

Scilab Textbook Companion for  
Heat Transfer (In SI Units)  
by J. P. Holman<sup>1</sup>

Created by  
Shivam Singh  
B.Tech  
Chemical Engineering  
IT BHU  
College Teacher  
Prakash Kotecha  
Cross-Checked by

August 10, 2013

<sup>1</sup>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:** Heat Transfer (In SI Units)

**Author:** J. P. Holman

**Publisher:** Tata McGraw - Hill Education, New Delhi

**Edition:** 9

**Year:** 2002

**ISBN:** 0-07-063451-3

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

**Exa** Example (Solved example)

**Eqn** Equation (Particular equation of the above book)

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

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

# Contents

List of Scilab Codes	4
1 Introduction	8
2 Steady State Conduction One Dimension	13
3 Steady State Conduction Multiple Dimension	28
4 Unsteady State Conduction	47
5 Principles of Convection	64
6 Empirical and Practical Relations for Forced Convection Heat Transfer	80
7 Natural Convection Systems	99
8 Radiation Heat Transfer	116
9 Condensation and Boiling Heat Transfer	158
10 Heat Exchangers	164
11 Mass Transfer	198

# List of Scilab Codes

Exa 1.1	conduction through copper plate . . . . .	8
Exa 1.2	convection calculation . . . . .	8
Exa 1.3	multimode heat transfer . . . . .	9
Exa 1.4	heat source and convection . . . . .	10
Exa 1.5	radiation heat transfer . . . . .	10
Exa 1.6	total heat loss by convection and radiation . . . . .	11
Exa 2.1	multilayer conduction . . . . .	13
Exa 2.2	multilayer cylindrical system . . . . .	14
Exa 2.3	heat transfer through a composite wall . . . . .	15
Exa 2.4	overall heat transfer coefficient for a tube . . . . .	17
Exa 2.5	critical insulation thickness . . . . .	18
Exa 2.6	heat source with convection . . . . .	19
Exa 2.7	influence of thermal conductivity on fin temperature profiles . . . . .	20
Exa 2.8	straight aluminium fin . . . . .	23
Exa 2.9	circumferential aluminium fin . . . . .	24
Exa 2.10	rod with heat sources . . . . .	25
Exa 2.11	influence of contact conductance on heat transfer . . . . .	26
Exa 3.1	buried pipe . . . . .	28
Exa 3.2	cubical furnace . . . . .	29
Exa 3.3	buried disk . . . . .	30
Exa 3.4	buried parallel disk . . . . .	30
Exa 3.5	Nine node problem . . . . .	31
Exa 3.6	Gauss seidal calculation . . . . .	33
Exa 3.7	numerical formulation with heat generation . . . . .	35
Exa 3.8	heat generation with non uniform nodal elements . . . . .	38
Exa 3.11	use of variable mesh size . . . . .	42
Exa 3.12	Three dimensional numerical formulation . . . . .	44

Exa 4.1	steel ball cooling in air . . . . .	47
Exa 4.2	Semi infinite solid with sudden change in surface conditions . . . . .	48
Exa 4.3	pulsed energy at surface of semi infinite solid . . . . .	49
Exa 4.4	heat removal from semi infinite solid . . . . .	50
Exa 4.5	sudden exposure of semi infinite solid slab to convection . . . . .	51
Exa 4.6	aluminium plate suddenly exposed to convection . . . . .	53
Exa 4.7	long cylinder suddenly exposed to convection . . . . .	54
Exa 4.8	semi infinite cylinder suddenly exposed to convection . . . . .	56
Exa 4.9	finite length cylinder suddenly exposed to convection . . . . .	57
Exa 4.10	heat loss for finite length cylinder . . . . .	59
Exa 4.12	implicit formulation . . . . .	61
Exa 5.1	water flow in a diffuser . . . . .	64
Exa 5.2	isentropic expansion of air . . . . .	65
Exa 5.3	mass flow and boundary layer thickness . . . . .	66
Exa 5.4	isothermal flat plate heated over entire length . . . . .	67
Exa 5.5	flat plate with constant heat flux . . . . .	68
Exa 5.6	plate with unheated starting length . . . . .	70
Exa 5.7	oil flow over heated flat plate . . . . .	71
Exa 5.8	drag force on a flat plate . . . . .	72
Exa 5.9	turbulent heat transfer from isothermal flat plate . . . . .	74
Exa 5.10	turbulent boundary layer thickness . . . . .	75
Exa 5.11	high speed heat transfer for a flat plate . . . . .	75
Exa 6.1	turbulent heat transfer in a tube . . . . .	80
Exa 6.2	heating of water in laminar tube flow . . . . .	81
Exa 6.3	heating of air in laminar tube flow for constant heat flux . . . . .	83
Exa 6.4	heating of air with isothermal tube wall . . . . .	85
Exa 6.5	heat transfer in a rough tube . . . . .	86
Exa 6.6	turbulent heat transfer in a short tube . . . . .	88
Exa 6.7	airflow across isothermal cylinder . . . . .	89
Exa 6.8	heat transfer from electrically heated . . . . .	90
Exa 6.9	heat transfer from sphere . . . . .	92
Exa 6.10	heating of air with in line tube bank . . . . .	93
Exa 6.11	alternate calculation method . . . . .	95
Exa 6.12	heating of liquid bismuth in tube . . . . .	97
Exa 7.1	constant heat flux from vertical plate . . . . .	99
Exa 7.2	heat transfer from isothermal vertical plate . . . . .	101
Exa 7.3	heat transfer from horizontal tube in water . . . . .	102

Exa 7.4	heat transfer from fine wire in air . . . . .	103
Exa 7.5	heated horizontal pipe in air . . . . .	104
Exa 7.6	cube cooling in air . . . . .	105
Exa 7.7	calculation with simplified relations . . . . .	106
Exa 7.8	heat transfer across vertical air gap . . . . .	108
Exa 7.9	heat transfer across horizontal air gap . . . . .	109
Exa 7.10	heat transfer across water layer . . . . .	110
Exa 7.11	reduction of convection in ar gap . . . . .	111
Exa 7.12	heat transfer across evacuated space . . . . .	113
Exa 7.13	combined free and forced convection with air . . . . .	114
Exa 8.1	transmission and absorption in a gas plate . . . . .	116
Exa 8.2	heat transfer between black surfaces . . . . .	117
Exa 8.3	shape factor algebra for open ends of cylinder . . . . .	118
Exa 8.4	shape factor algebra for truncated cone . . . . .	119
Exa 8.5	shape factor algebra for cylindrical reflector . . . . .	121
Exa 8.6	hot plates enclosed by a room . . . . .	122
Exa 8.7	surface in radiant balance . . . . .	124
Exa 8.8	open hemisphere in large room . . . . .	126
Exa 8.9	effective emissivity of finned surface . . . . .	128
Exa 8.10	heat transfer reduction with parallel plate shield . . . . .	129
Exa 8.11	open cylindrical shield in large room . . . . .	130
Exa 8.12	network for gas radiation between parallel plates . . . . .	132
Exa 8.13	cavity with transparent cover . . . . .	134
Exa 8.14	Transmitting and reflecting system for furnace opening . . . . .	134
Exa 8.15	numerical solution for enclosure . . . . .	137
Exa 8.16	numerical solution for parallel plates . . . . .	139
Exa 8.17	radiation from a hole with variable radiosity . . . . .	142
Exa 8.18	heater with constant heat flux and surrounding shields . . . . .	146
Exa 8.19	numerical solution for combined convection and radiation non linear system . . . . .	151
Exa 8.20	solar environment equilibrium temperatures . . . . .	154
Exa 8.21	influence of convection on solar equilibrium temperature . . . . .	155
Exa 8.23	temperature measurement error caused by radiation . . . . .	156
Exa 9.1	condensation on vertical plate . . . . .	158
Exa 9.2	condensation on tube tank . . . . .	159
Exa 9.3	boiling on brass plate . . . . .	160
Exa 9.4	Flow boiling . . . . .	161
Exa 9.5	water boiling in a pan . . . . .	162

Exa 9.6	heat flux comparisons . . . . .	163
Exa 10.1	overall heat transfer coefficient for pipe in air . . . . .	164
Exa 10.2	overall heat transfer coefficient for pipe exposed to steam	166
Exa 10.3	influence of fouling factor . . . . .	168
Exa 10.4	calculation of heat exchanger size from known temper- atures . . . . .	169
Exa 10.5	shell and tube heat exchanger . . . . .	170
Exa 10.6	design of shell and tube heat exchanger . . . . .	171
Exa 10.7	cross flow exchanger with one fluid mixed . . . . .	173
Exa 10.8	effects of off design flow rates for exchanger in previous example . . . . .	174
Exa 10.9	off design calculation using E NTU method . . . . .	176
Exa 10.10	off design calculation of exchanger in example 10 4 . . . . .	177
Exa 10.11	cross flow exchanger with both fluid unmixed . . . . .	179
Exa 10.12	comparison of single or two exchanger options . . . . .	181
Exa 10.13	shell and tube exchanger as air heater . . . . .	184
Exa 10.14	ammonia condenser . . . . .	185
Exa 10.15	crossflow exchanger as energy conservation device . . . . .	186
Exa 10.16	heat transfer coefficient in compact exchanger . . . . .	189
Exa 10.17	transient response of thermal energy storage system . . . . .	190
Exa 10.18	variable properties analysis of a duct heater . . . . .	194
Exa 11.1	diffusion coefficient for co2 . . . . .	198
Exa 11.2	diffusion coefficient for co2 . . . . .	199
Exa 11.3	Wet bulb temperature . . . . .	199
Exa 11.4	relative humidity of air stream . . . . .	200
Exa 11.5	water evaporation rate . . . . .	201



# Chapter 1

## Introduction

Scilab code Exa 1.1 conduction through copper plate

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 1.1\n\n\n");
4 // conduction through copper plate
5 // illustration1.1
6 // solution
7
8 k = 370; // [W/m] at 250 degree celsius
9 dt = 100-400; //[degree celsius] temperature
   difference
10 dx = 3*10^(-2); //[m] thickness of plate
11 //calculating heat transfer per unit area from
   fourier 's law
12 q = -k*dt/dx; //[MW/square meter]
13 printf("rate of heat transfer per unit area is %f MW
   /square meter",q/1000000);
```

---

Scilab code Exa 1.2 convection calculation

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 1.2\n\n\n");
4 // convection calculation
5 // illustration1.2
6 // solution
7
8 Twall = 250; //[degree celsius] wall temperature
9 Tair = 20; //[degree celsius] air temperature
10 h = 25; //[W/square meter] heat transfer coefficient
11 l = 75*10^(-2); //[m] length of plate
12 b = 50*10^(-2); //[m] width of plate
13 area = l*b; //[square meter] area of plate
14 dt = 250-20; //[degree celsius]
15 // from newton's law of cooling
16 q = h*area*dt; // [W]
17 printf("rate of heat transfer is %f kW",q/1000);

```

---

### Scilab code Exa 1.3 multimode heat transfer

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 1.3\n\n\n");
4 // multimode heat transfer
5 // illustration1.3
6 // solution
7
8 Qconv = 2156; // [W] from previous problem
9 Qrad = 300; // [W] given
10 dx = 0.02; // [m] plate thickness
11 l = 0.75; // [m] length of plate
12 w = 0.5; // [m] width of plate
13 k = 43; // [W/m] from table 1.1
14 area = l*w; //[square meter] area of plate
15 Qcond = Qconv+Qrad; // [W]

```

```

16 dt = Qcond*dx/(k*area);// [degree celsius]
    temperature difference
17 Ti = 250+dt;// inside temperature
18 printf("the inside plate temperature is therefore %f
    degree celsius",Ti);

```

---

#### Scilab code Exa 1.4 heat source and convection

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 1.4\n\n\n");
4 // heat source and convection
5 // illustration1.4
6 // solution
7
8 d = 1*10^(-3);//[m] diameter of wire
9 l = 10*10^(-2);//[m] length of wire
10 Sarea = 22*d*l/7;//[square meter] surface area of
    wire
11 h = 5000;//[W/square meter] heat transfer
    coefficient
12 Twall = 114;// [degree celsius]
13 Twater = 100;// [degree celsius]
14 //total convection loss is given by equation(1-8)
15 Q = h*Sarea*(Twall-Twater);// [W]
16 printf("heat transfer is therefore %f W",Q);
17 printf(" this is equal to the electric power which
    must be applied");

```

---

#### Scilab code Exa 1.5 radiation heat transfer

```

1 clear;
2 clc;

```

```

3 printf("\t\t\tExample Number 1.5\n\n\n");
4 // radiation heat transfer
5 // illustration1.5
6 // solution
7
8 sigma = 5.699*10^(-8); // [W/square meter*k^(4)]
   universal constant
9 T1 = 273.15+800; // [k] first plate temperature
10 T2 = 273.15+300; // [k] second plate temperature
11 //equation(1-10) may be employed for this problem
12 Q = sigma*((T1^(4))-(T2^(4))); // [W/square meter]
13 printf("heat transfer per unit area is %f kW/square
   meter",Q/1000);

```

---

**Scilab code Exa 1.6** total heat loss by convection and radiation

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 1.6\n\n\n");
4 // total heat loss by convection and radiation
5 // illustration1.6
6 // solution
7
8 d = 0.05; // [m] diameter of pipe
9 Twall = 50; // [degree celsius]
10 Tair = 20; // [degree celsius]
11 emi = 0.8; // emissivity
12 h = 6.5; // [W/square meter] heat transfer coefficient
   for free convection
13 Q1 = h*22*d*(Twall-Tair)/7; // [W/m] convection loss
   per unit length
14 sigma = 5.669*10^(-8); // [W/square meter*k^(4)]
   universal constant
15 T1 = 273.15+Twall; // [k]
16 T2 = 273.15+Tair; // [k]

```

```
17 Q2 = emi*22*d*sigma*((T1^(4))-(T2^(4)))/7; // [W/m]
    heat loss due to radiation per unit length
18 Qtotal = Q1+Q2; // [W/m] total heat loss per unit
    length
19 printf("total heat loss is therefore %f W/m",Qtotal)
    ;
```

---

## Chapter 2

# Steady State Conduction One Dimension

Scilab code Exa 2.1 multilayer conduction

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 2.1\n\n\n");
4 // multilayer conduction
5 // illustration2.1
6 // solution
7
8 dx1 = 0.1; // [m] thickness of layer of common brick
9 k1 = 0.7; // [W/m degree celsius] heat transfer
   coefficient of common brick
10 dx2 = 0.0375; // [m] thickness of layer of gypsum
   plaster
11 k2 = 0.48; // [W/m degree celsius] heat transfer
   coefficient gypsum plaster
12 Rb = dx1/k1; // [square meter degree celsius /W]
   thermal resistance of brick
13 Rp = dx2/k2; // [square meter degree celsius /W]
   thermal resistance of gypsum plaster
14 R = Rb+Rp; // [square meter degree celsius /W]
```

```

        thermal resistance without insulation
15 R1 = R/0.2; // [square meter degree celsius /W] with
    insulation
16 // heat loss with the rock-wool insulation is 20
    percent
17 Rrw = R1-R; // [square meter degree celsius /W]
18 k3 = 0.065; // [W/m degree celsius] heat transfer
    coefficient
19 dx3 = Rrw*k3; // [m]
20 printf("length of thickness is %f cm added to reduce
    the heat loss(or gain) through wall by 80
    percent", dx3*100);

```

---

#### Scilab code Exa 2.2 multilayer cylindrical system

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 2.2\n\n\n");
4 // multilayer cylindrical system
5 // illustration2.2
6 // solution
7
8 ID = 0.02; // [m] inner diameter of steel
9 OD = 0.04; // [m] outer diameter of steel
10 t = 0.03; // [m] thickness of asbestos insulation
11 // system is like three concentric cylinders
12 T1 = 600; // [degree celsius] inside wall temperature
13 T2 = 100; // [degree celsius] outside insulation
    temperature
14 Ks = 19; // [W/m degree celsius] heat transfer
    coefficient of steel
15 Ka = 0.2; // [W/m degree celsius] heat transfer
    coefficient of asbestos
16 // heat flow is given by per unit length
17 Q_1 = ((2*22*(T1-T2)/7)/((log(OD/ID)/Ks)+(log(0.1/OD

```

```

    )/Ka)); // [W/m]
18 // above calculated heat flow is used to calculate
    the interface temperature
19 // between the outside wall and the insulation
20 Ta = Q_1*(log(0.1/OD)/(2*3.14*Ka))+T2; // [degree
    celsius] Ta is interface temperature
21 printf("heat flow is given by %f W/m",Q_1);
22 printf("\n the interface temperature is %f degree
    celsius ",Ta);

```

---

### Scilab code Exa 2.3 heat transfer through a composite wall

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 2.3\n\n\n");
4 // heat transfer through a composite wall
5 // illustration2.3
6 // solution
7
8 // 1. heat transfer through studs for unit depth
9 l = 0.0413; // [m] length of wood studs
10 b = 1.0; // [m] unit depth
11 A = l*b; // [square meter] area of studs for unit
    depth
12 hi = 7.5; // [W/square meter per degree celsius]
    convectonal heat transfer coefficient
13 ho = 15; // [W/square meter per degree celsius]
    convectonal heat transfer coefficient
14 Kb = 0.69; // [W/m per degree celsius] heat transfer
    coefficient of brick
15 Kgi = 0.96; // [W/m per degree celsius] heat transfer
    coefficient of gypsum inner sheath
16 Ki = 0.04; // [W/m per degree celsius] heat transfer
    coefficient of insulation
17 Kws = 0.1; // [W/m per degree celsius] heat transfer

```



```

    coefficient of wood stud
18 Kgo = 0.48; // [W/m per degree celsius] heat transfer
    coefficient of gypsum outer sheath
19 Rair = 1/(ho*A); // [degree celsius /W] convection
    resistance outside of brick
20 dx_b = 0.08; // [m] thickness of brick
21 dx_os = 0.019; // [m] thickness of outer sheet
22 dx_ws = 0.0921; // [m] thickness of wood stud
23 dx_is = 0.019; // [m] thickness of inner sheet
24 Rb = dx_b/(Kb*A); // [degree celsius /W] conduction
    resistance in brick
25 Ros = dx_os/(Kgi*A); // [degree celsius /W]
    conduction resistance through outer sheet
26 Rws = dx_ws/(Kws*A); // [degree celsius /W]
    conduction resistance through wood stud
27 Ris = dx_is/(Kgo*A); // [degree celsius /W]
    conduction resistance through inner sheet
28 Ri = 1/(hi*A); // [degree celsius /W] convection
    resistance on inside
29 Rt = Rair+Rb+Ros+Rws+Ris+Ri; // [degree celsius /W]
    total thermal resistance through the wood stud
    section
30 printf("total thermal resistance through the wood
    stud section is %f degree celsius /W",Rt);
31 // 2. heat transfer through insulation section
32 A1 = 0.406-A; // [square meter] area of insulation
    section for unit depth
33 dx_ins = 0.0921; // [m] thickness of insulation
34 Rins = dx_ins/(Ki*A1); // [degree celsius /W]
    conduction resistance through insulation section
35 // five of the materials are same but resistance
    involve different area
36 // i.e. (40.6-4.13) cm instead of 4.13 cm
37 // so that each of the previous must be multiplied
    by a factor of (4.13/(40.6-4.13)) = 0.113
38 Rt_ins = (Rair+Rb+Ros+Ris+Ri)*0.113+Rins; // [degree
    celsius /W] total resistance through insulation
    section

```

```

39 printf("\n total thermal resistance through the
    insulation section is %f degree celsius /W",
    Rt_ins);
40 R_overall = 1/((1/Rt)+(1/Rt_ins));// [degree celsius
    /W] overall resistance for the section
41 // the value is related to overall heat transfer
    coefficient by
42 // Q = U*A*dt = dt/R_overall
43 // where A is area of total section
44 A_ = 0.406;// [square meter] area of total section
45 U = 1/(R_overall*A_);// [W/square meter degree
    celsius] overall heat transfer coefficient
46 // R value is somewhat different from thermal
    resistance and is given by
47 R_value = 1/U;// [degree celsius square meter/W] R
    value of system
48 printf("\n overall heat transfer coefficient is %f W
    /square meter per degree celsius",U);
49 printf("\n R value is %f square meter/W",R_value);

```

---

#### Scilab code Exa 2.4 overall heat transfer coefficient for a tube

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 2.4\n\n\n");
4 // overall heat transfer coefficient for a tube
5 // illustration2.3
6 // solution
7
8 ID = 0.025;// [m] inner diameter of steel
9 OD = ID+2*0.0008;//[m] outer diameter of steel
10 hi = 3500;// [W/square meter per degree celsius]
    convectonal heat transfer coefficient of inside
11 ho = 7.6;// [W/square meter per degree celsius]
    convectonal heat transfer coefficient of outside

```

```

12 L = 1.0; // [m] tube length
13 Ai = %pi*ID*L; // [square meter] inside crosssectional
    area
14 Ao = %pi*OD*L; // [square meter] outside
    crosssectional area
15 k = 16; // [W/square meter per degree celsius]
    thermal conductivity of tube
16 Ri = 1/(hi*Ai); // [degree celsius /W] convection
    resistance inside tube
17 Rt = log(OD/ID)/(2*3.14*k*L); // [degree celsius /W]
    thermal resistance
18 Ro = 1/(ho*Ao); // [degree celsius /W] convection
    resistance outside tube
19 R_total = Ri+Rt+Ro; // [degree celsius /W] total
    thermal and convection resistance
20 Uo = 1/(Ao*R_total); // [W/square meter degree
    celsius] overall heat transfer coefficient
21 printf("overall heat transfer coefficient is %f W/
    square meter degree celsius",Uo);
22 Tw = 50; // [degree celsius] water temperature
23 Ta = 20; // [degree celsius] surrounding air
    temperature
24 dt = Tw-Ta; // [degree celsius] temperature
    difference
25 q = Uo*Ao*dt; // [W] heat transfer
26 printf("\n heat loss per unit length is %f W(for 1m
    length)",q);

```

---

### Scilab code Exa 2.5 critical insulation thickness

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 2.5\n\n\n");
4 // critical insulation thickness
5 // illustration2.5

