wt% of moisture in the soap, Partial
   pressure of water in air(mm Hg)]
Data = [0 0;2.40 9.66;3.76 19.20;4.76 28.4;6.10
   37.2;7.83 46.4;9.90 55.0; 12.63 63.2;15.40
   71.9;19.02 79.5];
P = 760;// [mm Hg]
// Initial air
p1 = 12;// [mm Hg]
T = 273+75;// [K]
//******//

// Y = kg water/kg dry air
// X = kg water/kg dry soap
// E = Air water phase
// R = Soap water phase
Y = zeros(10);
X = zeros(10);
for i = 1:10
    Y(i) = Data(i,2)/(P-Data(i,2))*(18.02/29);
    X(i) = Data(i,1)/(100-Data(i,1));
end

printf('Illustration 5.2 (a)\n\n');

// Soln. (a)
// First operation
Y1 = p1/(P-p1);// [kg water/kg dry soap]
// Initial Soap
S1 = 16.7/(100-16.7);// [kg water/kg dry soap]
// Final soap
S2 = 13/(100-13);// [kg water/kg dry soap]
Rs = 10*(1-0.167);// [kg dry soap]
// Using ideal gas law
Es = 10*((760-p1)/760)*(273/T)*(29/22.41);// [kg dry
    air]
slopeOperat = -Rs/Es;

```

```
46 deff("[y] = f2(x)","y = slopeOperat*(x-S1)+Y1")
47 x = S1:-0.01:S2;
48
49 // Second Operation
50 X1 = S2;
51 scf(1);
52 deff("[y] = f3(S)","y = slopeOperat*(S-X1)+Y1");
53 S = 0:0.01:S1;
54 plot(X,Y,x,f2,S,f3);
55 xlabel('kg water / kg dry soap');
56 ylabel('kg water / kg dry air');
57 legend('Equilibrium line','First Process','Second
     Process');
58 a = get("current_axes");
59 tight_limits = "on";
60 a.data_bounds = [0 0;0.24 0.08];
61 xgrid();
62 title("Illustration 5.2(a)")
63 // Results for First Process
64 // The condition at abcissa S2 correspond to the end
      of first operation
65 printf("Conditions corresponding to First Operation
     \n")
66 printf("X = %f kg water/kg dry soap\n",S2);
67 printf("Y = %f kg water/kg dry air\n",f2(S2));
68
69 // Results for Second Process
70 //  The point at which the line meets the
     equilibrium line corresponds to the final value
71 X2 = 0.103;
72 Y2 = (X2/(1+X2));
73 printf("Final moisture content of soap is %f %%\n\n"
     ,Y2*100);
74
75 printf('Illustration 5.2 (b)\n\n');
76
77 // Solution (b)
78
```

```
Rs = 1*(1-0.167);// [kg dry soap/h]
// Entering soap
X1 = 0.20;// [kg water/kg dry soap]
// Leaving soap
x = 0.04;
X2 = x/(1-x);// [kg water/kg dry soap]
// Entering air
Y2 = 0.00996;// [from Illustration 5.2(a), kg water/
   kg dry air]
// The operating line of least slope giving rise to
   eqb. condition will indicate least amount of air
   usable.
// At X1 = 0.20; the eqb. condition:
Y1 = 0.0675;// [kg water/kg dry air]
scf(2);
deff('[y] = f4(x)','y = ((Y1-Y2)/(X1-X2))*(x-X1)+Y1'
   );
x = X2:0.01:0.24;
plot(X,Y,x,f4);
xlabel('kg water / kg dry soap');
ylabel('kg water / kg dry air');
a = get("current_axes");
tight_limits = "on";
a.data_bounds = [0 0;0.24 0.08];
xgrid();
title("Illustration 5.2(b)")
legend("Equilibrium line","Operating Line");
// By Eqn. 5.35
Es = Rs*(X1-X2)/(Y1-Y2);// [kg dry air/h]
Esv = (Es/29)*22.41*(P/(P-p1))*(T/273);// [cubic m/
   kg dry soap]
printf("Minimum amount of air required is %f cubic m
   /kg dry soap\n\n",Esv);

printf('Illustration 5.2 (c)\n\n');

// solution (c)

```

```
111 Esnew = 1.30*Es;// [kg dry air/h]
112 Y1 = Rs*((X1-X2)/Esnew)+Y2;
113 scf(3);
114 deff('[y] = f5(x)','y = ((Y1-Y2)/(X1-X2))*(x-X1)+Y1'
    );
115 x = X2:0.01:0.24;
116 plot(X,Y,x,f5);
117 xlabel('kg water / kg dry soap');
118 ylabel('kg water / kg dry air');
119 a = get("current_axes");
120 tight_limits = "on";
121 a.data_bounds = [0 0;0.24 0.08];
122 xgrid();
123 title("Illustration 5.2(c)")
124 legend("Equilibrium line","Operating Line");
125 // with final coordinates X = X1 & y = Y1
126 // From figure, Total number of eqb. stages = 3
127 N = 3;
128 printf("Moisture content of air leaving the drier is
     %f kg water/kg dry air\n",Y1);
129 printf("Total number of eqb. stages = %d\n",N);
```

Chapter 6

Equipment for Gas Liquid Operation

Scilab code Exa 6.1 Bubble Columns

```
1  clear;
2  clc;
3
4  // Illustration 6.1
5  // Page: 145
6
7  printf('Illustration 6.1 - Page: 145\n\n');
8
9  // solution
10
11 //****Data****//
12 // w = Gas flow rate per orifice
13 w = 0.055/50;// [kg/s]
14 L = 8*10^(-4);// [liquid flow rate, cubic m/s]
15 d = 0.003;// [diameter of the orifice,m]
16 viscocity_gas = 1.8*10^(-5);// [kg/m.s]
17 //******//
18
19 Re = 4*w/(%pi*d*viscocity_gas);
```

```
Dp = 0.0071*Re^(-0.05);// [m]
h = 3;// [height of vessel,m]
P_atm = 101.3;// [kN/square m]
Density_water = 1000;// [kg/cubic m]
g = 9.81;// [m/s^2]
Temp = 273+25;// [K]
P_orifice = P_atm+(h*Density_water*g/1000);// [kN/
   square m]
P_avg = P_atm+((h/2)*Density_water*g/1000);// [kN/
   square m]
Density_gas = (29/22.41)*(273/Temp)*(P_avg/P_atm);//
    [kg/cubic m]
D = 1;// [dia of vessel,m]
Area = (%pi*D^2)/4;// [square m]
Vg = 0.055/(Area*Density_gas);// [m/s]
Vl = L/Area;// [m/s]
sigma = 0.072;// [N/m]
// From fig. 6.2 (Pg 143)
abscissa = 0.0516;// [m/s]
Vg_by_Vs = 0.11;
Vs = Vg/Vg_by_Vs;// [m/s]
deff('[y] = f6(shi_g)','y = Vs-(Vg/shi_g)+(Vl/(1-
   shi_g))');
shi_g = fsolve(0.5,f6);
dp = ((Dp^3)*(P_orifice/P_avg))^(1/3);// [bubble
   diameter,m]
// From eqn. 6.9
a = 6*shi_g/dp;// [specific interfacial area,square
   m]
printf("The Specific Interfacial Area is %f square m
   /cubic m\n",a);

// For diffsion of Cl2 in H20
Dl = 1.44*10^(-9);// [square m/s]
viscocity_water = 8.937*10^(-4);// [kg/m.s]
Reg = dp*Vs*Density_water/viscocity_water;
Scl = viscocity_water/(Density_water*Dl);
// From Eqn.6.11
```

```
51 Shl = 2+(0.0187*(Reg^0.779)*(Scl^0.546)*(dp*(g^(1/3)
   )/(Dl^(2/3)))^0.116);
52 // For dilute soln. of Cl2 in H20
53 c = 1000/18.02;// [kmol/cubic m]
54 Fl = (c*Dl*Shl)/dp;// [kmol/square m.s]
55 printf("Mass Transfer coeffecient is %f kmol/square
   m.s\n",Fl);
```

Scilab code Exa 6.2 Mechanical Agitation

```
1  clear;
2  clc;
3
4  // Illustration 6.2
5  // Page: 157
6
7  printf('Illustration 6.2 - Page: 157\n\n');
8
9  // solution
10
11 //****Data****//
12 // a = N2 b = H2O
13 L = 9.5*10^(-4);// [cubic m/s]
14 G = 0.061;// [kg/s]
15 Temp = 273+25;// [K]
16 //*****//
17
18 printf("Construction Arrangement\n");
19 printf("Use 4 vertical wall baffles, 100 mm wide at
   90 degree intervals.\n");
20 printf("Use a 305 mm dameter, a six bladed disk flat
    blade turbine impeller, arranged axially, 300 mm
    from the bottom of vessel\n");
21 printf("The sparger underneath the impeller will be
   in the form of a 240 mm dameter ring made of 12.7
```

```
      mm tubing drilled in the top with 3.18 mm dia
      holes\n");
Di = 0.305;// [m]
Do = 0.00316;// [m]
viscocity_a = 1.8*10^(-5);// [kg/m.s]
Re_g = 35000;
Ma = 28.02;// [kg/kmol]
Mb = 18.02;// [kg/kmol]
// w = Gas flow rate per orifice
w = Re_g*%pi*Do*viscocity_a/4;// [kg/s]
N_holes = G/w;
Interval = %pi*240/round(N_holes);
printf("The number of holes is %d at approx %d mm
   interval around the sparger ring\n",round(N_holes
   ),round(Interval));

viscocity_b = 8.9*10^(-4);// [kg/m.s]
Sigma = 0.072;// [N/m]
Density_b = 1000;// [kg/cubic m]
D = 1;// [dia of vessel,m]
g = 9.81;// [m/s^2]
// From Eqn. 6.18
deff('[y] = f7(N)','y = (N*Di/(Sigma*g/Density_b)
   ^0.25)-1.22-(1.25*D/Di)');
N_min = fsolve(2,f7);// [r/s]
N = 5;// [r/s]
Re_l = ((Di^2)*N*Density_b/viscocity_b);
// From fig 6.5 (Pg 152)
Po = 5;
P = Po*Density_b*(N^3)*(Di^5);
h = 0.7;// [m]
P_atm = 101.33;// [kN/square m]
P_gas = P_atm+(h*Density_b*g/1000);// [kN/square m]
Qg = (G/Ma)*22.41*(Temp/273)*(P_atm/P_gas);// [cubic
    m/s]
// From Fig.6.7 (Pg 155)
abcissa = Qg/(N*(Di^3));
// abcissa is off scale
```

```
54 Pg_by_P = 0.43;
55 Pg = 0.43*P;// [W]
56 Vg = Qg/(%pi*(D^2)/4);// [superficial gas velocity,m
     /s]
57 check_value = (Re_l^0.7)*((N*Di/Vg)^0.3);
58 vl = %pi*(D^2)/4;// [cubic m]
59 // Since value<30000
60 // From Eqn. 6.21, Eqn.6.23 & Eqn. 6.24
61 K = 2.25;
62 m = 0.4;
63 Vt = 0.250;// [m/s]
64 shi = 1;
65 err = 1;
66 while (err>10^(-3))
67     a = 1.44*((Pg/vl)^0.4)*((Density_b/(Sigma^3))
         ^0.2)*((Vg/Vt)^0.5);// [square m/cubic m]
68     shin = (0.24*K*((viscocity_a/viscocity_b)^0.25)
        *((Vg/Vt)^0.5))^(1/(1-m));
69     Dp = K*((vl/Pg)^0.4)*((Sigma^3/Density_b)^0.2)*(
        shin^m)*((viscocity_a/viscocity_b)^0.25);// [
        m]
70     err = abs(shi-shin);
71     Vt = Vt-0.002;// [m/s]
72     shi = shin;
73 end
74
75 // For N2 in H2
76 Dl = 1.9*10^(-9);// [square m/s]
77 Ra = 1.514*10^(6);
78 // By Eqn. 6.25
79 Shl = 2.0+(0.31*(Ra^(1/3)));
80 // For dilute soln.
81 c = 1000/Mb;// [kmol/cubic m]
82 Fl = Shl*c*Dl/Dp;// [kmol/square m.s]
83 printf("The average gas-bubble diameter is %e m\n",
     Dp);
84 printf("Gas Holdup:%f\n",shi);
85 printf("Interfacial area:%e square m/cubic m \n",a);
```

```
86 printf("Mass transfer coeffecient:%e kmol/square m.s
   \n",Fl);
```

Scilab code Exa 6.3 Tray Towers

```
1  clear;
2  clc;
3
4  // Illustration 6.3
5  // Page: 174
6
7  printf('Illustration 6.3 - Page: 174\n\n');
8
9  // solution
10
11 //****Data****//
12 // a = methanol b = water
13 G = 0.100;// [kmol/s]
14 L = 0.25;// [kmol/s]
15 Temp = 273+95;// [K]
16 XaG = 0.18;// [mol % in gas phase]
17 MaL = 0.15;// [mass % in liquid phase]
18 //*****//
19
20 Ma = 32;// [kg/kmol]
21 Mb = 18;// [kg/kmol]
22 Mavg_G = XaG*Ma+((1-XaG)*Mb);// [kg/kmol]
23 Density_G = (Mavg_G/22.41)*(273/Temp);// [kg/cubic
   cm]
24 Q = G*22.41*(Temp/273);// [cubic cm/s]
25 Density_L = 961;// [kg/cubic cm]
26 Mavg_L = 1/((MaL/Ma)+(1-MaL)/Mb);// [kg/kmol]
27 q = L*Mavg_L/Density_L;
28
29 // Perforations
```

```
30 printf("Perforations\n");
31 printf("Do = 4.5mm on an equilateral triangle pitch
      12 mm between the hole centres, punched in sheet
      metal 2 mm thick\n");
32 Do = 0.0045;// [m]
33 pitch = 0.012;// [m]
34 // By Eqn.6.31
35 Ao_by_Aa = 0.907*(Do/pitch)^2;
36 printf("The ratio of Hole Area By Active Area is:%f\
      n",Ao_by_Aa);
37 printf("\n");
38
39 // Tower Diameter
40 printf("Tower Diameter\n");
41 t = 0.50;// [tray spacing,m]
42 printf("Tower Spacing:%f m\n",t);
43 // abcissa = (L/G)*(Density_G/Density_L)^0.5 = (q/Q)
      *(Density_L/Density_G)^0.5
44 abcissa = (q/Q)*(Density_L/Density_G)^0.5;
45 // From Table 6.2 (Pg 169)
46 alpha = (0.0744*t)+0.01173;
47 beeta = (0.0304*t)+0.015;
48 if (abcissa<0.1)
49     abcissa = 0.1;
50 end
51 sigma = 0.040;// [N/m]
52 // From Eqn.6.30
53 Cf = ((alpha*log10(1/abcissa))+beeta)*(sigma/0.02)
      ^0.2;
54 // From Eqn. 6.29
55 Vf = Cf*((Density_L-Density_G)/Density_G)^(1/2);// [
      m/s]
56 // Using 80% of flooding velocity
57 V = 0.8*Vf;// [m/s]
58 An = Q/V;// [square m]
59 // The tray area used by one downspout = 8.8%
60 At = An/(1-0.088);// [square m]
61 D = (4*At/%pi)^(1/2);// [m]
```

```
// Take D = 1.25 m
D = 1.25; //[m]
At = %pi*(D^2)/4;// [corrected At, square m]
W = 0.7*D;// [weir length ,m]
Ad = 0.088*At;// [square m]
// For a design similar to Fig 6.14 (Pg 168)
// A 40 mm wide supporting ring, beams between
   downspouts and a 50 mm wide disengaging &
   distributing zones these areas total 0.222 square
    m
Aa = At-(2*Ad)-0.222;
printf("Weir Length:%f\n",W);
printf("Area for perforated sheet: %f square m\n",Aa
   );
printf("\n");

// Weir crest h1 & Weir height hw
printf("Weir crest h1 & Weir height hw\n")
h1 = 0.025;// [m]
h1_by_D = h1/D;
D_by_W = D/W;
// From Eqn. 6.34
Weff_by_W = sqrt(((D_by_W)^2)-((((D_by_W)^2-1)^0.5)
   +(2*h1_by_D*D_by_W))^2);
// Set hw to 50 mm
hw = 0.05;// [m]
printf("Weir crest: %f m\n",h1);
printf("Weir height: %f m\n",hw);
printf("\n");

// Dry Pressure Drop
printf("Dry Pressure Drop\n");
l = 0.002;// [m]
// From Eqn. 6.37
Co = 1.09*(Do/l)^0.25;
Ao = 0.1275*Aa;// [square m]
Vo = Q/Ao;// [m/sec]
viscocity_G = 1.25*10^(-5);// [kg/m.s]
```

```
Re = Do*Vo*Density_G/viscocity_G;
// From "The Chemical Engineers Handbook," 5th
   Edition fig 5.26
fr = 0.008;
g = 9.81;// [m/s^2]
// From Eqn. 6.36
deff('[y] = f(hd)','y = (2*hd*g*Density_L/(Vo^2*
   Density_G))-(Co*(0.40*(1.25-(Ao/An))+(4*l*fr/Do)
   +(1-(Ao/An))^2))');
hd = fsolve(1,f);
printf("Dry Pressure Drop:%f m\n",hd);
printf("\n");

// Hydraulic head hl
printf("Hydraulic head hl");
Va = Q/Aa;// [m/s]
z = (D+W)/2;// [m]
// From Eqn. 6.38
hl = 6.10*10^(-3)+(0.725*hw)-(0.238*hw*Va*(Density_G
   )^0.5)+(1.225*q/z);// [m]
printf("Hydraulic head: %f m\n",hl);
printf("\n");

//Residual Pressure drop hr
printf("Residual Pressure drop hr\n");
// From Eqn. 6.42
hr = 6*sigma/(Density_L*Do*g);// m
printf("Residual Pressure Drop:%e m\n",hr);
printf("\n");

// Total Gas pressure Drop hg
printf("Total Gas pressure Drop hg\n")
// From Eqn. 6.35
hg = hd+hl+hr;// [m]
printf("Total gas pressure Drop: %f m\n",hg);
printf("\n");

// Pressure loss at liquid entrance h2
```

```
129 printf("Pressure loss at liquid entrance h2\n");
130 // Al: Area for the liquid flow under the apron
131 Al = 0.025*W;// [square m]
132 Ada = min(Al,Ad);
133 // From Eqn. 6.43
134 h2 = (3/(2*g))*(q/Ada)^2;
135 printf("Pressure loss at liquid entrance:%e m\n",h2)
    ;
136 printf("\n");
137
138 // Backup in Downspout h3
139 printf("Backup in Downspout h3\n");
140 // From Eqn.6.44
141 h3 = hg+h2;
142 printf("Backup in Downspout:%f m\n",h3);
143 printf("\n");
144
145 // Check on Flooding
146 printf("Check on Flooding\n");
147 if((hw+h1+h3)<(t/2))
148     printf("Choosen Tower spacing is satisfactory\n"
       );
149 else
150     printf("Choosen Tower spacing is  not
       satisfactory\n")
151 end
152 printf("\n");
153
154 // Weeping Velocity
155 printf("Weeping Velocity\n");
156 printf("For W/D ratio %f weir is set at %f m from
    the center from the tower\n",W/D,0.3296*D);
157 Z = 2*(0.3296*D);// [m]
158 // From Eqn.6.46
159 deff('[y] = f8(Vow)','y = (Vow*viscocity_G/(sigma))
    -(0.0229*((viscocity_G^2/(sigma*Density_G*Do))*(
    Density_L/Density_G))^0.379)*((1/Do)^0.293)*(2*Aa
    *Do/(sqrt(3)*(pitch^3)))^(2.8/((Z/Do)^0.724))');
```

```
160 Vow = fsolve(0.1,f8);// [m/s]
161 printf("The minimum gas velocity through the holes
      below which excessive weeping is likely: %f m/s\n
      ",Vow);
162 printf("\n");
163
164 // Entrainment
165 printf("Entrainment\n");
166 V_by_Vf = V/Vf;
167 // From Fig.6.17 (Pg 173), V/Vf = 0.8 & abcissa =
      0.0622
168 E = 0.05;
169 printf("Entrainment:%f\n",E);
```

Scilab code Exa 6.4 Efficiency of sieve tray

```
1  clear;
2  clc;
3
4  // Illustration 6.4
5  // Page: 183
6
7  printf('Illustration 6.4 - Page: 183\n\n');
8
9  // solution
10
11 //****Data****//
12 //From Illustrtion 6.3:
13 G = 0.100;// [kmol/s]
14 Density_G = 0.679;// [kg/cubic m]
15 q = 5*10^(-3);// [cubic m/s]
16 Va = 3.827;// [m/s]
17 z = 1.063;// [m]
18 L = 0.25;// [kmol/s]
19 hL = 0.0106;// [m]
```

```
hW = 0.05;// [m]
Z = 0.824;// [m]
E = 0.05;
ya = 0.18;// [mole fraction methanol]

// a:CH3OH b:H2O
Ma = 32;// [kg/kmol]
Mb = 18;// [kg/kmol]
// From Chapter 2:
ScG = 0.865;
Dl = 5.94*10^(-9);// [square m/s]
// From Eqn. 6.61:
NtG = (0.776+(4.57*hW)-(0.238*Va*Density_G^0.5)
   +(104.6*q/Z))/ScG^0.5;
DE = ((3.93*10^(-3))+(0.0171*Va)+(3.67*q/Z)+(0.1800*
   hW))^2;// [square m/s]
thethaL = hL*z*Z/q;// [s]
NtL = 40000*Dl^0.5*((0.213*Va*Density_G^0.5)+0.15)*
   thethaL;
// For 15 mass% methanol:
xa = (15/Ma)/((15/Ma)+(85/Mb));
// From Fig 6.23 (Pg 184)
mAC = -(NtL*L)/(NtG*G);// [Slope of AC line]
meqb = 2.50;// [slope of equilibrium line]
// From Eqn. 6.52:
NtoG = 1/((1/NtG)+(meqb*G/L)*(1/NtL));
// From Eqn. 6.51:
EOG = 1-exp(-NtoG);
// From Eqn. 6.59:
Pe = Z^2/(DE*thethaL);
// From Eqn. 6.58:
eta = (Pe/2)*((1+(4*meqb*G*EOG/(L*Pe)))^0.5-1);
// From Eqn. 6.57:
EMG = EOG*(((1-exp(-(eta+Pe)))/((eta+Pe)*(1+(eta+Pe)
   /eta)))+(exp(eta)-1)/(eta*(1+eta/(eta+Pe))));
// From Eqn. 6.60:
EMGE = EMG/(1+(EMG*E/(1-E)));
printf("Effeciency of Sieve trays: %f",EMGE);
```

Scilab code Exa 6.5 Packing

```
clear;
clc;

// Illustration 6.5
// Page: 200

printf('Illustration 6.5 - Page: 200\n\n');

// solution

// ****Data****//
G = 0.80;// [cubic m/s]
P = 10^2;// [kN/square m]
XaG = 0.07;
Temp = 273+30;// [K]
L = 3.8;// [kg/s]
Density_L = 1235;// [kg/cubic m]
viscocity_L = 2.5*10^(-3);// [kg/m.s]
//******//

// a = SO2 b = air

// Solution (a)

// Since the larger flow quantities are at the
   bottom for an absorber, the diameter will be
   choosen to accomodate the bottom condition
Mavg_G = XaG*64+((1-XaG)*29);// [kg/kmol]
G1 = G*(273/Temp)*(P/101.33)*(1/22.41);// [kmol/s]
G2 = G1*Mavg_G;// [kg/s]
Density_G = G2/G;// [kg/cubic m]
// Assuming Complete absorption of SO2
```

```
sulphur_removed = G1*XaG*64;// [kg/s]
abcissa = (L/G)*((Density_G/Density_L)^0.5);
//From Fig. 6.24, using gas pressure drop of 400 (N/
    square m)/m
ordinate = 0.061;
// For 25 mm ceramic Intalox Saddle:
Cf = 98;// [Table 6.3 Pg 196]
J = 1;
G_prime = (ordinate*Density_G*(Density_L-Density_G)
    /(Cf*viscocity_L^0.1*J))^0.5;// [kg/square m.s]
A = G2/G_prime;// [square m]
D = (4*A/%pi)^0.5;// [m]
printf("The Tower Diameter is %f m\n",D);

// Solution (b)

// Let
D = 1;// [m]
A = %pi*D^2/4;// [square m]
// The pressure drop for 8 m of irrigated packing
delta_p = 400*8;// [N/square m]
// For dry packing
G_prime = (G2-sulphur_removed)/A;// [kg/square m.s]
P = P-(delta_p/1000);// [kN/square m]
Density_G = (29/22.41)*(273/Temp)*(P/101.33);// [kg/
    cubic m]
// From Table 6.3 (Pg 196)
Cd = 241.5;
// From Eqn. 6.68
delta_p_by_z = Cd*G_prime^2/Density_G;// [N/square m
    for 1m of packing]
pressure_drop = delta_p+delta_p_by_z;// [N/square m]
V = 7.5;// [m/s]
head_loss = 1.5*V^2/2;// [N.m/kg]
head_loss = head_loss*Density_G;// [N/square m]
Power = (pressure_drop+head_loss)*(G2-
    sulphur_removed)/(Density_G*1000);// [kW]
eta = 0.6;
```

```
64 Power = Power/eta;// [kW]
65 printf("The Power for the fan motor is %f kW\n",
      Power);
```

Scilab code Exa 6.6 Mass Transfer Coeffecient for packed towers

```
1  clear;
2  clc;
3
4  // Illustration 6.6
5  // Page: 204
6
7  printf('Illustration 6.6 - Page: 204\n\n');
8
9  // solution
10
11 //****Data****//
12 // Gas
13 Mavg_G = 11;// [kg/kmol]
14 viscocity_G = 10^(-5);// [kg/m.s]
15 Pt = 107;// [kN/square m]
16 Dg = 1.30*10^(-5);// [square m/s]
17 Temp = 273+27;// [K]
18 G_prime = 0.716;// [kg/square m.s]
19
20 // Liquid:
21 Mavg_L = 260;
22 viscocity_L = 2*10^(-3);// [kg/m.s]
23 Density_L = 840;// [kg/cubic m]
24 sigma = 3*10^(-2);// [N/m]
25 Dl = 4.71*10^(-10);// [square m/s]
26 //******//
27
28 //Gas:
29 Density_G = (Mavg_G/22.41)*(Pt/101.33)*(273/Temp);//
```

```
    [kg/cubic m]
ScG = viscocity_G/(Density_G*Dg);
G = G_prime/Mavg_G;// [kmol/square m.s]

// Liquid:
L_prime = 2.71;// [kg/square m.s]
ScL = viscocity_L/(Density_L*Dl);

// Holdup:
// From Table 6.5 (Pg 206), L_prime = 2.71 kg/square
    m.s
Ds = 0.0472;// [m]
beeta = 1.508*Ds^0.376;
shiLsW = 5.014*10^(-5)/Ds^1.56;// [square m/cubic m]
shiLtW = (2.32*10^(-6))*(737.5*L_prime)^beeta/(Ds^2)
   ;// [square m/cubic m]
shiLoW = shiLtW-shiLsW;// [square m/cubic m]
H = (1404*(L_prime^0.57)*(viscocity_L^0.13)/((
   Density_L^0.84)*((3.24*L_prime^0.413)-1)))*(sigma
   /0.073)^(0.2817-0.262*log10(L_prime));
shiLo = shiLoW*H;// [square m/cubic m]
shiLs = 4.23*10^(-3)*(viscocity_L^0.04)*(sigma^0.55)
   /((Ds^1.56)*(Density_L^0.37));// [square m/cubic
   m]
shiLt = shiLo+shiLs;// [square m/cubic m]

// Interfacial Area:
// From Table 6.4 (Pg 205)
m = 62.4;
n = (0.0240*L_prime)-0.0996;
p = -0.1355;
aAW = m*((808*G_prime/(Density_G^0.5))^n)*(L_prime^p
   );// [square m/cubic m]
// From Eqn. 6.73
aA = aAW*shiLo/shiLoW;// [square m/cubic m]
// From Table 6.3 (Pg 196)
e = 0.75;
// From Eqn. 6.71
```

```
60 eLo = e-shiLt;
61 // From Eqn. 6.70
62 deff('[y] = f9(Fg)','y = ((Fg*ScG^(2/3))/G)-1.195*((
      Ds*G_prime)/(viscocity_G*(1-eLo)))^(-0.36)');
63 Fg = fsolve(1,f9);// [kmol/square m.s]
64 // From Eqn. 6.72:
65 deff('[y] = f10(Kl)','y = (Kl*Ds/Dl)-(25.1*(Ds*
      L_prime/viscocity_L)^0.45)*ScL^0.5');
66 Kl = fsolve(1,f10);// [(kmol/square m.s).(kmol/cubic
       m)]
67 // Since the value of Kl is taken at low conc., it
      can be converted into Fl
68 c = (Density_L/Mavg_L);// [kmol/cubic m]
69 Fl = Kl*c;// [kmol/cubic m]
70 printf("The volumetric coeffecients are\n");
71 printf("Based on Gas Phase %f kmol/cubic m.s\n",Fg*
      aA);
72 printf("based on Liquid Phase %f kmol/cubic m.s\n",
      Fl*aA);
```

Scilab code Exa 6.7 Volumetric Coeffecient for packed towers

```
1 clear;
2 clc;
3
4 // Illustration 6.7
5 // Page: 207
6
7 printf('Illustration 6.7 - Page: 207\n\n');
8
9 // solution
10
11 //****Data****//
12 // Air
13 G_prime = 1.10;// [kg/square m.s]
```

```
viscocity_G = 1.8*10^(-5);// [kg/m.s]
ScG = 0.6;// [for air water mixture]
Temp1 = 273+20;// [K]

// Water
L_prime = 5.5;// [kg/square m.s]
//*****//

// Air:
Ma = 29;// [kg/kmol]
G = G_prime/Ma;// [kmol/square m.s]
Density_G = (Ma/22.41)*(273/Temp1);
Cpa = 1005;// [N.m/kg.K]
PrG = 0.74;

// Liquid:
kth = 0.587;// [W/m.K]
Cpb = 4187;// [N.m/kg.K]
viscocity_L = 1.14*10^(-3);// [kg/m.s]

// From Table 6.5 (Pg 206)
Ds = 0.0725;// [m]
beeta = 1.508*(Ds^0.376);
shiLtW = (2.09*10^(-6))*(737.5*L_prime)^beeta/(Ds^2)
   ;// [square m/cubic m]
shiLsW = 2.47*10^(-4)/(Ds^1.21);// [square m/cubic m
   ]
shiLoW = shiLtW-shiLsW;// [square m/cubic m]
// From Table 6.4 (Pg 205)
m = 34.03;
n = 0;
p = 0.362;
aAW = m*(808*G_prime/Density_G^0.5)^(n)*L_prime^p;//
    [square m/cubic m]
// From Eqn. 6.75
aVW = 0.85*aAW*shiLtW/shiLoW;// [square m/cubic m]
// From Table 6.3
e = 0.74;
```

```
eLo = e-shiLtW;
// From Eqn. 6.70
deff('[y] = f11(Fg)','y = ((Fg*ScG^(2/3))/G)
   -1.195*((Ds*G_prime)/(viscocity_G*(1-eLo)))
   ^(-0.36)');
Fg = fsolve(1,f11);// [kmol/square m.s]
// Since the liquid is pure water. It has no mass
   trnsfer coeffecient.
// For such process we need convective heat transfer
    coeffecient for both liquid & gas.
// Asuming Jd = Jh
// From Eqn. 6.70
Jh = 1.195*((Ds*G_prime)/(viscocity_G*(1-eLo)))
   ^(-0.36);
Hg = Jh*Cpa*G_prime/(PrG^(2/3));// [W/square m.K]
PrL = Cpb*viscocity_L/kth;
// Heat transfer analog of Eqn. 6.72
Hl = 25.1*(kth/Ds)*(Ds*L_prime/viscocity_L)^0.45*PrL
   ^0.5;// [W/square m.K]
printf("The volumetric coeffecients are\n");
printf("Based on Gas Phase %f W/cubic m.K\n",Hg*aVW)
   ;
printf("based on Liquid Phase %f W/cubic m.K\n",Hl*
   aVW);
```

Chapter 7

Humidification Operation

Scilab code Exa 7.1 Interpolation Between Data

```
1  clear;
2  clc;
3
4  // Illustration 7.1
5  // Page: 222
6
7  printf('Illustration 7.1 - Page: 222\n\n');
8
9  // Solution
10
11 // ****Data****//
12 Temp1 = 273+26.1;// [K]
13 P1 = 100;// [mm Hg]
14 Temp2 = 273+60.6;// [K]
15 P2 = 400;// [mm Hg]
16 P = 200;// [mm Hg]
17 //*****//
18
19 deff('[y] = f12(T)','y = ((1/Temp1)-(1/T))/((1/Temp1
       )-(1/Temp2))-((log(P1)-log(P))/(log(P1)-log(P2)))
       ');
```

```
20 T = fsolve(37,f12);// [K]
21 printf("At %f 0C, the vapour pressure of benzene is
      200 mm Hg\n",T-273);
```

Scilab code Exa 7.2 Reference Substance Plots

```
1  clear;
2  clc;
3
4  // Illustration 7.2:
5  // Page: 223
6
7  printf('Illustration 7.2 - Page: 223\n\n');
8  printf('Illustration 7.2 (b)\n\n');
9
10 // Solution (b)
11
12 // At 100 OC,
13 PH2O = 760;// [Vapour pressure of water, mm of Hg]
14 // From Fig. 7.2 (Pg 224)
15 // At this value,
16 PC6H6 = 1400;// [Vapour pressure of benzene, mm of
      Hg]
17 printf("Vapour Pressure of benzene at 100 OC is %d
      mm of Hg\n\n", PC6H6);
18
19 printf('Illustration 7.2 (c)\n\n');
20
21 // Solution (c)
22
23 // Reference: H20
24 // At 25 OC
25 m = 0.775;
26 Mr = 18.02;// [kg/kmol]
27 lambdar = 2443000;// [N/m.kg]
```

```
28 M = 78.05;// [kg/kmol]
29 // From Eqn. 7.6:
30 lambda = m*lambdar*Mr/M;// [N/m.kg]
31 printf("Latent Heat of Vaporization at 25 OC is %f
      kN/m.kg\n",lambda/1000);
```

Scilab code Exa 7.3 Enthalpy

```
1 clear;
2 clc;
3
4 // Illustration 7.3
5 // Page: 226
6
7 printf('Illustration 7.3 - Page: 226\n\n');
8
9 // solution
10
11 // ****Data****//
12 m = 10;// [kg]
13 Cvap = 1.256;// [kJ/kg.K]
14 Cliq = 1.507;// [kJ/kg.K]
15 Temp1 = 100;// [OC]
16 Temp4 = 10;// [OC]
17 //******//
18
19 // Using Fig 7.2 (Pg 224):
20 Temp2 = 25;// [OC]
21 // Using the notation of Fig. 7.3:
22 H1_diff_H2 = Cvap*(Temp1-Temp2);// [kJ/kg]
23 // From Illustration 7.2:
24 H2_diff_H3 = 434;// [Latent Heat of Vaporisation, kJ
      /kg]
25 H3_diff_H4 = Cliq*(Temp2-Temp4);// [kJ/kg]
26 H1_diff_H4 = H1_diff_H2+H2_diff_H3+H3_diff_H4;// [kJ
```

```
    /kg]
27 H = m*H1_diff_H4;// [kJ]
28 printf("Heat evolved for 10 kg Benzene is %f kJ\n",H
    );
```

Scilab code Exa 7.4 Vapour Gas Mixture

```
1  clear;
2  clc;
3
4  // Illustration 7.4
5  // Page: 227
6
7  printf('Illustration 7.4 - Page: 227\n\n');
8
9  // solution
10
11 //****Data****//
12 // A = benzene vapour; B = Nitrogen Gas
13 P = 800;// [mm Hg]
14 Temp = 273+60;// [K]
15 pA = 100;// [mm Hg]
16 //******//
17
18 pB = P-pA;// [mm Hg]
19 MA = 78.05;// [kg/kmol]
20 MB = 28.08;// [kg/kmol]
21
22 // Mole Fraction
23 printf("On the Basis of Mole Fraction\n");
24 yAm = pA/P;
25 yBm = pB/P;
26 printf("Mole Fraction of Benzene is %f\n",yAm);
27 printf("Mole Fraction of Nitrogen is %f\n",yBm);
28 printf("\n");
```

```
29
30 // Volume Fraction
31 printf("On the Basis of Volume Fraction\n");
32 // Volume fraction is same as mole Fraction
33 yAv = yAm;
34 yBv = yBm;
35 printf("Volume Fraction of Benzene is %f\n",yAv);
36 printf("Volume Fraction of Nitrogen is %f\n",yBv);
37 printf("\n");
38
39 // Absolute Humidity
40 printf("On the basis of Absolute humidity\n")
41 Y = pA/pB;// [mol benzene/mol nitrogen]
42 Y_prime = Y*(MA/MB);// [kg benzene/kg nitrogen]
43 printf("The concentration of benzene is %f kg
     benzene/kg nitrogen\n",Y_prime);
```

Scilab code Exa 7.5 Saturated Vapour Gas Mixture

```
1 clear;
2 clc;
3
4 // Illustration 7.5
5 // Page: 228
6
7 printf('Illustration 7.5 - Page: 228\n\n');
8
9 printf('Illustration 7.5 (a)\n\n');
10 // solution(a)
11
12 //****Data****//
13 // A = benzene vapour; B = Nitrogen Gas
14 P = 1;// [atm]
15 //*****//
16
```

```
17 MA = 78.05;// [kg/kmol]
18 MB = 28.02;// [kg/kmol]
19 // Since gas is saturated, from Fig. 7.2 (Pg 224):
20 pA = 275/760;// [atm]
21 Y = pA/(P-pA);// [kmol benzene/kmol nitrogen]
22 Y_prime = Y*(MA/MB);// [kg benzene/kg nitrogen]
23 printf("The concentration of benzene is %f kg
      benzene/kg nitrogen\n\n",Y_prime);
24
25 printf('Illustration 7.5 (b)\n\n');
26 // solution(b)
27
28 // A = benzene vapour; B = CO2
29 MA = 78.05;// [kg/kmol]
30 MB = 44.01;// [kg/kmol]
31 // Since gas is saturated, from Fig. 7.2:
32 pA = 275/760;// [atm]
33 Y = pA/(P-pA);// [kmol benzene/kmol CO2]
34 Y_prime = Y*(MA/MB);// [kg benzene/kg CO2]
35 printf("The concentration of benzene is %f kg
      benzene/kg CO2\n",Y_prime);
```

Scilab code Exa 7.6 Air Water System

```
1 clear;
2 clc;
3
4 // Illustration 7.6
5 // Page: 234
6
7 printf('Illustration 7.6 - Page: 234\n\n');
8
9 // solution
10
11 //****Data****//
```

```
12 // A = water vapour; B = air
13 TempG = 55;// [OC]
14 P = 1.0133*10^(5);// [N/square m]
15 Y_prime = 0.030;// [kg water/kg dry air]
16 //******//
17
18 MA = 18.02;// [kg/kmol]
19 MB = 28.97;// [kg/kmol]
20
21 // Percent Humidity
22 // From psychrometric chart, at 55 OC
23 Ys_prime = 0.115;// [kg water/kg dry air]
24 percent_Humidity = (Y_prime/Ys_prime)*100;
25 printf("The sample has percent Humidity =  %f %%\n",
      percent_Humidity);
26
27 // Molal Absolute Humidity
28 Y = Y_prime*(MB/MA);// [kmol water/kmol dry air]
29 printf("Molal Absolute Humidity of the sample is %f
      kmol water/kmol dry air\n",Y);
30
31 // Partial Pressure
32 pA = Y*P/(1+Y);// [N/square m]
33 printf("The Partial Pressure Of Water is %f N/square
      m\n",pA);
34
35 // Relative Humidity
36 pa = 118*133.3;// [vapour pressure of water at 55 OC
      ,N/square m]
37 relative_Humidity = (pA/pa)*100;
38 printf("The sample has relative Humidity =  %f %%\n"
      ,relative_Humidity);
39
40 // Dew Point
41 // From psychrometric chart,
42 dew_point = 31.5;// [OC]
43 printf("Dew point Of the Sample is %f Oc\n",
      dew_point);
```

```
44
45 // Humid Volume
46 // At 55 OC
47 vB = 0.93;// [specific volume of dry air,cubic m/kg]
48 vsB = 1.10;// [specific volume of saturated air,
      cubic m/kg]
49 vH = vB+((vsB-vB)*(percent_Humidity/100));// [cubic
     m/kg]
50 printf("The Humid Volume of the Sample is %f cubic m
     /kg\n",vH);
51
52 // Humid Heat
53 CB = 1005;// [J/kg.K]
54 CA = 1884;// [J/kg.K]
55 Cs = CB+(Y_prime*CA);// [J/kg]
56 printf("The Humid Heat is %f J/kg dry air.K\n",Cs);
57
58 // Enthalpy
59 HA = 56000;// [J/kg dry air]
60 HsA = 352000;// [J/kg dry air]
61 H_prime = HA+((HsA-HA)*(percent_Humidity/100));// [J
     /kg dry air]
62 printf("The Enthalphy of the sample is %f J/kg dry
     air\n",H_prime);
```

Scilab code Exa 7.7 Air Water System

```
1 clear;
2 clc;
3
4 // Illustration 7.7
5 // Page: 236
6
7 printf('Illustration 7.7 - Page: 236\n\n');
8
```

```
9  // solution
10
11 //****Data****//
12 // A = water vapour; B = air
13 V = 100;// [m^3]
14 Tempi = 55;// [OC]
15 Tempf = 110;// [OC]
16 //*****//
17
18 // From Illustration 7.6
19 vH = 0.974;// [m^3/kg]
20 Cs = 1061.5;// [J/kg]
21 WB = V/vH;// [kg]
22 Q = WB*Cs*(Tempf-Tempi);// [J]
23 printf("Heat recquired is %e J\n",Q);
```

Scilab code Exa 7.8 Adiabatic Saturation Curves

```
1  clear;
2  clc;
3
4  // Illustration 7.8
5  // Page: 237
6
7  printf('Illustration 7.8 - Page: 237\n\n');
8
9  // solution
10
11 //****Data****//
12 Y_prime1 = 0.030;// [kg water/kg dry air]
13 Temp1 = 83;// [OC]
14 //*******//
15
16 // From the psychrometric chart, the condition at 90
       OC
```

```
17 Temp2 = 41.5;// [OC]
18 Y_prime2 = 0.0485;// [kg water/kg dry air]
19 printf("The Outlet Air condition are:\n");
20 printf("Temp. = %f OC\n",Temp2);
21 printf("Absolute Humidity = %f kg water/kg dry air\
   n",Y_prime2);
```

Scilab code Exa 7.9 Lewis Relation

```
1 clear;
2 clc;
3
4 // Illustration 7.9
5 // Page:240
6
7 printf('Illustration 7.9 - Page:240\n\n');
8
9 // solution
10
11 //****Data****//
12 Tempw = 35;// [OC]
13 Tempg = 65;// [OC]
14 //******//
15
16 // From psychrometric chart
17 lambda_w = 2419300;// [J/kg]
18 Y_prime_w = 0.0365;// [kg H2O/kg dry air]
19 // From fig 7.5(a)
20 hG_by_kY = 950;// [J/kg]
21 // From Eqn. 7.26
22 deff('[y] = f13(Y_prime)','y = (Tempg-Tempw)-((
   lambda_w*(Y_prime_w-Y_prime))/hG_by_kY)');
23 Y_prime = fsolve(2,f13);// [kg H2O/kg dry air]
24 printf("Humidity of air is %f kg H2O/kg dry air\n",
   Y_prime);
```

Scilab code Exa 7.10 Lewis Relation

```
clear;
clc;

// Illustration 7.10
// Page:241

printf('Illustration 7.10 - Page:241\n\n');

// solution

//****Data****//
Tg = 60;// [OC]
Y_prime = 0.050;// [kg toulene/kg air]
//*****//

// Wet Bulb temparature
Dab = 0.92*10^(-5);// [square m/s]
density_air = 1.060;// [kg/cubic cm];
viscocity_G = 1.95*10^(-5);// [kg/m.s]
Sc = viscocity_G/(density_air*Dab);
// From Eqn. 7.28
hG_by_kY = 1223*(Sc^0.567);// [J/kg.K]
// Soln. of Eqn. 7.26 by trial & error method:
// (Tg-Tw) = (Yas_prime-Y_prime)*(lambda_w/hG_by_kY)
Tw = 31.8;// [OC]
printf("Wet Bulb Temparature:%f OC\n",Tw);

// Adiabatic Saturation Temparature
C_air = 1005;// [J/kg.K]
C_toulene = 1256;// [J/kg.K]
Cs = C_air+(C_toulene*Y_prime);// [J/kg.K]
// Soln. of Eqn. 7.21 by trial & error method:
```

```
33 // (Tg-Tas) = (Yas_prime-Y_prime)*(lambda_as/Cs)
34 Tas = 25.7;// [OC]
35 printf("Adiabatic Saturation Temparature: %f OC\n",
      Tas);
```

Scilab code Exa 7.11 Adiabatic Operations

```
1  clear;
2  clc;
3
4  // Illustration 7.11
5  // Page: 249
6
7  printf('Illustration 7.11 - Page: 249\n\n');
8
9  // solution
10
11 //****Data****//
12 L_min = 2.27;// [kg/square m.s]
13 G_min = 2;// [kg/square m.s]
14 L2_prime = 15;// [kg/s]
15 Q = 270;// [W]
16 Templ2 = 45;// [OC]
17 Tempg1 = 30;// [OC]
18 Tempw1 = 24;// [OC]
19 Kya = 0.90;// [kg/cubic m.s]
20 //*******//
21
22 H1_prime = 72;// [kJ/kg dry air]
23 Y1_prime = 0.0160;// [kg water/kg dry air]
24 Templ1 = 29;// [OC]
25 Cal = 4.187;// [kJ/kg]
26 // Equilibrium Data:
27 // Data = [Temp.(OC),H_star(kJ/kg)]
28 Data_star = [29 100;32.5 114;35 129.8;37.5 147;40
```

```
    166.8;42.5 191;45 216];
// The operating line for least slope:
H2_star = 209.5;// [kJ/kg]
Data_minSlope = [Templ1 H1_prime;Templ2 H2_star];
deff('[y] = f14(Gmin)','y = ((L2_prime*Cal)/Gmin)-((
    H2_star-H1_prime)/(Templ2-Templ1))');
Gmin = fsolve(2,f14);// [kg/s]
Gs = 1.5*Gmin;// [kg/s]
// For the Operating Line:
y = deff('[y] = f15(H2)','y = ((H2-H1_prime)/(Templ2
    -Templ1))-((L2_prime*Cal)/Gs)');
H2 = fsolve(2,f15);// [kJ/kg dry air]
Data_opline = [Templ1 H1_prime;Templ2 H2];
scf(4);
plot(Data_star(:,1),Data_star(:,2),Data_minSlope
    (:,1),Data_minSlope(:,2),Data_opline(:,1),
    Data_opline(:,2));
xgrid();
legend('Equilibrium line','Minimum Flow Rate Line','
    Operating Line');
xlabel("Liquid Temperature, 0C");
ylabel("Enthalphy Of Air Water vapour, kJ / kg dry
    air");
// Tower cross section Area:
Al = L2_prime/L_min;// [square m]
Ag = Gs/G_min;// [square m]
A = min(Al,Ag);// [square m]
// Data from operating line:
// Data1 = [Temp.(OC),H_prime(kJ/kg)]
Data1 = [29 72;32.5 92;35 106.5;37.5 121;40
    135.5;42.5 149.5;45 163.5];
// Driving Force:
Data2 = zeros(7,2);
// Data2 = [Temp[OC],driving Force]
for i = 1:7
    Data2(i,1) = Data1(i,1);
    Data2(i,2) = 10^2/(Data_star(i,2)-Data1(i,2));
end
```

```
59 // The data for operating line as abcissa is plotted
       against driving force;
60 Area = 3.25;
61 // From Eqn. 7.54
62 deff('[y] = f16(Z)','y = Area-(Kya*Z/G_min)');
63 Z = fsolve(2,f16);
64 printf("The height of tower is %f m\n",Z);
65 NtoG = 3.25;
66 HtoG = G_min/Kya;// [m]
67
68 // Make up water
69 // Assuming the outlet air is essentially saturated:
70 Y2_prime = 0.0475;// [kg water/kg dry air]
71 E = G_min*(A)*(Y2_prime-Y1_prime);// [kg/s]
72 // Windage loss estimated as 0.2 percent
73 W = 0.002*L2_prime;// [kg/s]
74 ppm_blowdown = 2000;// [ppm]
75 ppm_makeup = 500;// [ppm]
76 // Since the weight fraction are proportional to the
       corresponding ppm values:
77 B = (E*ppm_makeup/(ppm_blowdown-ppm_makeup))-W;// [
     kg/s]
78 M = B+E+W;// [kg/s]
79 printf("The makeup water is estimated to be %f kg/s\
     n",M);
```

Scilab code Exa 7.12 Adiabatic Operations

```
1 clear;
2 clc;
3
4 // Illustration 7.12
5 // Page: 252
6
7 printf('Illustration 7.12 - Page: 252\n\n')
```

```
// solution

//****Data****//
Tempg1 = 32;// [OC]
Tempw1 = 28;// [OC]
//******//

H1 = 90;// [kJ/kg]
H1_prime = 72;// [kJ/kg dry air]
H2_prime = 163.6;// [kJ/kg dry air]
deff('y = f17(H2)','y = (H2-H1)-(H2_prime-H1_prime)');
H2 = fsolve(2,f17);// [kJ/kg dry air]
// Slope of Operating Line same as Operating Line as Illustration 7.11
slopeOperat = (163.5-72)/(45-29);
deff("[y] = f18(Temp)","y = slopeOperat*(Temp-Tempg1)+H1");
Temp = 30:0.01:45;
// Equilibrium Data:
// Data  = [Temp.(OC),H_star(kJ/kg)]
Data_star = [29 100;32.5 114;35 129.8;37.5 147;40 166.8;42.5 191;45 216];
scf(5);
plot(Data_star(:,1),Data_star(:,2),Temp,f18);
xgrid();
legend("Equilibrium Line","operating Line");
xlabel("Liquid Temperature, C");
ylabel("Enthalphy Of Air Water vapour, kJ/kg dry air");
// The Value for NtoG & HtoG will be same as in Illustration 7.11
NtoG = 3.25;
HtoG = 2.22;// [m]
// By hit & trial method:
Temp = 37.1;// [OC]
printf("The Temperature to which water is to be cooled is %f OC\n",Temp);
```

Scilab code Exa 7.13 Recirculating Liquid Gas Humidididification

```
clear;
clc;

// Illustration 7.13
// Page: 254

printf('Illustration 7.13\n\n');

// solution

// Given
Tempg1=65;// [OC]
Y1_prime=0.0170;// [kg water/kg dry air]
// Using adiabatic satursion line on Fig. 7.5 (Pg
   232)
Tempas=32;// [OC]
Yas_prime=0.0309;// [kg water/kg dry air]
Tempg2=45;// [OC]
Z=2;// [m]
//*******//

Y2_prime=0.0265;// [kg water/kg dry air]
deff('[y]=f19(Kya_by_Gs)','y=log((Yas_prime-Y1_prime
   )/(Yas_prime-Y2_prime))-(Kya_by_Gs*Z)');
Kya_by_Gs=fsolve(1,f19);// [1/m]

// For the extended chamber:
Z=4;// [m]
deff('[y]=f20(Y2_prime)','y=log((Yas_prime-Y1_prime)
   /(Yas_prime-Y2_prime))-(Kya_by_Gs*Z)');
Y2_prime=fsolve(0.029,f20);//[kg water/kg dry air]
// With the same adiabatic curve:
```

```
30 Tempg2=34;// [OC]
31 printf("The Outlet Conditions are:\n");
32 printf("Absolute Humidity is %f kg water/kg dry air\
   n",Y2_prime);
33 printf("Dry Bulb Temperature is %f OC\n",Tempg2);
```

Scilab code Exa 7.14 Dehumidification Of Air Water Mixture

```
1  clear;
2  clc;
3
4  // Illustration 7.14
5  // Page: 256
6
7  printf('Illustration 7.14 - Page: 256\n\n');
8
9  // solution
10
11 //****Data****//
12 // a = N2 b = CO
13 // Entering gas
14 Y1_prime = 0;// [kg water/kg dry air]
15 Pt = 1;// [atm]
16 Tempg1 = 315;// [OC]
17 G_prime = 5;// [square m/s]
18
19 // Temp of the tower:
20 Templ2 = 18;// [OC]
21 Density_L2 = 1000; //[kg/square m]
22 viscocity_L2 = 1.056*10^(-3);// [kg/m.s]
23 Tempg2 = 27;// [OC]
24
25 Mb = 28;// [kg/kmol]
26 Ma = 18.02;// [kg/kmol]
27 Density_G1 = (Mb/22.41)*(273/(Tempg1+273));// [kg/
```

```
    square m]
28 G1 = G_prime*(Density_G1);// [kg/s]
29
30 // Since the outlet gas is nearly saturated:
31 Y_prime = 0.024;// [kg water/kg dry air]
32 Y2_prime = 0.022;// [kg water/kg dry air, assumed]
33 G2 = G1*(1+Y2_prime);// [kg/s]
34 Mav = (1+Y2_prime)/((1/Mb)+(Y2_prime/Ma));// [kg/
    kmol]
35 Density_G2 = (Mav/22.4)*(273/(Templ2+273));// [kg/
    square m]
36 L2_by_G2 = 2;
37 abcissa = L2_by_G2*(Density_G2/(Density_L2-
    Density_G2))^(1/2);
38 // From Fig. 6.34:
39 // For a gas pressure drop of 400 N/square m/m
40 ordinate = 0.073;
41 // From Table 6.3:
42 Cf = 65;
43 J = 1;
44 deff('[y] = f21(G2_prime)','y = ((G2_prime^2)*Cf*(
    viscocity_L2^0.1)*J/(Density_G2*(Density_L2-
    Density_G2)))-ordinate');
45 // Tentative data:
46 G2_prime = fsolve(2,f21);// [kg/square m.s]
47 Area = G1/G2_prime;// [square m]
48 dia = sqrt(4*Area/%pi);// [m]
49
50 // Final data:
51 dia = 1.50;// [m]
52 Area = %pi*dia^2/4;// [square m]
53 Gs_prime = G1/Area;// [kg/square m.s]
54 G2_prime = G2/Area;// [kg/square m.s]
55 L2_prime = L2_by_G2*G2_prime;// [kg/square m.s]
56 // From Eqn. 7.29:
57 deff('[y] = f22(L1_prime)','y = (L2_prime-L1_prime)
    -(Gs_prime*(Y2_prime-Y1_prime))');
58 L1_prime = fsolve(2,f22);
```

```
59 Cb = 1089;// [J/kg.K]
60 Ca = 1884;// [J/kg.K]
61 Cs1 = Cb+(Y1_prime*Ca);// [J/(kg dry air).K]
62 Cs2 = Cb+(Y2_prime*Ca);// [J/(kg dry air).K]
63 Tempo = Templ2;// [base temp.,K]
64 lambda = 2.46*10^6;// [J/kg]
65 CaL = 4187;// [J/kg K]
66 // From Eqn. 7.31:
67 deff('[y] = f23(Templ1)','y = ((L2_prime*CaL*(Templ2
     -Tempo))+(Gs_prime*Cs1*(Tempg1-Tempo)))-((
     L1_prime*CaL*(Templ1-Tempo))+(Gs_prime*(Cs2*(
     Tempg2-Tempo))+(Y2_prime*lambda)))');
68 Templ1 = fsolve(2,f23);
69 // At Templ1 = 49.2 OC
70 viscocity_L = 0.557*10^(-3);// [kg/m.s]
71 Density_L = 989;// [kg/square m]
72 K = 0.64;// [w/m.K]
73 Prl = CaL*viscocity_L/K;
74
75 // For Entering Gas:
76 viscocity_G1 = 0.0288*10^(-3);// [kg*/m.s]
77 Dab = 0.8089*10^(-4);// [square m/s]
78 ScG = viscocity_G1/(Density_G1*Dab);
79 PrG = 0.74;
80
81 // From Illustration 6.7:
82 a = 53.1;// [square m/square m]
83 Fga = 0.0736;// [kmol/square m]
84 Hga = 4440;// [W/square m.K]
85 Hla = 350500;// [W/square m.K]
86 // At the bottom, by several trial:
87 Tempi = 50.3;// [OC]
88 pai = 93.9/760;// [atm]
89 paG = 0;// [atm]
90 // By Eqn. 7.64:
91 dY_prime_by_dZ = -(Ma*Fga/Gs_prime)*log((1-(pai/Pt))
     /(1-(paG/Pt)));// [(kg H2O/kg dry gas)/m]
92 Hg_primea = -(Gs_prime*Ca*dY_prime_by_dZ)/(1-exp((
```

```
    Gs_prime*Ca*dY_prime_by_dZ)/(Hga)));// [W/square
    m.K]
93  dTempg_by_dZ = -(Hg_primea*(Tempg1-Tempi)/(Gs_prime*
    Cs1));// [OC/m]
94  Tempi = (Templ1)+((Gs_prime*(Cs1*dTempg_by_dZ)+((Ca
    *(Tempg1))-(CaL*(Templ1))+(((CaL-Ca)*(Tempo))+
    lambda))*dY_prime_by_dZ)/((Gs_prime*CaL*
    dY_prime_by_dZ)-Hla));// [OC]
95  // Assume:
96  delta_Tempg = -30;// [OC]
97  delta_Z = delta_Tempg/(dTempg_by_dZ);// [m]
98  Tempg = Tempg1+delta_Tempg;// [OC]
99  Y_prime = Y1_prime+(dY_prime_by_dZ)*delta_Z;// [kg
    H2O/kg dry gas]
100 paG = Y_prime/(Y_prime+(Ma/Mb));// [atm]
101 Cs = Cb+Ca*(Y_prime);// [J/(kg dry air).K]
102 // Water balance, From Eqn. 7.29:
103 deff('[y] = f24(L_prime)','y = (L2_prime-L_prime)-(
    Gs_prime*(Y_prime-Y1_prime))');
104 L_prime = fsolve(2,f24);// [kg/square m.s]
105
106 deff('[y] = f25(Templ)','y = ((L_prime*CaL*(Templ-
    Tempo))+(Gs_prime*Cs1*(Tempg1-Tempo)))-((L1_prime
    *CaL*(Templ1-Tempo))+(Gs_prime*(Cs*(Tempg-Tempo))
    +(Y_prime*lambda)))');
107 Templ = fsolve(2,f25);
108 // This process is repeated several times until gas
    temp falls to Tempg2
109 // The value of Y2_prime was calculated to be 0.0222
     which is sufficiently close to the assumed value
    .
110 // Z = sum of all delta_Z
111 Z = 1.54;// [m]
112 printf("The diameter of tower is %f m\n",dia);
113 printf("The packed height is %f m\n",Z);
```

Scilab code Exa 7.15 Nonadiabatic Operation

```
clear;
clc;

// Illustration 7.15
// Page: 267

printf('Illustration 7.15 - Page: 267\n\n');

// solution

//***Data***//
w = 0.75;// [m]
OD = 19.05/1000;// [m]
l = 3.75;// [m]
n = 20;
t = 1.65/1000;// [m]
Ws = 2.3;// [kg/s]
Wal = 10;// [kg/s]
Wt = 4;// [kg/s]
Density = 800;// [kg/cubic m]
viscocity = 0.005;// [kg/m.s]
K = 0.1436;// [W/m.K]
Ct = 2010;// [J/kg.K]
Cal = 4187;// [J/kg.K]
Y1_prime = 0.01;// [kg H2O/kg dry air]
Y2_prime = 0.06;// [kg H2O/kg dry air]
TempT = 95;// [OC]
//*****//

Free_area = (w-(n*OD))*l;// [square m]
Gs_min = 2.3/Free_area;// [kg/square m.s]
Yav_prime = (Y1_prime+Y2_prime)/2;// [kg H2O/kg dry
```

```
    air]
// From Eqn. 7.86:
ky = 0.0493*(Gs_min*(1+Yav_prime))^0.905;// [kg/
    square m.s.delta_Y_prime]
// From Fig. 7.5:
H1_prime = 56000;// [J/kg]
Ao = 400*%pi*OD*l;// [square m]
// Cooling water is distributed over 40 tubes &
    since tubes are staggered
geta = Wal/(40*2*l);// [kg/m.s]
geta_by_OD = geta/OD;// [kg/square m.s]
// Assume:
TempL = 28;// [OC]
// From Eqn. 7.84:
hL_prime = (982+(15.58*TempL))*(geta_by_OD^(1/3));//
    [W/square m.K]
// From Eqn. 7.85:
hL_dprime = 11360;// [W/square m.K]
// From Fig. 7.5 (Pg 232)
m = 5000;// [J/kg.K]
Ky = 1/((1/ky)+(m/hL_dprime));
ID = (OD-(2*t));// [m]
Ai = %pi*(ID^2)/4;// [square m]
Gt_prime = Wt/(n*Ai);// [kg/square m.s]
Re = ID*Gt_prime/viscocity;
Pr = Ct*viscocity/K;
// From a standard correlation:
hT = 364;// [W/square m.K]
Dav = (ID+OD)/2;// [m]
Zm = (OD-ID)/2;// [m]
Km = 112.5;// [W/m.K]
// From Eqn. 7.67:
Uo = 1/((OD/(ID*hT))+((OD/Dav)*(Zm/Km))+(1/hL_prime)
    );// [W/square m.K]
// From Eqn. 7.75:
alpha1 = -(((Uo*Ao)/(Wt*Ct))+((Uo*Ao)/(Wal*Cal)));
alpha2 = m*Uo*Ao/(Wt*Ct);
// From Eqn. 7.76:
```

```
66 beeta1 = Ky*Ao/(Wal*Cal);
67 beeta2 = -((m*Ky*Ao/(Wal*Cal))-(Ky*Ao/Ws));
68 y = deff('[y] = f26(r)','y = (r^2)+((alpha1+beeta2)*
      r)+((alpha1*beeta2)-(alpha2*beeta1))');
69 r1 = fsolve(10,f26);
70 r2 = fsolve(0,f26);
71 beeta2 = 1.402;
72 // From Eqn. 7.83:
73 // N1-(M1*(r1+alpha1)/beeta1) =
      0.....(1)
74 // N2-(M2*(r2+alpha2)/beeta2) =
      0.....(2)
75 // From Eqn. 7.77:
76 // At the top:
77 x1 = 1;
78 // TempL2+(M1*exp(r1*x1))+(M2*exp(-(r2*x1))) = TempL
      .....(3)
79 // From Eqn. 7.78:
80 // At the bottom:
81 x2 = 0;
82 // H1_star-N1-N2 = H1_prime
      .....(4)

83 // From Eqn. 7.80:
84 // ((M1/r1)*(exp(r1)-1))+((M2*r2)*(exp(r2)-1)) = (
      Tempt-TempL).....(5)
85 // From Eqn. 7.81:
86 // ((N1/r1)*(exp(r1)-1))+((N2*r2)*(exp(r2)-1)) = (
      H1_star-H1_prime).....(6)
87 // From Eqn. 7.91 & Eqn. 7.92:
88 // Uo*Ao*(TempT-TempL)=Ky*Ao*(H1_star-H1_prime)
      .....(7)
89
90 // Elimination of M's & N's by solving Eqn. (1) to
      (4) and (7) simultaneously:
91 // and from Fig. 7.5 (Pg 232):
92 TempL1=28;// [OC]
93 H1_star=(Uo*Ao*(TempT-TempL)/(Ky*Ao))+H1_prime;// [J
```

```
    /kmol]
// Solving (1) to (4) simultaneously:
a = [1 -(r1+alpha1)/beeta1 0 0;0 0 1 -(r2+alpha1)/
    beeta1;0 exp(r1*x1) 0 exp(r2*x1);1 0 1 0];
b = [0;0;TempT-TempL1;H1_star-H1_prime];
soln = a\b;
N1 = soln(1);
M1 = soln(2);
N2 = soln(3);
M2 = soln(4);
// By Eqn. 5
delta_Temp = ((M1/r1)*(exp(r1)-1))+((M2*r2)*(exp(r2)
    -1));// [OC]
Q = Uo*delta_Temp*Ao;
TempT1 = TempT-(Q/(Wt*Ct));// [OC]
H2_prime = Q/(Ws)+H1_prime;// [J/kg]
printf("Temparature to which oil was cooled: %f OC\n
    ",TempT1);
```

Chapter 8

Gas Absorption

Scilab code Exa 8.1 Ideal Liquid Solution

```
1  clear;
2  clc;
3
4  // Illustration 8.1
5  // Page: 278
6
7  printf('Illustration 8.1 - Page: 278\n\n');
8
9  // solution
10
11 //****Data****//
12 P_star = 2*10^(5);// [N/square m]
13 X_methane = 0.6;
14 X_ethane = 0.2;
15 X_propane = 0.08;
16 X_nbutane = 0.06;
17 X_npentane = 0.06;
18 //******//
19
20 MoleFraction = [0.6 0.2 0.08 0.06 0.06]
21 Heading = ["Component" "Equilibrium Partial Pressure
```

```
    " "Vapour Pressue    " "Mole Fraction"];
22 Component = ["Methane" "Ethane    " "Propane" "n-
    Butane" "n-Pentane"];
23 VapPressure = [0 42.05 8.96 2.36 0.66];// [N/square
    m]
24 Sum = 0;
25 for i = 1:4
26     printf("%s \t",Heading(i));
27 end
28 printf("\n");
29 for i = 1:5
30     printf("%s \t ",Component(i));
31     printf("%e \t \t \t",(MoleFraction(i)*P_star));
32     printf("%e \t \t",(VapPressure(i)*10^(5)));
33     if  (VapPressure(i) =  = 0)
34         printf("\t \n");
35         Sum = Sum+0;
36     else
37          printf("%f \n",(MoleFraction(i)*P_star)/(
               VapPressure(i)*10^(5)));
38          Sum = Sum+(MoleFraction(i)*P_star)/(
               VapPressure(i)*10^(5));
39
40 end
41 end
42 printf("Mole Fraction Of solvent Oil is %f",1-Sum);
```

Scilab code Exa 8.2 Minimum Liquid Gas Ratio for absorbers

```
1 clear;
2 clc;
3
4 // Illustration 8.2
5 // Page: 286
6
```

```
printf('Illustration 8.2 - Page: 286\n\n');

// solution

//****Data****//
// Absorber:
G = 0.250;// [cubic m/s]
Temp1 = 273+26;// [K]
Pt = 1.07*10^(5);// [N/square m]
y1 = 0.02;
x2 = 0.005;
//******//

G1 = G*(273/Temp1)*(Pt/(1.0133*10^(5)))*(1/22.41);// [kmol/s]
Y1 = y1/(1-y1);// [kmol benzene/kmol dry gas]
Gs = G1*(1-y1);// [kmol dry gas/s]
// For 95% removal of benzene:
Y2 = Y1*0.05;
X2 = x2/(1-x2);// [kmol benzene/kmol oil]
// Vapour pressure of benzene:

P_star = 13330;// [N/square m]
X_star = zeros(20);
Y_star = zeros(20);
j = 0;
for i = 0.01:0.01:0.20
    j = j+1;
    x = i;
    X_star(j) = i;
    deff('[Y] = f27(y)','Y = (y/(1+y))-(P_star/Pt)*(x/(1+x))');
    Y_star(j) = fsolve(0,f27);
end
// For min flow rate:
X1 = 0.176;// [kmolbenzene/kmol oil]
DataMinFlow = [X2 Y2;X1 Y1];
scf(6);
```

```
43 plot(X_star,Y_star,DataMinFlow(:,1),DataMinFlow(:,2)
   );
44 minLs = (Gs*(Y1-Y2)/(X1-X2));// [kmol/s]
45 // For 1.5 times the minimum:
46 Ls = 1.5*minLs;// [kmol/s]
47 X1_prime = (Gs*(Y1-Y2)/Ls)+X2;// [kmol benzene/kmol
   oil]
48 DataOperLine = [X2 Y2;X1_prime Y1];
49 plot(X_star,Y_star,DataMinFlow(:,1),DataMinFlow(:,2)
   ,DataOperLine(:,1),DataOperLine(:,2));
50 xgrid();
51 xlabel("moles of benzene / mole wash oil");
52 ylabel("moles benzene / mole dry gas");
53 legend("Equlibrium Line","Min Flow Rate Line","
   Operating Line");
54 title("Absorption")
55 printf("The Oil circulation rate is %e kmol/s\n",Ls)
   ;
56
57 // Stripping
58 Temp2 = 122+273;// [K]
59 // Vapour pressure at 122 OC
60 P_star = 319.9;// [kN/square m]
61 Pt = 101.33;// [kN/square m]
62 X_star = zeros(7);
63 Y_star = zeros(7);
64 j = 0;
65 for i = 0:0.1:0.6
66     j = j+1;
67     x = i;
68     X_star(j) = i;
69     deff('[Y] = f28(y)','Y = (y/(1+y))-(P_star/Pt)*(
         x/(1+x))');
70     Y_star(j) = fsolve(0,f28);
71 end
72 X1 = X2;// [kmol benzene/kmol oil]
73 X2 = X1_prime;// [kmol benzene/kmol oil]
74 Y1 = 0;// [kmol benzene/kmol steam]
```

```
75 // For min. steam rate:
76 Y2 = 0.45;
77 DataMinFlow = [X2 Y2;X1 Y1];
78 minGs = Ls*(X2-X1)/(Y2-Y1);// [kmol steam/s]
79 slopeOperat = 1.5*(Y2-Y1)/(X2-X1);
80 deff('[y] = f29(x)','y = slopeOperat*(x-X1)+Y1');
81 x = 0:0.01:0.14;
82 scf(7);
83 plot(Y_star,X_star,DataMinFlow(:,1),DataMinFlow(:,2)
     ,x,f29);
84 xgrid();
85 xlabel("moles of benzene / mole wash oil");
86 ylabel("moles benzene / mole dry gas");
87 legend("Equlibrium Line","Min Flow Rate Line","
     Operating Line");
88 title("Stripping");
89 printf("The Steam circulation rate is %e kmol/s\n"
     ,1.5*minGs);
```

Scilab code Exa 8.3 Countercurrent Multistage Operation

```
1  clear;
2  clc;
3
4  // Illustration 8.3
5  // Page: 292
6
7  printf('Illustration 8.3 - Page: 292\n\n');
8
9  // solution
10
11 // Since tower is a tray device:
12 // Following changes in notation is made:
13 // L1 to LNp
14 // L2 to L0
```

```
15 // X1 to XNp
16 // X2 to X0
17 // G1 to GNpPlus1
18 // G2 to G1
19 // Y1 to YNpPlus1
20 // Y2 to Y1
21 // x1 to xNp
22 // x2 to x0
23 // y1 to yNpPlus1
24 // y2 to y1
25 // From Illustration 8.2:
26 yNpPlus1 = 0.02;
27 Y1 = 0.00102;
28 y1 = Y1/(1+Y1);
29 GNpPlus1 = 0.01075;// [kmol/s]
30 x0 = 0.005;
31 m = 0.125;// [m = y_star/x]
32 Ls = 1.787*10^(-3);// [kmol/s]
33 Gs = 0.01051;// [kmol/s]
34 XNp = 0.1190;
35 LNp = Ls*(1+XNp);// [kmol/s]
36 ANp = LNp/(m*GNpPlus1);
37 X0 = x0/(1-x0);
38 L0 = Ls*(1+X0);// [kmol/s]
39 G1 = Gs*(1+Y1);// [kmol/s]
40 A1 = L0/(m*G1);
41 A = (ANp*A1)^0.5;
42 // From Eqn. 5.55:
43 Np = (log((yNpPlus1-(m*x0))/(y1-(m*x0))*(1-(1/A))
      +(1/A)))/log(A);
44 printf("Absorber\n");
45 printf("From Analytical Method, no. of theoretical
      trays required is %f \n",Np);
46 // From Fig. 8.13 (Pg292):
47 Np = 7.6;
48 printf("From Graphical Method, no. of theoretical
      trays required is %f \n",Np);
49
```