```

```

6 // solution
7
8 k = 0.17; // [W/m per degree celsius] heat transfer
    coefficient of asbestos
9 Tr = 20; // [degree celsius] temperature of room air
10 h = 3; // [W/square meter per degree celsius]
    convectional heat transfer coefficient
11 Tp = 200; // [degree celsius] temperature of pipe
12 d = 0.05; // [m] diameter of pipe
13 // from equation (2-18) we calculate r_o as
14 r_o = k/h; // [m] critical radius of insulation
15 printf("critical radius of insulation for asbestos
    is %f cm ", r_o*100);
16 Ri = d/2; // [m] inside radius of insulation
17 // heat transfer is calculated from equation (2-17)
18 q_by_L = (2*3.14*(Tp-Tr))/(((log(r_o/Ri))/0.17)+(1/(
    h*r_o))); // [W/m] heat transfer per unit length
19 printf("\n heat loss when covered with critical
    radius of insulation is %f W/m", q_by_L);
20 // without insulation the convection from the outer
    surface of pipe is
21 q_by_L1 = h*2*3.14*Ri*(Tp-Tr); // [W/m] convection
    from outer surface without insulation
22 printf("\n heat loss without insulation is %f W/m",
    q_by_L1);
23 per_inc = ((q_by_L-q_by_L1)/q_by_L1)*100; //
    percentage increase in heat transfer
24 printf("\n so the addition of 3.17 of insulation
    actually increases the heat transfer by %f
    percent", per_inc);

```

---

### Scilab code Exa 2.6 heat source with convection

```

1 clear;
2 clc;

```

```

3 printf("\t\t\tExample Number 2.6\n\n\n");
4 // heat source with convection
5 // illustration2.6
6 // solution
7
8 // all the power generated in the wire must be
   dissipated by convection to the liquid
9 //  $P = i^2 R = q = h A \Delta t$ 
10 L = 100; // [cm] length of the wire
11 k = 19; // [W/m per degree celsius] heat transfer
   coefficient of steel wire
12 A = %pi*(0.15)^(2); // [square meter] crossectional
   area of wire
13 rho = 70*10^(-6); // [micro ohm cm] resistivity of
   steel
14 R = rho*L/A; // [ohm] resistance of wire
15 i = 200; // [ampere] current in the wire
16 P = i^2*R; // [W] power generated in the wire
17 Tl = 110; // [degree celsius] liquid temperature
18 d = 0.003; // [m] diameter of wire
19 l = 1; // [m] length of wire
20 Tw = (P/(4000*3.14*d*1))+110; // [degree celsius]
   wire temperature
21 // heat generated per unit volume q_dot is
   calculated as
22 //  $P = q_{dot} V = q_{dot} 3.14 r^2 l$ 
23 r = d/2; // [m] radius of wire
24 q_dot = P/(%pi*r^2*l); // [W/m^3]
25 // finally the center temperature of the wire is
   calculated from equation (2-26)
26 To = ((q_dot*(r^2))/(4*k))+Tw; // [degree celsius]
27 printf("center temperature of the wire is %f degree
   celsius",To);

```

---

Scilab code Exa 2.7 influence of thermal conductivity on fin temperature profiles

```

1  clear;
2  clc;
3  printf("\t\t\tExample Number 2.7\n\n\n");
4  // influence of thermal conductivity on fin
   temperature profiles
5  // illustration2.7
6  // solution
7
8  d = 0.02; // [m] diameter of rod
9  L = 0.1; // [m] length of rod
10 A = %pi*d^(2)/4; // [square meter] crossectional area
   of rod
11 h = 25; // [W/square meter per degree celsius]
   convectonal heat transfer coefficient
12 k_c = 385; // [W/m per degree celsius] heat transfer
   coefficient of copper
13 k_s = 17; // [W/m per degree celsius] heat transfer
   coefficient of steel
14 k_g = 0.8; // [W/m per degree celsius] heat transfer
   coefficient of glass
15 // calculating (h*P/(k*A)) and m and m*L for three
   different rod
16 P = %pi*d; // [m] perimeter of rod
17 printf("Material\t(hP/kA)\t\tm\t\tmL");
18 printf("\ncopper\t\t%f\t%f\t\t%f", (h*P/(k_c*A)), ((h*
   P/(k_c*A)))^(1/2), ((h*P/(k_c*A)))^(1/2)*L);
19 printf("\nstainless steel\t\t%f\t%f\t\t%f", (h*P/(k_s*A))
   , ((h*P/(k_s*A)))^(1/2), ((h*P/(k_s*A)))^(1/2)*L);
20 printf("\nglass\t\t\t%f\t%f\t\t%f", (h*P/(k_g*A)), ((h*P/(
   k_g*A)))^(1/2), ((h*P/(k_g*A)))^(1/2)*L);
21 //
22 Lc = L+d/4; // [m] corrected length
23 // the parameters of interest for the heat flow and
   efficiency comparisons are now tabulated as
24 printf("\nthe parameters of interest for the heat

```

```

        flow and efficiency comparisons are now tabulated
        as");
25 printf("\nMaterial\t\t(hPkA)\t\tmLc");
26 printf("\ncopper\t\t%f\t\t%f", (h*P*k_c*A), ((h*P/(k_c
    *A)))^(1/2)*Lc);
27 printf("\nstainless steel\t%f\t\t%f", (h*P*k_s*A), ((h
    *P/(k_s*A)))^(1/2)*Lc);
28 printf("\nglass\t\t%f\t\t%f", (h*P*k_g*A), ((h*P/(k_g*
    A)))^(1/2)*Lc);
29 // efficiency is calculated using equation(2-38) by
    using the above values of mLc
30 // to compare the heat flows we could either
    calculate the values from equation (2-36) for a
    unit value of theta_o
31 printf("\nMaterial\t\ttefficiency\tq relative to
    copper percentage");
32 printf("\ncopper\t\t%f\t\t%f", tanh(((h*P/(k_c*A)))
    ^ (1/2)*Lc)/(((h*P/(k_c*A)))^(1/2)*Lc), 100);
33 printf("\nstainless steel\t%f\t\t%f", tanh(((h*P/(k_s
    *A)))^(1/2)*Lc)/(((h*P/(k_s*A)))^(1/2)*Lc), ((tanh
    (((h*P/(k_s*A)))^(1/2)*Lc)/(((h*P/(k_s*A)))^(1/2)
    *Lc)))/(tanh(((h*P/(k_c*A)))^(1/2)*Lc)/(((h*P/(k_c
    *A)))^(1/2)*Lc)))*100);
34 printf("\nglass\t\t%f\t\t%f", tanh(((h*P/(k_g*A)))
    ^ (1/2)*Lc)/(((h*P/(k_g*A)))^(1/2)*Lc), ((tanh(((h*
    P/(k_g*A)))^(1/2)*Lc)/(((h*P/(k_g*A)))^(1/2)*Lc))
    /(tanh(((h*P/(k_c*A)))^(1/2)*Lc)/(((h*P/(k_c*A)))
    ^ (1/2)*Lc)))*100);
35 deff ('[y]=f1(x)', 'y=exp(-((h*P/(k_c*A)))^(1/2)*x)
    /(1+exp(-2*((h*P/(k_c*A)))^(1/2)*L))+exp(((h*P/(
    k_c*A)))^(1/2)*x)/(1+exp(2*((h*P/(k_c*A)))^(1/2)*
    L))');
36 deff ('[y]=f2(x)', 'y=exp(-((h*P/(k_s*A)))^(1/2)*x)
    /(1+exp(-2*((h*P/(k_s*A)))^(1/2)*L))+exp(((h*P/(
    k_s*A)))^(1/2)*x)/(1+exp(2*((h*P/(k_s*A)))^(1/2)*
    L))');
37 deff ('[y]=f3(x)', 'y=exp(-((h*P/(k_g*A)))^(1/2)*x)
    /(1+exp(-2*((h*P/(k_g*A)))^(1/2)*L))+exp(((h*P/(

```

```

        k_g*A)))^(1/2)*x)/(1+exp(2*((h*P/(k_g*A)))^(1/2)*
        L))');
38 x=0:0.01:0.1;
39 plot(x,f1,x,f2,x,f3);
40 legend("copper, k = 385 W/m degree celsius",
        stainless steel k = 17 W/m degree celsius", "glass
        k = 0.8 W/m degree celsius");
41 xlabel("x,m");
42 ylabel("theta / theta_O");
43 xgrid();

```

---

#### Scilab code Exa 2.8 straight aluminium fin

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 2.8\n\n\n");
4 // straight aluminium fin
5 // illustration2.8
6 // solution
7
8 t = 0.003; // [m] thickness of fin
9 L = 0.075; // [m] length of fin
10 Tb = 300; // [degree celsius] base temperature
11 Tair = 50; // [degree celsius] ambient temperature
12 k = 200; // [W/m per degree celsius] heat transfer
    coefficient of aluminium fin
13 h = 10; // [W/square meter per degree celsius]
    convectional heat transfer coefficient
14 // We Will use the approximate method of solution by
    extending the fin
15 // With a fictitious length t/2
16 // using equation(2-36)
17 Lc = L+t/2; // [m] corrected length
18 z = 1; // [m] unit depth
19 p = (2*z+2*t); // [m] perimeter of fin

```



```

20 A = z*t; // [square meter] crossectional area of fin
21 m = ((h*p)/(k*A))^(0.5);
22 // from equation(2-36)
23 dt = Tb-Tair; // [degree celsius] temperature
    difference
24 q = tanh(m*Lc)*((h*p*k*A)^(0.5))*dt; // [W/m] heat
    transfer per unit length
25 printf("heat loss from the fin per unit length is %f
    W/m",q);

```

---

### Scilab code Exa 2.9 circumferential aluminium fin

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 2.9\n\n\n");
4 // circumferential aluminium fin
5 // illustration2.9
6 // solution
7
8 t = 0.001; // [m] thickness of fin
9 L = 0.015; // [m] length of fin
10 Ts = 170; // [degree celsius] surface temperature
11 Tfluid = 25; // [degree celsius] fluid temperature
12 k = 200; // [W/m per degree celsius] heat transfer
    coefficient of aluminium fin
13 h = 130; // [W/square meter per degree celsius]
    convectional heat transfer coefficient
14 d = 0.025; // [m] tube diameter
15 Lc = L+t/2; // [m] corrected length
16 r1 = d/2; // [m] radius of tube
17 r2_c = r1+Lc; // [m] corrected radius
18 Am = t*(r2_c-r1); // [square meter] profile area
19 c = r2_c/r1; // constant to determine efficiency of
    fin from curve
20 c1 = ((Lc)^(1.5))*((h/(k*Am))^(0.5)); // constant to

```

```

        determine efficiency of fin from curve
21 // using c and c1 to determine the efficiency of the
    fin from figure (2-12)
22 // we get nf = 82 percent
23 // heat would be transferred if the entire fin were
    at the base temperature
24 // both sides of fin exchanging heat
25 q_max = 2*pi*(r2_c^(2)-r1^(2))*h*(Ts-Tfluid); // [W]
    maximum heat transfer
26 q_act = 0.82*q_max; // [W] actual heat transfer
27 printf("the actual heat transferred is %f W",q_act);

```

---

#### Scilab code Exa 2.10 rod with heat sources

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 2.10\n\n\n");
4 // rod with heat sources
5 // illustration2.10
6 // solution
7
8 // q_dot is uniform heat source per unit volume
9 // h is convection coefficient
10 // k is heat transfer coefficient
11 // A is area of crosssection
12 // P is perimeter
13 // Tinf is environment temperature
14 // we first make an energy balance on the element of
    the rod shown in figure(2-10)
15 // energy in left place + heat generated in element
    = energy out right face + energy lost by
    convection
16 // or
17 //  $-(k*A*dT\_by\_dx)+(q\_dot*A*dx) = -(k*A(dT\_by\_dx+(d^2T\_by\_dx^2)*dx))+h*P*dx*(T-Tinf)$ 

```

```

18 // simplifying we have
19 //  $d^2T_{by\_dx^2} - ((h*P)/(k*A)) * (T - T_{inf}) + q_{dot}/k = 0$ 
20 // replacing  $\theta = (T - T_{inf})$  and (square meter) =
    ((h*P)/(k*A))
21 //  $d^2\theta_{by\_dx^2} - (\text{square meter}) * \theta + q_{dot}/k = 0$ 
22 // we can make a further substitution as  $\theta' =$ 
     $\theta - (q_{dot}/(k * (\text{square meter})))$ 
23 // so that our differential equation becomes
24 //  $d^2\theta'_{by\_dx^2} - (\text{square meter}) * \theta' =$ 
25 // which has the general solution  $\theta' = C1 * \exp^{(-m*x)} + C2 * \exp^{(m*x)}$ 
26 // the two end temperatures are used to establish
    the boundary conditions:
27 //  $\theta' = \theta_1' = T1 - T_{inf} - q_{dot}/(k * (\text{square meter})) = C1 + C2$ 
28 //  $\theta' = \theta_2' = T2 - T_{inf} - q_{dot}/(k * (\text{square meter})) = C1 * \exp^{(-m*L)} + C2 * \exp^{(m*L)}$ 
29 // solving for the constants C1 and C2 gives
30 //  $((\theta_1' * \exp^{(2*m*L)} - \theta_2' * \exp^{(m*L)}) * \exp^{(-m*x)} + ((\theta_2' * \exp^{(m*L)} - \theta_1') * \exp^{(m*x)}) / (\exp^{(2*m*L)} - 1))$ 
31 printf("the expression for the temperature
    distribution in the rod is ");
32 printf("\n  $\theta' = ((\theta_1' * \exp^{(2*m*L)} - \theta_2' * \exp^{(m*L)}) * \exp^{(-m*x)} + ((\theta_2' * \exp^{(m*L)} - \theta_1') * \exp^{(m*x)}) / (\exp^{(2*m*L)} - 1))$ ");
33 printf("\n for an infinitely long heat generating
    fin with the left end maintained at T1, the
    temperature distribution becomes ");
34 printf("\n  $\theta' / \theta_1 = \exp^{(-m*x)}$ ");

```

---

**Scilab code Exa 2.11** influence of contact conductance on heat transfer

```

1 clear;
2 clc;

```

```

3 printf("\t\t\tExample Number 2.11\n\n\n");
4 // influence of contact conductance on heat transfer
5 // illustration2.11
6 // solution
7
8 d = 0.03; //[m] diameter of steel bar
9 l = 0.1; //[m] length of steel bar
10 A = (%pi*d^(2))/4; // [square meter] crossectional
    area of bar
11 k = 16.3; // [W/square meter per degree celsius]
    thermal conductivity of tube
12 hc = 1893.93; // [W/square meter per degree celsius]
    contact coefficient
13 // the overall heat flow is subjected to three
    thermal resistances
14 // one conduction resistance for each bar
15 // contact resistance
16 Rth = 1/(k*A); // [degree celsius /W]
17 // from table(2-2) the contact resistance is
18 Rc = 1/(hc*A); // [degree celsius /W]
19 Rt = 2*Rth+Rc; // [degree celsius /W] total
    resistance
20 dt = 100; // [degree celsius] temperature difference
21 q = dt/Rt; // [W] overall heat flow
22 printf("overall heat flow is %f W",q);
23 // temperature drop across the contact is found by
    taking the ratio
24 // of the contact resistance to the total thermal
    resistance
25 dt_c = (Rc/(2*Rth))*dt; // [degree celsius]
26 printf("\nthe temperature drop across the contact is
    %f degree celsius",dt_c);

```

---

# Chapter 3

## Steady State Conduction Multiple Dimension

Scilab code Exa 3.1 buried pipe

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 3.1\n\n\n");
4 // buried pipe
5 // illustration3.1
6 // solution
7
8 d = 0.15; // [m] diameter of pipe
9 r = d/2; // [m] radius of pipe
10 L = 4; // [m] length of pipe
11 Tp = 75; // [degree celsius] pipe wall temperature
12 Tes = 5; // [degree celsius] earth surface
    temperature
13 k = 0.8; // [W/m per degree celsius] thermal
    conductivity of earth
14 D = 0.20; // [m] depth of pipe inside earth
15 // We may calculate the shape factor for this
    situation using equation given in table 3-1
16 // since  $D < 3*r$ 
```

```

17 S = (2*pi*L)/acosh(D/r); // [m] shape factor
18 // the heat flow is calculated from
19 q = k*S*(Tp-Tes); // [W]
20 printf("heat lost by the pipe is %f W",q);

```

---

### Scilab code Exa 3.2 cubical furnace

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 3.2\n\n\n");
4 // cubical furnace
5 // illustration3.2
6 // solution
7
8 a = 0.5; // [m] length of side of cubical furnace
9 Ti = 500; // [degree celsius] inside furnace
   temperature
10 To = 50; // [degree celsius] outside temperature
11 k = 1.04; // [W/m per degree celsius] thermal
   conductivity of fireclay brick
12 t = 0.10; // [m] wall thickness
13 A = a*a; // [square meter] area of one face
14 // we compute the total shape factor by adding the
   shape factors for the walls, edges and corners
15 Sw = A/t; // [m] shape factor for wall
16 Se = 0.54*a; // [m] shape factor for edges
17 Sc = 0.15*t; // [m] shape factor for corners
18 // there are six wall sections, twelve edges and
   eight corners, so the total shape factor S is
19 S = 6*Sw+12*Se+8*Sc; // [m]
20 // the heat flow is calculated as
21 q = k*S*(Ti-To); // [W]
22 printf("heat lost through the walls is %f kW",q
   /1000);

```

---

### Scilab code Exa 3.3 buried disk

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 3.3\n\n\n");
4 // buried disk
5 // illustration3.3
6 // solution
7
8 d = 0.30; // [m] diameter of disk
9 r = d/2; // [m] radius of disk
10 Td = 95; // [degree celsius] disk temperature
11 Ts = 20; // [degree celsius] isothermal surface
    temperature
12 k = 2.1; // [W/m per degree celsius] thermal
    conductivity of medium
13 D = 1.0; // [m] depth of disk in a semi-infinite
    medium
14 // We have to calculate shape factor using relation
    given in table (3-1)
15 // We select the relation for the shape factor is
    for the case  $D/(2*r) > 1$ 
16 S = (4*pi*r)/((pi/2)-atan(r/(2*D))); // [m] shape
    factor
17 // heat lost by the disk is
18 q = k*S*(Td-Ts); // [W]
19 printf("heat lost by disk is %f W",q);
```

---

### Scilab code Exa 3.4 buried parallel disk

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

```

3 printf("\t\t\tExample Number 3.4\n\n\n");
4 // buried parallel disk
5 // illustration3.4
6 // solution
7
8 d = 0.50; // [m] diameter of both disk
9 r = d/2; // [m] radius of disk
10 Td1 = 80; // [degree celsius] first disk temperature
11 Td2 = 20; // [degree celsius] second disk temperature
12 k = 2.3; // [W/m per degree celsius] thermal
    conductivity of medium
13 D = 1.5; // [m] seperation of disk in a infinite
    medium
14 // We have to calculate shape factor using relation
    given in table (3-1)
15 // We select the relation for the shape factor is
    for the case  $D > 5r$ 
16 S = (4*pi*r)/((pi/2)-atan(r/D)); // [m] shape
    factor
17 q = k*S*(Td1-Td2); // [W]
18 printf("heat transfer between the disks is %f W",q);

```

---

### Scilab code Exa 3.5 Nine node problem

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 3.5\n\n\n");
4 // Example 3.5 (page no.-86-88)
5 // Nine-node problem
6 // solution
7
8 h = 10; // [W/square meter per degree celsius]
    convectional heat transfer coefficient
9 k = 10; // [W/m per degree celsius] heat transfer
    coefficient

```



```

10 dx = 1/3; // [m] length of small squares in x
    direction
11 dy = 1/3; // [m] length of small squares in y
    direction
12 y = h*dx/(k); // to use in equation (3-25) and (3-26)
13 // the nodal equation for nodes is following
14 // T2+T4-4*T1 = -600 FOR NODE 1
15 // T3+T1+T5-4*T2 = -500 FOR NODE 2
16 // 2*T2+T6-4.67*T3 = -567 FOR NODE 3
17 // T5+T7+T1-4*T4 = -100 FOR NODE 4
18 // T6+T4+T8+T2-4*T5 = 0 FOR NODE 5
19 // 2*T5+T3+T9-4.67*T6 = -67 FOR NODE 6
20 // 2*T4+T8-4.67*T7 = -167 FOR NODE 7
21 // 2*T5+T7+T9-4.67*T8 = -67 FOR NODE 8
22 // T6+T8-2.67*T9 = -67 FOR NODE 9
23 A = [-4 1 0 1 0 0 0 0 0;1 -4 1 0 1 0 0 0 0;0 2 -4.67
    0 0 1 0 0 0;1 0 0 -4 1 0 1 0 0;0 1 0 1 -4 1 0 1
    0;0 0 1 0 2 -4.67 0 0 1;0 0 0 2 0 0 -4.67 1 0;0 0
    0 0 2 0 1 -4.67 1;0 0 0 0 0 1 0 1 -2.67];
24 C = [-600;-500;-567;-100;0;-67;-167;-67;-67];
25 T = A^(-1)*C; // [degree celsius]
26 printf("The nodal temperature of node 1 to 9 is
    shown below respectively");
27 disp(T);
28 // the heat flows at the boundaries are computed in
    two ways:
29 // as conduction flows for the 100 and 500 degree
    celsius faces and
30 // as convection flows for the other two faces
31 // for the 500 degree face the heat flow into the
    face is q = sigma(k*dx*dT/dy)
32 // where dt is temperature difference and dy is
    length of small squares in y direction
33 q = k*dx*[500-T(1)+500-T(2)+(500-T(3))/2]/dy; // [W/m
    ]
34 // the heat flow out of the 100 degree face is q =
    sigma(k*dy*dT/dx)
35 q1 = k*dy*[T(1)-100+T(4)-100+(T(7)-100)/2]/dx; // [W/

```