```
50 // Stripper
51 SNp = 1/ANp;
52 S1 = 1/A1;
53 // Due to relative nonconstancy of the stripping
      factor,graphical method should be used.
54 printf("Stripper\n");
55 // From Fig. 8.11 (Pg 289):
56 Np = 6.7;
57 printf("From Graphical Method, no. of theoretical
      trays required is %f \n",Np);
58 // From Fig. 5.16 (Pg 129):
59 Np = 6.0;
60 printf("From Fig. 5.16, no. of theoretical trays
      required is %f \n",Np);
```

Scilab code Exa 8.4 Nonisothermal Operation

```
1  clear;
2  clc;
3
4  // Illustration 8.4
5  // Page: 295
6
7  printf('Illustration 8.4 - Page: 295\n\n');
8
9  // solution
10
11 //****Data****//
12 // a = CH4 b = C5H12
13 Tempg = 27;// [OC]
14 Tempo = 0;// [base temp,OC]
15 Templ = 35;// [OC]
16 xa = 0.75;// [mole fraction of CH4 in gas]
17 xb = 0.25;// [mole fraction of C5H12 in gas]
18 M_Paraffin = 200;// [kg/kmol]
```

```
hb = 1.884;// [kJ/kg K]
//********//

Ha = 35.59;// [kJ/kmol K]
Hbv = 119.75;// [kJ/kmol K]
Hbl = 117.53;// [kJ/kmol K]
Lb = 27820;// [kJ/kmol]
// M = [Temp (OC) m]
M = [20 0.575;25 0.69;30 0.81;35 0.95;40 1.10;43
   1.25];
// Basis: Unit time
GNpPlus1 = 1;// [kmol]
yNpPlus1 = 0.25;// [kmol]
HgNpPlus1 = ((1-yNpPlus1)*Ha*(Tempg-Tempo))+(
   yNpPlus1*(Hbv*(Tempg-Tempo)+Lb));// [kJ/kmol]
L0 = 2;// [kmol]
x0 = 0;// [kmol]
HL0 = ((1-x0)*hb*M_Paraffin*(Templ-Tempo))+(x0*hb*(
   Templ-Tempo));// [kJ/kmol]
C5H12_absorbed = 0.98*xb;// [kmol]
C5H12_remained = xb-C5H12_absorbed;
G1 = xa+C5H12_remained;// [kmol]
y1 = C5H12_remained/G1;// [kmol]
LNp = L0+C5H12_absorbed;// [kmol]
xNp = C5H12_absorbed/LNp;// [kmol]
// Assume:
Temp1 = 35.6;// [OC]
Hg1 = ((1-y1)*Ha*(Temp1-Tempo))+(y1*(Hbv*(Temp1-
   Tempo)+Lb));// [kJ/kmol]

// Eqn. 8.11:
Qt = 0;
deff('[y] = f30(HlNp)','y = ((L0*HL0)+(GNpPlus1*
   HgNpPlus1))-((LNp*HlNp)+(G1*Hg1)+Qt)');
HlNp = fsolve(2,f30);

deff('[y] = f31(TempNp)','y = HlNp-(((1-x0)*hb*
   M_Paraffin*(TempNp-Tempo))+(x0*hb*(TempNp-Tempo))
```

```
    )');
TempNp = fsolve(35.6,f31);
// At Temp = TempNp:
mNp = 1.21;
yNp = mNp*xNp;// [kmol]
GNp = G1/(1-yNp);// [kmol]
HgNp = ((1-yNp)*Ha*(TempNp-Tempo))+(yNp*(Hbv*(TempNp
    -Tempo)+Lb));// [kJ/kmol]
// Eqn. 8.13 with n = Np-1
deff('[y] = f32(LNpMinus1)','y = LNpMinus1+GNpPlus1
    -(LNp+GNp)');
LNpMinus1 = fsolve(2,f32);// [kmol]

// Eqn. 8.14 with n = Np-1
deff('[y] = f33(xNpMinus1)','y = ((LNpMinus1*
    xNpMinus1)+(GNpPlus1*yNpPlus1))-((LNp*xNp)+(GNp*
    yNp))');
xNpMinus1 = fsolve(0,f33);// [kmol]

// Eqn. 8.15 with n = Np-1
deff('[y] = f34(HlNpMinus1)','y = ((LNpMinus1*
    HlNpMinus1)+(GNpPlus1*HgNpPlus1))-((LNp*HlNp)+(
    GNp*HgNp))');
HlNpMinus1 = fsolve(0,f34);// [kJ/kmol]
deff('[y] = f35(TempNpMinus1)','y = HlNpMinus1-(((1-
    xNpMinus1)*hb*M_Paraffin*(TempNpMinus1-Tempo))+(
    xNpMinus1*hb*(TempNpMinus1-Tempo)))');
TempNpMinus1 = fsolve(42,f35);// [OC]

// Thecomputation are continued upward through the
    tower in this manner until the gas composition
    falls atleast to 0.00662.
// Results = [Tray No.(n) Tn(OC) xn yn]
Results = [4.0 42.3 0.1091 0.1320;3 39.0 0.0521
    0.0568;2 36.8 0.0184 0.01875;1 35.5 0.00463
    0.00450];
scf(8);
plot(Results(:,1),Results(:,4));
```

```
76 xgrid();
77 xlabel('Tray Number');
78 ylabel('mole fraction of C5H12 in gas');
79
80 scf(9);
81 plot(Results(:,1),Results(:,2));
82 xgrid();
83 xlabel('Tray Number');
84 ylabel('Temparature(OC)');
85
86 // For the cquired y1
87 Np = 3.75;
88 printf("The No. of trays will be %f",Np);
```

Scilab code Exa 8.5 Real Trays and Tray Efficiency

```
1 clear;
2 clc;
3
4 // Illustration 8.5
5 // Page: 299
6
7 printf('Illustration 8.5 - Page: 299\n\n');
8
9 // solution
10
11 //****Data****//
12 // a = NH3 b = H2 c = N2 w = water
13 P = 2;// [bars]
14 Temp = 30;// [OC]
15 L = 6.38;// [kg/s]
16 W = 0.53;// [weir length,m]
17 pitch = 12.5/1000;// [m]
18 D = 0.75;// [Tower diameter,m]
19 hW = 0.060;// [weir height,m]
```

```
t = 0.5;// [tray spacing,m]
//*******//

// From Geometry of Tray Arrangement:
At = 0.4418;// [Tower Cross section,square m]
Ad = 0.0403;// [Downspout Cross section,square m]
An = At-Ad;// [square m]
Ao = 0.0393;// [perforation area,square m]
Z = 0.5307;// [distance between downspouts,square m]
z = (D+W)/2;// [average flow width,m]
h1 = 0.04;// [weir crest,m]
// From Eqn. 6.34
Weff = W*(sqrt(((D/W)^2)-((((D/W)^2-1)^0.5)+((2*h1/D
   )*(D/W)))^2));// [m]
q = Weff*(1.839*h1^(3/2));//[cubic m/s]
// This is a recommended rate because it produces
   the liquid depth on the tray to 10 cm.
Density_L = 996;// [kg/s]
Mw = 18.02;// [kg/kmol]
L1 = 6.38/Mw;// [kmol/s]
Ma = 17.03;// [kg/kmol]
Mb = 28.02;// [kg/kmol]
Mc = 2.02;// [kg/kmol]
MavG = (0.03*Ma)+(0.97*(1/4)*Mb)+(0.97*(3/4)*Mc);//
   [kg/kmol]
Density_G = (MavG/22.41)*(P/0.986)*(273/(273+Temp));
   // [kg/cubic m]
G = 0.893;// [kg/s]
sigma = 68*10^(-3);// [N/m]
abcissa = (L/G)*(Density_G/Density_L)^0.5;
// From Table 6.2 (Pg169):
alpha = 0.04893;
beeta = 0.0302;
// From Eqn. 6.30
Cf = ((alpha*log10(1/abcissa))+beeta)*(sigma/0.02)
   ^0.2;
// From Eqn. 6.29
Vf = Cf*((Density_L-Density_G)/Density_G)^(1/2);// [
```

```
    m/s]
// 80% of flooding value:
V = 0.8*Vf;// [m/s]
G = 0.8*G;// [kg/s]
G1 = G/MavG;// [kmol/s]
Vo = V*An/Ao;// [m/s]
l = 0.002;// [m]
Do = 0.00475;// [m]
// From Eqn. 6.37
Co = 1.09*(Do/l)^0.25;
viscosity_G = 1.13*10^(-5);// [kg/m.s]
Reo = Do*Vo*Density_G/viscosity_G;
// At Reynold's No. = Reo
fr = 0.0082;
g = 9.81;// [m/s^2]
// From Eqn. 6.36
deff('[y] = f36(hD)','y = (2*hD*g*Density_L/(Vo^2*
    Density_G))-(Co*(0.40*(1.25-(Ao/An))+(4*l*fr/Do)
    +(1-(Ao/An))^2))');
hD = fsolve(1,f36);
// From Eqn. 6.31;
Aa = (Ao/0.907)*(pitch/Do)^2;// [square m]
Va = V*An/Aa;// [m/s]
// From Eqn. 6.38
hL = 6.10*10^(-3)+(0.725*hW)-(0.238*hW*Va*(Density_G
    )^0.5)+(1.225*q/z);// [m]
// From Eqn. 6.42
hR = 6*sigma/(Density_L*Do*g);// m
// From Eqn. 6.35
hG = hD+hL+hR;// [m]
Al = 0.025*W;// [square m]
Ada = min(Al,Ad);
// From Eqn. 6.43
h2 = (3/(2*g))*(q/Ada)^2;// [m]
// From Eqn.6.44
h3 = hG+h2;
// since hW+h1+h3 is essentially equal to t/2,
    flooding will not occur
```

```
abcissa = (L/G)*(Density_G/Density_L)^0.5;
V_by_Vf = V/Vf;
// From Fig.6.17, V/Vf = 0.8 & abcissa = 0.239
E = 0.009;

// At the prevailing conditions:
Dg = 2.296*10^(-5);// [square m/s]
viscosity_G = 1.122*10^(-5);// [kg/m.s]
ScG = viscosity_G/(Density_G*Dg)
Dl = 2.421*10^(-9);// [square m/s]

// From Henry's Law:
m = 0.850;
A = L1/(m*G1);

// From Eqn. 6.61:
NtG = (0.776+(4.57*hW)-(0.238*Va*Density_G^0.5)
   +(104.6*q/Z))/(ScG^0.5);
// From Eqn. 6.64:
thetha_L = hL*z*Z/q;// [s]
// From Eqn. 6.62:
NtL = 40000*(Dl^0.5)*((0.213*Va*Density_G^0.5)+0.15)
   *thetha_L;
// From Eqn. 6.52:
NtoG = 1/((1/NtG)+(1/(A*NtL)));
// From Eqn. 6.51:
EoG = 1-exp(-NtoG);
// From Eqn. 6.63:
DE = ((3.93*10^(-3))+(0.0171*Va)+(3.67*q/Z)+(0.1800*
   hW))^2;// [square m/s]
// From Eqn. 6.59:
Pe = Z^2/(DE*thetha_L);
// From Eqn. 6.58:
eta = (Pe/2)*((1+(4*m*G1*EoG/(L1*Pe)))^0.5-1);
// From Eqn. 6.57:
EMG = EoG*(((1-exp(-(eta+Pe)))/((eta+Pe)*(1+(eta+Pe)
   /eta)))+((exp(eta)-1)/(eta*(1+(eta/(eta+Pe))))));
// From Eqn. 6.60:
```

```
EMGE = EMG/((1+(EMG*(E/(1-E)))));
// From Eqn. 8.16:
EO = log(1+EMGE*((1/A)-1))/log(1/A);
Np = 14*EO;
yNpPlus1 = 0.03;
x0 = 0;
// From Eqn. 5.54(a):
deff('[y] = f37(y1)','y = ((yNpPlus1-y1)/(yNpPlus1-m
   *x0))-(((A^(Np+1))-A)/((A^(Np+1))-1))');
y1 = fsolve(0.03,f37);
printf("Mole Fraction Of NH3 in effluent is %e",y1);
```

Scilab code Exa 8.6 Continuous Contact Equipment

```
clear;
clc;

// Illustration 8.6
// Page: 304

printf('Illustration 8.6 - Page: 304\n\n');

// solution

//****Data****//
// Gas:
// In:
y_prime1 = 0.02;
Y_prime1 = 0.0204;// [mol/mol dry gas]
// Out:
y_prime2 = 0.00102;
Y_prime2 = 0.00102;// [mol/mol dry gas]
// Non absorbed gas:
MavG = 11;// [kg/kmol]
G = 0.01051;// [kmol/s nonbenzene]
```

```
22 Gm = 0.01075;// [kmol/s]
23 T = 26;// [OC]
24 viscosity_G = 10^(-5);// [kg/m.s]
25 DaG = 1.30*10^(-5);// [square m/s]
26
27 // Liquid:
28 // In:
29 x_prime2 = 0.005;
30 X_prime2 = 0.00503;// [mol benzene/mol oil]
31 // Out:
32 x_prime1 = 0.1063;
33 X_prime1 = 0.1190;// [mol benzene/mol oil]
34 // Benzene free oil:
35 MavL = 260;// [kg/kmol]
36 viscosity_L = 2*10^(-3);// [kg/kmol]
37 Density_L = 840;// [kg/cubic cm]
38 L = 1.787*10^(-3);// [kmol/s]
39 DaL = 4.77*10^(-10);// [square m/s]
40 sigma = 0.03;// [N/square m]
41 m = 0.1250;
42 //*******//
43
44 A = 0.47^2*%pi/4;// [square m]
45 // At the bottom:
46 L_prime1 = ((L*MavL)+(X_prime1*L*78))/A;// [kg/
      square m.s]
47 // At the top
48 L_prime2 = ((L*MavL)+(X_prime2*L*78))/A;// [kg/
      square m.s]
49 L_primeav = (L_prime1+L_prime2)/2;// [kg/square m.s]
50 // At the bottom
51 G_prime1 = ((G*MavG)+(Y_prime1*G*78))/A;// [kg/
      square m.s]
52 // At the top
53 G_prime2 = ((G*MavG)+(Y_prime2*G*78))/A;// [kg/
      square m.s]
54 G_primeav = (G_prime1+G_prime2)/2;// [kg/square m.s]
55
```

```
56 // From Illustration 6.6:
57 Fga = 0.0719;// [kmol/cubic cm.s]
58 Fla = 0.01377;// [kmol/cubic cm.s]
59 // Operating Line:
60 X_prime = [0.00503 0.02 0.04 0.06 0.08 0.10 0.1190];
61 x_prime = zeros(7);
62 Y_prime = zeros(7);
63 y_prime = zeros(7);
64 for i = 1:7
65     x_prime(i) = X_prime(i)/(1+X_prime(i));
66     deff('[y] = f38(Y_prime)','y = (G*(Y_prime1-
         Y_prime))-(L*(X_prime1-X_prime(i)))');
67     Y_prime(i) = fsolve(Y_prime1,f38);
68     y_prime(i) = Y_prime(i)/(1+Y_prime(i));
69 end
70 deff("[y] = f39(x)","y = m*x")
71 x = [0:0.01:0.14];
72
73 // Interface compositions are determined graphically
     and according to Eqn. 8.21:
74 yi = [0.000784 0.00285 0.00562 0.00830 0.01090
    0.01337 0.01580];
75 ylog = zeros(7);
76 y_by_yDiffyi = zeros(7);
77 for i = 1:7
78     ylog(i) = log10(yi(i));
79     y_by_yDiffyi(i) = y_prime(i)/(y_prime(i)-yi(i));
80 end
81 scf(10);
82 plot(x_prime,y_prime,x,f39,x_prime,yi);
83 legend("Operating Line","Equilibrium Line","
    Interface Composition");
84 xgrid();
85 xlabel("mole fraction of benzene in liquid");
86 ylabel("mole fraction of benzene in gas");
87 scf(11);
88 plot(ylog,y_by_yDiffyi);
89 xgrid();
```

```
xlabel("log y");
ylabel("y/(y-yi)");
title("Graphical Integration Curve");
// Area under the curve:
Ac = 6.556;
// Eqn. 8.28:
NtG = (2.303*Ac)+1.152*(log10((1-y_prime2)/(1-
   y_prime1)));
Gav = (Gm+(G/(1-Y_prime2)))/(2*A);// [kmol/square m.
   s]
HtG = Gav/Fga;// [m]
Z = HtG*NtG;// [m]
printf("The depth of packing recquired is %f m",Z);
```

Scilab code Exa 8.7 Overall height of Transfer Units

```
clear;
clc;

// Illustration 8.7
// Page: 312

printf('Illustration 8.7 - Page: 312\n\n');

// solution

// Fom Illustration 8.6:
y1 = 0.02;
y2 = 0.00102;
m = 0.125;
x2 = 0.005;
x1 = 0.1063;

// Number of transfer units:
// Method a:
```

```
y1_star = m*x1;
y2_star = m*x2;
yDiffy_star1 = y1-y1_star;
yDiffy_star2 = y2-y2_star;
yDiffy_starm = (yDiffy_star1-yDiffy_star2)/log(
   yDiffy_star1/yDiffy_star2);
// From Eqn. 8.48:
NtoG = (y1-y2)/yDiffy_starm;
printf("NtoG according to Eqn. 8.48: %f\n",NtoG);

// Mehod b:
// From Illustration 8.3:
A = 1.424;
NtoG = (log((((y1-(m*x2))/(y2-(m*x2)))*(1-(1/A)))
   +(1/A)))/(1-(1/A));
printf("NtoG according to Eqn. 8.50: %f\n",NtoG);

// Method c:
// Operating Line:
// From Illustration 8.3:
X_prime = [0.00503 0.02 0.04 0.06 0.08 0.10 0.1190];
x_prime = [0.00502 0.01961 0.0385 0.0566 0.0741
   0.0909 0.1063]
Y_prime = [0.00102 0.00357 0.00697 0.01036 0.01376
   0.01714 0.0204];
y_prime = [0.00102 0.00356 0.00692 0.01025 0.01356
   0.01685 0.0200];
deff("[y] = f2(x)","y = m*x")
x = [0:0.01:0.14];
scf(12);
plot(x_prime,y_prime,x,f2);
legend("Operating Line","Equilibrium Line",);
xgrid();
xlabel("mole fraction of benzene in liquid");
ylabel("mole fraction of benzene in gas");
// From graph:
NtoG = 8.7;
printf("NtoG from graph: %f\n",NtoG);
```

```
53
54 // Method d:
55 // from Fig 8.10:
56 Y_star = [0.000625 0.00245 0.00483 0.00712 0.00935
      0.01149 0.01347];
57 ordinate = zeros(7);
58 for i = 1:7
59     ordinate(i) = 1/(Y_prime(i)-Y_star(i));
60 end
61 scf(13);
62 plot(Y_prime,ordinate);
63 xgrid();
64 xlabel("Y");
65 ylabel("1/(Y-Y*)");
66 title("Graphical Integration");
67 // Area under the curve:
68 Ac = 8.63;
69 // From Eqn. 8.36:
70 NtoG = Ac+(1/2)*log((1+y2)/(1+y1));
71 printf("NtoG from graphical integration: %f\n",NtoG)
      ;
72
73 // Height of transfer units:
74 NtoG = 9.16;
75 // From Illustration 6.6:
76 Fga = 0.0719;// [kmol/cubic m.s]
77 Fla = 0.01377;// [kmol/cubic m.s]
78 Gav = 0.0609;// [kmol/square m.s]
79 L = 1.787*10^(-3);// [kmol/s]
80 X1 = x1/(1-x1);
81 X2 = x2/(1-x2);
82 Area = 0.1746;// [square m]
83 Lav = L*((1+X1)+(1+X2))/(2*Area);
84 // From Eqn. 8.24:
85 Htg = Gav/Fga;// [m]
86 // From Eqn. 8.31:
87 Htl = Lav/Fla;// [m]
88 // since Solutions are dilute:
```

```
89 HtoG = Htg+Htl/A;// [m]
90 printf("HtoG: %f m\n",HtoG);
91 Z = HtoG*NtoG;// [m]
92 printf("The depth of packing recquired is %f m",Z);
```

Scilab code Exa 8.8 Adiabatic Absorption and Stripping

```
1  clear;
2  clc;
3
4  // Illustration 8.8
5  // Page: 317
6
7  printf('Illustration 8.8 - Page: 317\n\n');
8
9  // Solution
10
11 //***Data***
12 // a:NH3 b:air c:H2O
13 ya = 0.416;// [mole fraction]
14 yb = 0.584;// [mole fraction]
15 G1 = 0.0339;// [kmol/square m.s]
16 L1 = 0.271;// [kmol/square m.s]
17 TempG1 = 20;// [OC]
18 //********//
19
20 // At 20 OC
21 Ca = 36390;// [J/kmol]
22 Cb = 29100;// [J/kmol]
23 Cc = 33960;// [J/kmol]
24 lambda_c = 44.24*10^6;// [J/kmol]
25 // Enthalpy base = NH3 gas, H2O liquid, air at 1 std
       atm.
26 Tempo = 20;// [OC]
27 lambda_Ao = 0;// [J/kmol]
```

```
lambda_Co = 44.24*10^6;// [J/kmol]

// Gas in:
Gb = G1*yb;// [kmol air/square m.s]
Ya1 = ya/(1-ya);// [kmol NH3/kmol air]
yc1 = 0;// [mole fraction]
Yc1 = yc1/(1-yc1);// [kmol air/kmol NH]
// By Eqn 8.58:
Hg1 = (Cb*(TempG1-Tempo))+(Ya1*(Ca*(TempG1-Tempo))+
   lambda_Ao)+(Yc1*(Cc*(TempG1-Tempo)+lambda_Co));//
    [J/kmol air]

// Liquid in:
xa1 = 0;// [mole fraction]
xc1 = 1;// [mole fraction]
Hl1 = 0;// [J/kmol air]

//Gas out:
Ya2 = Ya1*(1-0.99);// [kmol NH3/kmol air]
// Assume:
TempG2 = 23.9;// [OC]
yc2 = 0.0293;
deff('[y] = f(Yc2)','y = yc2-(Yc2/(Yc2+Ya2+1))');
Yc2 = fsolve(0.002,f);// [kmol H2O/kmol air]
Hg2 = (Cb*(TempG2-Tempo))+(Ya2*(Ca*(TempG2-Tempo))+
   lambda_Ao)+(Yc2*(Cc*(TempG2-Tempo)+lambda_Co));//
    [J/kmol air]

// Liquid out:
Lc = L1-(Yc1*Gb);// [kmol/square m.s]
La = Gb*(Ya1-Ya2);// [kmol/square m.s]
L2 = La+Lc;// [kmol/square m.s]
xa = La/L2;
xc = Lc/L2;
// At xa & tempo = 20 OC
delta_Hs = -1709.6*1000;// [J/kmol soln]

// Condition at the bottom of the tower:
```

```
// Assume:
TempL = 41.3;// {OC}
// At(TempL+TempG1)/2:
Cl = 75481;// [J/kmol]
deff('[y] = f40(Cl)','y = Hl1+Hg1-((Gb*Hg2)+(L2*(Cl
   *(TempL-Tempo)+delta_Hs)))');
Cl = fsolve(7,f40);// [J/kmol.K]

// For the Gas:
MavG = 24.02;// [kg/kmol]
Density_G = 0.999;// [kg/cubic m]
viscosity_G  =  1.517*10^(-5);// [kg/m.s]
kG  =  0.0261;// [W/m.K]
CpG  =  1336;// [J/kg.K]
Dab  =  2.297*10^(-5);// [square m/s]
Dac  =  3.084*10^(-5);// [square m/s]
Dcb  =  2.488*10^(-5);// [square m/s]
PrG  =  CpG*viscosity_G/kG;

// For the liquid:
MavL  =  17.97;// [kg/kmol]
Density_L  =  953.1;// [kg/cubic m]
viscosity_L  =  6.408*10^(-4);// [kg/m.s]
Dal  =  3.317*10^(-9);// [square m/s]
kl = 0.4777;// [W/m.K]
ScL = viscosity_L/(Density_L*Dal);
PrL = 5.72;
sigma = 3*10^(-4);
G_prime = G1*MavG;// [kg/square m.s]
L_prime = L2*MavL;// [kg/square m.s]
// From data of Chapter 6:
Ds = 0.0472;// [m]
a = 57.57;// [square m/cubic m]
shiLt = 0.054;
e = 0.75;
// By Eqn. 6.71:
eLo = e-shiLt;
// By Eqn. 6.72:
```

```
kL = (25.1*Dal/Ds)*(Ds*L_prime/viscosity_L)^0.45*ScL^0.5;// [m/s]
c = Density_L/MavL;// [kmol/cubic m]
Fl = kL*c;// [kmol/cubic m]
// The heat mass transfer analogy of Eqn. 6.72:
hL = (25.1*kl/Ds)*(Ds*L_prime/viscosity_L)^0.45*PrL^0.5;// [m/s]
// The heat transfer analogy of Eqn. 6.69:
hG = (1.195*G_prime*CpG/PrG^(2/3))*(Ds*G_prime/(viscosity_G*(1-eLo)))^(-0.36);// [W/square m.K]
// To obtain the mass transfer coeffecients:
Ra = 1.4;
Rc = 1-Ra;
// From Eqn. 8.83:
Dam = (Ra-ya)/(Ra*((yb/Dab)+((ya+yc1)/Dac))-(ya/Dac));// [square m/s]
Dcm = (Rc-yc1)/(Rc*((yb/Dcb)+((ya+yc1)/Dac))-(yc1/Dac));// [square m/s]
ScGa = viscosity_G/(Density_G*Dam);
ScGc = viscosity_G/(Density_G*Dcm);
// By Eqn. 6.69:
FGa = (1.195*G1/ScGa^(2/3))*(Ds*G_prime/(viscosity_G*(1-eLo)))^(-0.36);// [kmol/square m.K]
FGc = (1.195*G1/ScGc^(2/3))*(Ds*G_prime/(viscosity_G*(1-eLo)))^(-0.36);// [kmol/square m.K]
Ra = Ra-0.1;
// From Eqn. 8.80:
scf(14);
for i = 1:3
    deff('[yai] = f41(xai)','yai = Ra-(Ra-ya)*((Ra-xa)/(Ra-xai))^(Fl/FGa)');
    xai = xa:0.01:0.10;
    plot(xai,f41)
    Ra = Ra+0.1;
end
xgrid();
xlabel("Mole fraction NH3 in the liquid, xa");
ylabel("Mole fraction NH3 in the gas ya");
```

```
129 title("Operating Line curves");
130 Rc = Rc-0.1;
131 // From Eqn. 8.81:
132 scf(15);
133 for i = 1:3
134     deff('[yci] = f42(xci)','yci = Rc-(Rc-yc1)*((Rc-
        xc)/(Rc-xci))^(Fl/FGc)');
135     xci = xc:-0.01:0.85;
136     plot(xci,f42)
137     Rc = Rc+0.1;
138 end
139 xgrid();
140 xlabel("Mole fraction H2O in the liquid, xc");
141 ylabel("Mole fraction H2O in the gas, yc");
142 title("Operating line Curves");
143 // Assume:
144 Tempi = 42.7;// [OC]
145 // The data of Fig. 8.2 (Pg 279) & Fig 8.4 (Pg 319)
    are used to draw the eqb curve of Fig 8.25 (Pg
    320).
146 // By interpolation of operating line curves with
    eqb line and the condition: xai+xci = 1;
147 Ra = 1.38;
148 Rc = 1-Ra;
149 xai = 0.0786;
150 yai = f41(xai);
151 xci = 1-xai;
152 yci = f42(xci);
153 // From Eqn. 8.77:
154 dYa_By_dZ = -(Ra*FGa*a/Gb)*log((Ra-yai)/(Ra-ya));//
    [kmol H2O/kmol air]
155 // From Eqn. 8.78:
156 dYc_By_dZ = -(Rc*FGc*a/Gb)*log((Rc-yci)/(Rc-yc1));//
    [kmol H2O/kmol air]
157 // From Eqn. 8.82:
158 hGa_prime = -(Gb*((Ca*dYa_By_dZ)+(Cc*dYc_By_dZ)))
    /(1-exp(Gb*((Ca*dYa_By_dZ)+(Cc*dYc_By_dZ))/(hG*a)
    ));// [W/cubic m.K]
```

```
159 // From Eqn. 8.79:
160 dtG_By_dZ = -(hGa_prime*(TempG1-Tempi))/(Gb*(Cb+(Ya1
      *Ca)+(Yc1*Cc)));// [K/m]
161 // When the curves of Fig. 8.2 (pg 279) & 8.24 (Pg
      319) are interpolated for concentration xai and
      xci, the slopes are:
162 mar = 0.771;
163 mcr = 1.02;
164 lambda_c = 43.33*10^6;// [J/kmol]
165 // From Eqn. 8.3:
166 Hai = Ca*(Tempi-Tempo)+lambda_Ao-(mar*lambda_c);// [
      J/kmol]
167 Hci = Cc*(Tempi-Tempo)+lambda_Co-(mcr*lambda_c);// [
      J/kmol]
168 // From Eqn. 8.76
169 Tempi2 = TempL+(Gb/(hL*a))*(((Hai-Ca*(TempG1-Tempo)-
      lambda_Ao)*dYa_By_dZ)+((Hci-Cc*(TempG1-Tempo)-
      lambda_Co)*dYc_By_dZ)-((Cb+(Ya1*Ca)+(Yc1*Cc))*
      dtG_By_dZ));// [OC]
170 // The value of Tempi obtained is sufficiently close
       to the value assumed earlier.
171
172 deltaYa=-0.05;
173 // An interval of deltaYa up the tower
174 deltaZ = deltaYa/(dYa_By_dZ);// [m]
175 deltaYc = (dYc_By_dZ*deltaZ);
176 // At this level:
177 Ya_next = Ya1+deltaYa;// [kmol/kmol air]
178 Yc_next = Yc1+deltaYc;// [kmol H2O/kmol air]
179 tG_next = TempG1+(dtG_By_dZ*deltaZ);// [OC]
180 L_next = L1+Gb*(deltaYa+deltaYc);// [kmol/square m.s
      ]
181 xa_next = ((Gb*deltaYa)+(L1*xa))/L_next;// [mole
      fraction NH3]
182 Hg_next = (Cb*(tG_next-Tempo))+(Ya_next*(Ca*(tG_next
      -Tempo))+lambda_Ao)+(Yc_next*(Cc*(tG_next-Tempo)+
      lambda_Co));// [J/kmol air]
183 Hl_next = (L1*Hl1)+(Gb*(Hg_next-Hg2)/L_next);// [J/
```

```
    kmol]
// The calculation are continued where the specified
    gas outlet composition are reached.
// The packed depth is sum of all deltaZ
Z = 1.58;// [m]
printf("The packed depth is: %f m\n",Z);
```

Scilab code Exa 8.9 Multicomponent Sysems

```
clear;
clc;

// Illustration 8.9
// Page: 327

printf('Illustration 8.9 - Page: 327\n\n');

// solution

//****Data****//
// C1=CH4 C2=C2H6 C3=n-C3H8 C4=C4H10
Abs=0.15;// [Total absorption,kmol]
T=25;// [OC]
y1=0.7;// [mol fraction]
y2=0.15;// [mol fraction]
y3=0.10;// [mol fraction]
y4=0.05;// [mol fraction]
x1=0.01;// [mol fraction]
x_involatile=0.99;// [mol fraction]
L_by_G=3.5;// [mol liquid/mol entering gas]
//******//

LbyG_top=L_by_G/(1-y2);
LbyG_bottom=(L_by_G+y2)/1;
LbyG_av=(LbyG_top+LbyG_bottom)/2;
```

```
27 // The number of eqb. trays is fixed by C3
      absorption:
28 // For C3 at 25 OC;
29 m=4.10;
30 A=LbyG_av/m;
31 Frabs=0.7;// [Fractional absorption]
32 X0=0;
33 // From Eqn. 8.109:
34 deff('[y]=f43(Np)','y=Frabs-((A^Np)-A)/((A^Np)-1)');
35 Np=fsolve(2,f43);
36 printf("Number of trays required is %f \n",Np);
```

Chapter 9

Distillation

Scilab code Exa 9.1 Raoults law

```
clear;
clc;

// Illustration 9.1
// Page: 349

printf('Illustration 9.1 - Page: 349\n\n');

// solution

//****Data****//
// a:n-heptane b:n-octane
Pt = 760; // [mm Hg]
//*****//

Tempa = 98.4;// [boiling point of A,OC]
Tempb = 125.6;// [boiling point of B,OC]
x = zeros(6);
y_star = zeros(6);
alpha = zeros(6);
// Data =  [Temp Pa (mm Hg) Pb(mm Hg)]
```

```
22 Data = [98.4 760 333;105 940 417;110 1050 484;115
      1200 561;120 1350 650;125.6 1540 760];
23 for i = 1:6
24     x(i) = (Pt-Data(i,3))/(Data(i,2)-Data(i,3));// [
          mole fraction of heptane in liquid]
25     y_star(i) = (Data(i,2)/Pt)*x(i);
26     alpha(i) = Data(i,2)/Data(i,3);
27 end
28 printf("T(OC)\t\t\t Pa(mm Hg)\t\t\t Pb(mm Hg)\t\t\t
      x\t\t\t\t y*\t\t\t alpha\n");
29 for i = 1:6
30     printf("%f\t \t %d\t \t \t \t %d\t \t \t \t %f\t
          \t \t %3f\t \t %f\t \t\n",Data(i,1),Data(i
         ,2),Data(i,3),x(i),y_star(i),alpha(i));
31 end
```

Scilab code Exa 9.2 Azeotropes

```
1  clear;
2  clc;
3
4  // Illustration 9.2
5  // Page: 354
6
7  printf('Illustration 9.2 - Page: 354\n\n');
8
9  // solution
10
11 //****Data****//
12 // a:water b:ethylaniline
13 Pt = 760; // [mm Hg]
14 ma1 = 50;// [g]
15 mb1 = 50;// [g]
16 //*******//
17
```

```
18 // Data =  [Temp Pa(mm Hg) Pb(mm Hg)]
19 Data = [38.5 51.1 1;64.4 199.7 5;80.6 363.9 10;96.0
      657.6 20;99.15 737.2 22.8;113.2 1225 40];
20 Ma = 18.02;// [kg/kmol]
21 Mb = 121.1;// [kg/kmol]
22
23 for i = 1:6
24     p = Data(i,2)+Data(i,3);
25     if p =  = Pt
26         pa = Data(5,2);// [mm Hg]
27         pb = Data(i,3);// [mm Hg]
28         T = Data(i,1);// [OC]
29     end
30 end
31 ya_star = pa/Pt;
32 yb_star = pb/Pt;
33 ya1 = ma1/Ma;// [g mol water]
34 yb1 = mb1/Mb;// [g mol ethylalinine]
35 Y = ya1*(yb_star/ya_star);// [g mol ethylalinine]
36 printf("The original mixture contained %f g mol
      water and %f g mol ethylalinine\n",ya1,yb1);
37 printf("The mixture will continue to boil at %f OC,
      where the equilibrium vapour of the indicated
      composition,until all the water evaporated
      together with %f g mol ethylalinine\n",T,Y);
38 printf("The temparature will then rise to 204 OC,
      and the equilibrium vapour will be of pure
      ethylalinine");
```

Scilab code Exa 9.3 Multicomponent Sysems

```
1 clear;
2 clc;
3
4 // Illustration 9.3
```

```
// Page: 362

printf('Illustration 9.3 - Page: 362\n\n');

// solution

//****Data****//
// a:n-C3H8 b:n-C4H10 c:n-C5H12 d:n-C6H14
// Bubble Point Calculation
xa = 0.05;
xb = 0.30;
xc = 0.40;
xd = 0.25;
P = 350;// [kN/square m]
//******//

// Assume:
Temp = 60;// [OC]
x = [0.05 0.30 0.40 0.25];
m = [4.70 1.70 0.62 0.25];// [At 60 OC]
// Reference: C5H12
mref = m(3);
Sum = 0;
alpha = zeros(4)
alpha_x = zeros(4);
for i = 1:4
    alpha(i) = m(i)/m(3);
    alpha_x(i) = alpha(i)*x(i);
    Sum = Sum+alpha_x(i);
end
// From Eqn. 9.23:
SumF = Sum;
Sum = 0;
mref = 1/SumF;
// Corresponding Temparature from the nomograph:
Temp = 56.8;// [OC]
m = [4.60 1.60 0.588 0.235];// [At 56.8 OC]
for i = 1:4
```

```
43      alpha(i) = m(i)/m(3);
44      alpha_x(i) = alpha(i)*x(i);
45      Sum = Sum+alpha_x(i);
46  end
47  SumF = Sum;
48  mref = 1/SumF;
49  //  Corresponding Temparature from the nomograph:
50  Temp = 56.7;// [OC]
51  Bt = 56.8;// [OC]
52  yi = zeros(4);
53  for i = 1:4
54      yi(i) = alpha_x(i)/Sum;
55  end
56  printf("The Bubble Point is %f OC\n",Bt);
57  printf("Bubble point vapour composition \n");
58  printf("\t    yi\n");
59  printf("n-C3\t %f\n",yi(1));
60  printf("n-C4\t %f\n",yi(2));
61  printf("n-C5\t %f\n",yi(3));
62  printf("n-C6\t %f\n",yi(4));
63
64  printf("\n \n \n");
65
66  // Dew Point Calculation
67  // Asume:
68  ya = 0.05;
69  yb = 0.30;
70  yc = 0.40;
71  yd = 0.25;
72  Temp = 80;// [OC]
73  y = [0.05 0.30 0.40 0.25];
74  m = [6.30 2.50 0.96 0.43];// [At 60 OC]
75  // Reference: C5H12
76  mref = m(3);
77  Sum = 0;
78  alpha = zeros(4)
79  alpha_y = zeros(4);
80  for i = 1:4
```

```
81      alpha(i) = m(i)/m(3);
82      alpha_y(i) = y(i)/alpha(i);
83      Sum = Sum+alpha_y(i);
84  end
85
86  // From Eqn. 9.29:
87  SumF = Sum;
88  Sum = 0;
89  mref = SumF;
90  // Corresponding Temparature from the nomograph:
91  Temp = 83.7;// [OC]
92  m = [6.60 2.70 1.08 0.47];// [At 56.8 OC]
93  for i = 1:4
94      alpha(i) = m(i)/m(3);
95      alpha_y(i) = y(i)/alpha(i);
96      Sum = Sum+alpha_y(i);
97  end
98  SumF = Sum;
99  mref = 1/SumF;
100 //  Corresponding Temparature from the nomograph:
101 Temp = 84;// [OC]
102 Dt = 84;// [OC]
103 xi = zeros(4);
104 for i = 1:4
105     xi(i) = alpha_y(i)/Sum;
106 end
107 printf("The Dew Point is %f OC\n",Dt);
108 printf("Dew point liquid composition \n");
109 printf("\t    xi\n");
110 printf("n-C3\t %f\n",xi(1));
111 printf("n-C4\t %f\n",xi(2));
112 printf("n-C5\t %f\n",xi(3));
113 printf("n-C6\t %f\n",xi(4));
```

Scilab code Exa 9.4 Partial Condensation

```
clear;
clc;

// Illustration 9.4
// Page: 365

printf('Illustration 9.4 - Page: 365\n\n');

// solution

//****Data****//
// Basis:
F = 100;// [mol feed]
zF = 0.5;
D = 60;// [mol]
W = 40;// [mol]
//*******//

// From Illustration 9.1, Equilibrium data:
Data = [1 1;0.655 0.810;0.487 0.674;0.312
   0.492;0.1571 0.279;0 0];
Feed = [0 0;1 1];
// The operating line is drawn with a slope -(W/D)
   to cut the equilibrium line.
deff('[y] = f44(x)','y = -((W/D)*(x-zF))+zF');
x = 0.2:0.1:0.6;
scf(16);
plot(Data(:,1),Data(:,2),Feed(:,1),Feed(:,2),x,f44);
xgrid();
xlabel("Mole fraction of heptane in liquid");
ylabel("Mole fraction of heptane in vapour");
legend("Equilibrium Line","Feed Line","Operating
   Line");
// The point at which the operating line cuts the
   equilibrium line has the following composition*
   temparature:
yd = 0.575;// [mole fraction heptane in vapour phase
   ]
```

```
33 xW = 0.387;// [mole fraction heptane in liquid phase
   ]
34 Temp = 113;// [OC]
35 printf("mole fraction of heptane in vapour phase %f
   \n",yd);
36 printf("mole fraction of heptane in liquid phase %f\
   n",xW);
37 printf("Temparature is %d OC\n",Temp);
```

Scilab code Exa 9.5 Multicomponent Systems Ideal Solution

```
1 clear;
2 clc;
3
4 // Illustration 9.5
5 // Page: 366
6
7 printf('Illustration 9.5 - Page: 366\n\n');
8
9 // solution
10
11 //****Data****//
12 Pt = 760;// [mm Hg]
13 zFa = 0.5;// [mol fraction benzene]
14 zFb = 0.25;// [mol fraction toulene]
15 zFc = 0.25;// [mol fraction o-xylene]
16 //********//
17
18 // Basis:
19 F = 100;// [mol feed]
20 // For Summtion of Yd_star to be unity, W/D = 2.08
21 // The Eqn.are
22 // (1): W+D = F
23 // (2): W-2.08D = 0
24 a = [1 1;1 -2.08];
```

```
25 b = [F;0];
26 soln = a\b;
27 W = soln(1);
28 D = soln(2);
29 Sub = ['A','B','C'];
30 p = [1370 550 200];// [mm Hg]
31 m = zeros(3);
32 zF = [zFa zFb zFc];// [Given]
33 yd_star = zeros(3);
34 xW = zeros(3);
35 for i = 1:3
36     m(i) = p(i)/Pt;
37     yd_star(i) = zF(i)*((W/D)+1)/(1+(W/(D*m(i))));
38     xW(i) = yd_star(i)/m(i);
39 end
40 printf("\t \t \t \t \t \t \t \t  At W/D = 2.08\n\n\n
   ");
41 printf("Substance \t \t p(mm Hg)\t \t m\t \t \t \t
   zF\t \t \t \t yd*\t\t\t\txW\n");
42 for i = 1:3
43     printf("%c\t \t \t %d\t \t \t %f\t \t \t %f\t \t
          \t %f \t \t \t%f\n",Sub(i),p(i),m(i),zF(i),
       yd_star(i),xW(i));
44 end
```

Scilab code Exa 9.6 Differential Distillation

```
1 clear;
2 clc;
3
4 // Illustration 9.6
5 // Page: 370
6
7 printf('Illustration 9.6 - Page: 370\n\n');
8
```

```
// solution

//****Data****//
// Basis:
F = 100;// [mol]
xF = 0.5;
D = 0.6*100;// [mol]
//******//

W = F-D;// [mol]
// From Illustration 9.1:
alpha = 2.16;// [average value of alpha]
// From Eqn.9.46;
deff('[y] = f45(xW)','y = log(F*xF/(W*xW))-(alpha*
    log(F*(1-xF)/(W*(1-xW))))');
xW = fsolve(0.5,f45);// [mole fraction heptane]
deff('[y] = f46(yD)','y = F*xF-((D*yD)+(W*xW))');
yD = fsolve(100,f46);// [mole fraction heptane]
printf("Mole Fraction of heptane in the distillate
    is %f \n",yD);
printf("Mole Fraction of heptane in the residue is
    %f \n",xW);
```

Scilab code Exa 9.7 Multicomponent Systems Ideal Solution

```
clear;
clc;

// Illustration 9.7
// Page: 371

printf('Illustration 9.7 - Page: 371\n\n');

// solution

```

```
11 //****Data****//
12 // a:benzene b:toulene c:o-xylene
13 // Assume:
14 Bt = 100;//[OC]
15 pa = 1370;// [mm Hg]
16 pb = 550;// [mm Hg]
17 pc = 200;// [mm Hg]
18 xFa = 0.5;// [mole fraction]
19 xFb = 0.25;// [mole fraction]
20 xFc = 0.25;// [mole fraction]
21 // Basis:
22 F = 100;// [mol]
23 D = 32.5;// [mol]
24 //*******//
25
26 ref = pb;
27 alpha_a = pa/ref;
28 alpha_b = pb/ref;
29 alpha_c = pc/ref;
30 W = F-D;// [mol]
31 xbW = 0.3;// [mol]
32 xaW = 0.4;// [mol]
33 xcW = 0.3;// [mol]
34 err = 1;
35 while(err>(10^(-1)))
36     // From Eqn. 9.47:
37     deff('[y] = f47(xaW)','y = log(F*xFa/(W*xaW))-(
       alpha_a*log(F*xFb/(W*xbW)))');
38     xaW = fsolve(xbW,f47);
39     deff('[y] = f48(xcW)','y = log(F*xFc/(W*xcW))-(
       alpha_c*log(F*xFb/(W*xbW)))');
40     xcW = fsolve(xbW,f48);
41     xbW_n = 1-(xaW+xcW);
42     err = abs(xbW-xbW_n);
43     xbw = xbW_n;
44 end
45 // Material balance:
46 // for A:
```

```
47 deff('[y] = f49(yaD)','y = F*xFa-((D*yaD)+(W*xaW))')
   ;
48 yaD = fsolve(100,f49);// [mole fraction benzene]
49 // For B:
50 deff('[y] = f50(ybD)','y = F*xFb-((D*ybD)+(W*xbW))')
   ;
51 ybD = fsolve(100,f50);// [mole fraction toulene]
52 // For C:
53 deff('[y] = f51(ycD)','y = F*xFc-((D*ycD)+(W*xcW))')
   ;
54 ycD = fsolve(100,f51);// [mole fraction o-xylene]
55 printf("The residual compositions are:\n");
56 printf("Benzene:%f\n",xaW);
57 printf("Toulene:%f\n",xbW);
58 printf("o-xylene:%f\n",xcW);
59 printf("The composited distillate compositions are:\
   n");
60 printf("Benzene:%f\n",yaD);
61 printf("Toulene:%f\n",ybD);
62 printf("o-xylene:%f\n",ycD);
```

Scilab code Exa 9.8 Optimum Reflux Ratio

```
1  clear;
2  clc;
3
4  // Illustration 9.8
5  // Page: 388
6
7  printf('Illustration 9.8 - Page: 388\n\n');
8
9  // solution
10
11 //****Data*****//
12 // a:methanol b:water
```

```
13 Xa = 0.5;// [Wt fraction]
14 Temp1 = 26.7;// [OC]
15 Temp2 = 37.8;// [OC]
16 F1 = 5000;// [kg/hr]
17 //******//
18
19 //(a)
20 Ma = 32.04;// [kg/kmol]
21 Mb = 18.02;// [kg/kmol]
22 Xa = 0.5;// [Wt fraction]
23 Xb = 1-Xa;// [Wt fraction]
24 Temp1 = 26.7;// [OC]
25 Temp2 = 37.8;// [OC]
26 F1 = 5000;// [kg/hr];
27 // Basis: 1hr
28 F = (F1*Xa/Ma)+(F1*Xb/Mb);// [kmol/hr]
29 // For feed:
30 zF = (F1*Xa/Ma)/F;// [mole fracton methanol]
31 MavF = F1/F;// [kg/kmol]
32 // For distillate:
33 xD = (95/Ma)/((95/Ma)+(5/Mb));// [mole fraction
   methanol]
34 MavD = 100/((95/Ma)+(5/Mb));// [kg/kmol]
35 // For residue:
36 xW = (1/Ma)/((1/Ma)+(99/Mb));// [mole fraction
   methanol]
37 MavR = 100/((1/Ma)+(99/Mb));// [kg/kmol]
38 // (1): D+W = F [Eqn.9.75]
39 // (2): D*xD+W*xW = F*zF [Eqn. 9.76]
40 // Solvving simultaneously:
41 a = [1 1;xD xW];
42 b = [F;F*zF];
43 soln = a\b;
44 D = soln(1);// [kmol/h]
45 W = soln(2);// [kmol/h]
46 printf("Quantity of Distillate is %f kg/hr\n",D*MavD
   );
47 printf("Quantity of Residue is %f kg/hr\n",W*MavR);
```

```
48 printf("\n");
49
50 // (b)
51 // For the vapour-liquid equilibria:
52 Tempo = 19.69;// [Base Temp. according to "
      International Critical Tables"]
53 BtR = 99;// [Bubble point of the residue, OC]
54 hR = 4179;// [J/kg K]
55 hF = 3852;// [J/kg K]
56 deff('[y] = f52(tF)','y = (F1*hF*(tF-Temp1))-((W*
      MavR)*hR*(BtR-Temp2))');
57 tF = fsolve(Temp1,f52);// [OC]
58 BtF = 76;// [Bubble point of feed, OC]
59 // For the feed:
60 delta_Hs = -902.5;// [kJ/kmol]
61 Hf = ((hF/1000)*MavF*(tF-Tempo))+delta_Hs;// [kJ/
      kmol]
62 // From Fig 9.27:
63 HD = 6000;// [kJ/kmol]
64 HLo = 3640;// [kJ/kmol]
65 HW = 6000;// [kJ/kmol]
66 printf("The enthalpy of feed is %f kJ/kmol\n",Hf);
67 printf("The enthalpy of the residue is %f kJ/kmol\n"
      ,HW);
68 printf("\n");
69
70 // (c)
71 // From Fig.9.27:
72 // The miium reflux ratio is established by the tie
      line (x = 0.37 y = 0.71), which extended pass
      through F,the feed.
73 // At Dm:
74 Qm = 62570;// [kJ/kmol]
75 Hg1 = 38610;// [kJ/kmol]
76 // From Eqn. 9.65:
77 Rm = (Qm-Hg1)/(Hg1-HLo);
78 printf("The minimum reflux ratio is %f\n",Rm);
79 printf("\n");
```

```
80
81 // (d)
82 // From Fig. 9.28:
83 Np = 4.9;
84 // But it include the reboiler.
85 Nm = Np-1;
86 printf("The minimum number of theoretical trys
      required is %f \n",Nm);
87 printf("\n");
88
89 // (e)
90 R = 1.5*Rm;
91 // Eqn. 9.65:
92 deff('[y] = f53(Q_prime)','y = R-((Q_prime-Hg1)/(Hg1
      -HLo))');
93 Q_prime = fsolve(2,f53);// [kJ/kmol]
94 deff('[y] = f54(Qc)','y = Q_prime-(HD+(Qc/D))');
95 Qc = fsolve(2,f54);// [kJ/hr]
96 Qc = Qc/3600;// [kW]
97 printf("The Condensor heat load is %f kW\n",Qc);
98 // From Eqn. 9.77:
99 deff('[y] = f55(Q_dprime)','y = (F*Hf)-((D*Q_prime)
      +(W*Q_dprime))');
100 Q_dprime = fsolve(2,f55);
101 deff('[y] = f56(Qb)','y = Q_dprime-(HW-(Qb/W))');
102 Qb = fsolve(2,f56);// [kJ/hr]
103 Qb = Qb/3600;// [kW]
104 printf("The Reboiler heat load is %f kW\n",Qb);
105 printf("\n");
106
107 // (f)
108 // From Fig: 9.28
109 Np = 9;
110 // But it is including the reboiler
111 printf("No. of theoretical trays in tower is %d\n",
      Np-1);
112 G1 = D*(R+1);// [kmol/hr]
113 Lo = D*R;// [kmol/hr]
```

```
114 // From Fig. 9.28:
115 // At the feed tray:
116 x4 = 0.415;
117 y5 = 0.676;
118 x5 = 0.318;
119 y6 = 0.554;
120 // From Eqn. 9.64:
121 deff('[y] = f57(L4)','y = (L4/D)-((xD-y5)/(y5-x4))')
    ;
122 L4 = fsolve(2,f57);// [kmol/hr]
123 // From Eqn. 9.62:
124 deff('[y] = f58(G5)','y = (L4/G5)-((xD-y5)/(xD-x4))'
    );
125 G5 = fsolve(2,f58);// [kmol/hr]
126 // From Eqn. 9.74:
127 deff('[y] = f59(L5_bar)','y = (L5_bar/W)-((y6-xW)/(
    y6-x5))');
128 L5_bar = fsolve(2,f59);// [kmol/hr]
129 // From Eqn. 9.72:
130 deff('[y] = f60(G6_bar)','y = (L5_bar/G6_bar)-((y6-
    xW)/(x5-xW))');
131 G6_bar = fsolve(2,f60);// [kmol/hr]
132 // At the bottom:
133 // Material Balance:
134 // Eqn. 9.66:
135 // (1): L8_bar-GW_bar = W;
136 // From Fig. 9.28:
137 yW = 0.035;
138 x8 = 0.02;
139 // From Eqn. 9.72:
140 L8ByGW_bar = (yW-xW)/(x8-xW);
141 // (2): L8_bar-(L8ByGW_bar*Gw_bar) = 0
142 a = [1 -1;1 -L8ByGW_bar];
143 b = [W;0];
144 soln = a\b;
145 L8_bar = soln(1);// [kmol/h]
146 GW_bar = soln(2);// [kmol/h]
147 printf("The Liquid quantity inside the tower is %f
```

```
    kmol/hr\n",L8_bar);
148 printf("The vapour quantity inside the tower is %f
    kmol/hr\n",GW_bar);
149 printf("\n");
```

Scilab code Exa 9.9 Use of Open Steam

```
1  clear;
2  clc;
3
4  // Illustration 9.9
5  // Page: 395
6
7  printf('Illustration 9.9 - Page: 395\n\n');
8
9  // solution
10
11 //****Data****//
12 P = 695;// [kN/square m]
13 //********//
14
15 // a:methanol b:water
16 // From Illustration 9.8:
17 Ma = 32.04;// [kg/kmol]
18 Mb = 18.02;// [kg/kmol]
19 F = 216.8;// [kmol/h]
20 Tempo = 19.7;// [OC]
21 zF = 0.360;// [mole fraction methanol]
22 HF = 2533;// [kJ/kmol]
23 D = 84.4;// [kkmol/h]
24 zD = 0.915;// [mole fraction methanol]
25 HD = 3640;// [kJ/kmol]
26 Qc = 5990000;// [kJ/h]
27 // Since the bottom will essentially be pure water:
28 HW = 6094;// [kJ/kmol]
```

```
// From Steam tables:
Hs = 2699;// [enthalpy of saturated steam, kJ/kg]
hW = 4.2*(Tempo-0);// [enthalpy of water, kJ/kg]
HgNpPlus1 = (Hs-hW)*Mb;// [kJ/kmol]
// (1): GNpPlus1-W = D-F [From Eqn. 9.86]
// (2): (GNpPlus1*HgNpPlus1)-(W*HW) = (D*HD)+Qc-(F*
   HF) [From Eqn. 9.88]
a = [1 -1;HgNpPlus1 -HW];
b = [D-F;(D*HD)+Qc-(F*HF)];
soln = a\b;
GNpPlus1 = soln(1);// [kmol/h]
W = soln(2);// [kmol/h]
// From Eqn. 9.87:
deff('[y] = f61(xW)','y = (F*zF)-((D*zD)+(W*xW))');
xW = fsolve(2,f61);
// The enthalpy of the solution at its bubble point
   is 6048 kJ/kmol, sufficiently closed to 6094
   assumed earlier.
// For delta_w:
xdelta_w = W*xW/(W-GNpPlus1);
Q_dprime = ((W*HW)-(GNpPlus1*HgNpPlus1))/(W-GNpPlus1
   );// [kJ/kmol]
// From Fig 9.27 ad Fig. 9.28, and for the stripping
    section:
Np = 9.5;
printf("Steam Rate: %f kmol/h\n",GNpPlus1);
printf("Bottom Composition: xW: %f\n",xW);
printf("Number of theoretical stages: %f\n",Np);
```

Scilab code Exa 9.10 Optimum Reflux Ratio McCabe Thiele Method

```
clear;
clc;

// Illustration 9.10
```

```
5 // Page: 412
6
7 printf('Illustration 9.10 - Page: 412\n\n');
8
9 // solution
10
11 // a:methanol b:water
12 Ma = 32.04;// [kg/kmol]
13 Mb = 18.02;// [kg/kmol]
14 // Feed:
15 F1 = 5000;// [kg/h]
16 F = 216.8;// [kmol/h]
17 Tempo = 19.7;// [OC]
18 zF = 0.360;// [mole fraction methanol]
19 MavF = 23.1;// [kg/kmol]
20 Tempf = 58.3;// [OC]
21 // Distillate:
22 D1 = 2620;// [kg/h]
23 D = 84.4;// [kkmol/h]
24 xD = 0.915;// [mole fraction methanol]
25 // Residue:
26 R1 = 2380;// [kg/h]
27 R = 132.4;// [kmol/h]
28 xW = 0.00565;// [mole fraction methanol]
29
30 // From Fig. 9.42 (Pg 413):
31 BtF = 76.0;// [Bubble point if the feed, OC]
32 DtF = 89.7;// [Dew point of the feed, OC]
33 // Latent heat of vaporisation at 76 OC
34 lambda_a = 1046.7;// [kJ/kg]
35 lambda_b = 2284;// [kJ/kg]
36 ha = 2.721;// [kJ/kg K]
37 hb = 4.187;// [kJ/kg K]
38 hF = 3.852;// [kJ/kg K]
39 // If heats of solution is ignaored:
40 // Enthalpy of the feed at the bubble point referred
       to the feed temp.
41 HF = hF*MavF*(BtF-Tempf);// [kJ/kmol]
```

```
42 // enthalpy of the saturated vapour at dew point
   referred to the liquid at feed temp.
43 HL = (zF*((ha*Ma*(DtF-Tempf))+(lambda_a*Ma)))+((1-zF
   )*((hb*Mb*(DtF-Tempf))+(lambda_b*Mb)));// [kJ/
   kmol]
44 q = HL/(HL-HF);
45 slope = q/(q-1);
46 // In fig. 9.42: xD,xW & zF are  located on the 45
   degree diagonal & the q line is drawn with slope
   = 'slope' .
47 // The operating line for minimum reflux ratio in
   this case pass through the intersection of the q
   line and the equilibrium curve.
48 ordinate = 0.57;
49 deff('[y] = f62(Rm)','y = ordinate-(xD/(Rm+1))');
50 Rm = fsolve(0,f62);// [mole reflux/mole distillate]
51 // from fig. 9.42 (Pg 413):
52 // The minimum number of theoretical trays is
   determied using the 45 degree diagonal as
   operating line.
53 Np = 4.9;// [including the reboiler]
54 R = 1.5*Rm;// [mole reflux/mole distillate]
55 // From Eqn. 9.49:
56 L = R*D;// [kmol/h]
57 // From Eqn. 9.115:
58 G = D*(R+1);// [kmol/h]
59 // From Eqn. 9.126:
60 L_bar = (q*F)+L;// [kmol/h]
61 // From Eqn. 9.127:
62 G_bar = (F*(q-1))+G;// [kmol/h]
63 ordinateN = xD/(R+1);
64 // As in Fig. 9.43:
65 // The y-intercept = ordinateN and enriching and
   exhausting operating lines are plotted.
66 // Number of theoretical stages are determined.
67 NpN = 8.8;// [including the reboiler]
68 printf("Number of theoretical stages is %f\n",NpN-1)
   ;
```

Scilab code Exa 9.11 Suitable Reflux Ratio

```
clear;
clc;

// Illustration 9.11
// Page: 423

printf('Illustration 9.11 - Page: 423\n\n');

// solution

//****Data****//
// a:ethanol b:water
zF = 0.3;
xa = 0.3;// [mole fraction of ethanol]
Temp = 78.2;// [OC]
Ao = 0.0462;// [Area of perforations,square m]
t = 0.450;// [m]
//******//

Ma = 46.05;// [kg/kmol]
Mb = 18.02;// [kg/kmol]
xb = 1-xa;// [mole fraction of water]
ma = 0.3*Ma/((0.3*Ma)+(xb*Mb));// [mass fraction of
   ethanol]
mb = 1-ma;// [mass fraction of water]


// Feed:
F1 = 910;// [kg/h]
Xa = F1*ma/Ma;// [moles of ethanol]
Xb = F1*mb/Mb;// [moles of water]
F = Xa+Xb;// [Total moles]
```

```
// Distillate:
xD = 0.80;// [mole fraction of ethanol]
// If essentially all the ethanol is removed from
   the residue:
D = Xa/xD;// [kmol/h]
MavD = (xD*Ma)+((1-xD)*Mb);// [kg/kmol]
D1 = D*MavD;// [kg/h]
Density_G = (MavD/22.41)*(273/(273+Temp));// [kg/
   cubic meter]
Density_L = 744.9;// [kg/cubic meter]
sigma = 0.021;// [N/m]

// From Table 6.2,Pg 169:
alpha = (0.0744*t)+0.01173;
beeta = (0.0304*t)+0.015;
At = %pi*(0.760^2)/4;// [Tower cross sectional Area,
    square m]
WByT = 530/760;// [Table 6.1, Pg 162]
Ad = 0.0808*At;// [Downspout area,square m]
Aa = At-(2*Ad);//  [Active area,square m]
// abcissa = (L/G)*(density_G/Density_L)^0.5
// Assume:
abcissa = 0.1;
// From Eqn.6.30:
Cf = (alpha*log10(1/abcissa)+beeta)*(sigma/0.020)
   ^0.2;
// From Eqn. 6.29:
Vf = Cf*((Density_L-Density_G)/Density_G)^(1/2);// [
   m/s]
An = At-Ad;// [square m]
R = 3;// [Reflux Ratio]
G = D*(R+1);
G1 = (G*22.41/3600)*((273+Temp)/273);// [cubic meter
   /s]
V = G1/An;// [Vapour velocity,m/s]
percent = (V/Vf)*100;
// Vapour velocity is 58 percent of flooding
   velocity (amply safe)
```

```
63 L = R*D;// [kmol/h]
64 L1 = L*MavD;// [kg/h]
65 abcissa = (L1/(G1*3600*Density_G))*(Density_G/
     Density_L)^0.5;
66 // Since the value of abcissa is less than0.1, the
     calculaed value of Cf is correct.
67 // Since the feed is at the buubble point.
68 q = 1;
69 // From Eqn. 9.126:
70 L_bar = L+(q*F);// [kmol/h]
71 // From Eqn. 9.127:
72 G_bar = G+F*(q-1);// [kmol/h]
73 // The enthalpy of saturated steam,referred to 0 OC
     ,69 kN/square m:
74 HGNpPlus1 = 2699;// [kN m/kg]
75 // This will be the enthalpy as it enters the tower
     if expanded adiabatically to the tower pressure
76 // The enthalpy of steam at 1 std. atm:
77 HGsat = 2676;// [kN m/kg]
78 lambda = 2257;// [kN m/kg]
79 // From Eqn. 9.140:
80 deff('[y] = f63(GNpPlus1_bar)','y = G_bar-(
     GNpPlus1_bar*(1+((HGNpPlus1-HGsat)*Mb/(lambda*Mb)
     )))');
81 GNpPlus1_bar = fsolve(7,f63);
82 // From Eqn. 9.141:
83 LNp_bar = L_bar-(G_bar-GNpPlus1_bar);
84
85 // Tray Efficiencies:
86 // Consider the situation:
87 x = 0.5;
88 y_star = 0.962;
89 Temp = 79.8;// [OC]
90 // This is in the enriching section.
91 Density_L = 791;// [kg/cubic meter]
92 Density_G = 1.253;// [kg/cubic meter]
93 // From equilibrium data:
94 m = 0.42;
```

```
A = L/(m*G);
// From chapter 2:
ScG = 0.930;
Dl = 2.065*10^(-9);// [square m/s]
// For L = 38.73 kmol/h
q = 4.36*10^(-4);// [cubic meter/s]
// For G = 51.64 kmol/h
Va = 1.046;// [m/s]
// From tray dimensions:
z = 0.647;// [m]
Z = 0.542;// [m]
hW = 0.06;// [m]
// From Eqn. 6.61:
NtG = (0.776+(4.57*hW)-(0.238*Va*Density_G^0.5)
   +(104.6*q/Z))/(ScG^0.5);
// From Eqn. 6.38
hL = 6.10*10^(-3)+(0.725*hW)-(0.238*hW*Va*(Density_G
   )^0.5)+(1.225*q/z);// [m]
// From Eqn. 6.64:
thetha_L = hL*z*Z/q;// [s]
// From Eqn. 6.62:
NtL = 40000*(Dl^0.5)*((0.213*Va*Density_G^0.5)+0.15)
   *thetha_L;
// From Eqn. 6.52:
NtoG = 1/((1/NtG)+(1/(A*NtL)));
// From Eqn. 6.51:
EoG = 1-exp(-NtoG);
// From Eqn. 6.63:
DE = ((3.93*10^(-3))+(0.0171*Va)+(3.67*q/Z)+(0.1800*
   hW))^2;
// From Eqn. 6.59:
Pe = Z^2/(DE*thetha_L);
// From Eqn. 6.58:
eta = (Pe/2)*((1+(4*m*G1*EoG/(L1*Pe)))^0.5-1);
// From Eqn. 6.57:
EMG = EoG*(((1-exp(-(eta+Pe)))/((eta+Pe)*(1+(eta+Pe)
   /eta)))+((exp(eta)-1)/(eta*(1+(eta/(eta+Pe))))));
// Entrainment is neglible:
```

```
128 // Similarly for other x
129 // Value = [x Entrainment]
130 Value = [0 0.48;0.1 .543;0.3 0.74;0.5 EMG;0.7 0.72];
131
132 // Tray Calculation:
133 op_intercept = xD/(R+1);
134 // From Fig. 9.48:
135 // The exhausting section operating line, on this
      scale plot, for all practical purposes passes
      through the origin.
136 // The broken curve is located so that, at each
      concentration, vertical distances corresponding
      to lines BC and AC are in the ratio of EMG.
137 // This curve is used instead of equilibrium trays
      to locate the ideal trays.
138 // The feed tray is thirteenth.
139 x14 = 0.0150;
140 alpha = 8.95;
141 EMG = 0.48;
142 A_bar = L_bar/(alpha*G_bar);
143 // From Eqn. 8.16:
144 Eo = log(1+(EMG*((1/A_bar)-1)))/log(1/A_bar);
145 // The 6 real trays corresponds to:
146 NRp = 6*Eo;
147 xW = 0.015/((exp(NRp*log(1/A_bar))-A_bar)/(1-A_bar))
      ;// [mole fraction ethanol]
148 // This corresponds to ethanol loss of 0.5 kg/day.
149 printf("The Reflux ratio of %d will cause the
      ethanol loss of 0.5 kg/day\n",R);
150 printf("Larger reflux ratios would reduce this, but
      the cost of additional steam will probaby make
      them not worthwile.\n");
151 printf("Smaller values of R, with corresponding
      reduced steam cost and larger ethanol loss,
      should be considered, but care must be taken to
      ensure vapour velocities above the weeping
      velocities.");
```

Scilab code Exa 9.12 Dimension of Packed Section

```
clear;
clc;

// Illustration 9.12
// Page: 429

printf('Illustration 9.12 - Page: 429\n\n');

// solution

// a:methanol b:water
// Vapour and liquid quantities throughout the tower, as in Illustration 9.8, with the Eqn. 9.62, 9.64, 9.72, 9.74:
// Data = [x tL(OC) y tG(OC) Vapor(kmol/h) Vapor(kg/h) Liquid(kmol/h) Liquid(kg/h)]
Ma = 34.02;// [kg/kmol]
Mb = 18.02;// [kg/kmol]
Temp = 78.7;// [OC]
x = [0.915 0.600 0.370 0.370 0.200 0.100 0.02];
y = [0.915 0.762 0.656 0.656 0.360 0.178 0.032];
scf(17);
plot(x,y);
xgrid();
xlabel("mole fraction of methanol in liquid");
ylabel("mole fraction of methanol in vapour");
title("Operating Line curve");
//x = 0.370: the dividing point between stripping and enriching section
tL = [66 71 76 76 82 87 96.3];// [Bubble point, OC]
tG = [68.2 74.3 78.7 78.7 89.7 94.7 99.3];// [Dew Point, OC]
```

```
Vapor = [171.3 164.0 160.9 168.6 161.6 160.6 127.6]; // [kmol/h]
Vapor1 = [5303 4684 4378 4585 3721 3296 2360];// [kg/h]
Liquid = [86.7 79.6 76.5 301 294 293 260];// [kmol/h]
Liquid1 = [2723 2104 1779 7000 6138 5690 4767];// [kg/h]
Data = zeros(7,8);
for j = 1:7
        Data(j,1) = x(j);
        Data(j,2) = tL(j);
        Data(j,3) = y(j);
        Data(j,4) = tG(j);
        Data(j,5) = Vapor(j);
        Data(j,6) = Vapor1(j);
        Data(j,7) = Liquid(j);
        Data(j,8) = Liquid1(j);
end
// The tower diameter will be set by the conditions at the top of the stripping section because of the large liquid flow at this point.
// From Illustration 9.8:
G = Data(4,6);
L = Data(4,8);
Density_G = (Data(4,6)/(22.41*Data(4,5)))*(273/(273+Temp));// [kg/cubic m]
Density_L = 905;// [kg/cubic m]
// abcissa = (L/G)*(Density_L/Density_G)^0.5
abcissa = (Data(4,8)/Data(4,6))*(Density_G/Density_L)^0.5;
// From Fig. 6.34, choose a gas pressure drop of 450 N/square m/m
ordinate = 0.0825;
// From Table 6.3 (Pg 196):
Cf = 95;
viscosity_L = 4.5*10^(-4);// [kg/m.s]
sigma = 0.029;// [N/m]
```

```
57 J = 1;
58 G_prime = (ordinate*Density_G*(Density_L-Density_G)
     /(Cf*viscosity_L^0.1))^0.5;// [kg/square m.s]
59 A = G/(3600*G_prime);// [Tower ,cross section area,
     square m]
60 L_prime = L/(A*3600);// [kg/square m.s]
61 // Mass transfer will be computed for the same
     location:
62 // From Table 6.4 (Pg 205):
63 m = 36.4;
64 n = (0.0498*L_prime)-0.1013;
65 p = 0.274;
66 aAW = m*((808*G_prime/Density_G^0.5)^n)*L_prime^p;//
      [square m/cubic m]
67 // From Table 6.5 (Pg 206):
68 dS = 0.0530;// [m]
69 beeta = 1.508*dS^0.376;
70 shi_LsW = 2.47*10^(-4)/dS^1.21;
71 shi_LtW = ((2.09*10^(-6))*(737.5*L_prime)^beeta)/dS
     ^2;
72 shi_LOW = shi_LtW-shi_LsW;
73 shi_Ls = (0.0486*viscosity_L^0.02*sigma^0.99)/(dS
     ^1.21*Density_L^0.37);
74 H = ((975.7*L_prime^0.57*viscosity_L^0.13)/(
     Density_L^0.84*((2.024*L_prime^0.430)-1)))*(sigma
     /0.073)^(0.1737-0.262*log10(L_prime));// [m]
75 shi_Lo = shi_LOW*H;
76 shi_Lt = shi_Lo+shi_Ls;
77 // From Eqn. 6.73:
78 aA = aAW*(shi_Lo/shi_LOW);// [square m/cubic m]
79 // From Table 6.3 (Pg 196):
80 e = 0.71;
81 // From Eqn. 6.71:
82 eLo = e-shi_Lt;
83 // From Chapter 2:
84 ScG = 1;
85 MavG = 0.656*Ma+(1-0.656)*Mb;// [kg/kmol]
86 G = G_prime/MavG;
```

```
viscosity_G = 2.96*10^(-5);// [kg/m.s]
// From Eqn. 6.70:
Fg = (1.195*G/ScG^(2/3))*((dS*G_prime/(viscosity_G
   *(1-eLo)))^(-0.36));// [kmol/square m s (mole
   fraction)]
kY_prime = Fg;// [kmol/square m s (mole fraction)]
DL = 4.80*10^(-9);// [square m/s]
ScL = viscosity_L/(Density_L*DL);
// From Eqn. 6.72:
kL = (25.1*DL/dS)*((dS*L_prime/viscosity_L)^0.45)*
   ScL^0.5;// [kmol/square m s (kmol/cubic m)]
// At 588.33 OC
Density_W = 53.82;// [kg/cubic m]
kx_prime = Density_W*kL;// [kmol/square m s (mole
   fraction)]
// Value1 = [x G a ky_prime*10^3 kx_prime]
Value1 = [0.915 0.0474 20.18 1.525 0.01055;0.6
   0.0454 21.56 1.542 0.00865;0.370 0.0444 21.92
   1.545 0.00776;0.370 0.0466 38 1.640 0.0143;0.2
   0.0447 32.82 1.692 0.0149;0.1 0.0443 31.99 1.766
   0.0146;0.02 0.0352 22.25 1.586 0.0150];
// From Fig: 9.50
// At x = 0.2:
y = 0.36;
slope = -(Value1(5,5)/(Value1(5,4)*10^(-3)));
// The operating line drawn from(x,y) with slope.
   The point where it cuts the eqb. line gives yi.
// K = ky_prime*a(yi-y)
// For the enriching section:
// En = [y yi 1/K Gy]
En = [0.915 0.960 634 0.0433;0.85 0.906 532.8
   0.0394;0.8 0.862 481.1 0.0366;0.70 0.760 499.1
   0.0314;0.656 0.702 786.9 0.0292];
// For the Stripping section:
// St = [y yi 1/K Gy]
St = [0.656 0.707 314.7 0.0306;0.50 0.639 124.6
   0.0225;0.40 0.580 99.6 0.01787;0.3 0.5 89
   0.0134;0.2 0.390 92.6 0.00888;0.10 0.232 154.5
```