```

m]
36 // the convection heat flow out the right face is
    given by the convection relation  $q = \sigma(h*dy*($ 
     $T-T_{inf})$ )
37 q2 = h*dy*[T(3)-100+T(6)-100+(T(9)-100)/2]; // [W/m]
38 // the convection heat flow out the bottom face is
    given by the convection relation  $q = \sigma(h*dx*($ 
     $T-T_{inf})$ )
39 q3 = h*dx*[(100-100)/2+T(7)-100+T(8)-100+(T(9)-100)
    /2]; // [W/m]
40 // total heat flow out is
41 qt = q1+q2+q3;
42 printf(" heat conducted into the top face is %f W/m"
    ,q);
43 printf("\n total heat flow out is %f W/m",qt);
44 printf("\n this compares that heat flow into the
    system is equal to the heat flow out of the
    system ");

```

---

### Scilab code Exa 3.6 Gauss seidal calculation

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 3.6\n\n\n");
4 // Gauss-Seidal calculation
5 // Example 3.6 (page no.-97-98)
6 // solution
7
8 // it is useful to think in terms of a resistance
    formulation for this problem because all the
    connecting resistances between the nodes in
    figure 3-6(page no.-83) are equal; that is
9 //  $R = dy/(k*dy) = dx/(k*dy) = 1/k$ 
    (a)
10 // therefore , when we apply equation(3-32) to each

```

```

node, we obtain ( $q_i = 0$ )
11 //  $T_i = (\sum K_j * T_j) / (\sum K_j)$ 
                                         (b)
12 // because each node has four resistances connected
    to it and k is assumed constant,
13 //  $\sum K_j = 4 * k$ 
14 // and
15 //  $T_i = (1/4) * (\sum T_j)$ 
                                         (c)
16 // we are now making four nodal equations for
    iteration
17 // node 1 :  $T_1 = (1/4) * (100 + 500 + T_2 + T_3)$ 
18 // node 2 :  $T_2 = (1/4) * (500 + 100 + T_1 + T_4)$ 
19 // node 3 :  $T_3 = (1/4) * (100 + 100 + T_1 + T_4)$ 
20 // node 4 :  $T_4 = (1/4) * (T_3 + T_2 + 100 + 100)$ 
21 // we now set up an iteration table as shown in
    output
22 A=[4 -1 -1 0;-1 4 0 -1;-1 0 4 -1;0 -1 -1 4];
23 b=[600;600;200;200];
24 x=[300;300;200;200];
25 NumIters=6;
26 D=diag(A);
27 A=A-diag(D);
28 for i=1:4
29     D(i)=1/D(i);
30 end
31 n=length(x);
32 x=x(:);
33 y=zeros(n,NumIters);
34 for j=1:NumIters
35     for k=1:n
36         x(k)=(b(k)-A(k,:)*x)*D(k);
37     end
38     y(:,j)=x;
39 end
40 printf("the iteration table is shown as : \n\n");
41 disp(y);
42 printf("\n\n after five iterations the solution

```

```

        converges and the final temperatures are \n");
43 disp(y(1,6),"T1=");
44 disp(y(2,6),"T2=");
45 disp(y(3,6),"T3=");
46 disp(y(4,6),"T4=");

```

---

**Scilab code Exa 3.7** numerical formulation with heat generation

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 3.7\n\n");
4 // numerical formulation with heat generation
5 // Example 3.7 (page no.-99-100)
6 // solution
7
8 d = 4; // [mm] diameter of wire
9 Q = 500; // [MW/cubic meter] heat generation
10 Tos = 200; // [degree celsius] outside surface
    temperature of wire
11 k = 19; // [W/m degree celsius] thermal conductivity
12 // we shall make the calculations per unit length
13 dz = 1;
14 // because the system is one-dimensional, we take
15 dphai = 2*%pi;
16 dr = 0.5; // [mm]
17 // a summary of values for different nodes are
    following
18
19 // node 1.
20
21 rm1 = 0.25; // [mm]
22 Rmplus1 = (dr/2)/((rm1+dr/4)*dphai*dz*k); // [degree
    celsius/W]
23 // Rmminus1 = infinity
24 dV1 = rm1*dr*dphai*dz; // [cubic micro meter]

```

```

25 q1 = Q*dV1; // [W]
26
27 // node 2.
28
29 rm2 = 0.75; // [mm]
30 Rmplus2 = (dr/2)/((rm2+dr/4)*dphai*dz*k); // [degree
    celsius/W]
31 // Rmminus2 = infinity
32 dV2 = rm2*dr*dphai*dz; // [cubic micro meter]
33 q2 = Q*dV2; // [W]
34
35 // node 3.
36
37 rm3 = 1.25; // [mm]
38 Rmplus3 = (dr/2)/((rm3+dr/4)*dphai*dz*k); // [degree
    celsius/W]
39 // Rmminus3 = infinity
40 dV3 = rm3*dr*dphai*dz; // [cubic micro meter]
41 q3 = Q*dV3; // [W]
42
43 // node 4.
44
45 rm4 = 1.75; // [mm]
46 Rmplus4 = (dr/2)/((rm4+dr/4)*dphai*dz*k); // [degree
    celsius/W]
47 // Rmminus1 = infinity
48 dV4 = rm4*dr*dphai*dz; // [cubic micro meter]
49 q4 = Q*dV4; // [W]
50
51 // a summary of values of sum_one_by_Rij and Ti
    according to equation (3-32) is now given to be
    used in gauss seidal iteration scheme
52
53 // node 1
54
55 sum_one_by_Rij1 = (1/Rmplus1); // [degree celsius/W]
56 // the equations formed after putting values are
57 // T1 = 3.288+T2

```

```

58
59 // node 2
60
61 sum_one_by_Rij2 = (1/Rmplus2); // [degree celsius/W]
62 // the equations formed after putting values are
63 //  $T_2 = 3.289 + (1/3)*T_1 + (2/3)*T_3$ 
64
65 // node 3
66
67 sum_one_by_Rij3 = (1/Rmplus3); // [degree celsius/W]
68 // the equations formed after putting values are
69 //  $T_3 = 3.290 + 0.4*T_2 + 0.6*T_4$ 
70
71 // node 4
72
73 sum_one_by_Rij4 = (1/Rmplus4); // [degree celsius/W]
74 // the equations formed after putting values are
75 //  $T_4 = 2.193 + (2/7)*T_3 + 142.857$ 
76
77 // now we will solve these equations by iteration
78 A=[1 -1 0 0; -(1/3) 1 -(2/3) 0; 0 -0.4 1 -0.6; 0 0
    -(2/7) 1];
79 b=[3.288; 3.289; 3.290; 142.857+2.193];
80 x=[240; 230; 220; 210];
81 NumIters=13;
82 D=diag(A);
83 A=A-diag(D);
84 n=length(x);
85 x=x(:);
86 y=zeros(n, NumIters);
87 for j=1:NumIters
88     for z=1:n
89         x(z)=(b(z)-A(z,:)*x)*D(z);
90     end
91     y(:,j)=x;
92 end
93 printf("thirteen iterations are now tabulated :\n");
94 disp(y);

```

```

95 // the total heat loss from the wire may be
    calculated as the conduction through Rmplus at
    node 4. then
96 T4 = y(4,13); // [degree celsius]
97 q = (T4-Tos)/(Rmplus4); // [W/m]
98 // this must equal the heat generated in the wire,
    or
99 V = %pi*(d*10^(-3)/2)^(2); // [square meter]
100 q_exact = Q*10^(6)*V; // [W/m]
101 printf("\n\n the total heat loss from the wire by
    the conduction through Rmplus at node 4 is %f kW/
    m",q/1000);
102 printf("\n\n heat generated in the wire is %f kW/m",
    q_exact/1000);
103 printf("\n\n the difference between the two values
    results from the inaccuracy in determination of
    T4");

```

---

### Scilab code Exa 3.8 heat generation with non uniform nodal elements

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 3.8\n\n\n");
4 // heat generation with non uniform nodal elements
5 // Example 3.8 (page no.-100-103)
6 // solution
7
8 k = 0.8; // [W/m degree celsius] thermal conductivity
    of glass
9 d = 0.003; // [m] thickness of layer of glass
10 x = 0.001; // [m] thickness of electric conducting
    strip
11 Tinf = 30; // [degree celsius] environment
    temperature
12 h = 100; // [W/square meter degree celsius]

```

```

13 q1 = 40; // [W] heat generated by strips
14 q2 = 20; // [W] heat generated by strips
15 // the nodal network for a typical section of the
    glass is shown in figure. In this example we have
    not chosen  $dx = dy$ .
16 // because of symmetry  $T1 = T7$  , $T2 = T6$ , etc , and we
    only need to solve the temperatures of 16 nodes.
    we employ the resistance formulation. As shown,
    we have chosen
17 dx = 0.005; // [m]
18 dy = 0.001; // [m]
19 A = 0.005; // [square meter]
20 // various resistances may now be calculated:
21
22 // for nodes 1,2,3,4:
23 one_by_Rm_p1 = k*dy/(2*dx);
24 one_by_Rm_m1 = one_by_Rm_p1;
25 one_by_Rn_p1 = h*A;
26 one_by_Rn_m1 = k*dx/dy;
27
28 // for nodes 8,9,10,11,15,16,17,18:
29 one_by_Rm_p2 = k*dy/(dx);
30 one_by_Rm_m2 = one_by_Rm_p2;
31 one_by_Rn_m2 = k*dx/dy;
32 one_by_Rn_p2 = one_by_Rn_m2;
33
34 // for nodes 22,23,24,25:
35 one_by_Rm_p3 = k*dy/(2*dx);
36 one_by_Rm_m3 = one_by_Rm_p3;
37 one_by_Rn_p3 = k*dx/dy;
38 one_by_Rn_m3 = 0; // [insulated surface]
39
40 // from the above resistances we may calculate the
    sum_one_by_Rij as
41 // nodes : 1,2,3,4:
42 sum_one_by_Rij1 = 4.66;
43 // nodes : 8,9,10,11,15,16,17,18:
44 sum_one_by_Rij2 = 8.32;

```



```

45 // nodes : 22,23,24,25:
46 sum_one_by_Rij3 = 4.16;
47 // the nodal equations are obtained from equation
    (3-31)
48 // only node 4 has a heat generation term,  $Q_i = 0$ 
    for all other nodes.
49 // the equations are listed below
50 // for node 1 :  $T_8 \cdot \text{one\_by\_Rn\_m1} + T_2 \cdot \text{one\_by\_Rm\_p1} + 30 \cdot$ 
     $\text{one\_by\_Rn\_m1} - \text{sum\_one\_by\_Rij1} \cdot T_1 = 0;$ 
51 // for node 4 :  $T_5 \cdot \text{one\_by\_Rm\_p1} + T_3 \cdot \text{one\_by\_Rm\_m1} + 30 \cdot$ 
     $\text{one\_by\_Rn\_p1} + T_{11} \cdot \text{one\_by\_Rn\_m1} + Q - \text{sum\_one\_by\_Rij1} \cdot$ 
     $T_4 = 0$ 
52 // similar equations are obtained and we solve it by
    matrix method
53 Z = [-4.58 0.08 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
54      0.08 -4.66 0.08 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0;
55      0 0.08 -4.66 0.08 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0;
56      0 0 0.16 -4.66 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0;
57      4 0 0 0 -8.16 0.16 0 0 4 0 0 0 0 0 0 0 0 0 0;
58      0 4 0 0 0.16 -8.32 0.16 0 0 4 0 0 0 0 0 0 0 0 0;
59      0 0 4 0 0 0.16 -8.32 0.16 0 0 4 0 0 0 0 0 0 0 0;
60      0 0 0 4 0 0 0.32 -8.32 0 0 0 4 0 0 0 0 0 0 0;
61      0 0 0 0 4 0 0 0 -8.16 0.16 0 0 4 0 0 0 0 0 0;
62      0 0 0 0 0 4 0 0 0.16 -8.32 0.16 0 0 4 0 0 0 0 0;
63      0 0 0 0 0 0 4 0 0 0.16 -8.32 0.16 0 0 4 0 0 0 0;
64      0 0 0 0 0 0 0 4 0 0 0.32 -8.32 0 0 0 4 0 0 0;
65      0 0 0 0 0 0 0 0 4 0 0 0 -4.08 0.08 0 0 0 0;
66      0 0 0 0 0 0 0 0 0 4 0 0 0.08 -4.16 0.08 0 0;
67      0 0 0 0 0 0 0 0 0 0 4 0 0 0.08 -4.16 0.08;
68      0 0 0 0 0 0 0 0 0 0 0 4 0 0 0.16 -4.16];
69 C = [-15;-15;-15;-15-q2;0;0;0;0;0;0;0;0;0;0;0;0;0];
70 T1 = Z^(-1)*C;
71 printf("Nodes
    (1,2,3,4,8,9,10,11,15,16,17,18,22,23,24,25)
    temperature at 20 W/m respectively");
72 disp(T1);
73 Z1 = [-4.58 0.08 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
74      0.08 -4.66 0.08 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0;

```

```

75     0 0.08 -4.66 0.08 0 0 4 0 0 0 0 0 0 0 0 0;
76     0 0 0.16 -4.66 0 0 0 4 0 0 0 0 0 0 0 0;
77     4 0 0 0 -8.16 0.16 0 0 4 0 0 0 0 0 0 0;
78     0 4 0 0 0.16 -8.32 0.16 0 0 4 0 0 0 0 0 0;
79     0 0 4 0 0 0.16 -8.32 0.16 0 0 4 0 0 0 0 0;
80     0 0 0 4 0 0 0.32 -8.32 0 0 0 4 0 0 0 0;
81     0 0 0 0 4 0 0 0 -8.16 0.16 0 0 4 0 0 0;
82     0 0 0 0 0 4 0 0 0.16 -8.32 0.16 0 0 4 0 0;
83     0 0 0 0 0 0 4 0 0 0.16 -8.32 0.16 0 0 4 0;
84     0 0 0 0 0 0 0 4 0 0 0.32 -8.32 0 0 0 4;
85     0 0 0 0 0 0 0 0 4 0 0 0 -4.08 0.08 0 0;
86     0 0 0 0 0 0 0 0 0 4 0 0 0.08 -4.16 0.08 0;
87     0 0 0 0 0 0 0 0 0 0 4 0 0 0.08 -4.16 0.08;
88     0 0 0 0 0 0 0 0 0 0 0 4 0 0 0.16 -4.16];
89 C1 = [-15;-15;-15;-15-q1;0;0;0;0;0;0;0;0;0;0;0;0];
90 T2 = Z1^(-1)*C1;
91 printf("\n\n Nodes
      (1,2,3,4,8,9,10,11,15,16,17,18,22,23,24,25)
      temperature at 40 W/m respectively");
92 disp(T2);
93 // we know the numerical value that the convection
   should have
94 // the convection loss at the top surface is given
   by
95 qc1 = 2*h*[(dx/2)*(T1(1)-Tinf)+dx*(T1(2)+T1(3)-2*
   Tinf)+(dx/2)*(T1(4)-Tinf)]; // [W] for 20W/m, the
   factor of 2 accounts for both sides of section
96 qc2 = 2*h*[(dx/2)*(T2(1)-Tinf)+dx*(T2(2)+T2(3)-2*
   Tinf)+(dx/2)*(T2(4)-Tinf)]; // [W] for 40W/m
97 printf("\n\n the convection loss at the top surface
   is given by (for 20 W/m heat generation) %f W",
   qc1);
98 printf("\n\n the convection loss at the top surface
   is given by (for 40 W/m heat generation) %f W",
   qc2);

```

---

**Scilab code Exa 3.11** use of variable mesh size

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 3.11\n\n\n");
4 // use of variable mesh size
5 // Example 3.11 (page no.-108-110)
6 // solution
7
8 // using data given in figure example 3-11(page no
  .-109)
9 // nodes 5,6,8, and 9 are internal nodes with dx =
  dy and have nodal equations in the form of
  equation(3-24). Thus,
10 //  $600+T_6+T_8-4*T_5 = 0$ 
11 //  $500+T_5+T_7+T_9-4*T_6 = 0$ 
12 //  $100+T_5+T_9+T_{11}-4*T_8 = 0$ 
13 //  $T_8+T_6+T_{10}+T_{12}-4*T_9 = 0$ 
14 // For node 7 we can use a resistance formulation
  and obtain
15 //  $(1/R_{7_6}) = k$ 
16 //  $(1/R_{7_500\_degree}) = k*(dx/6+dx/2)/(dy/3) = 2*k$ 
17 //  $(1/R_{7_10}) = 2*k$ 
18 // and we find
19 //  $1000+T_6+2*T_{10}-5*T_7 = 0$ 
20 // similar resistance are obtained for node 10
21 //  $(1/R_{10_9}) = k$ 
22 //  $(1/R_{10_7}) = 2*k = (1/R_{10_1})$ 
23 // so that
24 //  $2*T_7+T_9+2*T_{11}-5*T_{10} = 0$ 
25 // for node 1,
26 //  $(1/R_{1_12}) = k*(dy/6+dy/2)/(dx/3) = 2*k$ 
27 //  $(1/R_{1_3}) = k*(dx/6+dx/2)/(dy) = 2*k/3$ 
28 //  $(1/R_{1_10}) = 2*k$ 
```

```

29 // and the nodal equation becomes
30 //  $3*T_{12}+3*T_{10}+T_3-7*T_1 = 0$ 
31 // for node 11,
32 //  $(1/R_{11-100\_degree}) = (1/R_{11-12}) = k*(dy/6+dy/2)$ 
//  $/(dx/3) = 2*k$ 
33 //  $(1/R_{11-8}) = k$ 
34 //  $(1/R_{11-13}) = k*(dx/3)/dy = k/3$ 
35 // and the nodal equation becomes
36 //  $600+6*T_{12}+3*T_8+T_{13}-16*T_{11} = 0$ 
37 // Similarly , the equation for node 12 is
38 //  $3*T_9+6*T_{11}+6*T_1+T_{14}-16*T_{12} = 0$ 
39 // for node 13,
40 //  $(1/R_{13-100\_degree}) = k*(dy)/(dx/3) = 3*k = 1/$ 
//  $R_{13-14}$ 
41 //  $(1/R_{13-11}) = (1/R_{13-100}) = k/3$ 
42 // and we obtain
43 //  $1000+9*T_{14}+T_{11}-20*T_{13} = 0$ 
44 // similarly for node 14,
45 //  $100+9*T_{13}+9*T_3+T_{12}-20*T_{14} = 0$ 
46 // finally , from resistances already found , the
// nodal equation for node 3 is
47 //  $200+9*T_{14}+2*T_1-13*T_3 = 0$ 
48 // we choose to solve the set of equations by the
// gauss-seidel iteration technique
49 A=[1 -1 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 -1 0 0 0 0 0 0 0
// 0 0 0 0;0 0 0 0 -4 1 0 1 0 0 0 0 0 0;0 0 0 0 1
-4 1 0 1 0 0 0 0 0;0 0 0 0 1 0 0 -4 1 0 1 0 0 0;0
// 0 0 0 0 1 0 1 -4 1 0 1 0 0;0 0 0 0 0 1 -5 0 0 2
0 0 0 0;2 0 0 0 0 0 2 0 1 -5 0 0 0 0;-7 0 1 0 0 0
// 0 0 0 3 0 3 0 0;0 0 0 0 0 0 0 0 3 0 0 -16 6 1 0;6
0 0 0 0 0 0 0 3 0 6 -16 0 1;0 0 0 0 0 0 0 0 0 0 0 1
// 0 -20 9;0 0 9 0 0 0 0 0 0 0 1 9 -20;2 0 -13 0
0 0 0 0 0 0 0 0 0 9];
50 b
// =[0;0;-600;-500;-100;0;-1000;0;0;-600;0;-1000;-100;-200];

51 T = A^(-1)*b;
52 printf("Nodal temperatures for node

```

```

    (1,2,3,4,5,6,7,8,9,10,11,12,13,14) are
    respectively as follows in degree celsius");
53 disp(T);

```

---

### Scilab code Exa 3.12 Three dimensional numerical formulation

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 3.12\n\n\n");
4 // Three-dimensional numerical formulation
5 // Example 3.12 (page no.-110-113)
6 // solution
7
8 Tinf = 10; // [degree celsius] environment
   temperature
9 h = 500; // [W/square meter degree celsius]
10 Ts = 100; // [degree celsius] four side temperature
11 k = 2; // [W/m degree celsius]
12 dx = 0.01; // [m]
13 dy = 0.01; // [m]
14 dz = 0.01; // [m]
15 // all of the interior nodes for Z-planes 2,3,4 have
   resistances of
16 A = dy*dz; // [square meter]
17 one_by_R = k*A/dx;
18 one_by_R_11_21 = one_by_R;
19 one_by_R_21_22 = one_by_R;
20 // the surface conduction resistances for surface Z-
   plane are
21 one_by_R_11_12 = k*A/dx;
22 one_by_R_11_14 = one_by_R_11_12;
23 // the surface convection resistances are
24 one_by_R_11_inf = h*A;
25 // for surfaces nodes like 11 the sum_one_by_R_ij
   term in equation (3-32) becomes