```
    0.00416;0.032 0.091 481 0.00124];
// Graphical Integration, according to Eqn.9.52::
scf(18);
plot(En(:,4),En(:,3));
xgrid();
xlabel("Gy");
ylabel("1 / (ky_prime*a*(yi-y))");
title("Graphical Integration for Enriching section");
// From Area under the curve:
Ze = 7.53;// [m]
// Graphical Integration:
scf(19);
plot(St(:,4),St(:,3));
xgrid();
xlabel("Gy");
ylabel("1 / (ky_prime*a*(yi-y))");
title("Graphical Integration for Stripping section");
// From Area under the curve:
Zs = 4.54;// [m]
// Since the equlibrium curve slope varies so
   greatly that the use of overall mass transfer
   coeffecient is not recommended:
printf("Height of Tower for enriching Section is %f
   m\n",Ze);
printf("Height of Tower for Stripping Section is %f
   m\n",Zs);
```

Scilab code Exa 9.13 Multicomponent Systems

```
clear;
clc;

// Illustration 9.13:
```

```
5
6  printf('Illustration 9.13\n\n');
7
8  //**************************Calculation Of Minimum
       Reflux ratio************************//
9  // Page: 436
10 printf('Page: 436\n\n');
11
12 //***Data***//
13 // C1:CH4 C2:C2H6 C3:n-C3H8 C4:n-C4H10 C5:n-C5H12 C6
       :n-C6H14
14 // zF = [zF(C1) zF(C2) zF(C3) zF(C4) zF(C5) zF(C6)]
15 zF = [0.03 0.07 0.15 0.33 0.30 0.12];// [mole
       fraction]
16 LF_By_F = 0.667;
17 Temp = 82;// [OC]
18 ylk = 0.98;
19 yhk = 0.01;
20 //**********//
21
22 // Data = [m HG HL(30 OC);m HG HL(60 OC);m HG HL(90
      OC);m HG HL(120 OC);]
23 Data1 = [16.1 12790 9770;19.3 13910 11160;21.8 15000
       12790;24.0 16240 14370];// [For C1]
24 Data2 = [3.45 22440 16280;4.90 24300 18140;6.25
       26240 19890;8.15 28140 21630];// [For C2]
25 Data3 = [1.10 31170 16510;2.00 33000 20590;2.90
       35800 25600;4.00 39000 30900];// [For C3]
26 Data4 = [0.35 41200 20350;0.70 43850 25120;1.16
       46500 30000;1.78 50400 35400];// [For C4]
27 Data5 = [0.085 50500 24200;0.26 54000 32450;0.50
       57800 35600;0.84 61200 41400];// [For C5]
28 Data6 = [0.0300 58800 27700;0.130 63500 34200;0.239
       68150 40900;0.448 72700 48150];// [For C6]
29
30 // T = [Temparature]
31 T = [30;60;90;120];
32
```

```
33 // Flash vaporisation of the Feed:
34 // Basis: 1 kmol feed throughout
35 // After Several trials, assume:
36 F = 1;// [kmol]
37 GF_By_F = 0.333;
38 LF_By_GF = LF_By_F/GF_By_F;
39 m82 = zeros(6);
40 y = zeros(6);
41 m82(1) = interpln([T';Data1(:,1)'],Temp);// [For C1]
42 m82(2) = interpln([T';Data2(:,1)'],Temp);// [For C2]
43 m82(3) = interpln([T';Data3(:,1)'],Temp);// [For C3]
44 m82(4) = interpln([T';Data4(:,1)'],Temp);// [For C4]
45 m82(5) = interpln([T';Data5(:,1)'],Temp);// [For C5]
46 m82(6) = interpln([T';Data6(:,1)'],Temp);// [For C6]
47 for i = 1:6
48     y(i) = zF(i)*(LF_By_GF+1)/(1+(2/m82(i)));
49 end
50 Sum = sum(y);
51 // Since Sum is sufficiently close to 1.0, therefore
     :
52 q = 0.67;// [LF_By_F]
53 // Assume:
54 // C3: light key
55 // C5: heavy key
56 zlkF = zF(3);// [mole fraction]
57 zhkF = zF(5);// [mole fraction]
58 ylkD = ylk*zF(3);// [kmol]
59 yhkD = yhk*zF(5);// [kmol]
60
61 // Estimate average Temp to be 80 OC
62 m80 = zeros(6);
63 alpha80 = zeros(6);
64 m80(1) = interpln([T';Data1(:,1)'],80);// [For C1]
65 m80(2) = interpln([T';Data2(:,1)'],80);// [For C2]
66 m80(3) = interpln([T';Data3(:,1)'],80);// [For C3]
67 m80(4) = interpln([T';Data4(:,1)'],80);// [For C4]
68 m80(5) = interpln([T';Data5(:,1)'],80);// [For C5]
69 m80(6) = interpln([T';Data6(:,1)'],80);// [For C6]
```

```
70 for i = 1:6
71     alpha80(i) = m80(i)/m80(5);
72 end
73 // By Eqn. 9.164:
74 yD_By_zF1 = (((alpha80(1)-1)/(alpha80(3)-1))*(ylkD/
      zF(3)))+(((alpha80(3)-alpha80(1))/(alpha80(3)-1))
      *(yhkD/zF(5)));// [For C1]
75 yD_By_zF2 = (((alpha80(2)-1)/(alpha80(3)-1))*(ylkD/
      zF(3)))+(((alpha80(3)-alpha80(2))/(alpha80(3)-1))
      *(yhkD/zF(5)));// [For C2]
76 yD_By_zF6 = (((alpha80(6)-1)/(alpha80(3)-1))*(ylkD/
      zF(3)))+(((alpha80(3)-alpha80(6))/(alpha80(3)-1))
      *(yhkD/zF(5)));// [For C6]
77 // The distillate contains:
78 yC1 = 0.03;// [kmol C1]
79 yC2 = 0.07;// [kmol C2]
80 yC6 = 0;// [kmol C6]
81 // By Eqn 9.165:
82 deff('[y] = g1(phi)','y = (((alpha80(1)*zF(1))/(
      alpha80(1)-phi))+((alpha80(2)*zF(2))/(alpha80(2)-
      phi))+((alpha80(3)*zF(3))/(alpha80(3)-phi))+((
      alpha80(4)*zF(4))/(alpha80(4)-phi))+((alpha80(5)*
      zF(5))/(alpha80(5)-phi))+((alpha80(6)*zF(6))/(
      alpha80(6)-phi)))-(F*(1-q))');
83 // Between alphaC3 & alphaC4:
84 phi1 = fsolve(3,g1);
85 // Between alphaC4 & alphaC5:
86 phi2 = fsolve(1.5,g1);
87 // From Eqn. 9.166:
88 // Val = D*(Rm+1)
89 // (alpha80(1)*yC1/(alpha80(1)-phi1))+(alpha80(2)*
      yC2/(alpha80(2)-phi1))+(alpha80(3)*ylkD/(alpha80
      (3)-phi1))+(alpha80(4)*yD/(alpha80(4)-phi1))+(
      alpha80(i)*yhkD/(alpha80(5)-phi1))+(alpha80(6)*
      yC6/(alpha80(6)-phi1)) = Val
      ......................(1)
90 // (alpha80(1)*yC1/(alpha80(1)-phi2))+(alpha80(2)*
      yC2/(alpha80(2)-phi2))+(alpha80(3)*ylkD/(alpha80
```

```
    (3)-phi2))+(alpha80(4)*yD/(alpha80(4)-phi2))+(
    alpha80(i)*yhkD/(alpha80(5)-phi2))+(alpha80(6)*
    yC6/(alpha80(6)-phi2)) = Val
    .....(2)
// Solving simultaneously:
a = [-alpha80(4)/(alpha80(4)-phi1) 1;-alpha80(4)/(
    alpha80(4)-phi2) 1];
b = [(alpha80(1)*yC1/(alpha80(1)-phi1))+(alpha80(2)*
    yC2/(alpha80(2)-phi1))+(alpha80(3)*ylkD/(alpha80
    (3)-phi1))+(alpha80(i)*yhkD/(alpha80(5)-phi1))+(
    alpha80(6)*yC6/(alpha80(6)-phi1));(alpha80(1)*yC1
    /(alpha80(1)-phi2))+(alpha80(2)*yC2/(alpha80(2)-
    phi2))+(alpha80(3)*ylkD/(alpha80(3)-phi2))+(
    alpha80(i)*yhkD/(alpha80(5)-phi2))+(alpha80(6)*
    yC6/(alpha80(6)-phi2))];
soln = a\b;
yC4 = soln(1);// [kmol C4 in the distillate]
Val = soln(2);
// For the distillate, at a dew point of 46 OC
ydD = [yC1 yC2 ylkD yC4 yhkD yC6];
D = sum(ydD);
yD = zeros(6);
m46 = zeros(6);
alpha46 = zeros(6);
m46(1) = interpln([T';Data1(:,1)'],46);// [For C1]
m46(2) = interpln([T';Data2(:,1)'],46);// [For C2]
m46(3) = interpln([T';Data3(:,1)'],46);// [For C3]
m46(4) = interpln([T';Data4(:,1)'],46);// [For C4]
m46(5) = interpln([T';Data5(:,1)'],46);// [For C5]
m46(6) = interpln([T';Data6(:,1)'],46);// [For C6]
for i = 1:6
    alpha46(i) = m46(i)/m46(5);
    yD(i) = ydD(i)/D;
    // Ratio = yD/alpha46
    Ratio1(i) = yD(i)/alpha46(i);
end
// mhk = mC5 at 46.6 OC, the assumed 46 OC is
    satisfactory.
```

```
116
117 // For the residue, at a dew point of 46 OC
118 xwW = [zF(1)-yC1 zF(2)-yC2 zF(3)-ylkD zF(4)-yC4 zF
      (5)-yhkD zF(6)-yC6];
119 W = sum(xwW);
120 xW = zeros(6);
121 m113 = zeros(6);
122 alpha113 = zeros(6);
123 m113(1) = interpln([T';Data1(:,1)'],113);// [For C1]
124 m113(2) = interpln([T';Data2(:,1)'],113);// [For C2]
125 m113(3) = interpln([T';Data3(:,1)'],113);// [For C3]
126 m113(4) = interpln([T';Data4(:,1)'],113);// [For C4]
127 m113(5) = interpln([T';Data5(:,1)'],113);// [For C5]
128 m113(6) = interpln([T';Data6(:,1)'],113);// [For C6]
129 for i = 1:6
130     alpha113(i) = m113(i)/m113(5);
131     xW(i) = xwW(i)/W;
132     // Ratio = yD/alpha46
133     Value(i) = alpha113(i)*xW(i);
134 end
135 // mhk = mC5 at 114 OC, the assumed 113 OC is
      satisfactory.
136 Temp_Avg = (114+46.6)/2;// [OC]
137 // Temp_avg is very close to the assumed 80 OC
138 Rm = (Val/D)-1;
139 printf("Minimum Reflux Ratio is %f mol reflux/mol
      distillate\n \n",Rm);
140 printf("*****************Distillate Composition
      ********************\n");
141 printf("C1\t \t \t \t: %f\n",yD(1));
142 printf("C2\t \t \t \t: %f\n",yD(2));
143 printf("C3\t \t \t \t: %f\n",yD(3));
144 printf("C4\t \t \t \t: %f\n",yD(4));
145 printf("C5\t \t \t \t: %f\n",yD(5));
146 printf("C6\t \t \t \t: %f\n",yD(6));
147 printf("\n");
148 printf("*****************Residue Composition
      ********************\n");
```

```
149 printf("C1\t \t \t \t: %f\n",xW(1));
150 printf("C2\t \t \t \t: %f\n",xW(2));
151 printf("C3\t \t \t \t: %f\n",xW(3));
152 printf("C4\t \t \t \t: %f\n",xW(4));
153 printf("C5\t \t \t \t: %f\n",xW(5));
154 printf("C6\t \t \t \t: %f\n",xW(6));
155 printf("\n");
156
157 //**********************Number of Theoretical stage
      ***********************//
158 // Page:440
159 printf('Page: 440\n\n');
160
161 for i = 1:6
162     alpha_av(i) = (alpha46(i)*alpha113(i))^0.5;
163 end
164 alphalk_av = alpha_av(3);
165 // By Eqn. 9.167:
166 xhkW = xwW(5);
167 xlkW = xwW(3);
168 Nm = log10((ylkD/yhkD)*(xhkW/xlkW))/log10(alphalk_av
      )-1;
169 // Ratio = yD/xW
170 for i = 1:6
171     Ratio2(i) = (alpha_av(i)^(Nm+1))*yhkD/xhkW;
172 end
173 // For C1:
174 // yC1D-Ratio(1)*xC1W = 0
175 // yC1D+xC1W = zF(1)
176 // Similarly for others
177 for i = 1:6
178     a = [1 -Ratio2(i);1 1];
179     b = [0;zF(i)];
180     soln = a\b;
181     yD2(i) = soln(1);// [kmol]
182     xW2(i) = soln(2);// [kmol]
183 end
184 D = sum(yD2);// [kmol]
```

```
185 W = sum(xW2);// [kmol]
186 // The distillate dew point computes to 46.6 OC and
      the residue bubble point computes to 113 OC,
      which is significantly close to the assumed.
187 printf("Minimum number of theoretical stage is: %f\n
      ",Nm);
188 printf("\n");
189
190 //***************Product composition at R =
      0.8************************//
191 // Page:441
192 printf('Page: 441\n\n');
193
194 // Since C1 and C2 do not enter in the residue nor
      C6 in the distillate, appreciably at total reflux
       or minimum reflux ratio, it will be assumed that
       they will not enter R = 0.8. C3 and C5
      distribution are fixed by specifications. Only
      that C4 remains to be estimated.
195 // R = [Infinte 0.8 0.58] [Reflux ratios For C4]
196 R = [%inf 0.8 0.58];
197 // Val = R/(R+1)
198 Val = R./R+1;
199 // ydD = [Inf 0.58]
200 y4D = [0.1255 0.1306];
201 yC4D = (((1-Val(2))/(1-Val(3)))*(y4D(2)-y4D(1)))+y4D
      (1);// Linear Interpolation
202 // For Distillate:
203 Sum1 = sum(Ratio1);
204 x0 = Ratio1./Sum1;
205 printf("For the reflux ratio of 0.8\n");
206 printf("*****************Distillate Composition
      **********************\n");
207 printf("\t\t\t Liquid reflux in equilibrium with the
       distillate vapour\n");
208 for i = 1:6
209     printf("C%d\t \t \t \t: %f\n",i,x0(i));
210 end
```

```
211 // For boiler:
212 Sum2 = sum(Value);
213 yNpPlus1 = Value./Sum2;
214 printf("**********Distillate Composition
      **********\n");
215 printf("\t\t\t Reboiler vapour in equilibrium with
      the residue\n");
216 for i = 1:6
217     printf("C%d\t \t \t \t: %f\n",i,yNpPlus1(i));
218 end
219 printf("\n");
220
221 //**********Number Of Theoretical Trays
      **********//
222 // Page: 443
223 printf('Page: 443\n\n');
224
225 R = 0.8;// [reflux ratio]
226 // From Eqn. 9.175
227 intersection = (zlkF-(ylkD/D)*(1-q)/(R+1))/(zhkF-(
      yhkD/D)*(1-q)/(R+1));
228 // Enriching Section:
229 y1 = zeros(5);
230 L = R*D;// [kmol]
231 G = L+D;// [kmol]
232 // Assume: Temp1 = 57 OC
233 // alpha57 = [C1 C2 C3 C4 C5]
234 alpha57 = [79.1 19.6 7.50 2.66 1];
235 // From Eqn. 9.177, n = 0:
236 for i = 1:5
237     y1(i) = (L/G)*x0(i)+((D/G)*yD(i));
238     Val57(i) = y1(i)/alpha57(i);
239 end
240 x1 = Val57/sum(Val57);
241 mC5 = sum(Val57);
242 Temp1 = 58.4; // [OC]
243 // Liquid x1's is in equilibrium with y1's.
244 xlk_By_xhk1 = x1(3)/x1(5);
```

```
245 // Tray 1 is not the feed tray.
246 // Assume: Temp2 = 63 OC
247 // alpha63 = [C1 C2 C3 C4 C5]
248 alpha63 = [68.9 17.85 6.95 2.53 1.00];
249 // From Eqn. 9.177, n = 1:
250 for i = 1:5
251     y2(i) = (L/G)*x1(i)+((D/G)*yD(i));
252     Val63(i) = y1(i)/alpha63(i);
253 end
254 mC5 = sum(Val63);
255 x2 = Val63/sum(Val63);
256 xlk_By_xhk2 = x2(3)/x2(5);
257 // The tray calculation are continued downward in
      this manner.
258 // Results for trays 5 & 6 are:
259 // Temp 75.4 [OC]
260 // x5 = [C1 C2 C3 C4 C5]
261 x5 = [0.00240 0.0195 0.1125 0.4800 0.3859];
262 xlk_By_xhk5 = x5(3)/x5(5);
263 // Temp6 = 79.2 OC
264 // x6 = [C1 C2 C3 C4 C5]
265 x6 = [0.00204 0.0187 0.1045 0.4247 0.4500];
266 xlk_By_xhk6 = x6(3)/x6(5);
267 // From Eqn. 9.176:
268 // Tray 6 is the feed tray
269 Np1 = 6;
270
271 // Exhausting section:
272 // Assume Temp = 110 OC
273 L_bar = L+(q*F);// [kmol]
274 G_bar = L_bar-W;// [kmol]
275 // alpha57 = [C3 C4 C5 C6]
276 alpha110 = [5 2.2 1 0.501];
277 // From Eqn. 9.178:
278 xNp = zeros(4);
279 k = 1;
280 for i = 3:6
281     xNp(k) = ((G_bar/L_bar)*yNpPlus1(i))+((W/L_bar)*
```

```
          xW(i));
282       Val110(k) = alpha110(k)*xNp(k);
283       k = k+1;
284   end
285   yNp = Val110/sum(Val110);
286   mC5 = 1/sum(Val110);
287   // yNp is in Eqb. with xNp:
288   xlk_By_xhkNp = xNp(1)/xNp(4);
289   // Results for Np-7 to Np-9 trays:
290   // For Np-7
291   // Temp =  95.7 OC
292   // xNpMinus7 = [C3 C4 C5 C6]
293   xNpMinus7 = [0.0790 0.3944 0.3850 0.1366];
294   xlk_By_xhkNpMinus7 = xNpMinus7(1)/xNpMinus7(3);
295   // For Np-8
296   // Temp =  94.1 OC
297   // xNpMinus8 = [C3 C4 C5 C6]
298   xNpMinus8 = [0.0915 0.3897 0.3826 0.1362];
299   xlk_By_xhkNpMinus8 = xNpMinus8(1)/xNpMinus8(3);
300   // For Np-9
301   // Temp =  93.6 OC
302   // xNpMinus9 = [C3 C4 C5 C6]
303   xNpMinus9 = [0.1032 0.3812 0.3801 0.1355];
304   xlk_By_xhkNpMinus9 = xNpMinus9(1)/xNpMinus9(3);
305   // From Eqn. 9.176:
306   // Np-8 is the feed tray.
307   deff('[y] = g2(Np)','y = Np-8-Np1');
308   Np = fsolve(7,g2);
309   printf("Number of theoretical Trays required for R =
          0.8: %d\n",Np);
310   printf("\n");
311
312   //**************Composition Correction
         *****************//
313   // Page: 446
314   printf('Page: 446\n\n');
315
316   // New Bubble Point:
```

```
317 // Temp = 86.4 OC
318 x6_new = x6*(1-xNpMinus8(4));
319 x6_new(6) = xNpMinus8(4);
320 // alpha86 = [C1 C2 C3 C4 C5 C6]
321 alpha86 = [46.5 13.5 5.87 2.39 1.00 0.467];
322 // From Eqn. 9.181:
323 xhkn = x5(5);
324 xhknPlus1 = x6_new(5);
325 xC65 = alpha86(6)*x6_new(6)*xhkn/xhknPlus1;
326 x5_new = x5*(1-xC65);
327 x5_new(6) = 1-sum(x5_new);
328 // Tray 5 has a bubble point of 80 OC
329 // Similarly , the calculations are continued upward
      :
330 // x2_new = [C1 C2 C3 C4 C5 C6]
331 x2_new = [0.0021 0.0214 0.1418 0.6786 0.1553
      0.00262];
332 // y2_new = [C1 C2 C3 C4 C5 C6]
333 y2_new = [0.0444 0.111 0.2885 0.5099 0.0458
      0.00034];
334 // x1_new = [C1 C2 C3 C4 C5 C6]
335 x1_new = [0.00226 0.0241 0.1697 0.7100 0.0932
      0.00079];
336 // y1_new = [C1 C2 C3 C4 C5 C6]
337 y1_new = [0.0451 0.1209 0.3259 0.4840 0.0239
      0.000090];
338 // x0_new = [C1 C2 C3 C4 C5 C6]
339 x0_new = [0.00425 0.0425 0.2495 0.6611 0.0425
      0.00015];
340 // yD_new = [C1 C2 C3 C4 C5 C6]
341 yD_new = [0.0789 0.1842 0.3870 0.3420 0.0079
      0.00001];
342 // From Eqn. 9.184:
343 // For C1 & C2
344 alphalkm = alpha86(3);
345 xlkmPlus1 = xNpMinus7(1);
346 xlkm = x6_new(3);
347 xC17 = x6_new(1)*alpha86(3)*xlkmPlus1/(alpha86(1)*
```

```
    xlkm);
xC27 = x6_new(2)*alpha86(3)*xlkmPlus1/(alpha86(2)*
    xlkm);
// Since xC17 = 1-xC27
// The adjusted value above constitute x7's.
// The new bubbl point is 94 OC
// The calculations are continued down in the same
    fashion.
// The new tray 6 has:
// xC1 = 0.000023 & xC2 = 0.00236
// It is clear that the conc. of these components
    are reducing so rapidly that there is no need to
    go an further.
printf("******Corrected Composition**********\n");
printf("Component\t \tx2\t \t \t y2\t \t \t x1\t \t
    \t y1\t \t \tx0\t \t \tyD\n");
for i = 1:6
    printf("C%d\t \t \t%f\t \t %f\t \t %f\t \t %f\t
        \t%f\t \t%f\n",i,x2_new(i),y2_new(i),x1_new(i
        ),y1_new(i),x0_new(i),yD_new(i));
end
printf("\n");

//************Heat Load of Condensor & Boiler & L/G
    ratio**********//
// Page 448
printf('Page: 448\n\n');

// Values of x0, yD & y1 are taken from the
    corrected concentration.
// HD46 = [C1 C2 C3 C4 C5 C6]
HD46 = [13490 23380 32100 42330 52570 61480];// [kJ/
    kmol]
for i = 1:6
    yDHD(i) = yD_new(i)*HD46(i);
end
HD = sum(yDHD);// [kJ]
// HL46 = [C1 C2 C3 C4 C5 C6]
```

```
375 HL46 = [10470 17210 18610 22790 27100 31050];// [kJ/
      kmol]
376 for i = 1:6
377     xHL(i) = x0_new(i)*HL46(i);
378 end
379 HL0 = sum(xHL);// [kJ]
380 // HG58 = [C1 C2 C3 C4 C5 C6]
381 HG58 = [13960 24190 37260 43500 53900 63500];// [kJ/
      kmol]
382 for i = 1:6
383     yHG1(i) = y1_new(i)*HG58(i);
384 end
385 HG1 = sum(yHG1);// [kJ]
386 // From Eqn. 9.54:
387 Qc = D*((R+1)*HG1-(R*HL0)-HD);// [kJ/kmol feed]
388 // Similarly:
389 HW = 39220;// [kJ]
390 HF = 34260;// [kJ]
391 // From Eqn. 9.55:
392 Qb = (D*HD)+(W*HW)+Qc-(F*HF);// [kJ/kmol feed]
393 // For tray n = 1
394 G1 = D*(R+1);// [kmol]
395 // With x1 & y2 from corrected composition;
396 // HG66 = [C1 C2 C3 C4 C5 C6]
397 HG66 = [14070 24610 33800 44100 54780 64430];// [kJ/
      kmol feed]
398 for i = 1:6
399     yHG2(i) = y2_new(i)*HG66(i);
400 end
401 HG2 = sum(yHG2);// [kJ]
402 // HL58 = [C1 C2 C3 C4 C5 C6]
403 HL58 = [11610 17910 20470 24900 29500 33840];// [kJ/
      kmol feed]
404 for i = 1:6
405     xHL1(i) = x1_new(i)*HL58(i);
406 end
407 HL1 = sum(xHL1);// [kJ]
408 // From Eqn. 9.185:
```

```
G2 = (Qc+D*(HD-HL1))/(HG2-HL1);// [kmol]
L2 = G2-D;// [kmol]
L2_By_G2 = L2/G2;
// Similarly, the calculations are made for other
   trays in enriching section.
// For tray, Np = 14:
// C1 & C2 are absent.
// HG113 = [C3 C4 C5 C6]
HG113 = [38260 49310 60240 71640];// [kJ/kmol feed]
k = 3;
for i = 1:4
    yHG15(i) = yNpPlus1(k)*HG113(i);
    k = k+1;
end
HG15 = sum(yHG15);
// HL107 = [C3 C4 C5 C6]
HL107 = [29310 31870 37680 43500];// [kJ/kmol feed]
for i = 1:4
    xHL14(i) = xNp(i)*HL107(i);
end
HL14 = sum(xHL14);// [kJ]
// Similarly:
HL13 = 36790;// [kJ]
HG14 = 52610;// [kJ]
// From Eqn. 9.186:
G15_bar = (Qb+(W*(HL14-HW)))/(HG15-HL14);// [kmol]
L14_bar = W+G15_bar;// [kmol]
G14_bar = (Qb+(W*(HL13-HW)))/(HG14-HL13);// [kmol]
L14_By_G14 = L14_bar/G14_bar;
printf("Condensor eat Load %f kJ:\n",HL0);
printf("Reboiler eat Load %f kJ:\n",HG15);
// For other Exhausting Section Trays:
// Result = [Tray No. L_By_G Temp(OC)]
// Tray 0: Condensor
// Tray 15: Reboiler
Result = [0,0.80 46.6;1 0.432 58.4;2 0.437 66;3
   0.369 70.4;4 0.305 74;5 0.310 80.3;6 1.53 86.4;7
   4.05 94.1;8 3.25 96.3;9 2.88 97.7;10 2.58 99;11
```

```
    2.48 100;12 2.47 102.9;13 2.42 104.6;14 2.18
    107.9;15 1.73 113.5];
444 printf("**************L/G*************\n")
445 printf("Tray No. \t\t L/G\t\t\t\t Temp(OC)\n");
446 for i = 1:16
447     printf("%d\t \t \t%f\t \t \t%2.2f\n",Result(i,1)
        ,Result(i,2),Result(i,3));
448 end
449 // These values are not final.
450 // They scatter eratically because they are based on
    the temp. and conc. computed with the assumption
    of constant L/G
451 printf("\n");
452
453 //**************Thiele Geddes Method
    ******************//
454 // Page:452
455 printf('Page: 452\n\n');
456
457 // Use the tray Temperature to obtain m.
458 // For C4:
459 // m = [0(Condensor) 1 2 3 4 5 6 7 8 9 10 11 12 13
    14 15(Reboiler)]
460 m = [0.50 0.66 0.75 0.81 0.86 0.95 1.07 1.22 1.27
    1.29 1.30 1.32 1.40 1.45 1.51 1.65];
461 for i = 1:7
462     A(i) = Result(i,2)/m(i);
463 end
464 for j = 1:9
465     i = i+1;
466     S(j) = 1/(Result(i,2)/m(i));
467 end
468 // f = Tray No. 6
469 f = 7;
470 // From Eqn. 9.196:
471 // Value1 = Gf*yf/(D*zD)
472 Sum = 0;
473 for i = 1:f-1
```

```
474       Val = 1;
475       for j = i:f-1
476           Val = Val*A(j);
477       end
478       Sum = Sum+Val;
479   end
480   Value1 = 1+Sum;
481   // From Eqn. 9.206:
482   // Value2 = Lf_bar*xf/(W*xW);
483   Sum = 0;
484   for i = 9:-1:1
485       Val = 1;
486       for j = i:-1:1
487           Val = Val*S(j);
488       end
489       Sum = Sum+Val;
490   end
491   Value2 = 1+Sum;
492   // From Eqn. 9.208:
493   // Value3 = W*xW/(D*zD)
494   Value3 = A(f)*Value1/Value2;
495   // From Eqn. 9.210:
496   DyD = F*zF(4)/(Value3+1);// [kmol,C4]
497   // From Eqn. 9.209:
498   WxW = (F*zF(4))-(DyD);// [kmol, C4]
499   // Similarly:
500   // For [C1; C2; C3; C4; C5; C6]
501   // Result2 = [Value1 Value2 Value3 DyD WxW]
502   Result2 = [1.0150 254*10^6 288*10^(-10) 0.03
         0;1.0567 8750 298*10^(-5) 0.07 0;1.440 17.241
         0.0376 0.1447 0.0053;1.5778 1.5306 1.475 0.1335
         0.1965;15580 1.1595 45.7 0.00643 0.29357;1080
         1.0687 7230 0.0000166 0.1198];
503   D = sum(Result2(:,4));// [kmol]
504   W = sum(Result2(:,5));// [kmol]
505   // In the Distillate:
506   DyD_C3 = Result2(3,4);// [kmol]
507   zFC3 = zF(3);// [kmol]
```

```
508 percentC3 = (DyD_C3/zFC3)*100;
509 DyD_C5 = Result2(5,4);// [kmol]
510 zFC5 = zF(5);// [kmol]
511 percentC5 = (DyD_C5/zFC5)*100;
512 // These do not quite meet the original speification
    .
513 // For Tray 2 & C4
514 // From Eqn. 9.195:
515 // Value4 = G2*y2/(D*zD)
516 n = 2;
517 Sum = 0;
518 for i = 1:n
519     Val = 1;
520     for j = i:n
521         Val = Val*A(j);
522     end
523     Sum = Sum+Val;
524 end
525 Value4 = 1+Sum;
526 // From The enthalpy Balnce:
527 G2 = 0.675;
528 // From Eqn. 9.211:
529 y2 = Value4*DyD/G2;
530 // Similarly:
531 // Value4 = [C1 C2 C3 C4 C5 C6]
532 Value4 = [1.0235 1.1062 1.351 2.705 10.18 46.9];
533 for i = 1:6
534     y2(i) = Result2(i,4)*Value4(i)/G2;
535 end
536 Y2 = sum(y2);
537 // Since Y2 is not equal to 1. THerefore the
    original temperature is incorrect. By adjusting
    y2 to unity.
538 // The dew point is 77 OC instead of 66 OC
539 // y2_adjusted = [C1 C2 C3 C4 C5 C6]
540 y2_adjusted = [0.0419 0.1059 0.2675 0.4939 0.0896
    0.00106];
541 printf("**********Composition By Thiele
```

```
    Geddes Method**********\n");
printf("Component\t \t \t y2\n")
for i = 1:6
    printf("C%d\t \t \t \t%f\n",i,y2_adjusted(i));
end
```

Chapter 10

Liquid Extraction

Scilab code Exa 10.1 Single Stage Extraction

```
1  clear;
2  clc;
3
4  // Illustration 10.1
5  // Page: 494
6
7  printf('Illustration 10.1 - Page: 494\n\n');
8
9  // solution
10
11 //****Data****//
12 // a:water b:isopropyl ether c:acetic acid
13 xF = 0.30;// [mol fraction]
14 yS = 0;// [mol fraction]
15 S1 = 40;// [kg]
16 B1 = 40;// [kg]
17 //*******//
18
19 // Equilibrium data at 20 OC:
20 // Wa: Wt. percent of a
21 // Wb: Wt. percent of b
```

```
// Wc: Wt. percent of c
// Data1 = [Wc Wa Wb]
// Data1: water layer
Data1 = [0.69 98.1 1.2;1.41 97.1 1.5;2.89 95.5
   1.6;6.42 91.7 1.9;13.30 84.4 2.3;25.50 71.1
   3.4;36.70 58.9 4.4;44.30 45.1 10.6;46.40 37.1
   16.5];
// Data2: isopropyl ether layer
Data2 = [0.18 0.5 99.3;0.37 0.7 98.9;0.79 0.8
   98.4;1.93 1 97.1;4.82 1.9 93.3;11.40 3.9
   84.7;21.60 6.9 71.5;31.10 10.8 58.1;36.20 15.1
   48.7];
scf(20);
plot(Data1(:,3)/100,Data1(:,1)/100,Data2(:,3)/100,
   Data2(:,1)/100);
xgrid();
xlabel("Wt fraction of isopropyl ether");
ylabel("Wt fraction of acetic acid");
// x: Wt fraction of acetic acid in water layer.
// y: Wt fraction of acetic acid in isopropyl layer.
legend("x Vs fraction ether","y Vs fraction ether");
// The rectangular coordinates of Fig 10.9(a) will
   be used but only upto x = 0.30
a = gca();
a.data_bounds = [0 0;1 0.3];
// Stage 1:
F = 100;// [kg]
// From Eqn. 10.4:
M1 = F+S1;// [kg]
// From Eqn. 10.5:
xM1 = ((F*xF)+(S1*yS))/M1;
// From Fig. 10.15 (Pg 495):
// Point M1 is located on the line FB and with the
   help of tie line passing through M1:
x1 = 0.258;// [mol fraction]
y1 = 0.117;// [mol fraction]
// From Eqn. 10.8:
E1 = (M1*(xM1-x1)/(y1-x1));// [kg]
```

```
51 // From Eqn. 10.4:
52 R1 = M1-E1;// [kg]
53
54 // Stage 2:
55 S2 = 40;// [kg]
56 B2 = 40;// [kg]
57 // From Eqn. 10.15:
58 M2 = R1+B2;// [kg]
59 // From Eqn. 10.16:
60 xM2 = ((R1*x1)+(S2*yS))/M2;
61 // Point M2 is located on the line R1B and the tie
     line passing through R2E2 through M2:
62 x2 = 0.227;
63 y2 = 0.095;
64 // From Eqn. 10.8:
65 E2 = (M2*(xM2-x2)/(y2-x2));// [kg]
66 // From Eqn. 10.4:
67 R2 = M2-E2;// [kg]
68
69 // Stage 3:
70 S3 = 40;// [kg]
71 B3 = 40;// [kg]
72 // From Eqn. 10.15:
73 M3 = R2+B3;// [kg]
74 // From Eqn. 10.16:
75 xM3 = ((R2*x2)+(S3*yS))/M3;
76 // Point M3 is located on the line R2B and the tie
     line passing through R3E3 through M3:
77 x3 = 0.20;// [mol fraction]
78 y3 = 0.078;// [mol fraction]
79 // From Eqn. 10.8:
80 E3 = (M3*(xM3-x3)/(y3-x3));// [kg]
81 // From Eqn. 10.4:
82 R3 = M3-E3;// [kg]
83 Ac = x3*R3;
84 printf("The composited extract is %f kg\n",(E1+E2+E3
     ));
85 printf("The acid content is %f kg\n",((E1*y1)+(E2*y2
```

```
   )+(E3*y3)));
printf("\n");