```

```

26 sum_one_by_R_11_j = 4*one_by_R_11_12+one_by_R+
    one_by_R_11_inf;
27 // while, for interior nodes, we have
28 sum_one_by_R_21_j = 6*one_by_R;
29 // for the insulated black surface nodes
30 sum_one_by_R_51_j = 4*one_by_R_11_12+one_by_R;
31 // there are 30 nodes in total; 6 in each z-plane.
    we could write the equations for all of them but
    prefer to take advantage of the symmetry of the
    problem as indicated in figure. thus,
32 // T11 = T13 = T14 = T16 And T12 = T15, etc
33 // we may then write the surface nodal equations as
34 // T11 = [0.05* Tinf+0.02*T21+(0.01)*(100+100+T14+T12
    )]/0.11
35 // T12 = [0.05* Tinf+0.02*T22+(0.01)*(100+T11+T15+T13
    )]/0.11
36 // inserting
37 Tinf = 10;// [degree celsius]
38 // following the same procedure for the other z-
    planes we obtain
39 // T21 = (200+T11+T31+T22)/5
40 // T22 = (100+T12+T32+2*T21)/5
41 // T31 = (200+T21+T41+T32)/5
42 // T32 = (100+T22+T42+2*T31)/5
43 // T41 = (200+T31+T51+T42)/5
44 // T42 = (100+T32+T52+2*T41)/5
45 // T51 = (2+0.02*T41+0.01*T52)/0.05
46 // T52 = (1+0.02*T42+0.02*T51)/0.05
47 // Solving the 10 equations
48 Z = [-0.1 0.01 0.02 0 0 0 0 0 0 0 0;
49      0.02 -0.1 0 0.02 0 0 0 0 0 0 0;
50      1 0 -5 1 1 0 0 0 0 0 0;
51      0 1 2 -5 0 1 0 0 0 0 0;
52      0 0 1 0 -5 1 1 0 0 0 0;
53      0 0 0 1 2 -5 0 1 0 0 0;
54      0 0 0 0 1 0 -5 1 1 0 0;
55      0 0 0 0 0 1 2 -5 0 1 0;
56      0 0 0 0 0 0 0.02 0 -0.05 0.01;

```

```

57     0 0 0 0 0 0 0 0 0.02 0.02 -0.05];
58 C = [-2.5; -1.5; -200; -100; -200; -100; -200; -100; -2; -1];
59 T = Z^(-1)*C;
60 T11 = T(1);
61 T12 = T(2);
62 T21 = T(3);
63 T22 = T(4);
64 T31 = T(5);
65 T32 = T(6);
66 T41 = T(7);
67 T42 = T(8);
68 T51 = T(9);
69 T52 = T(10);
70 printf("the following results for the temperature in
        each z-plane is ");
71 printf("\n\t\t z-plane\t\tNode 1\t\tNode 2");
72 printf("\n\t\t%f\t\t%f\t\t%f" ,1,T11,T12);
73 printf("\n\t\t%f\t\t%f\t\t%f" ,2,T21,T22);
74 printf("\n\t\t%f\t\t%f\t\t%f" ,3,T31,T32);
75 printf("\n\t\t%f\t\t%f\t\t%f" ,4,T41,T42);
76 printf("\n\t\t%f\t\t%f\t\t%f" ,5,T51,T52);
77 val = [1 2 3 4 5];
78 val1 = [T11 T21 T31 T41 T51];
79 val2 = [T12 T22 T32 T42 T52];
80 plot(val, val1, val, val2);
81 legend("T11", "T22");
82 xgrid();
83 xlabel("z-plane");
84 ylabel("Temperature (degree celsius)");

```

---

# Chapter 4

## Unsteady State Conduction

Scilab code Exa 4.1 steel ball cooling in air

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 4.1\n\n");
4 // steel ball cooling in air
5 // illustration4.1
6 // solution
7
8 h = 10; // [W/square meter per degree celsius]
          convective heat transfer coefficient
9 k = 35; // [W/m per degree celsius] heat transfer
          coefficient
10 c = 460; // [kJ/kg]
11 r = 0.05/2; // [m] diameter of ball
12 Tb = 450; // [degree celsius] ball temperature
13 Te = 100; // [degree celsius] environment temperature
14 A = 4*pi*r^(2);
15 V = 4*pi*r^(3)/3;
16 // We anticipate that the lumped capacity method
          will apply because of the low value of h and high
          value of k
17 // we check by using equation (4-6)
```



```

18 K = h*(V/A)/k;
19 // since the value of K is less than 0.1 so we will
    use equation (4-5)
20 T = 150; // [degree celsius] attained temperature by
    the ball
21 rho = 7800; // [kg/cubic meter] density of the ball
22 a = (h*A)/(rho*c*V);
23 t = log((T-Te)/(Tb-Te))/(-a); // [s] time required to
    attain the temperature of 150 degree celsius
24 printf("time required to attain the temperature of
    150 by degree celsius by the ball is %f h",t
    /3600);

```

---

**Scilab code Exa 4.2** Semi infinite solid with sudden change in surface conditions

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 4.2\n\n\n");
4 // Semi-infinite solid with sudden change in surface
    conditions
5 // illustration4.2
6 // solution
7
8 k = 45; // [W/m per degree celsius] thermal
    conductivity of steel block
9 alpha = 1.4*10(-5); // [square meter/s] constant
10 Tb = 35; // [degree celsius] block temperature
11 x = 0.025; // [m] depth at which temperature is
    calculated
12 t = 30; // [s] time after which temperature is to be
    calculated
13 // we can make use of the solutions for the semi-
    infinite solid given as equation (4-8) and (4-13a
    )

```

```

14 // for case A (by suddenly raising the surface
    temperature to 250 degree celsius)
15 To = 250; // [degree celsius]
16 T_x_t = To+(Tb-To)*(erf(x/(2*(alpha*t)^(1/2))));
17 printf("temperature at depth of 0.025 m after 30
    second for case 1 is %f degree celsius",T_x_t);
18 // for the constant heat flux case B we make use of
    equation (4-13a)
19 // since qo/A is given
20 q_by_A = 3.2*10^(5); // [W/square meter]
21 T_x_t1 = Tb+(2*q_by_A*(alpha*t/%pi)^(1/2)*exp(-x^(2)
    /(4*alpha*t))/k)-(q_by_A*x*(1-erf(x/(2*(alpha*t)
    ^ (1/2))))/k); // [degree celsius]
22 printf("\n temperature at depth of 0.025 m after 30
    second for case 2 is %f degree celsius",T_x_t1)
23 // for the constant heat flux case the surface
    temperature after 30 s would be evaluated with x
    = 0 in equation(4-13a)
24 x = 0; // [m] at the surface
25 T_x_o = Tb+(2*q_by_A*(alpha*t/%pi)^(1/2)*exp(-x^(2)
    /(4*alpha*t))/k)-(q_by_A*x*(1-erf(x/(2*(alpha*t)
    ^ (1/2))))/k); // [degree celsius]
26 printf("\n surface temperature after 30 second is %f
    degree celsius",T_x_o);

```

---

### Scilab code Exa 4.3 pulsed energy at surface of semi infinite solid

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 4.3\n\n\n");
4 // pulsed energy at surface of semi-infinite solid
5 // illustration4.3
6 // solution
7
8 rho = 7800; // [kg/cubic meter] density of slab

```

```

9 c = 460; // [J/kg degree celsius] constant
10 alpha = 0.44*10^(-5); // [square meter/s] constant
11 Ts = 40; // [degree celsius] initial temperature of
    of slab
12 x = 0.0; // [m] depth at which temperature is
    calculated
13 t = 2; // [s] time after which temperature is
    calculated
14 // this problem is a direct application of equation
    (4-13b)
15 // we have
16 Qo_by_A = 10^(7); // [J/square meter] heat flux
17 To = Ts+(Qo_by_A/(rho*c*(%pi*alpha*t)^(1/2)))*exp(-x
    ^2)/(4*alpha*t)); // [degree celsius] surface
    temperature at x = 0
18 printf("surface temperature at x = 0 and at t = 2
    second is %f degree celsius",To);
19 x = 0.002; // [m] depth at which temperature is
    calculated
20 T = Ts+(Qo_by_A/(rho*c*(%pi*alpha*t)^(1/2)))*exp(-x
    ^2)/(4*alpha*t)); // [degree celsius] temperature
    at depth x = 0.002
21 printf("\n temperature at depth 0.002 m and after 2
    second is %f degree celsius",T);

```

---

#### Scilab code Exa 4.4 heat removal from semi infinite solid

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 4.4\n\n\n");
4 // heat removal from semi-infinite solid
5 // illustration4.4
6 // solution
7
8 alpha = 8.4*10^(-5); // [square meter/s] constant

```

```

 9 Ts = 200; // [degree celsius] initial temperature of
    of slab
10 x = 0.04; // [m] depth at which temperature is
    calculated
11 T_x_t = 120; // [degree celsius] temperature at depth
    0.04 m
12 To = 70; // [degree celsius] surface temperature
    after lowering
13 k = 215; // [W/m degree celsius] heat transfer
    coefficient
14 // We first find the time required to attain the 120
    degree celsius temperature
15 // and then integrate equation(4-12) to find the
    total heat removed during this interval
16 t = (x/(erfinv(((T_x_t-To)/(Ts-To)))*2*sqrt(alpha)))
    ^2); // [s]
17 printf("time taken to attain the temperature of 120
    degree celsius %f s",t);
18 // the total heat removed at the surface is obtained
    by integrating equation(4-12):
19 // Qo_by_A = integrate('qo_by_A','dt',0,t)
20 // or Qo_by_A = integrate('k*(To-Ts)/(sqrt(%pi*alpha
    *t))','dt',0,t)
21 Qo_by_A = integrate('k*(To-Ts)/(sqrt(%pi*alpha*t))',
    't',0,t);
22 printf("\n the total heat removed from the surface
    is %e J/square meter",Qo_by_A);

```

---

**Scilab code Exa 4.5** sudden exposure of semi infinite solid slab to convection

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 4.5\n\n\n");
4 // sudden exposure of semi-infinite solid slab to

```

```

        convection
5 // illustration4.5
6 // solution
7
8 alpha = 8.4*10(-5); // [square meter/s] constant
9 Ts = 200; // [degree celsius] initial temperature of
    of slab
10 Te = 70; // [degree celsius] environment temperature
11 k = 215; // [W/m degree celsius] heat transfer
    coefficient of slab
12 h = 525; // [W/square meter degree celsius] heat
    transfer coefficient
13 x = 0.04; // [m] depth at which temperature is
    calculated
14 T_x_t = 120; // [degree celsius] temperature at depth
    0.04 m
15 // we can use equation (4-15) or figure (4-5) for
    solution of this problem
16 // by using figure it is easier to calculate it
    involves iterative method to solve because time
    appears in both the variables
17 //  $h\sqrt{\alpha t}/k$  and  $x/(2\sqrt{\alpha t})$ 
18 K = (T_x_t-Ts)/(Te-Ts);
19 // we seek the values of t such that the above value
    of K is equal to the value of K which comes out
    from graph
20 // we therefore try values of t and obtain other
    readings
21 printf("The iteration are listed below\n");
22 // at t = 1000s
23 t = 1000; // [s] time
24 A = h*sqrt(alpha*t)/k;
25 B = x/(2*sqrt(alpha*t));
26 printf(" t\t

```

```

30 B = x/(2*sqrt(alpha*t));
31 printf("\n %f\t\t %f \t %f \t\t 0.61",t,A,B);
32 t = 4000; // [s] time
33 A = h*sqrt(alpha*t)/k;
34 B = x/(2*sqrt(alpha*t));
35 printf("\n %f\t\t %f \t %f \t\t 0.68",t,A,B);
36 printf("\n consequently the time required is
    approximately 3000 second");

```

---

Scilab code Exa 4.6 aluminium plate suddenly exposed to convection

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 4.6\n\n\n");
4 // aluminium plate suddenly exposed to convection
5 // illustration4.6
6 // solution
7
8 alpha = 8.4*10^(-5); // [square meter/s] constant
9 Ts = 200; // [degree celsius] initial temperature of
    of plate
10 Te = 70; // [degree celsius] environment temperature
11 k = 215; // [W/m degree celsius] heat transfer
    coefficient of plate
12 h = 525; // [W/square meter degree celsius] heat
    transfer coefficient
13 x = 0.0125; // [m] depth at which temperature is
    calculated
14 t = 60; // [s] time after which plate temperature is
    calculated
15 L = 0.025; // [m] thickness of plate
16 theta_i = Ts-Te; // [degree celsius]
17 // then
18 Z = alpha*t/L^2;
19 X = k/(h*L);

```

```

20 x_by_L = x/L;
21 // from figure 4-7(page no.-144-145)
22 theta_o_by_theta_i = 0.61;
23 theta_o = theta_o_by_theta_i*theta_i;// [degree
    celsius]
24 // from figure 4-10(page no.-149) at x/L = 0.5,
25 theta_by_theta_o = 0.98;
26 theta = theta_by_theta_o*theta_o;// [degree celsius]
27 T = Te+theta;// [degree celsius]
28 // we compute the energy lost by the slab by using
    Figure 4-14(page no.-152). For this calculation
    we require the following properties of aluminium:
29 rho = 2700;// [kg/cubic meter]
30 C = 900;// [J/kg degree celsius]
31 // for figure 4-14(page no.-152) we need
32 V = h^2*alpha*t/(k^2);
33 B = h*L/k;
34 // from figure 4-14(page no.-152)
35 Q_by_Qo = 0.41;
36 // for unit area
37 Qo_by_A = rho*C*2*L*theta_i;// [J/square meter]
38 // so that the heat removed per unit surface area is
39 Q_by_A = Qo_by_A*Q_by_Qo;// [J/square meter]
40 printf("\n\n temperature at a depth of 1.25 cm from
    one of faces after 1 min of exposure of plate to
    the environment is %f degree celsius",T);
41 printf("\n\n energy removed per unit area from the
    plate in this time is %e J/square meter",Q_by_A);

```

---

**Scilab code Exa 4.7** long cylinder suddenly exposed to convection

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 4.7(Page no.-154-155)\n
    \n\n");

```

```

4 // long cylinder suddenly exposed to convection
5 // Example 4.7
6 // solution
7
8 d = 0.05; // [m] diameter of cylinder
9 Ti = 200; // [degree celsius] initial temperature of
    aluminium cylinder
10 Tinf = 70; // [degree celsius] temperature of
    environment
11 h = 525; // [W/square meter degree celsius] heat
    transfer coefficient
12 // we have
13 theta_i = Ti-Tinf; // [degree celsius]
14 alpha = 8.4*10^(-5); // [square meter/s]
15 ro = d/2; // [m]
16 t = 60; // [s]
17 k = 215; // [W/m degree celsius]
18 r = 0.0125; // [m]
19 rho = 2700; // [kg/cubic meter]
20 C = 900; // [J/kg degree celsius]
21 // we compute
22 Z = alpha*t/ro^2;
23 X = k/(h*ro);
24 r_by_ro = r/ro;
25 // from figure 4-8(page no.-146)
26 theta_o_by_theta_i = 0.38;
27 // and from figure 4-11(page no.-150) at r/ro = 0.5
28 theta_by_theta_o = 0.98;
29 // so that
30 theta_by_theta_i = theta_o_by_theta_i*
    theta_by_theta_o;
31 theta = theta_i*theta_by_theta_i; // [degree celsius]
32 T = Tinf+theta; // [degree celsius]
33 // to compute the heat lost , we determine
34 V = h^2*alpha*t/k^2;
35 B = h*ro/k;
36 // then from figure 4-15(page no.-153)
37 Q_by_Qo = 0.65;

```



```

38 // for unit length
39 Qo_by_L = rho*C*pi*ro^2*theta_i;// [J/m]
40 // and the actual heat lost per unit length is
41 Q_by_L = Qo_by_L*Q_by_Qo;// [J/m]
42 printf("temperature at a radius of 1.25 cm is %f
         degree celsius",T);
43 printf("\n\nheat lost per unit length 1 minute after
         the cylinder is exposed to the environment is %e
         J/m",Q_by_L);

```

---

**Scilab code Exa 4.8** semi infinite cylinder suddenly exposed to convection

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 4.8\n\n\n");
4 // semi-infinite cylinder suddenly exposed to
   convection
5 // illustration4.8
6 // solution
7
8 d = 0.05;// [m] diameter of aluminium cylinder
9 Ti = 200;// [degree celsius] initial temperature of
   of cylinder
10 Te = 70;// [degree celsius] environment temperature
11 k = 215;// [W/m degree celsius] heat transfer
   coefficient of plate
12 h = 525;// [W/square meter degree celsius]
   convection heat transfer coefficient
13 alpha = 8.4*10^(-5);// [square meter/s] constant
14 x = 0.10;// [m] distance at which temperature is
   calculated from end
15 t = 60;// [s] time after which temperature is
   measured
16 // so that the parameters for use with figure(4-5)
17 A = h*sqrt(alpha*t)/k;

```

```

18 B = x/(2*sqrt(alpha*t));
19 // from figure (4-5)
20 z = 1-0.036;
21 S_of_X = z;
22 // for the infinite cylinder we seek both the axis-
    and surface-temperature ratios.
23 // the parameters for use with fig(4-8) are
24 r_o = d/2;// [m] radius of aluminium cylinder
25 r = d/2;// [m] for surface temperature ratio
26 C = k/(h*r_o);
27 D = (alpha*t/r_o^(2));
28 y = 0.38;
29 // this is the axis temperature ratio.
30 // to find the surface-temperature ratio ,we enter
    figure (4-11), using
31 R = r/r_o;
32 u = 0.97;
33 // thus
34 w = y;// at r = 0
35 v = y*u;// at r = r_o
36 C_of_0_axis = w;// at r = 0
37 C_of_0_r_o = v;// at r = r_o
38 // combining the solutions for the semi-infinite
    slab and infinite cylinder , we have
39 t = S_of_X*C_of_0_axis;
40 s = S_of_X*C_of_0_r_o;
41 // the corresponding temperatures are
42 T_axis = Te+t*(Ti-Te);
43 T_r_o = Te+s*(Ti-Te);
44 printf("the temperature at the axis is %f degree
    celsius",T_axis);
45 printf("\n the temperature at the surface is %f
    degree celsius",T_r_o);

```

---

**Scilab code Exa 4.9** finite length cylinder suddenly exposed to convection

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 4.9\n\n\n");
4 // finite length cylinder suddenly exposed to
   convection
5 // illustration4.9
6 // solution
7
8 d = 0.05; // [m] diameter of aluminium cylinder
9 Ti = 200; // [degree celsius] initial temperature of
   of cylinder
10 Te = 70; // [degree celsius] environment temperature
11 k = 215; // [W/m degree celsius] heat transfer
   coefficient of plate
12 h = 525; // [W/square meter degree celsius]
   convection heat transfer coefficient
13 alpha = 8.4*10^(-5); // [square meter/s] constant
14 x1 = 0.00625; // [m] distance at which temperature is
   calculated from end
15 t = 60; // [s] time after which temperature is
   measured
16 r = 0.0125; // [m] radius at which temperature is
   calculated
17 // to solve this problem we combine the solutions
   from heisler charts for an infinite cylinder and
   an infinite plate in accordance with the
   combination shown in fig (4-18f)
18 // for the infinite plate problem
19 L = 0.05; // [m]
20 // the x position is measured from the center of the
   plate so that
21 x = L-x1; // [m]
22 A = k/(h*L);
23 B = (alpha*t/L^(2));
24 // from figures (4-17) and (4-10) respectively
25 theta_o_by_i = 0.75;
26 theta_by_i = 0.95;
27 // so that

```

```

28 theta_by_i_plate = theta_o_by_i*theta_by_i;
29 // for the cylinder
30 r_o = d/2;// [m] radius of the cylinder
31 R = r/r_o;
32 C = k/(h*r_o);
33 D = (alpha*t/r_o^(2));
34 // and from figures (4-8) and (4-11), respectively
35 theta_o_by_i_cyl = 0.38;
36 theta_by_o = 0.98;
37 // so that
38 theta_by_i_cyl = theta_o_by_i_cyl*theta_by_o;
39 // combining the solutions for the plate and
    cylinder gives
40 theta_by_i_short_cyl = theta_by_i_plate*
    theta_by_i_cyl;
41 // thus
42 T = Te+theta_by_i_short_cyl*(Ti-Te);
43 printf("the temperature at a radial position of
    0.0125 m and a distance of 0.00625m from one end
    of cylinder 60 second after exposure to
    environment is %f degree celsius",T);

```

---

**Scilab code Exa 4.10** heat loss for finite length cylinder

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 4.10\n\n\n");
4 // heat loss for finite-length cylinder
5 // illustration4.10
6 // solution
7
8 d = 0.05;// [m] diameter of aluminium cylinder
9 l = 0.1;// [m] length of aluminium cylinder
10 Ti = 200;// [degree celsius] initial temperature of
    of cylinder

```

```

11 Te = 70; // [degree celsius] environment temperature
12 k = 215; // [W/m degree celsius] heat transfer
    coefficient of plate
13 h = 525; // [W/square meter degree celsius]
    convection heat transfer coefficient
14 alpha = 8.4*10^(-5); // [square meter/s] constant
15 x1 = 0.00625; // [m] distance at which temperature is
    calculated from end
16 t = 60; // [s] time after which temperature is
    measured
17 r = 0.0125; // [m] radius at which temperature is
    calculated
18 // we first calculate the dimensionless heat-loss
    ratio for the infinite plate and infinite
    cylinder which make up the multidimensional body
19 // for the plate we have
20 L = 0.05; // [m]
21 A = h*L/k;
22 B = h^(2)*alpha*t/k^(2);
23 // from figure (4-14), for the plate, we read
24 Q_by_Q_o_plate = 0.22;
25 // for the cylinder
26 r_o = 0.025; // [m]
27 // so we calculate
28 C = h*r_o/k;
29 // and from figure(4-15) we have
30 Q_by_Q_o_cyl = 0.55;
31 // the two heat ratios may be inserted in equation
    (4-22) to give
32 Q_by_Q_o_tot = Q_by_Q_o_plate+Q_by_Q_o_cyl*(1-
    Q_by_Q_o_plate);
33 c = 896; // [J/kg degree celsius] specific heat of
    aluminium
34 rho = 2707; // [kg/cubic meter] density of aluminium
35 V = %pi*r_o^(2)*l; // [cubic meter]
36 Qo = rho*c*V*(Ti-Te); // [J]
37 Q = Qo*Q_by_Q_o_tot; // [J] the actual heat loss in
    the 1-minute

```

```
38 printf(" the actual heat loss in the 1-minute is %f
    kJ",Q/1000);
```

---

#### Scilab code Exa 4.12 implicit formulation

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 4.12\n\n\n");
4 // implicit formulation
5 // Example 4.12 (page no.-173-174)
6 // solution
7
8 // we are using the data of example 4.11 for this
  question
9 // we are inserting the value of Rij in equation
  (4-43) to write the nodal equations for the end
  of the first time increment, taking all  $T1^{(p)} =$ 
  200 degree celsius
10 // we use underscore to designate the temperatures
  at the end of the time increment. for node 1
11 //  $0.05302 * T1\_ = 200/70.731 + T2\_$ 
   $/70.731 + 40/84.833 + 0.01296 * 200$ 
12 // for node 2
13 //  $0.05302 * T2\_ = T1\_/70.731 + T3\_$ 
   $/70.731 + 40/84.833 + 0.01296 * 200$ 
14 // for node 3 and 4,
15 //  $0.05302 * T3\_ = T2\_/70.731 + T4\_$ 
   $/70.731 + 40/84.833 + 0.01296 * 200$ 
16 //  $0.02686 * T4\_ = T3\_$ 
   $/70.731 + 40/2829 + 40/169.77 + 0.00648 * 200$ 
17 // these equations can then be reduced to
18 //  $0.05302 * T1\_ - 0.01414 * T2\_ = 5.8911$ 
19 //  $-0.01414 * T1\_ + 0.05302 * T2\_ - 0.01414 * T3\_ = 3.0635$ 
20 //  $-0.01414 * T2\_ + 0.05302 * T3\_ - 0.01414 * T4\_ = 3.0635$ 
21 //  $-0.01414 * T3\_ + 0.02686 * T4\_ = 1.5457$ 
```

```

22 // These equations can be solved by matrix method
23 Z = [0.05302 -0.01414 0 0;-0.01414 0.05302 -0.01414
      0;0 -0.01414 0.05302 -0.01414;0 0 -0.01414
      0.02686];
24 C = [5.8911;3.0635;3.0635;1.5457];
25 T_ = Z^(-1)*C;
26 T1_ = T_(1);// [degree celsius]
27 T2_ = T_(2);// [degree celsius]
28 T3_ = T_(3);// [degree celsius]
29 T4_ = T_(4);// [degree celsius]
30 // we can now apply the backward-difference
    formulation a second time using the double
    underscore to designate the temperatures at the
    end of the second time increment:
31 // 0.05302*T1__ = 200/70.731+T2__
    /70.731+40/84.833+0.01296*145.81
32 // 0.05302*T2__ = T1__/70.731+T3__
    /70.731+40/84.833+0.01296*130.12
33 // 0.05302*T3__ = T2__/70.731+T4__
    /70.731+40/84.833+0.01296*125.43
34 // 0.02686*T4__ = T3__
    /70.731+40/2829+40/169.77+0.00648*123.56
35 // These equations can be solved by matrix method
36 X = [0.05302 -0.01414 0 0;-0.01414 0.05302 -0.01414
      0;0 -0.01414 0.05302 -0.01414;0 0 -0.01414
      0.02686];
37 V = [5.1888;2.1578;2.0970;1.0504];
38 T__ = X^(-1)*V;
39 T1__ = T__(1);// [degree celsius]
40 T2__ = T__(2);// [degree celsius]
41 T3__ = T__(3);// [degree celsius]
42 T4__ = T__(4);// [degree celsius]
43 printf(" temperatures after time increment 1 are:");
44 printf("\n\n\t\t T1'' == %f",T1_);
45 printf("\n\n\t\t T2'' == %f",T2_);
46 printf("\n\n\t\t T3'' == %f",T3_);
47 printf("\n\n\t\t T4'' == %f",T4_);
48 printf("\n\n temperatures after time increment 2 are

```

```
        :");  
49 printf("\n\n\t\t T1' ' ' == %f", T1__);  
50 printf("\n\n\t\t T2' ' ' == %f", T2__);  
51 printf("\n\n\t\t T3' ' ' == %f", T3__);  
52 printf("\n\n\t\t T4' ' ' == %f", T4__);
```

---



# Chapter 5

## Principles of Convection

Scilab code Exa 5.1 water flow in a diffuser

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 5.1\n\n\n");
4 // water flow in a diffuser
5 // illustration5.1
6 // solution
7
8 Tw = 20; // [degree celcius] water temperature
9 m_dot = 8; // [kg/s] water flow rate
10 d1 = 0.03; // [m] diameter at section 1
11 d2 = 0.07; // [m] diameter at section 2
12 A1 = %pi*d1^(2)/4; // [square meter] cross-sectional
    area at section 1
13 A2 = %pi*d2^(2)/4; // [square meter] cross-sectional
    area at section 2
14 gc = 1; // [m/s^(2)] acceleration due to gravity
15 rho = 1000; // [kg/cubic meter] density of water at
    20 degree celcius
16 // we may calculate the velocities from the mass-
    continuity relation
17 u1 = m_dot/(rho*A1); // [m/s]
```

```

18 u2 = m_dot/(rho*A2); // [m/s]
19 // the pressure difference is obtained by Bernoulli
    equation(5-7a)
20 p2_minus_p1 = rho*(u1^(2)-u2^(2))/(2*gc); // [Pa]
21 printf("the increase in static pressure between
    sections 1 and 2 is %f kPa",p2_minus_p1/1000);

```

---

### Scilab code Exa 5.2 isentropic expansion of air

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 5.2\n\n\n");
4 // isentropic expansion of air
5 // illustration5.2
6 // solution
7
8 Ta = 300+273.15; // [K] air temperature
9 Pa = 0.7; // [MPa] pressure of air
10 u2 = 300; // [m/s] final velocity
11 gc = 1; // [m/s^(2)] acceleration due to gravity
12 Y = 1.4; // gama value for air
13 Cp = 1005; // [J/kg degree celsius]
14 // the initial velocity is small and the process is
    adiabatic. in terms of temperature
15 T2 = Ta-u2^(2)/(2*gc*Cp);
16 printf("the static temperature is %f K",T2);
17 // we may calculate the pressure difference from the
    isentropic relation
18 p2 = Pa*((T2/Ta)^(Y/(Y-1)));
19 printf("\n\n static pressure is %f MPa",p2);
20 // the velocity of sound at condition 2 is
21 a2 = 20.045*T2^(1/2); // [m/s]
22 // so that the mach no. is
23 M2 = u2/a2;
24 printf("\n\n Mach number is %f",M2);

```

---

**Scilab code Exa 5.3** mass flow and boundary layer thickness

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 5.3\n\n\n");
4 // mass flow and boundary-layer thickness
5 // illustration 5.3
6 // solution
7
8 Ta = 27+273.15; // [K] air temperature
9 Pa = 101325; // [Pa] pressure of air
10 u = 2; // [m/s] air velocity
11 x1 = 0.2; // [m] distance from the leading edge of
    plate
12 x2 = 0.4; // [m] distance from the leading edge of
    plate
13 R = 287; // []
14 mu = 1.85*10^(-5); // [kg/m s] viscosity of air
15 // the density of air is calculated from
16 rho = Pa/(R*Ta); // [kg/cubic meter]
17 // the reynolds number is calculated as
18 Re_x1 = rho*u*x1/mu;
19 Re_x2 = rho*u*x2/mu;
20 // the boundary layer thickness is calculated from
    equation(5-21)
21 del_x1 = 4.64*x1/Re_x1^(1/2); // [m]
22 del_x2 = 4.64*x2/Re_x2^(1/2); // [m]
23 // to calculate the mass flow which enters the
    boundary layer from the free stream between x =
    0.2 m and x = 0.4 m
24 // we simply take the difference between the mass
    flow in the boundary layer at these two x
    positions.
25 // at any x position the mass flow in the boundary
```

```

    layer is given by the integral dm = integrate('
rho*u_y', 'y', 0, del);
26 // velocity is given by equation (5-19) u_y = u
    *[1.5*(y/del) - 0.5*(y/del)^(3)]
27 // after integration we get dm = 5*rho*u*del/8
28 dm = 5*rho*u*(del_x2-del_x1)/8
29 printf(" mass flow entering the boundary layer is %e
    kg/s", dm);

```

---

#### Scilab code Exa 5.4 isothermal flat plate heated over entire length

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 5.4\n\n\n");
4 // isothermal flat plate heated over entire length
5 // illustration5.4
6 // solution
7
8 // total heat transfer over a certain length of the
    plate is desired, so we wish to calculate average
    heat transfer coefficients.
9 // for this purpose we use equations (5-44) and
    (5-45), evaluating the properties at the film
    temperature :
10 Tp = 60+273.15; // [K] plate temperature
11 Ta = 27+273.15; // [K] air temperature
12 Tf = (Tp+Ta)/2; // [K]
13 u = 2; // [m/s] air velocity
14 // from appendix A the properties are
15 v = 17.36*10^(-6); // [square meter/s] kinematic
    viscosity
16 x1 = 0.2; // [m] distance from the leading edge of
    plate
17 x2 = 0.4; // [m] distance from the leading edge of
    plate

```

```

18 k = 0.02749; // [W/m K] heat transfer coefficient
19 Pr = 0.7; // prandtl number
20 Cp = 1006; // [J/kg K]
21 // at x = 0.2m
22 Re_x1 = u*x1/v; // reynolds number
23 Nu_x1 = 0.332*Re_x1^(1/2)*Pr^(1/3); // nusselt number
24 hx1 = Nu_x1*k/x1; // [W/square meter K]
25 // the average value of the heat transfer
    coefficient is twice this value , or
26 h_bar1 = 2*hx1; // [W/square meter K]
27 // the heat flow is
28 A1 = x1*1; // [square meter] area for unit depth
29 q1 = h_bar1*A1*(Tp-Ta); // [W]
30 // at x = 0.4m
31 Re_x2 = u*x2/v; // reynolds number
32 Nu_x2 = 0.332*Re_x2^(1/2)*Pr^(1/3); // nusselt number
33 hx2 = Nu_x2*k/x2; // [W/square meter K]
34 // the average value of the heat transfer
    coefficient is twice this value , or
35 h_bar2 = 2*hx2; // [W/square meter K]
36 // the heat flow is
37 A2 = x2*1; // [square meter] area for unit depth
38 q2 = h_bar2*A2*(Tp-Ta); // [W]
39 printf("the heat transferred in first case of the
    plate is %f W",q1);
40 printf("\n\n and the heat transferred in second case
    of the plate is %f W",q2);

```

---

**Scilab code Exa 5.5** flat plate with constant heat flux

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 5.5\n\n\n");
4 // flat plate with constant heat flux
5 // illustration5.5

```

```

6 // solution
7
8 u = 5; // [m/s] air velocity
9 l = 0.6; // [m] plate length
10 Ta = 27+273.15; // [K] temperature of airstream
11 // properties should be evaluated at the film
    temperature, but we do not know the plate
    temperature so for an initial calculation we take
    the properties at the free-stream conditions of
12 v = 15.69*10(-6); // [square meter/s] kinematic
    viscosity
13 k = 0.02624; // [W/m degree celsius] heat transfer
    coefficient
14 Pr = 0.7; // prandtl number
15 Re_l = l*u/v; // reynolds number
16 P = 1000; // [W] power of heater
17 qw = P/l(2); // [W/square meter] heat flux per unit
    area
18 // from equation (5-50) the average temperature
    difference is
19 Tw_minus_Tinf_bar = qw*l/(0.6795*k*(Re_l)(1/2)*(Pr)
    (1/3)); // [degree celsius]
20 // now, we go back and evaluate properties at
21 Tf = (Tw_minus_Tinf_bar+Ta+Ta)/2; // [degree celsius]
22 // and obtain
23 v1 = 28.22*10(-6); // [square meter/s] kinematic
    viscosity
24 k1 = 0.035; // [W/m degree celsius] heat transfer
    coefficient
25 Pr1 = 0.687; // prandtl number
26 Re_l1 = l*u/v1; // reynolds number
27 Tw_minus_Tinf_bar1 = qw*l/(0.6795*k1*(Re_l1)(1/2)*(
    Pr1)(1/3)); // [degree celsius]
28 // at the end of the plate (x = l = 0.6m) the
    temperature difference is obtained from equation
    (5-48) and (5-50) with the constant of 0.453
29 Tw_minus_Tinf_x_equal_l = Tw_minus_Tinf_bar1
    *0.6795/0.453; // [degree celsius]

```

```

30 printf("average temperature difference along the
    plate is %f degree celsius",Tw_minus_Tinf_bar);
31 printf("\n\n temperature difference at the trailing
    edge is %f degree celsius",
    Tw_minus_Tinf_x_equal_l);

```

---

**Scilab code Exa 5.6** plate with unheated starting length

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 5.6\n\n\n");
4 // plate with unheated starting length
5 // illustration5.6
6 // solution
7
8 u = 20; // [m/s] air velocity
9 l = 0.2; // [m] plate length as well as width (square
    )
10 p = 101325; // [Pa] air pressure
11 Ta = 300; // [K] temperature of airstream
12 Tw = 350; // [K] temperature of last half of plate
13 // First we evaluate the air properties at the film
    temperature
14 Tf = (Tw+Ta)/2; // [K]
15 // and obtain
16 v = 18.23*10(-6); // [square meter/s] kinematic
    viscosity
17 k = 0.02814; // [W/m degree celsius] heat transfer
    coefficient
18 Pr = 0.7; // prandtl number
19 // at the trailing edge of the plate the reynolds
    number is
20 Re_l = l*u/v; // reynolds number
21 // or laminar flow over the length of the plate
22 // heating does not start until the last half of the

```

```

    plate, or at position xo = 0.1m.
23 xo = 0.1; // [m]
24 // the local heat transfer coefficient is given by
    equation (5-41)
25 //  $h_x = 0.332 * k * Pr^{(1/3)} * (u/v)^{(1/2)} * x^{(-1/2)} * [1 - (x_0/x)^{(0.75)}]^{(-1/3)}$ ;
26 // the plate is 0.2 m wide so the heat transfer is
    obtained by integrating over the heated length  $x_0 < x < l$ 
27 q = 1*(Tw-Ta)*integrate(' (0.332*k*Pr^(1/3)*(u/v)
    ^ (1/2)*x^(-1/2)*[1-(xo/x)^(0.75)]^(-1/3)) ', 'x', xo
    ,l);
28 printf("the heat lost by the plate is %f W",q);
29 // the average value of the heat transfer
    coefficient over the heated length is given by
30 h = q*(Tw-Ta)*(1-xo)*l; // [W/square meter degree
    celsius]
31 printf("\n\n the average value of heat transfer
    coefficient over the heated length is given by %f
    W/square meter degree celsius",h);

```

---

### Scilab code Exa 5.7 oil flow over heated flat plate

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 5.7\n\n\n");
4 // oil flow over heated flat plate
5 // illustration5.7
6 // solution
7
8 u = 1.2; // [m/s] oil velocity
9 l = 0.2; // [m] plate length as well as width (square
    )
10 To = 20+273.15; // [K] temperature of engine oil
11 Tu = 60+273.15; // [K] uniform temperature of plate

```



```

12 // First we evaluate the film temperature
13 T = (To+Tu)/2; // [K]
14 // and obtain the properties of engine oil are
15 rho = 876; // [kg/cubic meter] density of oil
16 v = 0.00024; // [square meter/s] kinematic viscosity
17 k = 0.144; // [W/m degree celsius] heat transfer
    coefficient
18 Pr = 2870; // prandtl number
19 // at the trailing edge of the plate the reynolds
    number is
20 Re = l*u/v; // reynolds number
21 // because the prandtl no. is so large we will
    employ equation(5-51) for the solution.
22 // we see that hx varies with x in the same fashion
    as in equation(5-44) , i.e. hx is inversely
    proportional to the square root of x ,
23 // so that we get the same solution as in equation
    (5-45) for the average heat transfer coefficient.
24 // evaluating equation(5-51) at x = 0.2m gives
25 Nux = 0.3387*Re^(1/2)*Pr^(1/3)/[1+(0.0468/Pr)^(2/3)
    ]^(1/4);
26 hx = Nux*k/l; // [W/square meter degree celsius]
    heat transfer coefficient
27 // the average value of the convection coefficient
    is
28 h = 2*hx; // [W/square meter degree celsius]
29 // so that total heat transfer is
30 A = l^(2); // [square meter] area of the plate
31 q = h*A*(Tu-To); // [W]
32 printf("average value of the convection coefficient
    is %f W/square meter degree celsius",h);
33 printf("\n\n and the heat lost by the plate is %f W"
    ,q);

```

---

Scilab code Exa 5.8 drag force on a flat plate

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 5.8\n\n\n");
4 // drag force on a flat plate
5 // illustration 5.8
6 // solution
7
8 // data is used from example 5.4
9 // we use equation (5-56) to compute the friction
   coefficient and then calculate the drag force .
10 // an average friction coefficient is desired, so
   st_bar*pr^(2/3) = Cf_bar/2
11 p = 101325; // [Pa] pressure of air
12 x = 0.4; // [m] drag force is computed on first 0.4 m
   of the plate
13 R = 287; // []
14 Tf = 316.5; // [K]
15 u = 2; // [m/s] air velocity
16 Cp = 1006; // [J/kg K]
17 Pr = 0.7; // prandtl no.
18 rho = p/(R*Tf); // [kg/cubic meter] density at 316.5
   K
19 h_bar = 8.698; // [W/square meter K] heat transfer
   coefficient
20 // for the 0.4m length
21 st_bar = h_bar/(rho*Cp*u);
22 // then from equation (5-56)
23 Cf_bar = st_bar*Pr^(2/3)*2;
24 // the average shear stress at the wall is computed
   from equation(5-52)
25 tau_w_bar = Cf_bar*rho*u^(2)/2; // [N/square meter]
26 A = x*1; // [square meter] area per unit length
27 // the drag force is the product of this shear
   stress and the area ,
28 D = tau_w_bar*A; // [N]
29 printf("Drag force exerted on the first 0.4 m of the
   plate is %f mN",D*1000);

```

---

**Scilab code Exa 5.9** turbulent heat transfer from isothermal flat plate

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 5.9\n\n");
4 // turbulent heat transfer from isothermal flat
   plate
5 // illustration5.9
6 // solution
7
8 p = 101325; // [Pa] pressure of air
9 R = 287; // []
10 Ta = 20+273.15; // [K] temperature of air
11 u = 35; // [m/s] air velocity
12 L = 0.75; // [m] length of plate
13 Tp = 60+273.15; // [K] plate temperature
14 // we evaluate properties at the film temperature
15 Tf = (Ta+Tp)/2; // [K]
16 rho = p/(R*Tf); // [kg/cubic meter]
17 mu = 1.906*10^(-5); // [kg/m s] viscosity
18 k = 0.02723; // [W/m degree celsius]
19 Cp = 1007; // [J/kg K]
20 Pr = 0.7; // prandtl no.
21 // the reynolds number is
22 Rel = rho*u*L/mu;
23 // and the boundary layer is turbulent because the
   reynolds number is greater than 5*10^(5).
24 // therefore, we use equation(5-85) to calculate
   the average heat transfer over the plate:
25 Nul_bar = Pr^(1/3)*(0.037*Rel^(0.8)-871);
26 A = L*1; // [square meter] area of plate per unit
   depth
27 h_bar = Nul_bar*k/L; // [W/square meter degree
   celsius]
```

```
28 q = h_bar*A*(Tp-Ta); // [W] heat transfer from plate
29 printf("heat transfer from plate is %f W",q);
```

---

**Scilab code Exa 5.10** turbulent boundary layer thickness

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 5.10\n\n\n");
4 // turbulent-boundary-layer thickness
5 // illustration 5.10
6 // solution
7
8 // we have to use the data from example 5.8 and 5.9
9 Re1 = 1.553*10^6; // from previous example
10 L = 0.75; // [m] length of plate
11 // it is a simple matter to insert this value in
    equations(5-91) and (5-95) along with
12 x = L; // [m]
13 // turbulent-boundary-layer thickness are
14 // part a. from the leading edge of the plate
15 del_a = x*0.381*Re1^(-0.2); // [m]
16 // part b from the transition point at Recrit =
    5*10^(5)
17 del_b = x*0.381*Re1^(-0.2) - 10256*Re1^(-1); // [m]
18 printf("turbulent-boundary-layer thickness at the
    end of the plate from the leading edge of the
    plate is %f mm",del_a*1000);
19 printf("\n\n turbulent-boundary-layer thickness at
    the end of the plate from the transition point at
    Re_crit = 5*10^(5) is %f mm",del_b*1000);
```

---

**Scilab code Exa 5.11** high speed heat transfer for a flat plate

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 5.11\n\n\n");
4 // high speed heat transfer for a flat plate
5 // Example 5.11 (page no.-257-259)
6 // solution
7
8 L = 0.7; // [m] length of flat plate
9 W = 1; // [m] width of plate
10 // flow conditions are
11 M = 3;
12 p = 101325/20; // [Pa]
13 T = -40+273; // [degree celsius]
14 Tw = 35; // [degree celsius] temperature at which
    plate is maintained
15 Gamma = 1.4;
16 g_c = 1; // []
17 R = 287; // [] universal gas constant
18 // we have to consider laminar and turbulent
    portions of the boundary layer seperately
19 // the free-stream acoustic velocity is calculated
    from
20 a = sqrt(Gamma*g_c*R*T); // [m/s]
21 // so that free stream velocity is
22 u = M*a; // [m/s]
23 // the maximum reynolds number is estimated by
    making a computation based on properties
    evaluated at free stream conditions:
24 rho = p/(R*T); // [Kg/m^(3)]
25 mu = 1.434*10^(-5); // [Kg/m s]
26 Re_L = rho*u*L/mu;
27 // thus we conclude that both laminar and turbulent
    boundary layer heat transfer must be considered.
28 // we first determine the reference temperature for
    the two regimes and then evaluate properties at
    these temperatures.
29
30 // LAMINAR PORTION