// If an extraction to give the same final raffinate
    concentration were to be done in single stage,
   the point M would be at the intersection of tie
   line R3E3 and the line BF.
x = 0.20;// [mol fraction]
xM = 0.12;// [mol fraction]
// From Eqn. 10.6:
S = F*(xF-xM)/(xM-yS);// [kg]
printf("%f kg of solvent would be recquired if the
   same final raffinate concentration were to be
   obtained with one stage.\n",S);
```

Scilab code Exa 10.2 Insoluble Liquids

```
clear;
clc;

// Illustration 10.2
// Page: 497

printf('Illustration 10.2 - Page: 497\n\n');

printf('Illustration 10.2 (a)\n\n');

// solution (a)

//****Data****//
// a:water b:kerosene c:Nicotine
xF = 0.01;// [wt fraction nicotine]
F = 100;// [kg]
B = 150;// [kg]
//******//
```

```
19
20 // Equilibrium data:
21 // x_prime = kg nicotine/kg water
22 // y_prime = kg nicotine/kg kerosene
23 // Data = [x_prime y_prme]
24 Data = [0 0;0.001011 0.000807;0.00246
      0.001961;0.00502 0.00456;0.00751 0.00686;0.00998
      0.00913;0.0204 0.01870];
25 xF_prime = xF/(1-xF);// kg nicotine/kg water
26 A = F*(1-xF);// [kg]
27 AbyB = A/B;
28 scf(21);
29 deff('[y] = f64(x)','y = -AbyB*(x-xF)');
30 x = 0:0.001:0.01;
31 plot(Data(:,1),Data(:,2),x,f64);
32 xgrid();
33 legend("Equilibrium line","Operating Line");
34 xlabel("kg nicotine / kg water");
35 ylabel("kg nicotine / kg kerosene");
36 title("Solution 10.2(a)")
37 // The operating line and equilibrium line intersect
       at:
38 x1_prime = 0.00425;// [kg nicotine/kg water]
39 y1_prime = 0.00380;// [kg nicotine/kg water]
40 extract = A*(0.01011-x1_prime);
41 printf("%f %% of nicotine is extracted.\n\n",extract
      *100);
42
43 printf('Illustration 10.2 (b)\n\n');
44
45 // Solution (b)
46 B = 50;// [kg]
47 // For each stage:
48 AbyB = A/B;
49 deff('[y] = f65(x1)','y = -AbyB*(x1-xF)');
50 x1 = 0:0.001:0.01;
51 deff('[y] = f66(x2)','y = -AbyB*(x2-0.007)');
52 x2 = 0:0.001:0.01;
```

```
deff('[y] = f67(x3)','y = -AbyB*(x3-0.005)');
x3 = 0:0.001:0.01;
scf(22);
plot(Data(:,1),Data(:,2),x1,f65,x2,f66,x3,f67);
xgrid();
a=gca();
a.data_bounds=[0 0;0.012 0.010];
legend("Equilibrium line","Operating Line from xF","
   Operating Line from 0.007","Operating Line from
   0.005");
xlabel("kg nicotine / kg water");
ylabel("kg nicotine / kg kerosene");
title("Solution 10.2(b)")
// The final raffinate composition:
x3_prime = 0.0034;// [kg nicotine/kg water]
extract = A*(0.01011-x3_prime);
printf("%f %% of nicotine is extracted.\n",extract
   *100);
```

Scilab code Exa 10.3 Continuous Countercurrent Multistage Extraction

```
clear;
clc;

// Illustration 10.3
// Page: 502

printf('Illustration 10.3 - Page: 502\n\n');

// Solution

//****Data****//
// a:water b:isopropyl ether c:acetic acid
F = 8000;// [kg/h]
xF = 0.30;// [wt. fraction acetic acid]
```

```
15 //*******//
16
17 // From Illustration 10.1 (Pg 494)
18 // Equilibrium Data:
19 // Eqb = [y_star*100 x*100]
20 Eqb = [0.18 0.69;0.37 1.41;0.79 2.89;1.93 6.42;4.82
      13.30;11.40 25.50;21.60 36.70;31.10 44.30;36.20
      46.40];
21
22 // Solution(a)
23
24 // From Figure 10.23 (Pg 503):
25 // For minimum solvent rate:
26 y1 = 0.143;// [Wt fraction of acetic acid in
      isopropyl ether layer]
27 xM = 0.114;// [Wt fraction of acetic acid in water
      layer]
28 // From Eqn. 10.24:
29 Bm = (F*xF/xM)-F;// [kg/h]
30 printf("Minimm solvent rate: %f kg/h\n",Bm);
31 printf("\n");
32
33 // Solution (b)
34
35 B = 20000;// [kg solvent/h]
36 yS = 0;
37 S = B;
38 // From Eqn 10.24:
39 xM = ((F*xF)+(S*yS))/(F+S);
40 // From Fig. 10.23 (Pg 503):
41 y1 = 0.10;
42 // Operating curve data:
43 // Operat = [YsPlus1 Xs]
44 Operat = [0 0.02;0.01 0.055;0.02 0.09;0.04
      0.150;0.06 0.205;0.08 0.250;0.1 0.3];
45 scf(23);
46 plot(Eqb(:,2)/100,Eqb(:,1)/100,Operat(:,2),Operat
      (:,1));
```

```
47 xgrid();
48 a = gca();
49 a.data_bounds = [0 0;xF y1];
50 legend("Operating Line","Equilibrium Line");
51 xlabel("Wt. fraction acetic acid in water solution")
      ;
52 ylabel("Wt. fraction acetic acid in isopropyl ether
      solution");
53 title("Solution 10.3")
54 // From Figure scf(22):
55 xNp = 0.02;
56 Np = 7.6;
57 // By acid balance:
58 M = B+F;
59 E1 = M*(xM-xNp)/(y1-xNp);// [kg/h]
60 RNp = M-E1;// [kg/h]
61 printf("Number of theoretical Stages: %f\n",Np);
62 printf("Weight of the extract: %d kg/h\n",E1);
63 printf("Weight of the raffinate %d kg/h\n",RNp);
```

Scilab code Exa 10.4 Continuous Countercurrent Multistage Extraction Insoluble Liquids

```
1  clear;
2  clc;
3
4  // Illustration 10.4
5  // Page: 506
6
7  printf('Illustration 10.4 - Page: 506\n\n');
8
9  // Solution
10
11 //****Data****//
12 // a:water b:kerosene c:Nicotine
```

```
13 F = 1000;// [kg/h]
14 xF = 0.01;// [wt. fraction acetic acid]
15 //*******//
16
17 // Equilibrium data:
18 // x_prime = kg nicotine/kg water
19 // y_prime = kg nicotine/kg kerosene
20 // Eqb = [x_prime y_prme]
21 Eqb = [0 0;0.001011 0.000807;0.00246
      0.001961;0.00502 0.00456;0.00751 0.00686;0.00998
      0.00913;0.0204 0.01870];
22
23 // Solution (a)
24
25 A = 1000*(1-xF);// [kg water/h]
26 yS = 0;
27 yS_prime = 0;
28 y1_prime = 0;
29 xF_prime = xF/(1-xF);// [kg nicotine/kg water]
30 // For xF_prime = 0.0101:
31 yk = 0.0093;
32 xNp = 0.001;// [wt. fraction acetic acid]
33 xNp_prime = xNp/(1-xNp);// [kg nicotine/kg water]
34 // For infinite stages:
35 // Operating Line should pass through (xNp_prime,
      y1_prime) & (xF_prime,yk)
36 Operat = [xNp_prime y1_prime;xF_prime yk];
37 scf(24);
38 plot(Eqb(:,1),Eqb(:,2),Operat(:,1),Operat(:,2));
39 xgrid();
40 legend("equilibrium Line","Operating Line");
41 xlabel("kg nicotine / kg water");
42 ylabel("kg nicotine / kg kerosene");
43 title("Solution 10.4(a)")
44 a = gca();
45 a.data_bounds = [0 0;0.012 0.01];
46 AbyBm = (yk-y1_prime)/(xF_prime-xNp_prime);
47 Bm = A/AbyBm;// [kg kerosene/h];
```

```
48 printf("Mininmum kerosene rate: %f kg kerosene/h \n"
      ,Bm);
49
50 // Solution (b)
51
52 B = 1150;// [kg/h]
53 AbyB = A/B;
54 // From Eqn. 10.36:
55 y2_prime = ((xF_prime-xNp_prime)*AbyB)+yS_prime;// [
      kg nicotine/kg kerosene]
56 // Operating Line should pass through (xNp_prime,
      y1_prime) & (xF_prime,y2_prime)
57 Operat = [xNp_prime y1_prime;xF_prime y2_prime];
58 scf(25);
59 plot(Eqb(:,1),Eqb(:,2),Operat(:,1),Operat(:,2));
60 xgrid();
61 legend("equilibrium Line","Operating Line");
62 xlabel("kg nicotine/kg water");
63 ylabel("kg nicotine/kg kerosene");
64 title("Solution 10.4(b)")
65 a = gca();
66 a.data_bounds = [0 0;0.012 0.01];
67 // From Figure:
68 Np = 8.3;
69 printf("Number of theoretical stages:%f \n",Np);
```

Scilab code Exa 10.5 Continuous Countercurrent Extraction with Reflux

```
1 clear;
2 clc;
3
4 // Illustration 10.5
5 // Page: 510
6
7 printf('Illustration 10.5 - Page: 510\n\n');
```

```
8
9  // solution
10
11 //****Data****//
12 // a:ethylbenzne b:diethylene glycol c:styrene
13 F = 1000;// [kg/h]
14 xF = 0.5;// [Wt. fraction styrene]
15 xPE = 0.9;// [kg styrene/kg hydrocarbon]
16 xRNp = 0.1;// [kg styrene/kg hydrocarbon]
17 //******//
18
19 // X: kg styrene/kg hydrocarbon
20 // Y: kg styrene/kg hydrocarbon
21 // N:kg glycol/kg hydrocarbon
22 // Equilibrium data:
23 // Hydrocarbon rich solutions:
24 // Eqb1 = [X N]
25 Eqb1 = [0 0.00675;0.0870 0.00817;0.1833
      0.00938;0.288 0.01010;0.384 0.01101;0.458
      0.01215;0.464 0.01215;0.561 0.01410;0.573
      0.01405;0.781 0.01833;1 0.0256];
26 // Solvent rich solutions:
27 // Eqb2 = [Y_star N]
28 Eqb2 = [0 8.62;0.1429 7.71;0.273 6.81;0.386
      6.04;0.480 5.44;0.557 5.02;0.565 4.95;0.655
      4.46;0.674 4.37;0.833 3.47;1 2.69];
29 scf(26);
30 plot(Eqb1(:,1),Eqb1(:,2),Eqb2(:,1),Eqb2(:,2));
31 xgrid();
32 legend("X Vs N","Y Vs N");
33 xlabel("kg styrene / kg hydrocarbon");
34 ylabel("kg diethylene glycol / kg hydrocarbon");
35 title("Equilibrium Data")
36 // In Fig. 10.31 (Pg 512):
37 // Point E1 is located.
38 NE1 = 3.10;
39
40 // solution (a)
```

```
41
42 // From Fig. 10.30 (Pg 511):
43 Np = 9.5;
44 printf("Minimum number of theoretical stages: %f\n",
     Np);
45 printf("\n");
46
47 // Solution (b)
48
49 // The tie line when extended passes through F
     provides the minimum reflux ratio.
50 // From the plot:
51 N_deltaEm = 20.76;
52 // From Eqn. 10.48:
53 Ratiom = (N_deltaEm-NE1)/NE1;// [kg reflux/kg
     extract product]
54 printf("Minimum extract reflux ratio: %f kg reflux/
     kg extract product\n",Ratiom);
55 printf("\n");
56
57 // Solution (c)
58
59 Ratio = 1.5*Ratiom;// [kg reflux/kg extract product]
60 // From Eqn. 10.48;
61 N_deltaE = (Ratio*NE1)+NE1;
62 // Point deltaE is plotted.
63 // A straight line from deltaE through F intersects
     line X = 0.10 at deltaR.
64 N_deltaR = -29.6;
65 // In Fig. 10.31 (Pg 512):
66 // Random lines are drawn from deltaE for the
     concentrations to the right of F, and from deltaR
      for those to the left ,and intersection of these
     with the solubility curves provide the
     coordinates of the opeating curve.
67 // The tie line data are plotted directly to provide
      the equilibrium curve.
68 // From Fig. 10.32 (Pg 513):
```

```
Np = 15.5;
// Feed is to be introduced in the seventh from the
   extract product end of cascade.
// From Fig. 10.31 (Pg 512):
XRNp = 0.10;
NRNp = 0.0082;
// Basis:1 hour.
// Overall plant balance:
// (1): PE_prime+RNp_prime = F
// C Balance
// (2): PE_prime*(1-XRNp)+RNp_prime*XRNp = F*xF
// Solving (1) & (2) simultaneously:
a = [1 1;(1-XRNp) XRNp];
b = [F;F*xF];
soln = a\b;
PE_prime = soln(1);// [kg/h]
RNp_prime = soln(2);// [kg/h]
RO_prime = Ratio*PE_prime;// [kg/h]
// From Eqn 10.39:
E1_prime = RO_prime+PE_prime;// [kg/h]
BE = E1_prime*NE1;// [kg/h]
E1 = BE+E1_prime;// [kg/h]
RNp = RNp_prime*(1+NRNp);// [kg/h]
S = BE+(RNp_prime*NRNp);// [kg/h]
printf("Number of theoretical stages: %f\n",Np);
printf("Extract Flow Rate: %f kg/h\n",E1);
printf("solvent Flow Rate: %f kg/h\n",S);
```

Scilab code Exa 10.6 Fractional Extraction

```
clear;
clc;

// Illustration 10.6
// Page: 516
```

```
6
7  printf('Illustration 10.6 - Page: 516\n\n');
8
9  // solution
10
11 //****Data****//
12 // a:heptane b:p-chloronitrobenzene c:o-
      chloronitrobenzene d:aq. methanol
13 xb = 0.4;// [Wt fraction]
14 xC = 0.60;// [Wt fraction]
15 F = 100;// [kg]
16 // The para isomer(b) favours the heptane(a) and the
       ortho isomer(c) favours the methanol(d).
17 // Basis: 1 hour.
18 A = 2400;// [kg/h]
19 D = 2760;// [kg/h]
20 xbW = 0.8;// [Wt fraction]
21 xbZ = 0.15;// [Wt fraction]
22 kb=1.35;
23 kc=0.835;
24 //*******//
25
26 B = xb*F;// [kg]
27 C = F-B;// [kg]
28 // W = kg A rich product, after solvent removal
29 // Z = kg D rich product, after solvent removal
30 // B balance:
31 // (1): (0.80*W)+(0.15*Z) = B
32 // C balance:
33 // (2): (0.20*W)+(0.85*Z) = C
34 // Solving (1) & (2) simultaneously:
35 a = [0.80 0.15;0.20 0.85];
36 b = [B;C];
37 soln = a\b;
38 W = soln(1);
39 Z = soln(2);
40 Wb = xbW*W;// [kg]
41 Wc = W-Wb;// [kg]
```

```
42 Zb = xbZ*Z;// [kg]
43 Zc = Z-Zb;// [kg]
44 xB1_prime = Zb/D;
45 xC1_prime = Zc/D;
46 yB1_prime = Wb/D;
47 yC1_prime = Wc/D;
48 DbyA = D/A;
49 // Equilibrium curve:
50 // First distribution coeffecient: yB_star/xB_prime
     = 1.35
51 deff('[y] = f68(x1)','y = kb*x1');
52 x1 = 0:0.01:0.06;
53 // Second distribution coeffecient: yC_star/xC_prime
     = 0.835
54 deff('[y] = f69(x2)','y = kc*x2');
55 x2 = 0:0.01:0.06;
56 // Operating Line, corresponding to First
     distribution coeffecient:
57 deff('[y] = f70(x3)','y = (DbyA*x3)+yB1_prime');
58 x3 = 0:0.01:0.06;
59 deff('[y] = f71(x4)','y = DbyA*(x4-xB1_prime)');
60 x4 = 0:0.01:0.06;
61 // Operating Line, corresponding to Second
     distribution coeffecient:
62 deff('[y] = f72(x5)','y = (DbyA*x5)+yC1_prime');
63 x5 = 0:0.01:0.06;
64 deff('[y] = f73(x6)','y = (DbyA)*(x6-xC1_prime)');
65 x6 = 0:0.01:0.06;
66 scf(27);
67 plot(x1,f68,x3,f70,x4,f71);
68 xgrid();
69 legend("Equilibrium curve","Operating curve","
     Operating curve");
70 xlabel("xB_prime");
71 ylabel("yB_prime");
72 title("yB_star/xB_prime = 1.35");
73 a1 = gca();
74 a1.data_bounds = [0 0;0.05 0.07];
```

```
75 scf(28);
76 plot(x2,f69,x5,f72,x6,f73);
77 xgrid();
78 legend("Equilibrium curve","Operating curve","
      Operating curve");
79 xlabel("xC_prime");
80 ylabel("yC_prime");
81 title("yC_star/xC_prime = 0.835");
82 a2 = gca();
83 a2.data_bounds = [0 0;0.06 0.07];
84 // The stages are constructed.
85 // The feed matching is shown on Fig. 10.37 (Pg 518)
      :
86 f_prime = 6.6;
87 fstage = 4.6;
88 printf("Number of ideal stage is %f\n",fstage+
      f_prime-1);
89 printf("The feed stage is %fth from the solvent-D
      inlet\n",fstage);
```

Scilab code Exa 10.7 Stage Type Extractors

```
1 clear;
2 clc;
3
4 // Illustration 10.7
5 // Page: 525
6
7 printf('Illustration 10.7 - Page: 525\n\n');
8 // solution
9
10 //****Data****//
11 // c:Water d:Toulene
12 Density_c = 998;// [kg/cubic m]
13 viscosity_c = 0.95*10^(-3);// [kg/m.s]
```

```
14 Dc = 2.2*10^(-9);// [square m/s]
15 Density_d = 865;// [kg/cubic m]
16 viscosity_d = 0.59*10^(-3);// [kg/m.s]
17 Dd = 1.5*10^(-9);// [square m/s]
18 sigma = 0.022;// [N/m]
19 Dist = 20.8;// [Distribution Coeffecient]
20 d = 0.5;// [m]
21 h = 0.5;// [m]
22 di = 0.15;// [m]
23 N = 13.3;// [r/s]
24 g = 9.81;// [m/s^2]
25 qC = 3*10^(-3);// [cubic m/s]
26 qD = 3*10^(-4);// [cubic m/s]
27 //********//
28
29 V = %pi*h*d^2/4;// [Vessel volume,cubic m]
30 phi_DF = qD/(qD+qC);// [Volume fraction toulene]
31 // Assume:
32 phi_Dbyphi_DF = 0.9;
33 phi_D = phi_Dbyphi_DF*phi_DF;
34 phi_W = 1-phi_D;
35 // From Eqn. 10.56:
36 Density_M = (Density_c*phi_W)+(Density_d*phi_D);// [
      kg/cubic m]
37 if phi_W>0.4
38     viscosity_M = (viscosity_c/phi_W)*(1+(6*
          viscosity_d*phi_D/(viscosity_d+viscosity_c)))
          ;// [kg/m s]
39 else
40     viscosity_M = (viscosity_c/phi_D)*(1-(1.5*
          viscosity_c*phi_W/(viscosity_d+viscosity_c)))
          ;// [kg/m s]
41 end
42 // Impeller Reynold's Number:
43 IRe = (di^2*N*Density_M/viscosity_M);
44 // From Fig 6.5 (Pg 152), curve g:
45 Po = 0.72;
46 P = Po*Density_M*N^3*di^5;// [W]
```

```
47 // From Eqn. 10.61:
48 Value1 = P*qD*viscosity_c^2/(V*sigma^3);
49 Value2 = viscosity_c^3/(qD*Density_c^2*sigma);
50 Value3 = Density_c/(Density_c-Density_d);
51 Value4 = sigma^3*Density_c/(viscosity_c^4*g);
52 Value5 = viscosity_d/viscosity_c;
53 phi_Dbyphi_DF = 3.39*Value1^0.247*Value2^0.427*
      Value3^0.430*Value4^0.401*Value5^0.0987;
54 // The value of phi_Dbyphi_DF is sufficiently close
      to the value 0.90 assumed earlier.
55 phi_D = phi_Dbyphi_DF*phi_DF;
56 // From Eqn. 10.6:
57 Value6 = viscosity_c/Density_c;// [square m/s]
58 Value7 = P/(V*Density_M);
59 Value8 = sigma/Density_c;
60 dp = 10^(-2.066+(0.732*phi_D))*Value6^0.0473*Value7
      ^(-0.204)*Value8^(0.274);// [m]
61 a = 6*phi_D/dp;// [square m/cubic m]
62 Sca = viscosity_c/(Density_c*Dc);
63 // From Eqn. 10.65:
64 Shc = 65.3;
65 kLc = Shc*Dc/dp;// [kmol/square m s (kmol/cubic m)]
66 thetha = V/(qD+qC);// [s]
67 // From Table 10.1 (Pg 524):
68 // lambda = [lambda1 lambda2 lambda3]
69 lambda = [1.359 7.23 17.9];
70 // B = [B1 B2 B3]
71 B = [1.42 0.603 0.317];
72 Val = zeros(1,3);
73 Sum = 0;
74 for n = 1:3
75     Val(n) = (B(n)^2)*exp((-lambda(n))*64*Dd*thetha/
         dp^2);
76     Sum = Sum+Val(n);
77 end
78 // From Eqn. 10.66:
79 kLd = -(dp/(6*thetha))*log((3/8)*Sum);
80 mCD = 1/Dist;
```

```
// From Eqn. 10.67:
KLd = 1/((1/kLd)+(1/(mCD*kLc)));// [kmol/square m s
   (kmol/cubic m)]
Z = 0.5;// [m]
Vd = qD/(%pi*Z^2/4);// [m/s]
// From Eqn.10.70:
NtoD = Z/(Vd/(KLd*a));
// From Eqn. 10.71:
EMD = NtoD/(NtoD+1);
printf("Expected stage efficiency: %f\n",EMD);
```

Scilab code Exa 10.8 Sieve Tray Tower

```
clear;
clc;

// Illustration 10.8
// Page: 539

printf('Illustration 10.8 - Page: 539\n\n');

// solution

//****Data****//
// a:acetic acid c:Water d:Isopropylether layer
// Water solution (continuous):
C = 8000;// [kg/h]
xCn = 0.175;// [mass fraction]
Density_c = 1009;// [kg/cubic m]
viscosity_c = 3.1*10^(-3);// [kg/m.s]
Dc = 1.24*10^(-9);// [square m/s]

// Isopropyl Ethr Layer:
D = 20000;// [kg/h]
xDnPlus1 = 0.05;// [mass fraction]
```

```
Density_d = 730;// [kg/cubic m]
viscosity_d = 0.9*10^(-3);// [kg/m.s]
Dd = 1.96*10^(-9);// [square m/s]

sigma = 0.013;// [/N/m]
m = 2.68;// [Distributon coeffecient]
//*******//

Ma = 60.1;
g = 9.81;// [m/square s]
cCn = xCn*Density_c/Ma;// [kmol/cubic m]
cDnPlus1 = xDnPlus1*Density_d/Ma;// [kmol/cubic m]
mCD = m*(Density_c/Density_d);// [(kmol/cubic min
   ether)/(kmol/cubic m in water)]

// Perforations:
Do = 0.006;// [m]
pitch = 0.015;// [m]
qD = D/(3600*Density_d);// [cubic m/s]
delta_Density = Density_c-Density_d;// [kg/cubic m]
Value1 = Do/(sigma/(delta_Density*g))^0.5;
if Value1<0.1785
    // From Eqn. 10.74(a):
    doBydj = (0.485*Value1^2)+1;
else
    // From Eqn. 10.74(b)
    doBydj = (1.51*Value1)+0.12;
end
dj = Do/doBydj;// [m]
Vomax = 2.69*((dj/Do)^2)*(sigma/(dj*((0.5137*
   Density_d)+(0.4719*Density_c))))^0.5;// [m/s]
// Since Vomax is less than 0.1:
Vo = 0.1;// [m/s]
Ao = qD/Vo;// [square m]
No = Ao/(%pi*Do^2/4);// [square m]
// From Eqn. 6.30:
// Plate area for perforation:
Aa = Ao/(0.907*(Do/pitch)^2);// [square m]
```

```
59
60 // Downspout:
61 dp = 0.0007;// [m]
62 // From Eqn. 10.75:
63 U = Density_c^2*sigma^3/(g*viscosity_c^4*
      delta_Density);
64 // From Fig. 10.47 (Pg 534):
65 ordinate = 1.515;
66 abcissa = 0.62;
67 deff('[y] = f74(Vt)','y = abcissa-(dp*Vt*Density_c/(
      viscosity_c*U^0.15))');
68 Vt = fsolve(7,f74);// [m/s]
69 Vd = Vt;// [m/s]
70 qC = C/(Density_c*3600);// [cubic m/s]
71 Ad = qC/Vd;// [square m]
72 // From Table 6.2 (Pg 169):
73 // Allowing for supports and unperforated area:
74 At = Aa/0.65;// [square m]
75 T = (At*4/%pi)^0.5;// [m]
76 An = At-Ad;// [square m]
77
78
79 // Drop Size:
80 alpha1 = 10.76;
81 alpha2 = 52560;
82 alpha3 = 1.24*10^6;
83 alpha4 = 3.281;
84 abcissa = (alpha2*sigma*Do/delta_Density)+(alpha3*Do
      ^1.12*Vo^0.547*viscosity_c^0.279/delta_Density
      ^1.5);
85 Parameter = alpha1*Density_d*Vo^2/(delta_Density);
86 ordinate = 0.024;
87 dp = ordinate/alpha4;
88
89 // Coalesced layer:
90 Vn = qD/An;// [m/s]
91 // From Eqn. 10.80:
92 ho = (Vo^2-Vn^2)*Density_d/(2*g*0.67^2*delta_Density
```

```
    );// [m]
hD = ho;
// From Eqn. 10.82:
hC = 4.5*Vd^2*Density_c/(2*g*delta_Density);// [m]
// From Eqn. 10.78:
h = hC+hD;
// Since this is very shallow, increase it by
    placing an orifice at the bottom of the downspout
    .
// VR: Velocity through the restriction.
// hR: Corresponding depth of the coalesced layer.
// Assume:
Vr = 0.332;// [m/s]
hr = (Vr^2-Vd^2)*Density_c/(2*0.67^2*delta_Density);
Ar = qC/Vr;// [square m]
dr = (4*Ar/%pi)^0.5;// [m]
h = h+hr;// [m]
// The above results are satisfacyory.
Z = 0.35;// [m]
// Lead the downspout apron to within 0.1 m of the
    tray below.

// Dispersed-phase holdup:
// From Eqn. 10.48:
Vsphi_D = Vn;
// From Fig. 10.47 (Pg 534):
ordinate = 165.2;
abcissa = 30;
deff('[y] = f75(Vt)','y = abcissa-(dp*Vt*Density_c/(
    viscosity_c*U^0.15))');
Vtl = fsolve(7,f75);// [m/s]
// For solids:
// From Fig. 10.48 (Pg 536):
abcissa = dp/(3*viscosity_c^2/(4*Density_c*
    delta_Density*g))^(1/3);
phi_D = [0 0.1 0.2 0.3];
// Corresponding ordinates, from Fig. 10.48 (Pg 536)
    :
```

```
124 ordinate1 = [8.8 5.9 4.3 3.0];
125 Value1 = 1/(4*viscosity_c*delta_Density*g/(3*
      Density_c^2))^(1/3);
126 Val = zeros(4,6);
127 // Val = [phi_D ordinate Vs(1-phi_D) (Vs for solids)
      Vs/Vt (Vs for liquids) (Vs*phi_D (for liquids))]
128 for i = 1:4
129     Val(i,1) = phi_D(i);
130     Val(i,2) = ordinate1(i);
131     Val(i,3) = Val(i,2)/Value1;
132     Val(i,4) = Val(i,3)/(1-Val(i,1));
133     Val(i,5) = Val(i,4)/Val(1,4);
134     Val(i,6) = Vtl*Val(i,5);
135     Val(i,7) = Val(i,6)*Val(i,1);
136 end
137
138 //  By Interpolation:
139 Phi_D = 0.1;
140 // Mass transfer:
141 thetha_f = (%pi*(dp^3)/6)/(qD/No);// [s]
142 // From Eqn. 10.87:
143 const = 1.5;
144 kLDf = const*(Dd/(%pi*thetha_f))^0.5;// [m/s]
145 // From Eqn. 10.86
146 KLDf = 1/((1/kLDf)*(1+((1/mCD)*(Dd/Dc)^0.5)));// [m/
      s]
147 // The ordinate of Fig. 10.47 for the drops larger
      than 70. Hence mass transfer coeffecient during
      drop rise is given by Eqn. 10.89:
148 // From Eqn. 10.91:
149 b = 1.052*dp^0.225;
150 // From Eqn. 10.90:
151 omega = (1/(2*%pi))*sqrt(192*sigma*b/(dp^3*((3*
      Density_d)+(2*Density_c))));// [1/s]
152 del = 0.2;
153 kLDr = sqrt((4*Dd*omega/%pi)*(1+del+(1/2)*del^2));
154 KLDr = 1/1/((1/kLDr)*(1+((1/mCD)*(Dd/Dc)^0.5)));// [
      m/s]
```

```
155 // From Eqn. 10.98:
156 EMD = ((4.4*KLDf/Vo)*(dp/Do)^2)+(6*KLDr*Phi_D*(Z-h)
       /(dp*Vn))/(1+((0.4*KLDf/Vo)*(dp/Do)^2)+(3*KLDr*
       Phi_D*(Z-h)/(dp*Vn)));
157 printf("Stage Efficiency: %f",EMD);
```

Scilab code Exa 10.9 Number Of Transfer Unit Dilute Solutions

```
1  clear;
2  clc;
3
4  // Illustration 10.9
5  // Page: 551
6
7  printf('Illustration 10.9 - Page: 551\n\n');
8
9  // solution
10
11 //****Data****//
12 B = 20000;// [kg/h]
13 //******//
14
15 // x and y are taken in weight fraction acetic acid.
16 x1 = 0.30;// [Wt fraction]
17 xF = 0.30;// [Wt fraction]
18 y2 = 0;// [Wt fraction]
19 x2 = 0.02;// [Wt fraction]
20 y1 = 0.10;// [Wt fraction]
21 // The operating diagram is plotted in Fig. 10.23:
22 // Data = [x x_star]
23 // From Fig. 10.23 (Pg 503):
24 Data = [0.30 0.230;0.25 0.192;0.20 0.154;0.15
       0.114;0.10 0.075;0.05 0.030;0.02 0];
25 Val = zeros(7);
26 for i = 1:7
```

```
27      Val(i) = 1/(Data(i,1)-Data(i,2));
28  end
29  scf(29);
30  plot(Data(:,1),Val);
31  xgrid();
32  a = gca();
33  a.Data_bounds = [0.02 0;0.30 50];
34  xlabel("x");
35  ylabel("1/(x-x*)");
36  title("Graphical Integration");
37  // From Area Under the curve:
38  Area = 8.40;
39  // The mutual solubility of water and isopropyl
       ether is very small.
40  Ma = 18;// [kg/kmol water]
41  Mb = 60;// [kg/kmol isopropyl ether]
42  r = Ma/Mb;
43  // From Eqn. 10.110:
44  NtoR = Area+(1/2)*log(1-x2/(1-x1))+(1/2)*log(x2*(r
       -1)+1/(x1*(r-1)+1));
45  // Since the operating line and equilibrium line are
        parallel:
46  Np = NtoR;
47  printf("Number of theoretical Units: %f\n",NtoR);
```

Scilab code Exa 10.10 Number Of Transfer Unit Dilute Solutions

```
1  clear;
2  clc;
3
4  // Illustration 10.10
5  // Page: 552
6
7  printf('Illustration 10.10 - Page: 552\n\n');
8
```

```
// Solution

//****Data****//
B = 1150;// [kg/h]
//*******//

// x and y are taken in weight ratio.
x1_prime = 0.0101;// [Wt. fraction]
xF_prime = 0.0101;// [Wt. fraction]
y2_prime = 0;// [Wt. fraction]
x2_prime = 0.001001;// [Wt. fraction]
y1_prime = 0.0782;// [Wt. fraction]
// From Illustration 10.4:
A = 990;// [kg/h]
// At the dilute end:
m1_prime = 0.798;
Value1 = m1_prime*B/A;
// At the concentrated end:
m2_prime = 0.953;
Value2 = m2_prime*B/A;
ValueAv = (Value1*Value2)^0.5;
// From Eqn. 10.116:
// Since y2_prime = 0
Value3 = x2_prime/x1_prime;
NtoR = (log((1/Value3)*(1-(1/ValueAv))+(1/ValueAv)))
   /(1-(1/ValueAv));
printf("Number of theoretical Unit : %f\n",NtoR);
```

Chapter 11

Adsorption and Ion Exchange

Scilab code Exa 11.1 Adsorption Equilibria

```
1  clear;
2  clc;
3
4  // Illustration 11.1
5  // Page: 575
6
7  printf('Illustration 11.1 - Page: 575\n\n');
8
9  // Solution
10
11 //*****Data*****//
12 Temp = 30;// [OC]
13 //*************//
14
15 // From Fig. 11.5 (Pg 572)
16 // The isosteres for various concentrations are
      straight and their slopes are measured with the
      help of milimeter rule.
17 // Data = [X(kg acetone/kg carbon) lambda(slope of
      isostere)]
18 Data = [0.05 1.170;0.10 1.245;0.15 1.3;0.20
```

```
    1.310;0.25 1.340;0.30 1.327];// [kg acetone/kg
    carbon]
lambdar = 551;// [reference at 30 OC,kJ/kg]
Val = zeros(6,5);
for i = 1:6
    Val(i,1) = Data(i,1);// [kg acetone/kg carbon]
    Val(i,2) = Data(i,2);// [slope of isostere]
    Val(i,3) = -Data(i,2)*lambdar;// [kJ/kg acetone]
end
scf(30);
plot(Val(:,1),Val(:,3));
xgrid();
xlabel("X (kg carbon / kg acetone)");
ylabel("Differential heat of adsorption (kJ / kg
    acetone)");
title("Graphical Integration");
// Area: The area under the curve between X = 0 to X
    = X
// Corresponding to Data(:,1):
Area = [-29.8 -63.0 -97.9 -134.0 -170.5 -207.5];
for i = 1:6
    Val(i,4) = Area(i);
    Val(i,5) = Area(i)+(lambdar*Val(i,1));
end
printf("X(kg acetone/kg carbon) Slope of isostere
      Differential heat of adsorption(kJ/kg acetone)
     deltaH_prime(vapour(kJ/kg carbon))  deltaH(
    liquid(kJ/kg carbon))\n");
for i = 1:6
    printf("%f \t \t \t %f \t \t %f \t \t \t \t \t
       %f \t \t \t %f\n",Val(i,1),Val(i,2),Val(i,3),
       Val(i,4),Val(i,5));
end
```

Scilab code Exa 11.2 Freundlich Equation

```
clear;
clc;

// Illustration 11.2
// Page: 596

printf('Illustration 11.2 - Page: 596\n\n');

// solution

//*****Data*****//
// x:kg carbon/kg soln
// y_star: Equilibrium colour, units/kg soln.
// X:adsorbate concentration, units/kg carbon
// Data = [x Y_star]
Data = [0 9.6;0.001 8.6;0.004 6.3;0.008 4.3;0.02
   1.7;0.04 0.7];
Yo = 9.6;// [units of colour/kg soln]
Y1 = 0.1*Yo;// [units of colour/kg soln]
Ls = 1000;// [kg soln]
//****************//

Data1 = zeros(5);
Val = zeros(5);
for i = 1:5
    Data1(i,1) = Data(i+1,1);
    Data1(i,2) = Data(i+1,2);
    Val(i) = (Data(1,2)-Data1(i,2))/Data1(i,1);
end
scf(31);
plot2d1("gll",Val,Data1(:,2));
xlabel("units of colour/kg carbon");
ylabel("units of colour/kg solution");
title("Equilibium Data(on log scale)");
xgrid();
n = 1.66;// [slope of line]
// At X = 663, Y_star = 4.3
// From eqn. 11.5
```