```

```

31
32 T_o = T*(1+((Gamma-1)/2)*M^(2)); // [K]
33 Pr = 0.7 // prandtl number(assuming)
34 // we have
35 r = sqrt(Pr);
36 T_aw = r*(T_o-T)+T; // [K]
37 // then the reference temperature from equation
    (5-124) is
38 T_star = T+0.5*(Tw-(T-273))+0.22*(T_aw-T); // [K]
39 // checking the prandtl number at this temperature
40 Pr_star = 0.697;
41 // so that the calculation is valid.because Pr_star
    and the value of Pr used to determine the
    recovery factor are almost same
42 // the other properties to be used in the laminar
    heat transfer analysis are
43 rho_star = p/(R*T_star); // [Kg/m^(3)]
44 mu_star = 2.07*10^(-5); // [Kg/m s]
45 k_star = 0.03; // [W/m degree celsius]
46 Cp_star = 1009; // [J/Kg degree celsius]
47
48 // TURBULENT PORTION
49
50 // Assuming
51 Pr = 0.7;
52 r = Pr^(1/3);
53 T_aw1 = r*(T_o-T)+T; // [K]
54 // then the reference temperature from equation
    (5-124) is
55 T_star = T+0.5*(Tw-(T-273))+0.22*(T_aw-T); // [K]
56 // checking the prandtl number at this temperature
57 Pr_star1 = 0.695;
58 // the agreement between Pr_star and the assumed
    value is sufficiently close.
59 // the other properties to be used in the turbulent
    heat transfer analysis are
60 rho_star1 = p/(R*T_star); // [Kg/m^(3)]
61 mu_star1 = 2.09*10^(-5); // [Kg/m s]

```

```

62 k_star1 = 0.0302; // [W/m degree celsius]
63 Cp_star1 = 1009; // [J/Kg degree celsius]
64
65 // LAMINAR HEAT TRANSFER
66
67 // we assume
68 Re_star_crit = 5*10^(5);
69 x_c = Re_star_crit*mu_star/(rho_star*u); // [m]
70 Nu_bar = 0.664*(Re_star_crit)^(1/2)*(Pr_star)^(1/3);
71 h_bar = Nu_bar*k_star/x_c; // [W/m^(2) degree celsius
    ] average heat transfer coefficient
72 // heat transfer is calculated as
73 A = x_c*1; // [m^(2)]
74 q = h_bar*A*(Tw-(T_aw-273)); // [W]
75
76 // TURBULENT HEAT TRANSFER
77
78 // to determine the turbulent heat transfer we must
    obtain an expression for the local heat transfer
    coefficient from
79 //  $St_x * Pr_{star1}^{2/3} = 0.0296 * Re_{star_x}^{-1/5}$ 
80 // and then integrate from  $x = 0.222m$  to  $x = 0.7m$  to
    determine the total heat transfer
81 h_x = integrate('Pr_star1^(-2/3)*rho_star1*u*
    Cp_star1*0.0296*(rho_star1*u*x/mu_star1)^(-1/5)',
    'x',0.222,0.7); // [W/m^(2) degree celsius]
82 // the average heat transfer coefficient in the
    turbulent region is determined from integral h_x
    dx divided by integral dx limit from 0.222 to 0.7
83 h_bar1 = h_x/(integrate('1', 'x',0.222,0.7)); // [W/m
    ^ (2) degree celsius]
84 // using this value we may calculate the heat
    transfer in the turbulent region of the flat
    plate:
85 A1 = (L-0.222); // [m^(2)]
86 q1 = h_bar1*A1*(Tw-(T_aw1-273)); // W
87
88 // the total amount of cooling required is the sum

```

```
    of the heat transfers for the laminar and
    turbulent portions
89 Total_cooling = abs(q)+abs(q1);// [W]
90 printf("the total amount of cooling required is the
    sum of the heat transfers for the laminar and
    turbulent portions is %f W",Total_cooling);
```

---



## Chapter 6

# Empirical and Practical Relations for Forced Convection Heat Transfer

Scilab code Exa 6.1 turbulent heat transfer in a tube

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 6.1\n\n");
4 // turbulent heat transfer in a tube
5 // illustration6.1
6 // solution
7
8 p = 2*101325; // [Pa] pressure of air
9 Ta = 200+273.15; // [K] temperature of air
10 d = 0.0254; // [m] diameter of tube
11 R = 287; // [J/kgK] gas constant
12 u = 10; // [m/s] velocity of air
13 dT = 20; // [degree celsius] temperature difference
    between wall and air
14 // we first calculate the reynolds number to
    determine if the flow is laminar or turbulent ,
    and then select the appropriate empirical
```

```

correlation to calculate the heat transfer
15 // the properties of air at a bulk temperature of
    473 K are
16 rho = p/(R*Ta); // [kg/cubic meter] density of gas
17 mu = 2.57*10^(-5); // [kg/m s] viscosity
18 k = 0.0386; // [W/m degree celsius]
19 Cp = 1025; // [J/kg K]
20 Pr = 0.681; // prandtl no.
21 Re_d = rho*u*d/mu; // reynolds number
22 disp(Re_d, "reynolds number is" );
23 disp("so that the flow is turbulent");
24 // we therefore use equation (6-4a) to calculate the
    heat transfer coefficient
25 Nu_d = 0.023*Re_d^(0.8)*Pr^(0.4); // nusselt no.
26 h = Nu_d*k/d; // [W/m^2 degree celsius] heat transfer
    coefficient
27 // the heat transfer per unit length is then
28 q_by_L = h*pi*d*(dT); // [W/m]
29 L = 3; // [m]
30 // we can now make an energy balance to calculate
    the increase in bulk temperature in a 3 m length
    of tube :
31 // q = m_dot*Cp*dT_b = L*(q_byL)
32 m_dot = rho*u*pi*d^(2)/4; // [kg/s] gas flow rate
33 // so that we insert the numerical values in the
    energy balance to obtain
34 dT_b = L*q_by_L/(m_dot*Cp); // [degree celsius]
35 printf("\n heat transfer per unit length is %f W/m",
    q_by_L);
36 printf("\n\n bulk temperature increase over the
    length of 3 m on tube is %f degree celsius ",dT_b
    );

```

---

Scilab code Exa 6.2 heating of water in laminar tube flow

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 6.2\n\n\n");
4 // heating of water in laminar tube flow
5 // illustration6.2
6 // solution
7
8 Tw = 60; // [degree celsius] temperature of water
9 d = 0.0254; // [m] diameter of tube
10 R = 287; // [] gas constant
11 u = 0.02; // [m/s] velocity of water
12 Tw = 80; // [degree celsius] temperature of wall
13 L = 3; // [m] length of the tube
14 // we first calculate the reynolds number at the
    inlet bulk temperature to determine the flow
    regime
15 // the properties of water at temperature of 333.15
    K are
16 rho = 985; // [kg/cubic meter] density of gas
17 mu = 4.71*10^(-4); // [kg/m s] viscosity
18 k = 0.651; // [W/m degree celsius]
19 Cp = 4.18*10^3; // [J/kg K]
20 Pr = 3.02; // prandtl no.
21 Re_d = rho*u*d/mu; // reynolds number
22 disp(Re_d, "reynolds number is" );
23 disp("so that the flow is laminar");
24 // so the flow is laminar, calculating the
    additional parameter, we have
25 B = Re_d*Pr*d/L ;
26 // since the value of B is greater than 10, so
    equation(6-10) is applicable.
27 // firstly making the calculation on the basis of 60
    degree celsius, determine the exit bulk
    temperature
28 // the energy balance becomes  $q = h \cdot \pi \cdot d \cdot L \cdot (T_w - (T_{b1} + T_{b2}) / 2) = \dot{m} \cdot C_p \cdot (T_{b2} - T_{b1})$  say equation a
29 // at the wall temperature of 80 degree celsius
30 mu_w = 3.55*10^(-4); // [kg/m s]

```

```

31 // from equation (6-10)
32 Nu_d = 1.86*(B)^(1/3)*(mu/mu_w)^(0.14);
33 h = k*Nu_d/d;
34 // the mass flow rate is
35 m_dot = rho*pi*d^(2)*u/4; // [kg/s]
36 // inserting the values in equation a
37 Tb1 = 60; // [degree celsius]
38 deff(' [y] = f(Tb2)', 'y = (h*pi*d*L*(Tw-(Tb1+Tb2)/2)
      -m_dot*Cp*(Tb2-Tb1))')
39 Tb2 = fsolve(1,f);
40 Tb_mean = (Tb1+Tb2)/2; // [degree celsius]
41 // we obtain the properties at Tb_mean
42 rho1 = 982; // [kg/m^3] density of gas
43 mu1 = 4.36*10^(-4); // [kg/m s] viscosity
44 k1 = 0.656; // [W/m degree celsius]
45 Cp1 = 4.185*10^3; // [J/kg K]
46 Pr1 = 2.78; // prandtl no.
47 Re_d1 = Re_d*mu/mu1;
48 C = Re_d1*Pr1*d/L ;
49 Nu_d1 = 1.86*(C)^(1/3)*(mu1/mu_w)^(0.14);
50 h = k1*Nu_d1/d;
51 // we insert this value of h back into equation a to
    get
52 deff(' [y] = f(Tb2)', 'y = (h*pi*d*L*(Tw-(Tb1+Tb2)/2)
      -m_dot*Cp*(Tb2-Tb1))')
53 Tb2 = fsolve(1,f);
54 printf("\n the exit water temperature is %f degree
      celsius", Tb2);

```

---

**Scilab code Exa 6.3** heating of air in laminar tube flow for constant heat flux

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 6.3\n\n\n");

```

```

4 // heating of air in laminar tube flow for constant
   heat flux
5 // illustration6.3
6 // solution
7
8 p = 101325; // [Pa] pressure of air
9 Ta = 27; // [degree celsius] temperature of air
10 d = 0.005; // [m] diameter of tube
11 R = 287; // [] gas constant
12 u = 3; // [m/s] velocity of air
13 L = 0.1; // [m] length of tube
14 Tb = 77; // [degree celsius] exit bulk temperature
15 // we first must evaluate the flow regime and do so
   by taking properties at the average bulk
   temperature
16 Tb_bar = (Ta+Tb)/2; // [degree celsius]
17 v = 18.22*10(-6); // [square meter/s] kinematic
   viscosity
18 k = 0.02814; // [W/m degree celsius]
19 Cp = 1006; // [J/kg K]
20 Pr = 0.703; // prandtl no.
21 Re_d = u*d/v; // reynolds number
22 disp(Re_d, "reynolds number is" );
23 disp("so that the flow is laminar");
24 // so that the flow is laminar
25 //the tube length is short, so we expect a thermal
   entrance effect and shall consult figure(6-5)
26 // the inverse Graetz number is computed as
27 Gz_inverse = L/(Re_d*Pr*d);
28 // therefore, for qw = constant, we obtain the
   nusselt number at exit from figure (6-5) as
29 Nu = 4.7;
30 // the total heat transfer is obtained in terms of
   the overall energy balance
31 // at entrance
32 rho = 1.1774; // [kg/cubic meter] density
33 // mass flow is
34 m_dot = rho*pi*d(2)*u/4; // [kg/s]

```

```

35 q = m_dot*Cp*(Tb-Ta); // [W]
36 // thus we may find the heat transfer without the
    actually determining wall temperatures or values
    of h. However, to determine Tw we must compute qw
    for insertion in equation(b). we have
37 qw = q/(%pi*d*L); // [W]
38 // now
39 Tw = Tb+(qw*d/(Nu*k)); // [degree celsius]
40 // and the heat transfer coefficient is
41 h = qw/(Tw-Tb); // [W/square meter degree celsius]
42 printf("\n total heat transfer is %f W",q);
43 printf("\n\n exit wall temperature is %f degree
    celsius",Tw);
44 printf("\n\n heat transfer coefficient is %f W/
    square meter degree celsius",h);

```

---

#### Scilab code Exa 6.4 heating of air with isothermal tube wall

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 6.4\n\n\n");
4 // heating of air with isothermal tube wall
5 // illustration6.4
6 // solution
7
8 p = 101325; // [Pa] pressure of air
9 Ta = 27; // [degree celsius] temperature of air
10 d = 0.005; // [m] diameter of tube
11 R = 287; // [] gas constant
12 u = 3; // [m/s] velocity of air
13 L = 0.1; // [m] length of tube
14 Tb = 77; // [degree celsius] exit bulk temperature
15 // we first must evaluate the flow regime and do so
    by taking properties at the average bulk
    temperature

```

```

16 Tb_bar = (Ta+Tb)/2; // [degree celsius]
17 v = 18.22*10^(-6); // [square meter/s] kinematic
    viscosity
18 k = 0.02814; // [W/m degree celsius]
19 Cp = 1006; // [J/kg K]
20 Pr = 0.703; // prandtl no.
21 Re_d = u*d/v; // reynolds number
22 disp(Re_d, "reynolds number is" );
23 disp("so that the flow is laminar");
24 // so that the flow is laminar
25 // now we determine Nu_d_bar for Tw = constant. for
    Gz_inverse = 0.0346 we read
26 Nu_d = 5.15;
27 // we thus calculate the average heat transfer
    coefficient as
28 h_bar = Nu_d*k/d; // [W/square meter degree celsius]
29 // we base the heat transfer on a mean bulk
    temperature of Tb_bar, so that
30 Tw = 3.49/(h_bar*%pi*d*L)+Tb_bar; // [degree celsius]
31 printf("\n exit wall temperature is %f degree
    celsius",Tw);

```

---

### Scilab code Exa 6.5 heat transfer in a rough tube

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 6.5\n\n\n");
4 // heat transfer in a rough tube
5 // illustration6.5
6 // solution
7
8 Tw = 90; // [degree celsius] temperature of tube wall
9 d = 0.02; // [m] diameter of tube
10 u = 3; // [m/s] velocity of air
11 Tw_i = 40; // [degree celsius] entering water

```

```

    temperature
12 Tw_f = 60; // [degree celsius] leaving water
    temperature
13 Cp = 4.174*10^3; // [J/kg K]
14 Tb_bar = (Tw_i+Tw_f)/2; // [degree celsius]
15 // we first calculate the heat transfer from q =
    m_dot*Cp*dTb
16 q = 989*3*%pi*0.01^(2)*4174*(Tw_f-Tw_i); // [W]
17 // for the rough tube condition , we may employ the
    Petukhov relation , equation (6-7) The mean film
    temperaturee is
18 Tf = (Tw+Tb_bar)/2; // [degree celsius]
19 // and the fluid properties are
20 rho = 978; // [kg/cubic meter] density of gas
21 mu = 4.0*10^(-4); // [kg/m s] viscosity
22 k = 0.664; // [W/m degree celsius]
23 Pr = 2.54; // prandtl no.
24 // also
25 mu_b = 5.55*10^(-4); // [kg/m s] viscosity
26 mu_w = 2.81*10^(-4); // [kg/m s] viscosity
27 // the reynolds number is thus
28 Re_d = rho*u*d/mu;
29 // consulting figure(6-14), we find the friction
    factor as
30 f_f = 0.0218;
31 // because Tw>Tb, we take
32 n = 0.11;
33 // and obtain
34 Nu_d = ((f_f*Re_d*2.54)/((1.07+12.7*(f_f/8)^(0.5)
    *(2.54^(2/3)-1))*8))*(mu_b/mu_w)^(n);
35 h = Nu_d*k/d; // [W/square meter degree celsius]
36 // the tube length is obtained from energy balance
37 L = q/(h*%pi*d*(Tw-Tb_bar)); // [m]
38 printf("the length of tube necessary to accomplish
    the heating is %f m",L);

```

---



**Scilab code Exa 6.6** turbulent heat transfer in a short tube

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 6.6\n\n");
4 // turbulent heat transfer in a short tube
5 // illustration6.6
6 // solution
7
8 p = 101325; // [Pa] pressure of air
9 Ta = 300; // [K] temperature of air
10 d = 0.02; // [m] diameter of tube
11 u = 40; // [m/s] velocity of air
12 L = 0.1; // [m] length of tube
13 dT = 5; // [degree celsius] rise in temperature
14 // we first must evaluate the air properties at 300
    K
15 v = 15.69*10^(-6); // [square meter/s] kinematic
    viscosity
16 k = 0.02624; // [W/m degree celsius]
17 Cp = 1006; // [J/kg K]
18 Pr = 0.70; // prandtl no.
19 rho = 1.18; // [kg/cubic meter]
20 Re_d = u*d/v; // reynolds number
21 disp(Re_d,"reynolds number is ");
22 disp("so the flow is turbulent");
23 // consulting figure (6-6) for this value of Re_d
    and L/d = 5 we find
24 Nu_x_by_Nu_inf = 1.15;
25 // or the heat transfer coefficient is about 15
    percent higher that it would be for thermally
    developed flow.
26 // we calculate heat-transfer for developed flow
    using
```

```

27 Nu_d = 0.023*Re_d^(0.8)*Pr^(0.4);
28 // and
29 h = k*Nu_d/d;// [W/square meter degree celsius]
30 // increasing this value by 15 percent
31 h = 1.15*h;// [W/square meter degree celsius]
32 // the mass flow is
33 Ac = %pi*d^(2)/4;// [square meter]
34 m_dot = rho*u*Ac;// [kg/s]
35 // so the total heat transfer is
36 A = %pi*d*L;// [square meter]
37 q_by_A = m_dot*Cp*dT/A;// [W/square meter]
38 printf("\n\n the constant heat flux that must be
    applied at the tube surface to result in an air
    temperature rise of 5 degree celsius is %f W/
    square meter",q_by_A);
39 Tb_bar = (Ta+(Ta+dT))/2;// [K]
40 Tw_bar = Tb_bar+q_by_A/h;// [K]
41 printf("\n\n average wall temperature is %f K",
    Tw_bar);

```

---

### Scilab code Exa 6.7 airflow across isothermal cylinder

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 6.7\n\n\n");
4 // airflow across isothermal cylinder
5 // illustration6.7
6 // solution
7
8 p = 101325;// [Pa] pressure of air
9 Ta = 35+273.15;// [K] temperature of air
10 d = 0.05;// [m] diameter of tube
11 R = 287;// [] gas constant
12 u = 50;// [m/s] velocity of air
13 Tc = 150+273.15;// [degree celsius] cylinder

```

```

    temperature
14 // we first find the reynolds number and then find
    the applicable constants from table(6-2) for use
    with equation (6-17)
15 // the properties of air are evaluated at the film
    temperature:
16 Tf = (Ta+Tc)/2; // [K]
17 rho_f = p/(R*Tf); // [kg/cubic meter]
18 mu_f = 2.14*10^(-5); // [kg/m s]
19 k_f = 0.0312; // [W/m degree celsius]
20 Pr_f = 0.695; // prandtl number
21 Re_f = rho_f*u*d/mu_f; // reynolds number
22 // from table (6-2) table
23 C = 0.0266;
24 n = 0.805;
25 // so from equation (6-17)
26 h = C*(Re_f)^(n)*(Pr_f)^(1/3)*k_f/d; // [W/square
    meter degree celsius] heat transfer coefficient
27 // the heat transfer per unit length is
28 q_by_L = h*pi*d*(Tc-Ta); // [W/m]
29 printf("heat loss per unit length of cylinder is %f
    W/m", q_by_L);

```

---

**Scilab code Exa 6.8** heat transfer from electrically heated

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 6.8\n\n\n");
4 // heat transfer from electrically heated
5 // illustration6.8
6 // solution
7
8 p = 101325; // [Pa] pressure of air
9 Tw = 25+273.15; // [K] temperature of air
10 d = 3.94*10^(-5); // [m] diameter of wire

```

```

11 R = 287; // [] gas constant
12 u = 50; // [m/s] velocity of air perpendicular to the
    air
13 Tr = 50+273.15; // [degree celsius] rise in surface
    temperature
14 // we first obtain the properties at the film
    temperature :
15 Tf = (Tw+Tr)/2; // [K]
16 v_f = 16.7*10^(-6); // [square meter/s]
17 k = 0.02704; // [W/m degree celsius]
18 Pr_f = 0.706; // prandtl number
19 Re_d = u*d/v_f; // reynolds number
20 // the Peclet number is
21 Pe = Re_d*Pr_f;
22 // and we find that equations (6-17),(6-21), or
    (6-19) apply.
23 // let us make the calculation with both the
    simplest expression, (6-17),and the most complex
    ,(6-21), and compare results.
24 // using equation (6-17) with
25 C = 0.683;
26 n = 0.466;
27 // we have
28 Nu_d = 0.683*Re_d^(n)*Pr_f^(1/3);
29 // the value of heat transfer coefficient is
30 h = Nu_d*k/d; // [W/square meter degree celsius]
31 // the heat transfer per unit length is then
32 q_by_L = %pi*d*h*(Tr-Tw); // [W/m]
33 // using equation (6-21), we calculate the nusselt
    no as
34 Nu_d1 = 0.3+(((0.62*Re_d^(1/2)*Pr_f^(1/3))/((1+(0.4/
    Pr_f)^(2/3))^(1/4)))*((1+(Re_d/282000)^(5/8))
    ^((4/5)))));
35 h1 = Nu_d1*k/d; // [W/square meter degree celsius]
36 // and
37 q_by_L1 = h1*%pi*d*(Tr-Tw); // [W/m]
38 printf("heat lost per unit length by the wire is %f
    W/m",q_by_L1);

```

---

**Scilab code Exa 6.9** heat transfer from sphere

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 6.9\n\n\n");
4 // heat transfer from sphere
5 // illustration 6.9
6 // solution
7
8 p = 101325; // [Pa] pressure of air
9 Ta = 27+273.15; // [K] temperature of air
10 d = 0.012; // [m] diameter of sphere
11 u = 4; // [m/s] velocity of air
12 Ts = 77+273.15; // [degree celsius] surface
    temperature of sphere
13 // consulting equation (6-30) we find that the
    reynolds number is evaluated at the free-stream
    temperature.
14 // we therefore need the following properties at Ta
    = 300.15K
15 v = 15.69*10^(-6); // [square meter/s]
16 k = 0.02624; // [W/m degree celsius]
17 Pr = 0.708; // prandtl number
18 mu_inf = 1.8462*10^(-5); // [kg/m s]
19 // at Ts = 350K
20 mu_w = 2.075*10^(-5); // [kg/m s]
21 Re_d = u*d/v; // reynolds number
22 // from equation (6-30),
23 Nu_bar = 2+((0.4)*(Re_d)^(1/2)+0.06*(Re_d)^(2/3))*
    (Pr^(0.4))*((mu_inf/mu_w)^(1/4));
24 // and
25 h_bar = Nu_bar*k/d; // [W/square meter degree celsius
    ] heat transfer coefficient
26 // the heat transfer is then
```

```

27 A = 4*pi*d^(2)/4; // [square meter] area of sphere
28 q = h_bar*A*(Ts-Ta); // [W]
29 // for comparison purposes let us also calculate the
    heat-transfer coefficient using equation(6-25).
    the film temperature is
30 Tf = (Ta+Ts)/2; // [K]
31 v_f = 18.23*10^(-6); // [square meter/s]
32 k_f = 0.02814; // [W/m degree celsius]
33 // reynolds number is
34 Re_d1 = u*d/v_f;
35 // from equation (6-25)
36 Nu_f = 0.37*(u*d/v_f)^(0.6);
37 // and h_bar is calculated as
38 h_bar = Nu_f*k_f/d; // [W/square meter degree celsius
    ]
39 printf("heat lost by the sphere is %f W",q);

```

---

**Scilab code Exa 6.10** heating of air with in line tube bank

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 6.10\n\n\n");
4 // heating of air with in-line tube bank
5 // Example 6.10(page number-300-302)
6 // solution
7
8 p = 101325; // [Pa] pressure of air
9 Ta = 10+273.15; // [K] temperature of air
10 d = 0.0254; // [m] diameter of tubes
11 Sp = 0.0381; // [m] spacing between tubes in normal
    and parallel direction
12 Sn = Sp;
13 R = 287; // [] universal gas constant
14 u = 7; // [m/s] velocity of air
15 Ts = 65+273.15; // [K] surface temperature of tubes

```

```

16 // the constants for use with equation (6-17) may be
    obtained from table 6-4(page no.-298),using
17 Sp_by_d = Sp/d;
18 Sn_by_d = Sn/d;
19 // so that
20 C = 0.278;
21 n = 0.620;
22 // the properties of air are evaluated at the film
    temperature, which at entrance to the tube bank
    is
23 Tf = (Ta+Ts)/2;// [K]
24 rho_f = p/(R*Tf);// [kg/cubic meter]
25 mu_f = 1.894*10(-5);// [kg/m s]
26 k_f = 0.027;// [W/m degree celsius]
27 Pr_f = 0.706;// prandtl number
28 Cp = 1007;// [J/Kg degree celsius]
29 // the maximum velocity is therefore
30 u_max = u*Sn/(Sn-d);// [m/s]
31 // the reynolds number is computed by using the
    maximum velocity
32 Re = rho_f*u_max*d/mu_f;
33 // the heat transfer coefficient is calculated by
    using equation(6-17)
34 h = C*Re(n)*Pr_f(1/3)*k_f/d;// [W/square meter
    degree celsius]
35 // multiplying by 0.92 from table 6-5 (page no.-298)
    to correct for only five tube rows gives
36 h = 0.92*h;// [W/square meter degree celsius]
37 // the total surface area for heat transfer,
    considering unit length of tubes is
38 N = 15*5;// ttal no. of tubes
39 A = N*pi*d*1;// [square meter/m]
40 // before calculating the heat transfer, we must
    recognize that the air temperature increases as
    the air flows through the tube bank.
41 // therefore, this must be taken into account when
    using q=h*A*(Ts-Ta)
42 // as a good approximatoin, we can use an arithmetic

```

```

        average value of Tinf and write for the energy
        balance
43 // say the equation A is       $q = h \cdot A \cdot (T_s - (T_{inf1} +$ 
        Tinf2)/2) = m_dot * Cp * (Tinf2 - Tinf1)
44 // where now the subscripts 1 and 2 designate
        entrance and exit to the tube bank.
45 // the mass flow to the entrance to the 15 tubes is
46 rho_inf = p/(R * Ta); // [Kg/m^(3)]
47 m_dot = rho_inf * u * 15 * Sn; // [kg/s]
48 // so that equation A becomes after inserting the
        values and solving
49 Tinf1 = Ta; // [K]
50 deff(' [y] = f1(Tinf2)', 'y = (h * A * (Ts - (Tinf1 + Tinf2)
        /2) - m_dot * Cp * (Tinf2 - Tinf1))')
51 Tinf2 = fsolve(1, f1);
52 // the heat transfer is then obtained from the right
        side of equation A
53 q = m_dot * Cp * (Tinf2 - Ta); // [W/m]
54 printf("the exit air temperature is %f degree
        celsius", Tinf2 - 273);
55 printf("\n\n heat transfer per unit length for the
        tube bank is %f kW/m", q/1000);

```

---

### Scilab code Exa 6.11 alternate calculation method

```

1 clear;
2 clc;
3 printf("\t\t\t Example Number 6.11\n\n\n");
4 // alternate calculation method
5 // example 6.10 (page no. -302)
6 // solution
7
8 // data for this example is taken from previous
        example (6-10)
9 // properties for use in equation (6-34) are

```



```

        evaluated at free-atream conditions of 10 degree
        celsius
10 v = 14.2*10(-6); // [square meter/s]
11 k = 0.0249; // [W/m degree celsius]
12 Pr = 0.712; // prandtl number
13 Pr_w = 0.70; // prandtl number
14 u = 7; // [m/s] velocity of air
15 Sp = 0.0381; // [m] spacing between normal and
    parallel direction to the flow
16 Sn = 0.0381; // spacing between normal and parallel
    direction to the flow
17 d = 0.0254; // [m] diameter of tube
18 //maximum velocity is
19 u_max = u*(Sn/(Sn-d)); // [m/s]
20 // the reynolds number is
21 Re_d_max = u_max*d/v;
22 // so that the constants for equation (6-34) are
23 C = 0.27;
24 n = 0.63;
25 // inserting values we obtain
26 h = C*Re_d_max(n)*(Pr/Pr_w)(1/4)*k/d; // [W/square
    meter degree celsius] heat transfer coefficient
27 // multiplying by 0.92 from table 6-7 (page no.-300)
    to correct for only five tube rows gives
28 h = 0.92*h; // [W/square meter degree celsius]
29 printf("the value of heat transfer coefficient is %f
    W/square meter degree celsius",h);
30 h_in = 163.46432; // [W/square meter degree celsius]
    from previous example
31 printf("\n\n the value of heat transfer coefficient
    for previous problem is %f W/square meter degree
    celsius",h_in);
32 P = (h-h_in)*100/h_in;
33 printf("\n\n percentage increase in value of h is %f
    ",P);

```

---

Scilab code Exa 6.12 heating of liquid bismuth in tube

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 6.12\n\n\n");
4 // heating of liquid bismuth in tube
5 // example 6.11 (page no.-305-6)
6 // solution
7
8 m_dot = 4.5; // [Kg/s] flow rate of bismuth
9 d = 0.05; // [m] diameter of steel tube
10 Ti = 415; // [degree celsius] initial temperature of
    bismuth
11 Tf = 440; // [degree celsius] final temperature of
    bismuth
12 // because a constant heat flux is maintained, we
    may use equation 6-47 to calculate the heat
    transfer coefficient.
13 // the properties of bismuth are evaluated at the
    average bulk temperature of
14 Ta = (Ti+Tf)/2; // [degree celsius]
15 mu = 1.34*10^(-3); // [Kg/m s] viscosity
16 Cp = 149; // [J/Kg degree celsius] heat
17 k = 15.6; // [W/m degree celsius]
18 Pr = 0.013; // prandtl number
19 // the total transfer is calculated from
20 q = m_dot*Cp*(Tf-Ti); // [W]
21 // we calculate reynolds and pecllet number as
22 G = m_dot/(%pi*d^(2)/4);
23 Re_d = d*G/mu;
24 Pe = Re_d*Pr;
25 // the heat transfer coefficient is calculated from
    equation 6-47
26 Nu_d = 4.82+0.0185*Pe^(0.827);
```

```
27 h = Nu_d*k/d;// [W/square meter degree celsius]
28 // the total required surface area of the tube may
    now be computed from  $q=h*A*DT$ 
29 // where we use the temperature difference of
30 DT = 20;// [degree celsius]
31 A = q/(h*DT);// [square meter]
32 // the area in turn can be expressed in terms of
    tube length
33 L = A/(%pi*d);// [m]
34 printf("Length of tube required to effect the heat
    transfer is %f m",L);
```

---

# Chapter 7

## Natural Convection Systems

Scilab code Exa 7.1 constant heat flux from vertical plate

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 7.1\n\n");
4 // constant heat flux from vertical plate
5 // Example 7.1 (page no.-330-331)
6 // solution
7
8 q_w = 800; // [W/square meter] radiant energy flux
9 H = 3.5; // [m] height of metal plate surface
10 W = 2; // [m] width of metal plate
11 Ta = 30; // [degree celsius] surrounding air
    temperature
12 // we treat this problem as one with constant heat
    flux on the surface since we do not know the
    surface temperature, we must make an estimate for
    determining Tf and the air properties.
13 // an approximate value of h for free convection
    problems is
14 h = 10; // [W/square meter degree celsius]
15 dT = q_w/h; // [degree celsius]
16 // then
```

```

17 Tf = (dT/2)+Ta;// [degree celsius] approximately
18 // at Tf the properties of air are
19 v = 2.005*10(-5);// [square meter/s]
20 k = 0.0295;// [W/m degree celsius]
21 Pr = 0.7;// prandtl number
22 Beta = 1/(Tf+273);// [K(-1)]
23 // from equation (7-30), with
24 x = 3.5;// [m]
25 g = 9.8;// [square meter/s] acceleration due to
    gravity
26 Gr_x = (g*Beta*q_w*x(4))/(k*v(2));
27 // we may therefore use equation (7-32) to evaluate
    h_x
28 h_x = (k*0.17*(Gr_x*Pr)(1/4))/x;// [W/square meter
    degree celsius]
29 // in the turbulent heat transfer governed by
    equation (7-32), we note that
30 // Nu_x = h*x/k ~ (Gr_x)(1/4) ~ x
31 // or h_x doest noy vary with x, and we may take
    this as the average value. the value of h
32 h = 5.41;// [W/square meter degree celsius]
33 // is less than the approximate value we used to
    estimate Tf, recalculating dT, we obtain
34 dT1 = q_w/h_x;// [degree celsius]
35 // our new film temperature would be
36 Tf1 = Ta+dT1/2;// [degree celsius]
37 // at Tf the properties of air are
38 v1 = 2.354*10(-5);// [square meter/s]
39 k1 = 0.0320;// [W/m degree celsius]
40 Pr1 = 0.695;// prandtl number
41 Beta1 = 1/(Tf1+273);// [K(-1)]
42 // then
43 Gr_x1 = (g*Beta1*q_w*x(4))/(k1*v1(2));
44 // and h_x is caalculated from
45 h_x1 = (k1*0.17*(Gr_x1*Pr1)(1/4))/x;// [W/square
    meter degree celsius]
46 // our new temperature difference is calculated as
47 dT2 = q_w/h_x1;// [degree celsius]

```

```

48 // the average wall temperature is therefore
49 T_w_avg = dT2+Ta;// [degree celsius]
50 printf("the average wall temperature is therefore %f
        degree celsius",T_w_avg);

```

---

**Scilab code Exa 7.2** heat transfer from isothermal vertical plate

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 7.2\n\n\n");
4 // heat transfer from isothermal vertical plate
5 // Example 7.2 (page no.-332)
6 // solution
7
8 H = 4;// [m] height of vertical plate
9 Tp = 60;// [degree celsius] plate temperature
10 Ta = 10;// [degree celsius] atmospheric temperature
11 // we first determine the film temperature as
12 Tf = (Tp+Ta)/2;// [degree celsius]
13 // the properties of interest are thus
14 v = 16.5*10^(-6);// [square meter/s]
15 k = 0.02685;// [W/m degree celsius]
16 Pr = 0.7;// prandtl number
17 Beta = 1/(Tf+273);// [K^(-1)]
18 g = 9.8;// [square meter/s] acceleration due to
    gravity
19 // and
20 Gr_into_Pr = (g*Beta*(Tp-Ta)*H^(3)*Pr)/(v^(2));
21 // we then may use equation (7-29) to obtain
22 Nu_bar_root = 0.825+(0.387*(Gr_into_Pr)^(1/6))
    /(1+(0.492/Pr)^(9/16))^(8/27);
23 Nu_bar = (Nu_bar_root)^(2);
24 // the heat transfer coefficient is
25 h_bar = Nu_bar*k/H;// [W/square meter degree celsius
    ]

```

```

26 // the heat transfer is
27 A = H*10; // [square meter] for 10 m wide plate
28 q = h_bar*A*(Tp-Ta); // [W]
29 // as an alternative, we could employ the simpler
    relation
30 Nu = 0.1*(Gr_into_Pr)^(1/3);
31 printf("heat transfer if the plate is 10 m wide is
    %f W", q);

```

---

**Scilab code Exa 7.3** heat transfer from horizontal tube in water

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 7.3\n\n\n");
4 // heat transfer from horizontal tube in water
5 // Example 7.3 (page no.-333)
6 // solution
7
8 d = 0.02; // [m] diameter of heater
9 Ts = 38; // [degree celsius] surface temperature of
    heater
10 Tw = 27; // [degree celsius] water temperature
11 // the film temperature is
12 Tf = (Ts+Tw)/2; // [degree celsius]
13 // from appendix A the properties of water are
14 k = 0.630; // [W/m degree celsius] thermal
    conductivity
15 // and the following term is particularly useful in
    obtaining the product GrPr product when it is
    multiplied by d^(3)*DT
16 // g*Beta*rho^(2)*Cp/(mu*k) = 2.48*10^(10) [1/m^(3)
    degree celsius]
17 K = 2.48*10^(10); // [1/m^(3) degree celsius]
18 Gr_into_Pr = K*(Ts-Tw)*d^(3);
19 // using table 7-1 (page number -328), we get

```

```

20 C = 0.53;
21 m = 1/4;
22 // so that
23 Nu = C*(Gr_into_Pr)^(1/4);
24 h = Nu*k/d; // [W/square meter degree celsius]
      convection heat transfer coefficient
25 // the heat transfer is thus
26 q_by_L = h*pi*d*(Ts-Tw); // [W/m]
27 printf("free-convection heat loss per unit length of
      heater is %f W/m", q_by_L);

```

---

**Scilab code Exa 7.4** heat transfer from fine wire in air

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 7.4\n\n\n");
4 // heat transfer from fine wire in air
5 // Example 7.4 (page no.-333-334)
6 // solution
7
8 d = 0.00002; // [m] diameter of wire
9 L = 0.5; // [m] length of wire whose temperature is
      maintained
10 Ts = 54; // [degree celsius] surface temperature of
      wire
11 Pa = 101325; // [Pa] pressure of air
12 Ta = 0; // [degree celsius] temperature of air
13 // we first determine the film temperature as
14 Tf = (Ts+Ta)/2; // [degree celsius]
15 // the properties of interest are thus
16 v = 15.69*10^(-6); // [square meter/s]
17 k = 0.02624; // [W/m degree celsius]
18 Pr = 0.708; // prandtl number
19 Beta = 1/(Tf+273); // [K^(-1)]
20 g = 9.8; // [square meter/s] acceleration due to

```



```

        gravity
21 // and
22 Gr_into_Pr = (g*Beta*(Ts-Ta)*d^(3)*Pr)/(v^(2));
23 // from table 7-1 we find
24 C = 0.675;
25 m = 0.058;
26 // so that
27 Nu_bar = C*(Gr_into_Pr)^(m);
28 h_bar = Nu_bar*k/d; // [W/square meter degree celsius
    ]
29 // the heat required is
30 A = %pi*d*L; // [square meter] surface area of wire
31 q = h_bar*A*(Ts-Ta); // [W]
32 printf("electric power necessary to maintain the the
    wire temperature if the length is 0.5 m is %f W"
    ,q);

```

---

#### Scilab code Exa 7.5 heated horizontal pipe in air

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 7.5\n\n\n");
4 // heated horizontal pipe in air
5 // Example 7.5 (page no.-334-335)
6 // solution
7
8 d = 0.3048; // [m] diameter of pipe
9 Ts = 250; // [degree celsius] surface temperature of
    pipe
10 Ta = 15; // [degree celsius] temperature of air
11 // we first determine the Grashof-prandtl number
    product and then select the appropriate constants
    from table 7-1(page no.-328) for use with
    equation (7-25)
12 // the properties of air are evaluated at the film

```

```

    temperature:
13 Tf = (Ts+Ta)/2; // [degree celsius]
14 // the properties of interest are thus
15 v = 26.54*10^(-6); // [square meter/s]
16 k = 0.03406; // [W/m degree celsius]
17 Pr = 0.687; // prandtl number
18 Beta = 1/(Tf+273); // [K^(-1)]
19 g = 9.8; // [square meter/s] acceleration due to
    gravity
20 Gr_d_into_Pr = g*Beta*(Ts-Ta)*d^(3)*Pr/(v^(2));
21 // from table 7-1
22 C = 0.53;
23 m = 1/4;
24 Nu_d = C*(Gr_d_into_Pr)^(m);
25 h = Nu_d*k/d; // [W/square meter degree celsius]
26 // the heat transfer per unit length is then
    calculated from
27 q_by_L = h*%pi*d*(Ts-Ta); // [W/m]
28 printf("free-convection heat loss per unit length is
    %f kW/m", q_by_L/1000);

```

---

#### Scilab code Exa 7.6 cube cooling in air

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 7.6\n\n\n");
4 // cube cooling in air
5 // Example 7.6 (page no.-336)
6 // solution
7
8 L = 0.2; // [m] side length of cube
9 Ts = 60; // [degree celsius] surface temperature of
    cube
10 Ta = 10; // [degree celsius] air temperature
11 // this is an irregular solid so we use the

```

```

        information in the last entry of table 7-1(page
        no.-328) in the absence of a specific correlation
        for this geometry.
12 // the properties were evaluated as
13 v = 17.47*10^(-6); // [square meter/s]
14 k = 0.02685; // [W/m degree celsius]
15 Pr = 0.70; // prandtl number
16 Beta = 3.25*10^(-3); // [K^(-1)]
17 g = 9.8; // [square meter/s] acceleration due to
        gravity
18 // the characteristic length is the distance a
        particle travels in the boundary layer, which is
        L/2 along the bottom plus L along the side plus L
        /2 on the top or
19 Gr_into_Pr = (g*Beta*(Ts-Ta)*(2*L)^(3)*Pr)/(v^(2));
20 // from the last entry in table 7-1 we find
21 C = 0.52;
22 n = 1/4;
23 // so that
24 Nu = C*(Gr_into_Pr)^(n);
25 h_bar = Nu*k/(2*L); // [W/square meter degree celsius
        ]
26 // the cube has six sides so the area is
27 A = 6*L^(2); // [square meter]
28 // the heat required is
29 q = h_bar*A*(Ts-Ta); // [W]
30 printf("heat transfer is %f W",q);

```

---

**Scilab code Exa 7.7** calculation with simplified relations

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 7.7\n\n\n");
4 // calculation with simplified relations
5 // Example 7.7 (page no.-338-339)

```

```

6 // solution
7
8 // this example is calculation of heat transfer with
  simplified relations for example (7.5) so we use
  the data of example 7.5
9
10 d = 0.3048; // [m] diameter of pipe
11 Ts = 250; // [degree celsius] surface temperature of
  pipe
12 Ta = 15; // [degree celsius] temperature of air
13 // we first determine the Grashof-prandtl number
  product and then select the appropriate constants
  from table 7-1(page no.-328) for use with
  equation (7-25)
14 // the properties of air are evaluated at the film
  temperature:
15 Tf = (Ts+Ta)/2; // [degree celsius]
16 // the properties of interest are thus
17 v = 26.54*10(-6); // [square meter/s]
18 k = 0.03406; // [W/m degree celsius]
19 Pr = 0.687; // prandtl number
20 Beta = 1/(Tf+273); // [K(-1)]
21 g = 9.8; // [square meter/s] acceleration due to
  gravity
22 // in example (7.5) we found that a rather large
  pipe with a substantial temperature difference
  between the surface and air still had a GrPr
  product of 1.57*10(8) < 10(9), so laminar
  equation is selected from table 7-2(page no.-339)
  . the heat transfer coefficient is given by
23 h = 1.32*((Ts-Ta)/d)(1/4); // [W/square meter degree
  celsius]
24 // the heat transfer is then
25 q_by_L = h*pi*d*(Ts-Ta); // [W/m]
26 printf("heat transfer is %f kW/m", q_by_L/1000);

```

---

**Scilab code Exa 7.8** heat transfer across vertical air gap

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 7.8\n\n\n");
4 // heat transfer across vertical air gap
5 // Example 7.8 (page no.-345)
6 // solution
7
8 L = 0.5; // [m] side length vertical square plate
9 d = 0.015; // [m] distance between plates
10 p = 101325; // [Pa] pressure of air
11 R = 287; // [] universal gas constant
12 T1 = 100; // [degree celsius] temperature of first
    plate
13 T2 = 40; // [degree celsius] temperature of second
    plate
14 E = 0.2; // emissivity of both surfaces
15 // the properties of air is evaluated at the mean
    temperature
16 Tf = (T1+T2)/2; // [degree celsius]
17 rho = p/(R*(Tf+273)); // [Kg/m^(3)] density
18 k = 0.0295; // [W/m degree celsius]
19 Pr = 0.70; // prandtl number
20 Beta = 1/(Tf+273); // [K^(-1)]
21 mu = 2.043*10^(-5); // [Kg/m s] viscosity
22 g = 9.8; // [square meter/s] acceleration due to
    gravity
23 // the Grashof-prandtl number product is now
    calculated as
24 Gr_into_Pr = (g*rho^(2)*Beta*(T1-T2)*(d)^(3)*Pr)/(mu
    ^2);
25 // we may now use equation (7-64) to calculate the
    effective thermal conductivity , with
```