```
X = 663;
Y_star = 4.3;
m = Y_star/X^n;
// Freundlich Equation:
deff('[Y] = f76(X)','Y = m*X^n');
X = 0:1:1000;
scf(32);
plot(X,f76);
xgrid();
xlabel("units of colour/kg carbon");
ylabel("units of colour/kg solution");
title("Equilibium Data(on arithmetic scale)");

// Single Stage Operation:
// Since fresh carbn is used:
Xo = 0;// [units/kg carbon]
// From scf(30):
X1 = 270;// [units/kg carbon]
Data2 = [Xo Yo;X1 Y1];
scf(33);
plot(X,f76,Data2(:,1),Data2(:,2));
xgrid();
xlabel("units of colour/kg carbon");
ylabel("units of colour/kg solution");
legend("Equilbrium curve","Operating line curve");
title("Single stage operation");
// From Eqn. 11.4:
Ss = Ls*((Yo-Y1)/(X1-Xo));// [kg carbon/kg soln]
printf("Quantity of fresh carbon recquired for
   single stage operation: %f kg carbon/1000 kg
   solution\n",Ss);

// Two stage cross current operation:
// For the minimumamount of carbon:
X1 = 565;// [units/kg carbon]
Y1 = 3.30;// [units of colour/kg soln]
X2 = 270;// [units/kg carbon]
Y2 = 0.96;// [units of colour/kg soln]
```

```
74 Data3 = [Xo Yo;X1 Y1];
75 Data4 = [0 Y1;X2 Y2];
76 scf(34);
77 plot(X,f76,Data3(:,1),Data3(:,2),Data4(:,1),Data4
     (:,2));
78 xgrid();
79 xlabel("units of colour/kg carbon");
80 ylabel("units of colour/kg solution");
81 legend("Equilbrium curve","First of two Cocurrent","
     Second of two Cocurrent");
82 title("Two stage Cross current operation");
83 // From Eqn. 11.8:
84 Ss1 = Ls*(Yo-Y1)/(X1-Xo);// [kg]
85 Ss2 = Ls*(Y1-Y2)/(X2-Xo);// [kg]
86 Ss = Ss1+Ss2;// [kg]
87 printf("Quantity of fresh carbon recquired for two
     stage crosscurrent operation: %f kg carbon/1000
     kg solution\n",Ss);
88
89 // Two Stage counter current operation:
90 Yo = 9.6;
91 Y2 = 0.96;
92 // By trial and error:
93 XNpPlus1 = 0;
94 X1 = 675;
95 Data5 = [X1 Yo;XNpPlus1 Y2];
96 scf(35);
97 plot(X,f76,Data5(:,1),Data5(:,2));
98 xgrid();
99 xlabel("units of colour/kg carbon");
100 ylabel("units of colour/kg solution");
101 legend("Equilbrium curve","Two stage Counter Current
     ");
102 title("Two stage Counter Current operation");
103 // By eqn 11.14:
104 Ss = Ls*(Yo-Y2)/(X1-XNpPlus1);
105 printf("Quantity of fresh carbon recquired for two
     stage Counter Current operation: %f kg carbon
```

```
/1000 kg solution\n",Ss);
```

Scilab code Exa 11.3 Agitated Vessel for Liquid Solid Contact

```
clear;
clc;

// Illustration 11.3
// Page: 602

printf('Illustration 11.3 - Page: 602\n\n');

// Solution

//***Data***//
T = 1; //[m]
di = 0.203;// [m]
n = 1;// [for one impeller]
Density_S = 2300;// [kg/cubic m]
Density_p = 2300;// [kg/cubic m]
C = 0.150;// [m]
S = 50;// [kg]
g = 9.807;// [m/s]
dp = 8*10^(-4);// [m]
N = 8.33; //[r/s]
Temp=25;// [OC]
//*************//

// Assume:
Po = 5;
viscosity_L = 8.94*10^(-4);// [kg/m.s]
Density_L = 998;// [kg/cubic m]
delta_Density = Density_S-Density_L;// [kg/cubic m]
// By Eqn. 11.23:
Vts = g*dp^2*delta_Density/(18*viscosity_L);// [m/s]
```

```
32 // By defn. of power number:
33 // P = Po*Density_m*di^5*Ni^3
34 // vm = %pi*T^2*(Z+C)/4
35 // Solid Volume = S/Density_p;
36 // If these are substituted in Eqn. 11.22
37 deff('[y] = f(Z)','y = (((Z+C)^(1/3))*exp(4.35*Z/(T
   -0.1))) -((1.0839*Po*di^(11/2)*N^3*Density_p^(2/3)
   )/(g*Vts*T^(7/6)*S^(2/3)))');
38 Z = fsolve(7,f);// [m]
39 phi_Sm = 4*S/(%pi*T^2*(Z+C)*Density_p);
40 Density_m = (phi_Sm*Density_p)+((1-phi_Sm)*Density_L
   );// [kg/cubic m]
41 phi_Ss = 0.6;
42 viscosity_m = viscosity_L/(1-(phi_Sm/phi_Ss))^1.8;//
    [kg/m.s]
43 Re = di^2*N*Density_m/viscosity_m;
44 P = Po*Density_m*N^3*di^5;// [W]
45 printf("Agitator Power required: %f W\n",P);
```

Scilab code Exa 11.4 Agitated Power

```
1 clear;
2 clc;
3
4 // Illustration 11.4
5 // Page: 604
6
7 printf('Illustration 11.4 - Page: 604\n\n');
8
9 //****Data*****//
10 // b: kerosene c:water
11 // c:kg water/cubic m liquid
12 Density_l = 783;// [kg/cubic m]
13 viscosity_l = 1.7*10^(-3);// [kg/m.s]
14 Mb = 200;// [kg/kmol]
```

```
15 Density_p = 881;// [kg/cubic m]
16 m = 0.522;// [(kg water/cubic m kerosene)/(kg water/
      kg gel)]
17 Xo = 0;// [kg H2O/kg gel]
18 //**************//
19
20 // Solution (a)
21 co = Density_l*4*10^(-5);// [kg water/cubic m]
22 c1 = Density_l*5*10^(-6);// [kg water/cubic m]
23 // For Ss minimum:
24 X1 = c1/m;// [kg H2O/kg gel]
25 // By Water Balance:
26 SsminByVl = (co-c1)/(X1-Xo);// [kg gel/cubic m
      kerosene]
27 printf("Minimum Solid/Liquid ratio used: %f kg gel/
      cubic m kerosene",SsminByVl);
28 printf("\n");
29
30 // Solution (b)
31 // Basis: 1 batch,1.7 cubic m kerosene
32 Vl = 1.7;// [cubic m]
33 Ss = 16*1.7;// [kg gel]
34 V = Ss/Density_p;// [Vol. solid, cubic m]
35 Vt = 1.7+V;// [Total batch volume, cubic m]
36 // Take Z = T
37 T = (Vt*4/%pi)^(1/3);// [m]
38 // To allow for the adequate free board:
39 h = 1.75;// [Vessel height,m]
40 // Use a six-blade disk impeller.
41 // From Fig. 11.26:
42 // dp corresponding to 14 mesh:
43 dp = 1.4/1000;// [m]
44 TBydi1 = 2;
45 Value1 = (Density_p-Density_l)/Density_l;
46 // From Fig. 11.26:
47 TBydi2 = 4.4;
48 TBydiAv = (TBydi1+TBydi2)/2;
49 di = T/TBydiAv;// [m]
```

```
fr = 0.6;// [settled volume fraction of solids]
Vs = V/fr;// [cubic m]
depth = Vs/((%pi*(T^2))/4);// [m]
// The depth of settled solid is negligible.
// Locate the turbine 150mm from the bottom of the
   tank.
C = 0.150;// [m]

// Power:
// Use the sufficient agitator power to lift the
   solids to 0.6 m above the bottom of the vessel.
Z_prime = 0.6-C;// [m]
// The properties of the slurry in 0.6 m above the
   bottom of the vessel.
Vm = 0.6*%pi*T^2/4;// [square m]
phi_Sm = V/Vm;// [vol fraction solid]
// From Eqn. 11.24:
Density_m = (phi_Sm*Density_p)+((1-phi_Sm)*Density_l
   );// [kg/cubic m]
// From Eqn. 11.25:
phi_Ss = 0.8;
viscosity_m = viscosity_l/(1-(phi_Sm/phi_Ss))^1.8;//
    [kg/m.s]
g = 9.81;// [m/s^2]
// From Eqn. 11.23:
delta_Density = Density_p-Density_l;// [kg/cubic m]
Vts = g*dp^2*delta_Density/(18*viscosity_l);// [m/s]
// From Eqn. 11.22:
n = 1;
P = (g*n*Density_m*Vm*Vts)*(phi_Sm^(2/3))*(TBydiAv
   ^(1/2))*exp((4.35*Z_prime/T)-0.1);// [W]
// Assume:
Po = 5;
N = (P/(Po*Density_m*di^5))^(1/3);// [r/s]
// Use:
N1 = 2;// [r/s]
Re = di^2*N1*Density_m/viscosity_m;
// From fig. 6.5: Po = 5
```

```
82 // Hence our assumption was right.
83 printf("Power delivered to the slurry: %f W\n",P*(N1
   /N)^3);
84 printf("Power to the motor will be larger, depending
    on the efficiency of the motor and speed reducer
   .\n");
85
86 // Mass transfer:
87 // From Eqn. 11.28:
88 Rep = (dp^(4/3))*(P/Vl)^(1/3)*(Density_l^(2/3)/
   viscosity_l);
89 // From Eqn. 2.44:
90 Temp = 298;// [K]
91 phi = 1;
92 Va = 0.0756;// [Chapter 2 notation]
93 Dl = ((117.3*10^(-18))*((phi*Mb)^0.5)*Temp)/(
   viscosity_l*(Va^(0.6)));
94 ScL = viscosity_l/(Density_l*Dl);
95 if  dp<(2/1000)
96     // From Eqn. 11.29:
97     ShL = 2+(0.47*Rep^0.62*(1/TBydiAv^0.17)*ScL
        ^0.36);
98 else
99     // From Eqn. 11.30:
100    ShL = 0.222*Rep^(3/4)*ScL^(1/3);
101 end
102 kL = ShL*Dl/dp;// [m/s]
103 apS = (%pi*dp^2)/(%pi*dp^3*Density_p/6);
104 apL = apS*16;// [square m/cubic m liquid]
105 Ratio = Ss/(Vl*m);
106 // From Eqn. 11.40:
107 thetha = log((co/c1)/(1+(1/Ratio)-(1/Ratio)*(co/c1))
   )/((1+(1/Ratio))*kL*apL);
108 printf("Contacting Time required: %f min\n",thetha
   /60);
```

Scilab code Exa 11.5 Continuous cocurrent adsorption and liquid and solid mass transfer resistances

```
clear;
clc;

// Illustration 11.5
// Page: 606

printf('Illustration 11.5 - Page: 606\n\n');

// Solution

//*****Data******//
Vl = 1.1*10^(-4);// [cubic m/s]
Ss = 0.0012;// [kg/s]
Density_p = 1120;// [kg/cubic m]
dp = 8*10^(-4);// [m]
Ds = 2*10^(-11);// [square m/s]
Dl = 7.3*10^(-10);// [square m/s]
m = 0.2;// [(kg Cu2+/cubic m soln)/(kg Cu2+/kg resin
    )]
T = 1;// [m]
//********************//

Z = T;// [m]
// The particles will be lifted to the top of the
    vessel.
Z_prime = 0.5;// [m]
viscosity_l = 8.94*10^(-4);// [kg/m.s]
Density_l = 998;// [kg/cubic m]
delta_Density = Density_p-Density_l;// [kg/cubic m]
g = 9.80;// [m/square s]
// From Eqn. 11.23:
```

```
Vts = g*dp^2*delta_Density/(18*viscosity_l);
Vm = %pi*T^2*Z/4;// [cubic m]
Vs = Ss/Density_p;// [cubic m/s]
phi_Sm = Vs/(Vs+Vl);// [vol fraction]
// From eqn. 11.24:
Density_m = (phi_Sm*Density_p)+((1-phi_Sm)*Density_l
   );// [kg/cubic m]
// From Eqn. 11.22:
n = 1;
di = 0.3;// [m]
P = (g*n*Density_m*Vm*Vts)*(phi_Sm^(2/3))*((T/di)
   ^(1/2))*exp((4.35*Z_prime/T)-0.1);// [W]
// To estimate the impeller speed:
// Assume:
Po = 5;
N = (P/(Po*Density_m*di^5))^(1/3);// [r/s]
Re = di^2*N*Density_m/viscosity_l;
// From fig. 6.5: Assumption of Po was correct.
printf("Speed of the impeller: %f r/s\n",N);
vT = (%pi/4)*T^2*Z;// [cubic m]
vL = vT*(1-phi_Sm);
// From Eqn. 11.28:
Rep = (dp^(4/3))*(P/vL)^(1/3)*(Density_l^(2/3)/
   viscosity_l);
ScL = viscosity_l/(Density_l*Dl);
if  dp<(2/1000)
    // From Eqn. 11.29:
    ShL = 2+(0.47*Rep^0.62*((di/T)^0.17)*ScL^0.36);
else
    // From Eqn. 11.30:
    ShL = 0.222*Rep^(3/4)*ScL^(1/3);
end
ShL = 130.3;// Value wrong in book
kL = ShL*Dl/dp;// [m/s]
// Since the dispersion is uniform throughout the
   vessel, the residence time for both liquid and
   solid is same.
thetha = vL*(1-phi_Sm)/Vl;// [s]
```

```
63 // From Fig. 11.27:
64 abcissa = m*kL*dp/(2*Ds*Density_p);
65 Parameter = 2*m*kL*thetha/(dp*Density_p);
66 co = 100*Density_l/10^6;// [kg/cubic m]
67 EMS = 0.63;
68 Xo = 0;
69 // From Eqn. 11.44:
70 // (1): X1-(EMS/m)*c1 = 0
71 // Solute balance:
72 // (2): (Ss*X1)+(vL*c1) = (vL*co)+(Xo*Ss)
73 a = [1 -(EMS/m);Ss Vl];
74 b = [0;((Vl*co)+(Xo*Ss))];
75 soln = a\b;
76 X1 = soln(1);
77 c1 = soln(2);
78 printf("Effluent Cu2+ conc. %f ppm\n",c1*10^(6)/
   Density_l);
```

Scilab code Exa 11.6 Continuous Countercurrent Isothermal Adsorber

```
1 clear;
2 clc;
3
4 // Illustration 11.6
5 // Page: 616
6
7 printf('Illustration 11.6 - Page: 616\n\n');
8
9 // Solution
10
11 //*****Data*****//
12 // a: air b:silica
13 Density_a = 1.181;// [kg/cubic m]
14 Density_b = 671.2;// [kg/cubic m]
15 kSap = 0.965;// [kg H2O/square m s]
```

```
16 Y1 = 0.005;// [kg H2O/kg dry air]
17 Y2 = 0.0001;// [kg H2O/kg dry air]
18 Ss = 0.680;// [square m/s]
19 Gs = 1.36;// [kg/square m.s]
20 X2 = 0;// [kg H2O/kg dry air]
21 // Equilibrium function:
22 m = 0.0185;
23 //************//
24 X1 = (Gs*(Y1-Y2)/Ss)+X2;// [kg H2O/kg dry air]
25 deff('[Y] = f77(X)','Y = m*X');
26 Y2_star = f77(X2);// [kg H2O/kg dry gel]
27 Y1_star = f77(X1);// [kg H2O/kg dry gel]
28 deltaY = ((Y1-Y1_star)-(Y2-Y2_star))/log((Y1-Y1_star
   )/(Y2-Y2_star));
29 NtoG = (Y1-Y2)/deltaY;
30 // If the fixed bed data are to be used for
   estimating the mass transfer coeffecient for a
   moving bed of solids
31 va = Ss/Density_b;// [m/s]
32 vb = Gs/Density_a;// [m/s]
33 rel_v = va+vb;// [relative velocity,m/s]
34 G_prime = rel_v*Density_a;// [relative mass velocity
    of air,kg/square m s]
35 HtG = Gs/(31.6*G_prime^0.55);// [m]
36 HtS = Ss/kSap;// [m]
37 // By Eqn. 11.52:
38 HtoG = HtG+(m*Gs/Ss)*HtS;// [m]
39 Z = NtoG*HtoG;// [m]
40 printf("Height of continuous countercurrent
   isothermal absorber for drying: %f m\n",Z);
```

Scilab code Exa 11.7 Fractionation

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

```
3
4 // Illustration 11.7
5 // Page: 619
6
7 printf('Illustration 11.7 - Page: 619\n\n');
8
9 // Solution
10
11 //*****Data*****//
12 // a: C2H4 b:C3H8
13 // The equlibrium curve is plotted in Fig.11.33 (Pg
     620)
14 // C3H8 is more strongly adsorbed component and
     composition in the gas and adsorbate are
     expressed as weight fraction C3H8.
15 Ma = 28;// [kg/kmol]
16 Mb = 44.1;// [kg/kmol]
17 xaF = 0.6;// [mole fraction]
18 xbF = 0.4;// [mole fraction]
19 xa1 = 0.05;// [mole fraction]
20 xa2 = 0.95;// [mole fraction]
21 //***************//
22
23 xF = xbF*Mb/((xbF*Mb)+(xaF*Ma));// [wt. fraction
     C3H8]
24 xb1 = 1-xa1;// [mole fraction]
25 x1 = xb1*Mb/((xb1*Mb)+xa1*Ma);// [wt. fraction C3H8]
26 xb2 = 1-xa2;// [mole fraction]
27 x2 = xb2*Mb/((xb2*Mb)+(xa2*Ma));// [wt. fraction
     C3H8]
28 // Basis: 100 kg feed gas
29 F = 100;// [kg]
30 // (1): R2+PE = F [From Eqn. 11.63]
31 // (2): (R2*x2)+(PE*x1) = (F*xF) [From Eqn. 11.64]
32 // Solving simultaneously:
33 a = [1 1;x2 x1];
34 b = [F;(F*xF)];
35 soln = a\b;
```

```
R2 = soln(1);// [kg]
PE = soln(2);// [kg]
// Point F at xF and point E1 at x1 are located on
   the diagram.
// From the diagram:
N1 = 4.57;// [kg carbon/kg adsorbate]
// The minimum reflux ratio is found as it is for
   the extraction.
delta_Em = 5.80;
Ratio = (delta_Em/N1)-1;// [kg reflux gas/kg product
   ]
R1_m = Ratio*PE;// [kg]
E1_m = R1_m+PE;// [kg]
B_m = N1*E1_m;// [kg carbon/100 kg feed]
Ratio1 = 2*Ratio;
// From Eqn. 11.58:
N_deltaE = (Ratio1+1)*N1;// [kg carbon/kg adsorbate]
// Point deltaE is located on the diagram:
R1 = Ratio1*PE;// [kg]
E1 = R1+PE;// [kg]
B = N1*E1;// [kg]
N_deltaR = -(B/R2);// [kg carbon/kg adsorbate]
// Random lines such as the delta_RK are drawn from
   detaR, and the intersection of equilibrium curves
    are projected downward in the manner shown to
   provide the adsorption section operating curve.
// Similarly random lines such as delta_EJ are drawn
    from deltaE, and the intersections are projected
    downwards to provide the enriching section
   operating curve.
// Data = [x x_star]
Data = [0.967 0.825;0.90 0.710;0.80 0.60;0.70
   0.50;0.60 0.43;0.512 0.39;0.40 0.193;0.30
   0.090;0.20 0.041;0.0763 0.003];
Val = zeros(10);
for i = 1:10
    Val(i) = 1/((Data(i,1))-Data(i,2));
end
```

```
63 scf(36);
64 plot(Data(:,1),Val);
65 xgrid();
66 xlabel("x");
67 ylabel("1 / (x-x*)");
68 title("Graphical Integraion");
69 // The area under the curve between x1 & xF, for the
       enriching section:
70 Area1 = 2.65;
71 // The area under the curve between xF & x2, for the
       adsorption section:
72 Area2 = 2.67;
73 r = Ma/Mb;
74 // From Eqn.11.66:
75 // For the enriching section:
76 NtoG1 = Area1-log((1+(r-1)*x1)/(1+(r-1)*xF));
77 // For the adsortion section:
78 NtoG2 = Area2-log((1+(r-1)*x1)/(1+(r-1)*xF));
79 NtoG = NtoG1+NtoG2;
80 printf("Number of transfer units: %f",NtoG);
```

Scilab code Exa 11.8 Unsteady State Fixed Bed Absorbers

```
1 clear;
2 clc;
3
4 // Illustration 11.8
5 // Page: 627
6
7 printf('Illustration 11.8 - Page: 627\n\n');
8
9 // Solution
10
11 //******Data******//
12 rate = 0.1;// [kg/s]
```

```
13 conc = 3;// [kg vapour/100cubic m]
14 Density_p = 720;// [kg/cubic m]
15 Density_bed = 480;// [kg/cubic m]
16 capablity = 0.45;// [kg vapour/kg carbon]
17 dp = 0.0028;// [m]
18 time = 3;// [h]
19 //*********************//
20
21 Vap_adsorbed = time*3600*rate;// [kg]
22 C_required = Vap_adsorbed/capablity;
23 // Two beds will be needed: one adsorbing and
     another regenerated.
24 totC_required = 2*C_required;// [kg]
25 printf("Amount of caron required: %d kg\n",
     totC_required);
26 Vol = (C_required/Density_bed);
27 // Assume:
28 Z = 0.5;// [m]
29 Area = Vol/Z;// [square m]
30 // From Eqn. 6.66:
31 T = 35;// [OC]
32 viscosity_air = 1.82*10^(-5);// [kg/m.s]
33 Density_air = (29/22.41)*(273/(T+273));
34 e = 1-(Density_bed/Density_p);
35 G = rate*(100/conc)*(Density_air/(Area));// [kg/
     square m.s]
36 Re = dp*G/viscosity_air;
37 Z = 0.5;// [m]
38 deff('[y] = f78(delta_p)','y = ((delta_p/Z)*(e^3*dp*
     Density_air)/((1-e)*G^2))-(150*(1-e)/Re)-1.75');
39 delta_p = fsolve(7,f78);
40 printf("The pressure drop is: %f N/square m\n",
     delta_p);
```

Scilab code Exa 11.9 Time Required to reach Breakpoint

```
clear;
clc;

// Illustration 11.9
// Page: 636

printf('Illustration 11.9 - Page: 636\n\n');

// Solution

//*****Data******//
Yo = 0.00267;// [kg H2O/kg dry air]
Yb = 0.0001;// [kg H2O/kg dry air]
Ye = 0.024;// [kg H2O/kg dry air]
Z = 0.61;// [m]
G_prime = 0.1295;// [kg/square m.s]
//******************//

// The equilicrium data is plotted in Fig. 11.45 (Pg 637)
// The gel is initially "dry" and the effluent air initially of so low a humidity asto be substantially dry, so that the operating line passes through the origin of the figure
// The operating line is then drawn to intersect the equilibrium curve.
// Data = [Y[kg H2O/kg dry air] Y_star[kg H2O/kg dry air]]
Data = [0.0001 0.00003;0.0002 0.00007;0.0004 0.00016;0.0006 0.00027;0.0008 0.00041;0.0010 0.00057;0.0012 0.000765;0.0014 0.000995;0.0016 0.00123;0.0018 0.00148;0.0020 0.00175;0.0022 0.00203;0.0024 0.00230];
Val1 = zeros(13);
// Val1 = [1/(Y-Y_star)]
for i = 1:13
    Val1(i) = 1/(Data(i,1)-Data(i,2));
end
```

```
// Graphical Integration:
scf(37);
plot(Data(:,1),Val1);
xgrid();
xlabel("Y(kg H20 / kg dry air)");
ylabel("1 / (Y-Y_star)");
title("Graphical Integration");
// Area under The curve between Y = Yb and Y = Y:
Area = [0 0.100 2.219 2.930 3.487 3.976 4.438 4.915
   5.432 6.015 6.728 7.716 9.304];
// The total number of transfer unit corresponding
   to adsorption zone:
NtoG = 9.304;
Val2 = zeros(13);
Val3 = zeros(13);
// Val2 = [(w-wb)/wo]
// Val3 = [Y/Yo]
for i = 1:13
    Val2(i) = Area(i)/NtoG;
    Val3(i) = Data(i,1)/Yo;
end
// Eqn. 11.74 can be arranged as follows:
// f = integrate((1-(Y/Yo)),(w-wb)/wa,0,1)
scf(38);
plot(Val2,Val3);
xgrid();
xlabel("(w-wb) / wo");
ylabel("Y / Yo");
title("Break through curve");
// From area above the curve of scf(2):
f = 0.530;

Gs = G_prime;// [kg/square m.s]
// From Illustration: 11.6
kYap = 31.6*G_prime^0.55;// [kg H2O/cubic m s
   delta_Y]
kSap = 0.965;// [kg H2O/cubic m s delta_X]
// From Fig. 11.48:
```

```
Xt = 0.0858;// [kg H2O/kg gel]
// From Eqn. 11.76:
Ss = Yo*Gs/Xt;// [kg/square m.s]
m = 0.0185;// [average slope of equilibrium curve]
// From Eqn. 11.51 & Eqn. 11.52:
HtG = Gs/kYap;// [m]
HtS = Ss/kSap;// [m]
HtoG = HtG+(m*Gs/Ss)*HtS;// [m]
// From Eqn. 11.79:
Za = NtoG*HtoG;// [m]
// From Eqn. 11.74:
Degree = (Z-(f*Za))/Z;
Density_bed = 671.2;// [Illustration 11.6, kg/cubic
   m]
mass_gel = Z*Density_bed;// [kg/square m]
// At saturation point the gel contins:
Y1 = mass_gel*Degree*Xt;// [kg H2O/square m cross
   section]
// The air introduces:
Y2 = Gs*Yo;// [kg/square m s]
printf("Time to reach breakpoint is: %f h\n",(Y1/(Y2
   *3600)));
```

Scilab code Exa 11.10 Calculation of Bed depth

```
clear;
clc;

// Illustration 11.10
// Page: 640

printf('Illustration 11.10 - Page: 640\n\n');

// Solution

```

```
11 //*****Data******//
12 // a:N2 b:H2O
13 Mb = 18;// [kg/kmol]
14 Ma = 29;// [kg/kmol]
15 Z = 0.268;// [m]
16 Xo_solid = 0.01;// [kg H20/kg solid]
17 Density_bed = 712.8;// [kg/cubic m]
18 T = 28.3;// [OC]
19 P = 593;// [kN/square m]
20 Gs = 4052;// [kg/square m.h]
21 Xo_gas = 1440*10^(-6);// [mole fraction]
22 //*********************//
23
24 // Yo_star is in equilibrium with Xo:
25 Xo = 0;// [kg H20/kg solid]
26 Yo_star = 0;// [kg H20/kg N2]
27 thetha_t = 12.8;// [h]
28 thetha_b = 9;// [h]
29 // The breakthrough data are plotted in the manner
      of Fig. 11.47 (Pg 639) and thetha_s is dtermined:
30 thetha_s = 10.9;// [h]
31 Xt = 0.21;// [kg H20/kg solid]
32 // From Eqn. 11.81:
33 LUB = (Z/thetha_s)*(thetha_s-thetha_b);
34 // For thetha_b = 15 h
35 thetha_b = 15;// [h]
36 Yo = (Xo_gas/(1-Xo_gas))*(Mb/Ma);// [kg H20/kg N2]
37 // From Eq. 11.82:
38 Zs = Gs*(Yo-Yo_star)*thetha_b/(Density_bed*(Xt-
      Xo_solid));// [m]
39 // From Eqn. 11.85:
40 Z = LUB+Zs;
41 printf("Height of adsorbent column: %f m\n",Z);
```

Scilab code Exa 11.11 Ion Exchange

```
clear;
clc;

// Illustration 11.11
// Page: 645

printf('Illustration 11.11 - Page: 645\n\n');

// Solution

//****Data****//
// For collection of Cu2+:
V = 37850;// [l/h]
c1 = 20;// [meq Cu2+/l]
c2 = 0.01*c1;// [meq Cu2+/l]
Mass_rate = 2;// [meq Cu2+/g resin h (meq Cu2+/l)]
exchanged = V*(c1-c2);// [meq/h]
X2 = 0.30;// [meq Cu2+/g]
//************///

// The point(c2,X2) is plotted in Fig. 11.48(a), Pg
   645:
// For the minimum resin/solution ratio and an
   infinitely tall tower, the operating line pass
   though point P.
X = 4.9;// [meq Cu2+/g]
MinRate = exchanged/(X-X2);// [g/h]
Rate = 1.2*MinRate;// [g/h]
// Copper balance:
X1 = (exchanged/Rate)+X2;// [meq Cu2+/g resin]
// The point (c1,x1) is ploted in Fig. 11.48(a) and
   operating line drawn can be straight line at this
    low conc.
// Adapting Eqn. 11.48 and rearranging:
// S*Z*Density_s = (V/Mass_rate)*integrate(1/(c-
   c_star),c,c1,c2)
// Mass_rate = KL_prime*ap/Density_s
// From the equilibrium curve:
```

```
33 // Data = [c c_star]
34 Data = [20 2.4;16 1.9;12 0.5;8 0.25;4 0.10;2 0.05;1
      .02;0.2 0];
35 Val = zeros(8);
36 for i = 1:8
37     Val(i) = 1/(Data(i,1)-Data(i,2));
38 end
39 scf(39);
40 plot(Data(:,1),Val);
41 xgrid();
42 xlabel("c");
43 ylabel("1 / (c-c*)");
44 title("Graphical Integration");
45 // From Graphical Integration:
46 Area = 5.72;
47 // holdup = S*Z*Density_s
48 holdup = V*Area/(Mass_rate);
49 printf("Resin Holdup: %f g\n",holdup);
50
51 // Regeneration of resin:
52 // For 70% utilisation of 2N acid, feed must contain
      :
53 V = exchanged;
54 F = V/(0.70*2000);// [l/h]
55 c1 = 0;// [meq Cu2+/l]
56 c2 = V/F;// [meq Cu2+/l]
57 X1 = 0.30;// [meq Cu2+/g resin]
58 X2 = 4.12;// [meq cu2+/g resin]
59 // The points (c1,X1) and (c2,X2) are plotted on Fig
      11.48(b), Pg 645
60 c1_star = 120;// [meq Cu2+/l]
61 c2_star = 1700;// [meq Cu2+/l]
62 logmean = ((c1_star-c1)-(c2_star-c2))/log((c1_star-
      c1)/(c2_star-c2));
63 Mass_rate = 0.018;// [meq Cu2+/g resin h (meq Cu2+/l
      )]
64 // Substituting in equation:
65 deff('[y] = f79(holdup)','y = (V*(c2-c1))-(Mass_rate
```

```
    *holdup*logmean)');
66 holdup = fsolve(7,f79);
67 printf("Resin Holdup in the regeneration Tower is %e
    g\n",holdup);
```

Chapter 12

Drying

Scilab code Exa 12.1 Moisture Evaporated

```
1  clear;
2  clc;
3
4  // Illustration 12.1
5  // Page: 660
6
7  printf('Illustration 12.1 - Page: 660\n\n');
8
9  // Solution
10
11 //****Data****//
12 F=1000;// [kg]
13 Xo=0.8;// [wt. fraction water]
14 X1=0.05;// [wt. fraction water]
15 //************//
16
17 Yo=Xo/(1-Xo);// [kg water/kg dry solid]
18 Y1=X1/(1-X1);// [kg water/kg dry solid]
19 solid=F*(1-X1);// [kg]
20 printf("Moisture to be evaporated: %f kg\n",solid*(
     Yo-Y1));
```

Scilab code Exa 12.2 Batch Drying

```
clear;
clc;

// Illustration 12.2
// Page: 665

printf('Illustration 12.2 - Page: 665\n\n');

// Solution

// ***Data***//
Y1 = 0.05;// [kg water/kg dry air]
Yair = 0.01;// [kg water/kg dry air]
TempG1 = 95;// [OC]
width = 1;// [m]
apart = 100/1000;// [m]
deep = 38/1000;// [m]
Rate_evaporation=7.5*10^(-3);// [kg/s]
//*****************//

// From Table 7.1: (Pg 234)
vH = (0.00283+(0.00456*Y1))*(TempG1+273);// [cubic m
   /kg dry air]
freeArea = width*(apart-deep)*11;// [square m]
// Rate of air flow at 1:
Rate_air1 = 3*freeArea/vH;// [square m]
Y2 = Y1+(Rate_evaporation/Rate_air1);// [kg water/kg
    dry air]
// Assuming adiabatic drying:
// From adiabatic saturation curve, Fig 7.5: (Pg
   232)
TempG2 = 86;// [OC]
```

```
30 // Overall Water Balance:
31 G = Rate_evaporation/(Y1-Yair);// [kg dry air/s]
32 // Rate of air flow at 3:
33 Rate_air3 = Rate_air1+G;// [kg dry air/s]
34 // Rate of air flow at 4:
35 Rate_air4 = Rate_air3;// [kg dry air/s]
36 // Volumetric Rate through fan:
37 Rate_fan = Rate_air3/vH;// [cubic m/s]
38 printf("Percentage of air recycled is: %f %%\n",(
      Rate_air1/Rate_air3)*100);
39 printf("\n");
40
41 // From Fig. 7.5 (page 232):
42 // Saturated enthalpy at adiabatic saturation temp.
43 Enthalpy1 = 233;// [kJ/kg dry air]
44 Enthalpy2 = 233;// [kJ/kg dry air]
45 // Enthalpy of fresh air:
46 Enthalpy_air = 50;// [kJ/kg dry air]
47 // Assuming complete mixing, by Enthalpy mixing:
48 Enthalpy3 = ((Enthalpy1*Rate_air1)+(Enthalpy_air*G))
      /Rate_air3;// [kJ/kg dry air]
49 Enthalpy4 = Enthalpy3;// [kJ/kg dry air]
50 // From table 7.1: (Pg 234)
51 Temp_dry = ((Enthalpy3*1000)-(2502300*Y1))
      /(1005+(1884*Y1));
52 Power = (Enthalpy2-Enthalpy3)*Rate_air3;// [kW]
53 // From Fig. 7.5, (Pg 232)
54 DewPoint1 = 40.4;// [OC]
55 DewPoint2 = 41.8;// [OC]
56 DewPoint3 = 40.4;// [OC]
57 DewPoint4 = 40.4;// [OC]
58 printf("At Point 1\n")
59 printf("Enthalpy of air: %f kJ/kg dry air\n",
      Enthalpy1);
60 printf("Dew Point of air: %f OC\n",DewPoint1);
61 printf("\n");
62 printf("At Point 2\n")
63 printf("Enthalpy of air: %f kJ/kg dry air\n",
```

```
    Enthalpy2);
64 printf("Dew Point of air: %f OC\n",DewPoint2);
65 printf("\n");
66 printf("At Point 3\n")
67 printf("Enthalpy of air: %f kJ/kg dry air\n",
    Enthalpy3);
68 printf("Dew Point of air: %f OC\n",DewPoint3);
69 printf("\n");
70 printf("At Point 4\n")
71 printf("Enthalpy of air: %f kJ/kg dry air\n",
    Enthalpy4);
72 printf("Dew Point of air: %f OC\n",DewPoint4);
73 printf("\n");
74 printf("Dry bulb temparature of air: %f OC\n",
    Temp_dry);
75 printf("Power delivered by heater: %f kW\n",Power);
```

Scilab code Exa 12.3 Time of Drying

```
1  clear;
2  clc;
3
4  // Illustration 12.3
5  // Page: 671
6
7  printf('Illustration 12.3 - Page: 671\n\n');
8
9  // Solution
10
11 // ***Data***//
12 SsByA = 40;
13 x1 = 0.25;// [moisture fraction]
14 x2 = 0.06;// [moisture fraction]
15 //***********//
16
```

```
17 X1 = x1/(1-x1);// [kg moisture/kg dry solid]
18 X2 = x2/(1-x2);// [kg moisture/kg dry solid]
19 // Fig. 12.10 (Pg 668) indicates that both constant
      and falling rate periods are involved.
20
21 // Constant Rate period:
22 // From Fig. 12.10 (Pg 668):
23 Xc = 0.200;// [kg moisture/kg dry solid]
24 Nc = 0.3*10^(-3);// [kg/square m.s]
25 // From Eqn. 12.4:
26 thetha1 = SsByA*(X1-Xc)/Nc;// [s]
27
28 // Falling Rate Period:
29 // From Fig. 12.10 (Pg 668):
30 // Data=[x N*10^3]
31 Data = [0.2 0.3;0.18 0.266;0.16 0.239;0.14
      0.208;0.12 0.180;0.10 0.150;0.09 0.097;0.08
      0.070;0.07 0.043;0.064 0.025];
32 Val = zeros(10);
33 // Val=[(1/N)*10^(-3)]
34 for i = 1:10
35     Val(i) = 1/Data(i,2);
36 end
37 scf(40);
38 plot(Data(:,1),Val);
39 xgrid();
40 xlabel("x [kg moisture / kg dry solid]");
41 ylabel("10^(-3) / N");
42 title("Graphical Integration Falling Rate Period");
43 // Area under the curve:
44 Area = 1060;
45 // From Eqn. 12.3:
46 thetha2 = SsByA*Area;// [s]
47 thetha = thetha1+thetha2;// [s]
48 printf("Total Drying Time: %f h\n",thetha/3600);
```

Scilab code Exa 12.4 Cross Circulation Drying

```
clear;
clc;

// Illustration 12.4
// Page: 676

printf('Illustration 12.4 - Page: 676\n\n');

// Solution (a)

//***Data***//
// For rectangular pan:
l = 0.7;// [m]
b = 0.7;// [m]
zS = 0.025;// [m]
zM = 0.0008;// [m]
d = 0.1;// [m]
Y1 = 0.01;// [kg water/kg dry air]
TempG = 65;// [OC]
v = 3;// [m/s]
TempR = 120;// [OC]
//*************//

// From Table 7.1: (Pg 234)
vH = (0.00283+(0.00456*Y1))*(TempG+273);// [cubic m/
    kg dry air]
Density_G = (1+Y1)/vH;// [kg/cubic m]
G = v*Density_G;// [kg/square m.s]
de = 4*d*l/(2*(l+d));// [m]
// From Eqn. 12.20:
hc = 5.90*G^0.71/de^0.29;// [W/square m.K]
// Assume:
```

```
32 e = 0.94;
33 // Estimate:
34 TempS = 38;// [OC]
35 // From Eqn. 12.14:
36 hR = e*5.729*10^(-8)*((273+TempR)^4-(273+TempS)^4)
      /((273+TempR)-(273+TempS));
37 A = l*b;// [square m]
38 Am = A;// [square m]
39 As = 4*l*zS;// [square m]
40 Au = Am+As;// [square m]
41 // Thermal Coductivities:
42 kM = 45;// [W/m.K]
43 kS = 3.5;// [W/m.K]
44 // By Eqn. 12.16:
45 Uk = 1/(((1/hc)*(A/Au))+((zM/kM)*(A/Au))+((zS/kS)*(A
      /Am)));// [W/squre m.K]
46 // From Table 7.1: (Pg 234)
47 Cs = 1005+(1884*Y1);// [kJ/kg]
48 // At estimated 38 OC
49 lambdaS = 2411.4;// [kJ/kg]
50 // From Eqn. 12.18:
51 // (Ys-Y1)*lambdaS*10^3/Cs = ((1+(Uk/hc))*(TempG-
      Temps))+((hR/hC)*(TempR-TempS))
52 // On Simplifying:
53 // Ys = 0.0864-(10.194*10^(-4)*TempS)
54 // The eqn. is solved simultaneously with the
      saturated humidity curve of the psychometric
      chart for the air water mixture.
55 // From Fig. 12.12: (Pg 677)
56 Ys = 0.0460;// [kg water/kg dry air]
57 TempS = 39;// [OC]
58 // At 39 OC
59 lambdaS = 2409.7;// [kJ/kg]
60 // From Eqn. 12.17:
61 Nc = (((hc+Uk)*(TempG-TempS))+(hR*(TempR-TempS)))/(
      lambdaS*10^3);// [kg water evaporated/square m.s]
62 printf("The Evaporation Rate: %e kg/s\n",Nc*A);
63
```

```
64 // Solution (b)
65 // When no radiation or conduction of heat through
      the solid occurs, the drying surface assumes wet
      bulb temparature of the air.
66 // From Fig. 12.12 (Pg 677)
67 TempS = 28.5;// [OC]
68 Ys = 0.025;// [kg water/kg dry air]
69 lambdaS = 2435;// [kJ/kg]
70 // From Eqn. 12.17:
71 Nc = hc*(TempG-TempS)/(lambdaS*10^3);// [kg/aquare m
      .s]
72 printf("The Evaporation Rate: %e kg/s\n",Nc*A);
```

Scilab code Exa 12.5 Drying of Bound Moisture

```
1 clear;
2 clc;
3
4 // Illustration 12.5
5 // Page: 684
6
7 printf('Illustration 12.5 - Page: 684\n\n');
8
9 // Solution
10
11 //***Data***//
12 x1 = 0.025;// [moisture fraction]
13 x2 = 0.001;// [moisture fraction]
14 zS = 0.018;// [m]
15 dp = 2*10^(-4);// [m]
16 Density_S = 1350;// [kg dry solid/cubic m]
17 //***********//
18
19 X1 = x1/(1-x1);// [kg water/kg dry air]
20 X2 = x2/(1-x2);// [kg water/kg dry air]
```

```
// From Fig 7.5 (Pg 232)
Y1 = 0.0153;// [kg water/kg dry air]
Tempas = 24;// [OC]
Yas = 0.0190;// [kg water/kg dry air]
Gs = 0.24;// [kg dry air/square m.s]
Gav = Gs+(Gs*(Y1+Yas)/2);// [kg dry air/square m.s]
// From Eqn. 12.26:
Nmax = Gs*(Yas-Y1);// [kg evaporated/square m.s]
viscosity_air = 1.8*10^(-5);// [kg/m.s]
Value = integrate('1/(Nmax*(1-exp(-(0.273/dp^0.35)
   *((dp*Gav/viscosity_air)^0.215)*(Density_S*zS*X)
   ^0.64)))','X',X2,X1);
// From Eqn. 12.3:
thetha = Density_S*zS*Value;// [s]
printf("The time for drying: %f min\n",thetha/60);
```

Scilab code Exa 12.6 Constant Rate Period

```
clear;
clc;

// Illustration 12.6
// Page: 685

printf('Illustration 12.6 - Page: 685\n\n');

// Solution

//***Data***//
Y1 = 0.01;// [kg water/kg dry air]
Gs = 1.1;// [kg dry air/square m.s]
dia = 13.5/1000;// [m]
l = 13/1000;// [m]
zS = 50/1000;// [m]
Density_S = 600;// [kg dry solid/square m.s]
```

```
18 a = 280;// [square m/cubic m]
19 //************//
20
21 // From Fig 7.5 (Pg 232)
22 Yas = 0.031;// [kg water/kg dry air]
23 Gav = Gs+(Gs*(Y1+Yas)/2);// [kg/square m.s]
24 viscosity_air = 1.9*10^(-5);// [kg/m.s]
25 Area = (2*%pi*dia^2/4)+(%pi*dia*l);// [square m]
26 dp = (Area/%pi)^0.5;// [m]
27 // From Table 3.3 (Pg 74)
28 Re = dp*Gav/viscosity_air;
29 e = 1-(dp*a/6);// [fraction voids]
30 jD = (2.06/e)*Re^(-0.575);
31 // For air water mixture:
32 Sc = 0.6;
33 // From Eqn. 12.33:
34 kY = jD*Gs/Sc^(2/3);// [kg H2O/square m.s.deltaX]
35 // From Eqn. 12.30:
36 NtG = kY*a*zS/Gs;
37 // From Eqn. 12.25:
38 Nmax = Gs*(Yas-Y1);// [kg/square m.s]
39 // From Eqn. 12.31:
40 N = Nmax*(1-exp(-NtG));// [kg water evaporated/
      square m.s]
41 Y2 = (Yas-Y1)*(N/Nmax)+Y1;// [kg water/kg dry air]
42 // From Fig 7.5 (Pg 232)
43 Tempas = 33;// [OC]
44 // From eqn. 12.2:
45 Rate = N/(Density_S*zS);// [kg H2O/(kg dry solid).s]
46 printf("Humidity of the exit air: %f kg water/kg dry
       air\n",Y2);
47 printf("Temparature of exit air: %d OC\n",Tempas);
48 printf("Rate of Drying: %e kg H2O/(kg dry solid).s\n
      ",Rate);
```

Scilab code Exa 12.7 Material And Enthalpy Balances

```
clear;
clc;

// Illustration 12.7
// Page: 700

printf('Illustration 12.7 - Page: 700\n\n');

// Solution

//***Data***//
x1 = 3.5;// [percent moisture]
x2 = 0.2;// [percent moisture]
dia = 1.2;// [m]
l = 6.7;// [m]
Rate_prod = 900;// [kg/h]
y2 = 0.5;// [Humidity]
TempG2 = 90;// [OC]
TempG1 = 32;// [OC]
TempS1 = 25;// [OC]
TempS2 = 60;// [OC]
//***********//

X1 = x1/(100-x1);// [kg H2O/kg dry solid]
X2 = x2/(100-x2);// [kg H2O/kg dry solid]
Ss = Rate_prod*(1-X2);// [kg dry solid/h]
Rate_drying = Ss*(X1-X2);// [kg water evaporated/h]
Y2 = (y2/(1-y2))/100;// [kg water/kg dry air]
Tempo = 0;// [Base temp,OC]
// From Table 7.1: (Pg 234)
// Enthalpy of air entering the drier:
HG2 = (1005+(1884*Y2))*(TempG2-Tempo)+(2502300*Y2);
   // [J/kg dry air]
// For the outlet air:
// HG1 = (1005+(1884*Y1))*(TempG1-Tempo)+(2502300*Y1
   ); [J/kg dry air]
```

```
// HG1 = (1005*TempG1)+((1884+TempG1)+2502300)*Y1; [J/kg dry air]
CsNH4 = 1507;// [J/kg.K]
CsH2O = 4187;// [J/kg.K]
// From Eqn. 11.45:
HS2 = CsNH4*(TempS2-Tempo)+(X2*CsH2O*(TempS2-Tempo));// [J/kg dry air]
HS1 = CsNH4*(TempS1-Tempo)+(X1*CsH2O*(TempS1-Tempo));// [J/kg dry air]
// The estimated combined natural convection and radiation heat transfer coeffecient from the drier to the surrounding:
h = 12;// [W/square m.K]
deltaTemp = ((TempG2-TempS1)+(TempG1-TempS1))/2;// [OC]
Ae = %pi*dia*l;// [square m]
Q = h*3600*Ae*deltaTemp;// [kJ/h]
// Moisture Balance, Eqn. 12.39:
// Ss*(X1-X2) = Gs(Y1-Y2)
// (Gs*Y1)-(Gs*Y2) = (Ss*(X1-X2)) ........(1)
// Enthalapy Balance, Eqn. 12.40:
// (Ss*HS1)+(Gs*HG2) = (Ss*HG2)+(Gs*HG1)+Q
// Gs*(HG2-HG1) = (Ss*HS2)+Q-(Ss*HS1)
// Gs*(HG2-((1005*TempG1)+((1884+TempG1)+2502300)*Y1)) = (Ss*HS2)+Q-(Ss*HS1)
// Gs*(HG2-(1005*TempG1))-(Gs*Y1*((1884+TempG1)+2502300)) = (Ss*HS2)+Q-(Ss*HS1) ........ (2)
// Solving Simultaneously:
a = [(HG2-(1005*TempG1)),-((1884+TempG1)+2502300);(-Y2) 1];
b = [(Ss*HS2)+Q-(Ss*HS1);(Ss*(X1-X2))];
soln = inv(a)*b;
Gs = soln(1);// [kg dry air/h]
Y1 = soln(2)/soln(1);// [kg water/kg dry air]
// From Fig. 7.5 (Pg 232)
Enthalpy_air = 56;// [kJ/kg dry air]
HeatLoad = Gs*(HG2-Enthalpy_air*1000);// [W]
printf("Air Flow Rate: %f kg/h\n",Gs);
```

```
64 printf("Moisture content of air: %f kg water/kg dry
      air \n",Y1);
65 printf("Heat Load of drier: %f kW",HeatLoad/1000);
```

Scilab code Exa 12.8 Rate of Drying for Continuous Direct Heat Driers

```
1  clear;
2  clc;
3
4  // Illustration 12.8
5  // Page: 705
6
7  printf('Illustration 12.8 - Page: 705\n\n');
8
9  // Solution
10
11 //***Data***//
12 x1 = 8;// [percent moisture]
13 x2 = 0.5;// [percent moisture]
14 Rate_prod = 0.63;// [kg/s]
15 // Drying Gas:
16 xCO2 = 0.025;// [mole fraction]
17 xO2 = 0.147;// [mole fraction]
18 xN2 = 0.760;// [mole fraction]
19 xH2O = 0.068;// [mole fraction]
20 TempG2 = 480;// [OC]
21 Cs = 0.837;// [kJ/kg.K]
22 Temp1 = 27;// [OC]
23 Temp2 = 150;// [OC]
24 dp = 200*10^(-6);// [m]
25 Density_S = 1300;// [kg/cubic m]
26 //***********//
27
28 X1 = x1/(100-x1);// [kg water/kg dry solid]
29 X2 = x2/(100-x2);// [kg water/kg dry solid]
```

```
30 Ss = Rate_prod*(1-X2);// [kg dry solid/s]
31 Water_evap = Ss*(X1-X2);// [kg/s]
32 // Basis: 1 kmol of dry gas:
33 xDry = 1-xH2O;// [kmol]
34 XCO2 = 44*xCO2;// [kg]
35 XO2 = 32*xO2;// [kg]
36 XN2 = 28*xN2;// [kg]
37 Xdry = XCO2+XO2+XN2;// [kg]
38 cCO2 = 45.6;// [kJ/kmol.K]
39 cO2 = 29.9;// [kJ/kmol.K]
40 cN2 = 29.9;// [kJ/kmol.K]
41 cH2O = 4.187;// [kJ/kg.K]
42 Mav = Xdry/xDry;// [kg/kmol]
43 Y2 = xH2O*18.02/(xDry*Mav);// [kg water/kg dry gas]
44 cav = ((xCO2*cCO2)+(xO2*cO2)+(xN2*cN2))/(xDry*Mav);
      // [kJ/kmol.K]
45 // Assume:
46 TempG1 = 120;// [OC]
47 cDry = 1.005;// [kJ/kmol.K]
48 Tempo = 0;// [Base Temp,OC]
49 // By Eqn. 7.13:
50 HG2 = (cav+(1.97*Y2))*(TempG2-Tempo)+(2502.3*Y2);//
      [kJ/kg dry air]
51 // For the outlet air:
52 // HG1 = (1.005+(1.884*Y1))*(TempG1-Tempo)+(2502.3*
      Y1); [kJ/kg dry air]
53 // HG1 = (1.005*TempG1)+((1.884+TempG1)+2502.3)*Y1;
      [kJ/kg dry air]
54 // By Eqn. 11.45:
55 HS1 = (Cs*(Temp1-Tempo))+(cH2O*X1*(Temp1-Tempo));//
      [kJ/kg dry air]
56 HS2 = (Cs*(Temp2-Tempo))+(cH2O*X2*(Temp2-Tempo));//
      [kJ/kg dry air]
57 // Q = 0.15*HG2*Gs; [kJ/s]
58 // Moisture Balance, Eqn. 12.39:
59 // Ss*(X1-X2) = Gs(Y1-Y2)
60 // (Gs*Y1)-(Gs*Y2) = (Ss*(X1-X2)) ........(1)
61 // Enthalapy Balance, Eqn. 12.40:
```

```
// (Ss*HS1)+(Gs*HG2) = (Ss*HG2)+(Gs*HG1)+Q
// Gs*(HG2-HG1) = (Ss*HS2)+(0.15*HG2*Gs)-(Ss*HS1)
// Gs*(HG2-(0.15*HG2) -((1.005*TempG1)+((1.884+TempG1
    )+2502.3)*Y1)) = (Ss*HS2)+Q-(Ss*HS1)
// Gs*(HG2-(0.15*HG2) -(1.005*TempG1))-(Gs*Y1
    *((1.884+TempG1)+2502.3)) = (Ss*HS2)+Q-(Ss*HS1)
    ..... (2)
a = [(HG2-(0.15*HG2)-(1.005*TempG1)),-((1.884+TempG1
    )+2502.3);(-Y2) 1];
b = [(Ss*HS2)-(Ss*HS1);(Ss*(X1-X2))];
soln = inv(a)*b;
Gs = soln(1);// [kg dry air/s]
Y1 = soln(2)/soln(1);// [kg water/kg dry gas]
HG1 = (1.005+(1.884*Y1))*(TempG1-Tempo)+(2502.3*Y1);
    // [kJ/kg dry air]
Q = 0.15*HG2*Gs;// [kJ/s]
// Assuming the sychrometric ratio of the gas as
    same as that of air:
// For Zone II:
Tempw = 65;// [OC]
Temp_A = 68;// [OC]
// At point A, Fig. 12.28 (Pg 702)
Enthalpy_A = Cs*(Temp_A-Tempo)+(X1*cH2O*(Temp_A-
    Tempo));// [kJ/kg dry air]
// At point B, Fig. 12.28 (Pg 702)
Temp_B = Temp_A;// [OC]
Enthalpy_B = Cs*(Temp_B-Tempo)+(X2*cH2O*(Temp_B-
    Tempo));// [kJ/kg dry air]

// Assuming that the heat losses in the three zones
    are propotional to the number of transfer units
    in each zone and to the average temp. difference
    between the gas and the surrounding air.
// Fractional heat loss in each Zone:
fr1 = 0.14;
fr2 = 0.65;
fr3 = 0.20;
// Calculations for zone III:
```

```
89  Cs3 = cav+(1.97*Y2);// [kJ/(kg dry gas).K]
90  // Heat balance:
91  deff('[y]=f1(TempGD)','y=(Gs*Cs3*(TempG2-TempGD))-(
       Ss*(HS2-Enthalpy_B)+(fr3*Q))');
92  TempGD = fsolve(7,f1);// [OC]
93  delta_TempG = Ss*(HS2-Enthalpy_B)/(Gs*Cs3);// [OC]
94  delta_TempM = ((TempG2-Temp2)+(TempGD-Temp_A))/2;//
       [OC]
95  NtoG3 = delta_TempG/delta_TempM;
96
97  // Calculations for zone I:
98  Cs1 = 1.005+(1.884*Y1);// [kJ/(kg dry gas).K]
99  // Heat balance:
100 deff('[y]=f2(TempGC)','y=(Gs*Cs1*(TempGC-TempG1))-(
       Ss*(Enthalpy_A-HS1)+(fr1*Q))');
101 TempGC = fsolve(7,f2);// [OC]
102 delta_TempG = Ss*(Enthalpy_A-HS1)/(Gs*Cs1);// [OC]
103 delta_TempM = ((TempGC-Temp_A)+(TempG1-Temp1))/2;//
       [OC]
104 NtoG1 = delta_TempG/delta_TempM;
105
106 // Calculations for zone II:
107 Cs2 = (cav+Cs1)/2;// [kJ/(kg dry gas).K]
108 // Heat balance:
109 True_deltaTemp = TempGD-TempGC;// [OC]
110 delta_Temp = fr2*Q/(Cs1*Gs);// [Change in temp
       resulting from heat loss,OC]
111 delta_TempG = True_deltaTemp-delta_Temp;// [OC]
112 delta_TempM = ((TempGD-Temp_A)-(TempGC-Temp_A))/log
       ((TempGD-Temp_A)/(TempGC-Temp_A));// [OC]
113 NtoG2 = delta_TempG/delta_TempM;
114
115 NtoG = NtoG1+NtoG2+NtoG3;
116
117 // Standard diameters are availaible at 1, 1.2 & 1.4
       m.
118 Td = 1.2;// [m]
119 Area = %pi*Td^2/4;// [square m]
```

```
Gs = Gs/Area;// [kg/square m.s]
Ss = Ss/Area;// [kg/square m.s]
Gav = Gs*(1+(Y1+Y2)/2);// [kg/square m.s]
// From Eqn. 12.47:
Ua = 237*Gav^0.417/Td;// [W/square m.K]
HtoG = Gs*Cs2*1000/Ua;// [m]
Z = NtoG*HtoG;// [m]
// Assume:
v = 0.35;// [m/s]
N = v/(%pi*Td);// [1/s]
// From Eqn. 12.37:
K = 0.6085/(Density_S*dp^(1/2));
// Take:
phi_D = 0.05;
// From Eqn. 12.35:
phi_DO = phi_D-(K*Gav);
// From Eqn. 12.35:
s = 0.3344*Ss/(phi_DO*Density_S*N^0.9*Td);// [m/s]
printf("Height of the drier: %f m\n",Z);
printf("Drier Slope: %f m/m \n",s);
```

Scilab code Exa 12.9 Drying at low temperature

```
clear;
clc;

// Illustration 12.9
// Page: 709

printf('Illustration 12.9 - Page: 709\n\n');

// Solution

//***Data***//
x1 = 0.46;// [fraction moisture]
```

```
x2 = 0.085;// [fraction moisture]
Y1 = 0.08;// [kg water/kg dry solid]
Y2 = 0.03;// [kg water/kg dry solid]
G = 1.36;// [kg/square m.s]
//**********//

X1 = x1/(1-x1);// [kg water/kg dry solid]
X2 = x2/(1-x2);// [kg water/kg dry solid]
// By water balance:
SsByGs = (Y1-Y2)/(X1-X2);// [kg dry solid/kg air]
// Since the initial moisture content of the rayon
   is less than the critical, drying takes place
   entirely within zone III.
// Comparing with Eqn. 12.22:
// (kY*A/(Ss(Xc-X*)))=0.0137*G^1.47
// thetha=integrate('(1/(0.0137*G^1.47))*(1/((X-
   X_star)*(Yw-Y)))','X',X2,X1) // [s]
X = [X1 0.80 0.60 0.40 0.20 X2];// [kg water/kg dry
   solid]
Y = zeros(6);
for i = 1:6
    // From Eqn. 12.54:
    Y(i) = Y2+((X(i)-X2)*SsByGs);// [kg water/kg dry
       gas]
end
// From Fig. 7.5 (Pg 232):
Yw = [0.0950 0.0920 0.0790 0.0680 0.0550 0.0490];//
   [kg water/kg dry gas]
X_star = zeros(6);
Val = zeros(6);
P = 51780;// [vapour pressure, kN/square m]
for i = 1:6
    // From Eqn 7.8:
    deff('[y]=f(p)','y=Y(i)-((p/(101330-p))*(18/29))
       ');
    p = fsolve(7,f);// [kN/square m]
    RH(i) = (p/P)*100;
    X_star(i) = (RH(i)/4)/(100-(RH(i)/4));// [kg
```

```
        water/kg dry solid]
44      Val(i) = 1/((X(i)-X_star(i))*(Yw(i)-Y(i)));
45  end
46  scf(41);
47  plot(X,Val);
48  xgrid();
49  xlabel("X kg water/kg dry solid");
50  ylabel("1/((X-X*)*(Yw-Y))");
51  title("Graphical Integration");
52  // Area Under the curve:
53  Area = 151.6;
54  // From Eqn. 12.59:
55  thetha = Area/(0.0137*G^1.47);
56  printf("Time required for drying: %f h\n",thetha
    /3600);
```

Chapter 13

Leaching

Scilab code Exa 13.1 Unsteady State Operation

```
1  clear;
2  clc;
3
4  // Illustration 13.1
5  // Page: 722
6
7  printf('Illustration 13.1 - Page: 722\n\n');
8
9  // Solution
10
11 //***Data***//
12 Density_L = 1137;// [kg/cubic m]
13 Density_S = 960;// [kg/cubic m]
14 Density_p = 1762;// [kg/cubic m]
15 A_prime = 16.4;// [square m/kg]
16 g = 9.81;// [square m/s]
17 sigma = 0.066;// [N/m]
18 Z = 3;// [m]
19 dia = 1;// [m]
20 //**********//
21
```

```
e = 1-(Density_S/Density_p);// [fraction void]
ap = A_prime*Density_S;// [square m/cubic m]
// By Eqn. 6.67:
dp = 6*(1-e)/ap;// [m]
// By Eqn. 13.6:
K = dp^2*e^3*g/(150*(1-e)^2);// [cubic m/s]
check = K*Density_L*g/(g*sigma);
if check<0.02
    // By Eqn. 13.3:
    So = 0.075;
else
    // By Eqn. 13.4:
    So = 0.0018/(check)
end
// By Eqn. 13.2:
ZD = (0.275/g)/((K/g)^0.5*(Density_L/sigma));// [m]
// By Eqn. 13.1:
Sav = ((Z-ZD)*So/Z)+(ZD/Z);
// VolRatio=Vol liquid retained/Vol bed.
VolRatio = Sav*e;
printf("Vol liquid retained/Vol bed : %f cubic m/
   cubic m\n",VolRatio);
Mass = VolRatio*%pi*dia^2*Z*Density_L/4;// [kg]
// Mass ratio=Mass Liquid/Mass dry solid
MassRatio = VolRatio*Density_L/(Density_S);
printf("Mass liquid/Mass dry solid: %f kg/kg\n",
   MassRatio);
```

Scilab code Exa 13.2 Multistage Crosscurrent Leaching

```
clear;
clc;

// Illustration 13.2
// Page: 749
```

```
6
7  printf('Illustration 13.2 - Page: 749\n\n');
8
9  // Solution
10
11 //***Data***//
12 // Eqb=[x(Wt fraction NaOH in clear solution) N(kg
      CaCO3/kg soln in settled sludge) y*(wt fraction
      NaOH in soln of settled sludge)]
13 // a=H2O b=CaCO3 c=NaOH
14 Eqb = [0.090 0.495 0.0917;0.0700 0.525 0.0762;0.0473
       0.568 0.0608;0.0330 0.600 0.0452;0.0208 0.620
      0.0295;0.01187 0.650 0.0204;0.00710 0.659
      0.01435;0.00450 0.666 0.01015];
15 deff('[y]=f80(x)','y=0');
16 x = 0:0.01:0.12;
17 Mass_c = 0.1;// [kg]
18 Mass_b = 0.125;// [kg]
19 Mass_a = 0.9;// [kg]
20 //**************//
21
22 scf(42);
23 plot(x,f80,Eqb(:,3),Eqb(:,2));
24 xgrid();
25 xlabel("x,y Wt. fraction of NaOH in loquid");
26 ylabel("N kg CaCO3 / kg solution");
27 legend("N Vs x","N Vs Y");
28 title("Equilibrium Plot")
29 // Basis: 1 kg soln in original mixture.
30 // As in Fig. 13.27 (Pg 750)
31 // The original mixture corresponds to M1:
32 NM1 = 0.125;// [kg CaCO3/kg soln]
33 yM1 = 0.1;// [kg NaOH/kg solution]
34 // The tie line through M1 is drawn. At point E1
      representing the settled sludge:
35 N1 = 0.47;// [kg CaCO3/kg soln]
36 y1 = 0.100;// [kg NaOH/kg solution]
37 E1 = Mass_b/N1;// [kg soln. in sludge]
```

```
38 Ro = 1-E1;// [kg clear soln drawn]
39
40 // Stage 2:
41 xo = 0;// [kg NaOH/kg soln]
42 // By Eqn. 13.11:
43 M2 = E1+Ro;// [kg liquid]
44 // By Eqn. 13.12:
45 NM2 = Mass_b/(E1+Ro);// [kg CaCO3/kg soln]
46 // M2 is located on line RoE1. At this value of N,
     and the tie line through M2 is drawn. At E2:
47 N2 = 0.62;// [kg CaCO3/kg soln]
48 y2 = 0.035;// [kg NaOH/kg solution]
49 E2 = Mass_b/N2;// [kg soln. in sludge]
50 Ro = 1-E2;// [kg clear soln drawn]
51
52 // Stage 3:
53 xo = 0;// [kg NaOH/kg soln]
54 // By Eqn. 13.11:
55 M3 = E2+Ro;// [kg liquid]
56 // By Eqn. 13.12:
57 NM3 = Mass_b/M3;// [kg CaCO3/kg soln]
58 // Tie line E3R3 is located through M3.At E3:
59 N3 = 0.662;// [kg CaCO3/kg soln]
60 y3 = 0.012;// [kg NaOH/kg solution]
61 // By Eqn. 13.8:
62 E3 = Mass_b/N3;// [kg soln. in sludge]
63 printf("The fraction of original NaOH in the slurry:
     %f \n",E3*y3/Mass_c);
```

Scilab code Exa 13.3 Multistage Countercurrent Leaching

```
1 clear;
2 clc;
3
4 // Illustration 13.3
```

```
5 // Page: 754
6
7 printf('Illustration 13.3 - Page: 754\n\n');
8
9 // Solution (a)
10
11 //***Data***//
12 // a=H2O b=CaCO3 c=NaOH
13 mass_c = 400;// [kg/h]
14 x1 = 0.1;// [wt fraction NaOH in overflow]
15 //**********//
16
17 Mb = 100;// [kg/kmol]
18 Mc = 40;// [kg/kmol]
19 rate_c = mass_c/Mc;// [kmol/h]
20 rate_b = rate_c/2;// [kmol/h]
21 mass_b = rate_b*Mb;// [kg/h]
22 // After trial calculations:
23 y3 = 0.01;// [kg NaOH/kg solution]
24 N3 = 0.666;// [kg CaCO3/kg solution]
25 E3 = mass_b/N3;// [kg/h]
26 lost_c = E3*y3;// [kg/h]
27 sludge_a = E3-lost_c;// [kg/h]
28 overflow_c = mass_c-lost_c;// [kg NaOH/kg solution]
29 R1 = overflow_c/x1;// [kg overflow/h]
30 R1_a = R1-overflow_c;// [kg/h]
31 RNpPlus1 = R1_a+sludge_a;// [kg/h]
32 // For purpose of calculation, it may be imagined
   that agitators are not present in the flowsheet
   and the first thickner is fed with the dry
   mixture of the reaction products, CaCO3 and NaOH,
    together with overflow from the second thickner.
33 F = 400;// [kg NaOH/h]
34 NF = mass_b/F;// [kg CaCO3/kg NaOH]
35 yF = 1;// [wt fraction NaOH in dry solid, CaCO3 free
    basis]
36 // Points R1, E3, RNpPlus1 and F are plotted as in
   Fig 13.30 (Pg 755) and locate the point deltaR at
```

```
     the intersection of lines FR1 and E3RNpPlus1
    extended. The coordinates of point deltaR are
    NdeltaR=-0.1419, ydeltaR=-0.00213. Further
    computation must be done on enlarged section of
    the equilibrium diagram (Fig 13.31 (Pg 755)).
    Point deltaR is plotted and the stages stepped
    off in a usual manner. The construction are
    projected on the xy diagram. Three stages produce
     a value:     y3=0.001
printf("The NaOH lost in sludge: %f%%\n",(lost_c/
   mass_c)*100);
printf("\n");

// Solution (b)
//*** Data***//
lost_c = 0.001*mass_c;// [kg/h]
//***********//

NNp_by_yNp = mass_b/lost_c;// [kg CaCO3/kg NaOH in
   final sludge]
// In order to determine the liquid content of the
   final sludge:
// Eqb=[N y_star]
Eqb = [0.659  0.01435;0.666 0.01015;0.677
   0.002;0.679 0.001;0.680 0.0005];
N_by_ystar = zeros(5);
for i = 1:5
    N_by_ystar(i) = Eqb(i,1)/(Eqb(i,2));
end
scf(43);
plot(Eqb(:,1),Eqb(:,2));
xgrid();
xlabel("x Wt fraction of NaOH");
ylabel("N kg CaCO3 / kg solution");
title("Equilibrium plot")
// By Interpolation, for N_by_ystar=NNp_by_yNp:
NNp = interpln([(N_by_ystar)';Eqb(:,1)'],NNp_by_yNp)
   ;// [kg CaCO3/kg soln]
```

```
yNp = NNp/NNp_by_yNp;// [wt fraction NaOH in the
   liquid of the final sludge]
ENp = mass_b/NNp;// [kg/h]
ENp_a = ENp-lost_c;// [kg/h]
overflow_c = mass_c-lost_c;// [kg/h]
R1 = overflow_c/0.1;// [kg/h]
R1_a = R1-overflow_c;// [kg/h]
RNpPlus1 = R1_a+sludge_a;// [kg/h]
// On the operating diagram (Fig 13.32 (Pg 757))
   point deltaR is located and stages were
   constructed.
// Beyond the fourth stage, the ratio of the
   overflow to the liquid in the sludge become
   substantially constant.
R_by_E = RNpPlus1/ENp;
// This is the initial slope of the operating line
   on the lower part of the figure.
// From Illustration 13.2:
m = 0.01015/0.00450;
Value1 = R_by_E/m;
xNpPlus1 = 0;// [kg NaOH/kg solution]
y4 = 0.007;// [wt fraction NaOH in the liquid]
Value2 = (yNp-(m*xNpPlus1))/(y4-(m*xNpPlus1));
// From Fig 5.16: (Pg 129):
// An Additional 2.3 stages beyond 4 are computed
   graphically are required.
// An additional two stage will make yNp/y4=0.099:
yNp = 0.099*y4;// [wt fraction NaOH in the liquid]
printf("%f kg NaOH was lost if 6 thickners were used
   \n",yNp*ENp);
// An additional three stage will make yNp/y4
   =0.0365:
yNp = 0.0365*y4;// [wt fraction NaOH in the liquid]
printf("%f kg NaOH was lost if 7 thickners were used
   \n",yNp*ENp);
```

Scilab code Exa 13.4 Multistage Countercurrent Leaching

```
clear;
clc;

// Illustration 13.4
// Page: 758

printf('Illustration 13.4 - Page: 758\n\n');

// Solution

//***Data***//
// a:oil b:soyabean c:hexane
// Data=[100y*(Wt % oil in soln) 1/N(kg soln
   retained/kg insoluble solid)]
Data = [0 0.58;20 0.66;30 0.70];
// Soyabean feed:
percent_b = 20;// [soluble]
yF = 1;// [mass fraction oil,solid free basis]
// Solvent:
RNpPlus1 = 1;// [hexane,kg]
xNpPlus1 = 0;// [mass fraction oil]
// Leached Solids:
leached = 0.005;// [fraction of oil to be leached]
// Miscella:
percent_miscella = 10;// [percent of insoluble solid
   ]
//**********//

N = zeros(3);
ystar_By_N = zeros(3);
for i = 1:3
    N(i) = 1/Data(i,2);// [kg insoluble solid/kg
```

```
        soln retained]
31      ystar_By_N(i) = Data(i,1)/(100*N(i));// [kg oil/
        kg insoluble solid]
32  end
33  // Basis: 1 kg flakes introduced
34  // Soyabean feed:
35  mass_b = 1-(percent_b/100);// [insoluble,kg]
36  F = 1-mass_b;// [kg]
37  NF = mass_b/F;// [kg insoluble solid/kg oil]
38
39  // Leached Solids:
40  Ratio = leached/(1-leached);// [kg oil/kg insoluble
    solid]
41  // By interpolation:
42  Np = interpln([ystar_By_N';N'],Ratio);
43  miscella_b = (percent_miscella/100)*mass_b;// [
    Insoluble solid lost to miscella,kg]
44  leached_b = (1-(percent_miscella/100))*mass_b;// [
    Insoluble solid in miscella,kg]
45  ENp = leached_b/Np;// [kg soln retained]
46  retained_a = Ratio*leached_b;// [oil retained,kg]
47  retained_c = ENp-retained_a;// [Hexane retained,kg]
48  yNp = retained_a/ENp;// [mass fraction of oil in
    retained liquid]
49
50  // Miscella:
51  mass_c = 1-retained_c;// [kg]
52  mass_a = F-retained_a;// [kg]
53  R1 = mass_c+mass_a;// [clear miscella,kg]
54  x1 = mass_a/R1;// [mass fraction of oil in the
    liquid]
55  NR1 = miscella_b/R1;// [kg insoluble solid/kg soln]
56
57  // The operating diagram is shown in Fig 13.33 (Pg
    759).
58  // Point R1 represents the cloudy miscella and is
    therefore is displaced from the axis of he graph
    at NR1. Point deltaR is located as usual and the
```

```
    stages determined with the N=0 axis for all the
    stages but the first.
59 printf("Between 4 and 5 stages are required\n");
```