```

26 L = 0.5; // [m]
27 del = 0.015; // [m]
28 // and the constants taken from table 7-3(page no
   .-344):
29 Ke_by_K = 0.197*(Gr_into_Pr)^(1/4)*(L/del)^(-1/9);
30 // the heat transfer may now be calculated with
   equation (7-54). the area is
31 A = L^(2); // [square meter]
32 q = Ke_by_K*k*A*(T1-T2)/del; // [W]
33 // the radiation flux is calculated with equation
   (7-67), taking
34 T1 = 373; // [K]
35 T2 = 313; // [K]
36 E1 = E;
37 E2 = E;
38 sigma = 5.669*10^(-8); // [W/square meter K^(4)]
39 q_A = sigma*(T1^(4)-T2^(4))/((1/E1)+(1/E2)-1); // [W/
   square meter]
40 q_rad = A*q_A; // [W]
41 printf("free-convection heat transfer across the air
   space is %f W",q);
42 printf("\n\nradiation heat transfer across the air
   space is %f W",q_rad);

```

---

**Scilab code Exa 7.9** heat transfer across horizontal air gap

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 7.9\n\n\n");
4 // heat transfer across horizontal air gap
5 // Example 7.9 (page no.-346)
6 // solution
7
8 a = 0.2; // [m] side length of plate
9 d = 0.01; // [m] seperation between two plates

```

```

10 p = 101325; // [Pa] pressure of air
11 R = 287; // [] universal gas constant
12 T1 = 100; // [degree celsius] temperature of first
    plate
13 T2 = 40; // [degree celsius] temperature of second
    plate
14 // the properties are the same as given in example
    (7.8)
15 Tf = (T1+T2)/2; // [degree celsius]
16 rho = p/(R*(Tf+273)); // [Kg/m^(3)] density
17 k = 0.0295; // [W/m degree celsius]
18 Pr = 0.70; // prandtl number
19 Beta = 1/(Tf+273); // [K^(-1)]
20 mu = 2.043*10^(-5); // [Kg/m s] viscosity
21 g = 9.8; // [square meter/s] acceleration due to
    gravity
22 // the GrPr product is evaluated on the basis of the
    separating distance, so we have
23 Gr_into_Pr = (g*rho^(2)*Beta*(T1-T2)*(d)^(3)*Pr)/(mu
    ^2);
24 // consulting table 7-3(page no.-344) we find
25 C = 0.059;
26 n = 0.4;
27 m = 0;
28 Ke_by_K = C*(Gr_into_Pr)^(n)*(a/d)^(m);
29 A = a^(2); // [square meter] area of plate
30 q = Ke_by_K*k*A*(T1-T2)/d; // [W]
31 printf("heat transfer across the air space is %f W",
    q);

```

---

**Scilab code Exa 7.10** heat transfer across water layer

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 7.10\n\n");

```

```

4 // heat transfer across water layer
5 // Example 7.10 (page no.-346-347)
6 // solution
7
8 L = 0.5; // [m] length of square plate
9 d = 0.01; // [m] separation between square plates
10 T1 = 100; // [degree F] temperature of lower plate
11 T2 = 80; // [degree F] temperature of upper plate
12 // we evaluate properties at mean temperature of 90
    degree F and obtain, for water
13 k = 0.623; // [W/m degree celsius]
14 // and the following term is particularly useful in
    obtaining the product GrPr
15 //  $g \cdot \beta \cdot \rho^2 \cdot C_p / (\mu \cdot k) = 2.48 \cdot 10^{10}$  [1/m3)
    degree celsius]
16 // the Grashof-prandtl number product is now
    evaluated using the plate spacing of 0.01 m as
    the characteristic dimension
17 K = 2.48*10(10); // [1/m3) degree celsius]
18 Gr_into_Pr = K*(T1-T2)*(5/9)*d(3);
19 // now, using equation 7-64 and consulting table
    7-3(page no.-344) we obtain
20 C = 0.13;
21 n = 0.3;
22 m = 0;
23 // therefore, equation (7-64) becomes
24 Ke_by_K = C*Gr_into_Pr(n);
25 // the effective thermal conductivity is thus
26 ke = k*Ke_by_K; // [W/m degree celsius]
27 // and the heat transfer is
28 A = L(2); // [square meter] area of plate
29 q = ke*A*(T1-T2)*(5/9)/d; // [W]
30 printf("heat lost by the lower plate is %f W",q);

```

---

Scilab code Exa 7.11 reduction of convection in ar gap



```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 7.11 \n\n\n");
4 // reduction of convection in ar gap
5 // Example 7.11 (page no.-347)
6 // solution
7
8 Tm = 300; // [K] mean temperature of air
9 dT = 20; // [degree celsius] temperature difference
10 R = 287; // [] universal gas constant
11 g = 9.81; // [m/s^(2)] acceleration due to gravity
12 p_atm = 101325; // [Pa] atmospheric pressure
13 // consulting table 7-13(page no.-344), we find that
    for gases, a value Grdel_into_Pr < 2000 is
    necessary to reduce the system to one of pure
    conduction.
14 // at 300 K the properties of air are
15 k = 0.02624; // [W/m degree celsius]
16 Pr = 0.7; // prandtl no.
17 mu = 1.846*10^(-5); // [Kg/m s]
18 Beta = 1/300;
19 // we have
20 Grdel_into_Pr = 2000;
21
22 // Part A for spacing of 1cm
23
24 del = 0.01; // [m] spacing between plate
25 p = sqrt((Grdel_into_Pr*((R*Tm)^(2))*mu^(2))/(g*Beta
    *dT*del^(3)*Pr)); // [Pa]
26 // or vacuum
27 vacuum = p_atm-p; // [Pa]
28 printf("vacuum necessary for glass spacings of 1 cm
    is %f Pa",vacuum);
29
30 // Part B for spacing of 2cm
31
32 del1 = 0.02; // [m] spacing between plate
33 p1 = sqrt(Grdel_into_Pr*(R*Tm)^(2)*mu^(2)/(g*Beta*dT

```

```

        *del1^(3)*Pr)); // [Pa]
34 // or vacuum
35 vacuum1 = p_atm-p1; // [Pa]
36 printf("\n\n vacuum necessary for glass spacings of
        2 cm is %f Pa",vacuum1);

```

---

**Scilab code Exa 7.12** heat transfer across evacuated space

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 7.12\n\n\n");
4 // heat transfer across evacuated space
5 // Example 7.12 (page no.-351-352)
6 // solution
7
8 E = 0.06; // emmisvity of polished aluminium plate
9 d = 0.025; // [m] seperation between plates
10 p = 101325*10^(-6); // [Pa] pressure of air between
    plates
11 T1 = 100; // [degree celsius] temperature of plate 1
12 T2 = 30; // [degree celsius] temperature of plate 2
13 // we first calculate the mean free path to
    determine if low-density effects to be important.
14 // evaluating properties at the mean air temperature
    of 65 degree celsius , we have
15 lambda = (2.27*10^(-5)*((T1+T2)/2+273))/(p); // [m]
16 // since the plate spacing is only 2.5 cm, we should
    expect low-density effects to be important.
17 // evaluating properties at the mean temperature of
    65 degree celsius , we have
18 k = 0.0291; // [W/m degree celsius]
19 Gamma = 1.40;
20 Pr = 0.7;
21 alpha = 0.9; // from table 7-4(page no.-350)
22 // combining equations (7-75)with the central

```

```

    temperature gradient relation gives
23 // inserting the appropriate properties gives
24 deff('y = f(dT)', 'y = dT - ((2-alpha)/alpha) * (2*Gamma
    /(Gamma+1)) * (lambda/Pr) * ((T1-T2-2*dT)/d)');
25 dT = fsolve(1,f);
26 // the conduction heat transfer is thus
27 q_by_A = k*((T1-T2-2*dT)/d); // [W/square meter]
28 printf("conduction heat transfer through the air gap
    is %f W/square meter", q_by_A);
29 // at normal atmospheric pressure the conduction
    would be
30 q_by_A1 = k*((T1-T2)/d); // [W/square meter]
31 // the radiation heat transfer is calculated with
    equation (8-42), taking E1=E2=0.06 for polished
    aluminium:
32 sigma = 5.669*10^(-8); // []
33 q_by_A_rad = sigma*(((T1+273)^(4)-(T2+273)^(4))/((2/
    E)-1)); // [W/square meter]
34 printf("\n\n thus, at the low density condition the
    radiation heat transfer is almost %f times as
    large as the conduction", q_by_A_rad/q_by_A);

```

---

**Scilab code Exa 7.13** combined free and forced convection with air

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 7.13\n\n\n");
4 // combined free and forced convection with air
5 // Example 7.12 (page no.-353-355)
6 // solution
7
8 p = 101325; // [Pa] pressure of air
9 Ta = 27; // [degree celsius] temperature of air
10 d = 0.025; // [m] diameter of tube
11 u = 0.3; // [m/s] velocity of air

```

```

12 Tw = 140; // [degree celcius] temperature of tube
    wall
13 L = 0.4; // [m] length of tube
14 R = 287; // [] universal gas constant
15 // the properties of air are evaluated at the film
    temperature:
16 Tf = (Tw+Ta)/2; // [degree celcius]
17 // the properties of interest are thus
18 kf = 0.0305; // [W/m degree celcius]
19 Pr = 0.695; // prandtl number
20 Beta = 1/(Tf+273); // [K(-1)]
21 g = 9.8; // [square meter/s] acceleration due to
    gravity
22 mu_f = 2.102*10(-5); // [Kg/m s]
23 mu_w = 2.337*10(-5); // [Kg/m s]
24 rho_f = p/(R*(Tf+273)); // [Kg/cubic meter]
25 // let us take the bulk temperature as 27 degree
    celsius for evaluating mu_b; then
26 mu_b = 1.8462*10(-5); // [Kg/m s]
27 // the significant parameters are calculated as
28 Re_f = rho_f*u*d/mu_f;
29 Gr = rho_f(2)*g*Beta*(Tw-Ta)*d(3)/mu_f(2);
30 Z = Gr*Pr*d/L; // constant
31 // according to figure(7-14)(page no.-354), the
    mixed convection flow regime is encountered. thus
    we must use equation(7-77).
32 // The graetz number is calculated as
33 Gz = Re_f*Pr*d/L;
34 // and the numerical calculation for equation(7-77)
    becomes
35 Nu = 1.75*(mu_b/mu_w)(0.14)*[Gz+0.012*(Gz*Gr(1/3))
    (4/3)](1/3);
36 // the average heat transfer coefficient is
    calculated as
37 h_bar = Nu*kf/d; // [W/square meter degree celsius]
38 printf("heat transfer coefficient is %f W/square
    meter degree celsius",h_bar);

```

---

# Chapter 8

## Radiation Heat Transfer

Scilab code Exa 8.1 transmission and absorption in a gas plate

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 8.1\n\n");
4 // transmission and absorption in a gas plate
5 // Example 8.1 (page no.-381)
6 // solution
7
8 T = 2000+273; // [K] furnace temperature
9 L = 0.3; // [m] side length of glass plate
10 t1 = 0.5; // transmissivity of glass between lambda1
    to lambda2
11 lambda1 = 0.2; // [micro m]
12 lambda2 = 3.5; // [micro m]
13 E1 = 0.3; // emissivity of glass upto lambda2
14 E2 = 0.9; // emissivity of glass above lambda2
15 t2 = 0; // transmissivity of glass except in the
    range of lambda1 to lambda2
16 sigma = 5.669*10^(-8); // [W/square meter K^(4)]
17 A = L^(2); // [square meter] area of glass plate
18 // calculating constants to use table 8-1(page no
    .-379-380)
```

```

19 K1 = lambda1*T;// [micro m K]
20 K2 = lambda2*T;// [micro m K]
21 // from table 8-1
22 Eb_0_lam1_by_sigmaT4 = 0;
23 Eb_0_lam2_by_sigmaT4 = 0.85443;
24 Eb = sigma*T^(4);// [W/square meter]
25 // total incident radiation is
26 // for 0.2 micro m to 3.5 micro m
27 TIR = Eb*(Eb_0_lam2_by_sigmaT4-Eb_0_lam1_by_sigmaT4)
    *A;// [W]
28 TRT = t1*TIR;// [W]
29 RA1 = E1*TIR;// [W] for 0<lambda<3.5 micro m
30 RA2 = E2*(1-Eb_0_lam2_by_sigmaT4)*Eb*A;// [W] for
    3.5 micro m <lambda< infinity
31 TRA = RA1+RA2;// [W]
32 printf("total energy absorbed in the glass is %f kW"
    ,TRA/1000);
33 printf("\n\n total energy transmitted by the glass
    is %f kW" ,TRT/1000);

```

---

### Scilab code Exa 8.2 heat transfer between black surfaces

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 8.2\n\n\n");
4 // heat transfer between black surfaces
5 // Example 8.2 (page no.-389-390)
6 // solution
7
8 L = 1;// [m] length of black plate
9 W = 0.5;// [m] width of black plate
10 T1 = 1000+273;// [K] first plate temperature
11 T2 = 500+273;// [K] second plate temperature
12 sigma = 5.669*10^(-8);// [W/square meter K^(4)]
13 // the ratios for use with figure 8-12(page no.-386)

```

```

        are
14 Y_by_D = W/W;
15 X_by_D = L/W;
16 // so that
17 F12 = 0.285; // radiation shape factor
18 // the heat transfer is calculated from
19 q = sigma*L*W*F12*(T1^(4)-T2^(4));
20 printf("net radiant heat exchange between the two
        plates is %f kW",q/1000);

```

---

**Scilab code Exa 8.3** shape factor algebra for open ends of cylinder

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 8.3\n\n\n");
4 // shape-factor algebra for open ends of cylinder
5 // Example 8.3 (page no.-395)
6 // solution
7
8 d1 = 0.1; // [m] diameter of first cylinder
9 d2 = 0.2; // [m] diameter of second cylinder
10 L = 0.2; // [m] length of cylinder
11 // we use the nomenclature of figure 8-15(page no
    .-388) for this problem and designate the open
    ends as surfaces 3 and 4.
12 // we have
13 L_by_r2 = L/(d2/2);
14 r1_by_r2 = 0.5;
15 // so from figure 8-15 or table 8-2(page no.-389) we
    obtain
16 F21 = 0.4126;
17 F22 = 0.3286;
18 // using the reciprocity relation (equation 8-18) we
    have
19 F12 = (d2/d1)*F21;

```

```

20 // for surface 2 we have F12+F22+F23+F24 = 1.0
21 // and from symmetry F23 = F24 so that
22 F23 = (1-F21-F22)/2;
23 F24 = F23;
24 // using reciprocity again ,
25 A2 = %pi*d2*L; // [m^2]
26 A3 = %pi*(d2^2-d1^2)/4; // [m^2]
27 F32 = A2*F23/A3;
28 // we observe that F11 = F33 = F44 = 0 and for
    surface 3 F31+F32+F34 = 1.0
29 // so, if F31 can be determined, we can calculate
    the desired quantity F34. for surface 1 F12+F13+
    F14 = 1.0
30 // and from symmetry F13 = F14 so that
31 F13 = (1-F12)/2;
32 F14 = F13;
33 // using reciprocity gives
34 A1 = %pi*d1*L; // [square meter]
35 F31 = (A1/A3)*F13;
36 // then
37 F34 = 1-F31-F32;
38 printf("shape factor between the open ends of the
    cylinder is %f ",F34);

```

---

**Scilab code Exa 8.4** shape factor algebra for truncated cone

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 8.4\n\n\n");
4 // shape-factor algebra for truncated cone
5 // Example 8.4 (page no.-396)
6 // solution
7
8 d1 = 0.1; // [m] diameter of top of cone
9 d2 = 0.2; // [m] diameter of bottom of cone

```



```

10 L = 0.1; // [m] height of cone
11 // we employ figure 8-16(page no.-390) for solution
    of this problem and take the nomenclature as
    shown, designating the top as surface 2,
12 // the bottom as surface 1, and the side as surface
    3. thus the desired quantities are F23 and F33.
    we have
13 Z = L/(d2/2);
14 Y = (d1/2)/L;
15 // thus from figure 8-16(page no.-390)
16 F12 = 0.12;
17 // from reciprocity(equatin 8-18)
18 A1 = %pi*d2^(2)/4; // [square meter]
19 A2 = %pi*d1^(2)/4; // [square meter]
20 F21 = A1*F12/A2;
21 //and
22 F22 = 0;
23 // so that
24 F23 = 1-F21;
25 // for surface 3 F31+F32+F33 = 1, so we must find
    F31 and F32 in order to evaluate F33. since F11 =
    0 we have
26 F13 = 1-F12;
27 // and from reciprocity
28 A3 = %pi*((d1+d2)/2)*[(d1/2-d2/2)^(2)+L^(2)]^(1/2);
    // [square meter]
29 // so from above equation
30 F31 = A1*F13/A3;
31 // a similar procedure is applies with surface 2 so
    that
32 F32 = A2*F23/A3;
33 // finally from above equation
34 F33 = 1-F32-F31;
35 printf("shape factor between the top surface and the
    side is %f ",F23);
36 printf("\nshape factor between the side and itself
    is %f ",F33);

```

---

**Scilab code Exa 8.5** shape factor algebra for cylindrical reflector

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 8.5\n\n\n");
4 // shape-factor algebra for cylindrical reflector
5 // Example 8.5 (page no.-397-398)
6 // solution
7
8 d = 0.6; // [m] diameter of long half-circular
   cylinder
9 L = 0.2; // [m] length of square rod
10 // we have given figure example 8-5(page no.-397)
   for solution of this problem and take the
   nomenclature as shown,
11 // from symmetry we have
12 F21 = 0.5;
13 F23 = F21;
14 // in general,  $F_{11}+F_{12}+F_{13} = 1$ . to aid in the
   analysis we create the fictitious surface 4 shown
   in figure example 8-5 as dashed line.
15 // for this surface
16 F41 = 1.0;
17 // now, all radiation leaving surface 1 will arrive
   either at 2 or at 3. likewise, this radiation will
   arrive at the imaginary surface 4, so that  $F_{41} =$ 
    $F_{12}+F_{13}$  say eqn a
18 // from reciprocity
19 A1 = %pi*d/2; // [square meter]
20 A4 = L+2*sqrt(0.1^(2)+L^(2)); // [square meter]
21 A2 = 4*L; // [square meter]
22 // so that
23 F14 = A4*F41/A1; // say eqn b
24 // we also have from reciprocity
```

```

25 F12 = A2*F21/A1; // say eqn c
26 // combining a,b,c, gives
27 F13 = F14-F12;
28 // finally
29 F11 = 1-F12-F13;
30 printf("value of F12 is %f ",F12);
31 printf("\nvalue of F13 is %f ",F13);
32 printf("\nvalue of F11 is %f ",F11);

```

---

### Scilab code Exa 8.6 hot plates enclosed by a room

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 8.6\n\n\n");
4 // hot plates enclosed by a room
5 // Example 8.6 (page no.-402-404)
6 // solution
7
8 w = 0.5; // [m] width of plate
9 L = 1; // [m] length of plate
10 t = 0.5; // [m] separation between two plates
11 sigma = 5.669*10^(-8); // [W/square meter K^(4)]
12 // this is a three-body problem, the two plates and
    the room, so the radiation network is shown in
    figure(8-27) page no.-401.
13 // from the data of the problem
14 T1 = 1000+273; // [K] temperature of first plate
15 T2 = 500+273; // [K] temperature of second plate
16 T3 = 27+273; // [K] temperature of walls of plates
17 A1 = w*L; // [square meter] area of plate
18 A2 = A1; // [square meter] area of plate
19 E1 = 0.2; // emissivity of plate 1
20 E2 = 0.5; // emissivity of plate 2
21 // because the area of the room A3 is very large,
    the resistance (1-E3)/(E3*A3) may be taken as

```

```

    zero and we obtain  $E_{b3} = J_3$ .
22 // the shape factor F12 was given in example 8-2:
23 F12 = 0.285;
24 F21 = F12;
25 F13 = 1-F12;
26 F23 = 1-F21;
27 // the resistance in the network are calculated as
28 R1 = (1-E1)/(E1*A1);
29 R2 = (1-E2)/(E2*A2);
30 R3 = 1/(A1*F12);
31 R4 = 1/(A1*F13);
32 R5 = 1/(A2*F23);
33 // taking the resistance  $(1-E3)/(E3*A3)$  as zero, we
    have the network (as shown in figure example 8-6(
    page no.-403)).
34 // to calculate the heat flows at each surface we
    must determine the radiosities J1 and J2. the
    network is solved by setting the sum of the heat
    currents entering nodes J1 and J2 to zero
35
36 // node J1:
37 //  $(E_{b1}-J_1)/R_1+(J_2-J_1)/R_3+(E_{b3}-J_1)/R_4 = 0$ 
    (a)
38
39 // node J2:
40 //  $(J_1-J_2)/R_3+(E_{b3}-J_2)/R_5+(E_{b2}-J_2)/R_2 = 0$ 
    (b)
41
42 // now
43 Eb1 = sigma*T1^(4); // [W/square meter]
44 Eb2 = sigma*T2^(4); // [W/square meter]
45 Eb3 = sigma*T3^(4); // [W/square meter]
46 J3 = Eb3; // [W/square meter]
47 // inserting the values of Eb1, Eb2, and Eb3 into
    equations (a) and (b), we have two equations and
    two unknowns J1 and J2 that may be solved
    simultaneously to give
48 // on simplifying we get  $J_1 = (J_2-R_3*[(E_{b3}-J_2)/R_5+($ 

```

```

Eb2-J2)/R2])
49 // putting this value in equation (a) and solve for
    J2
50 def('y' = f3(J2)', 'y = (Eb1-(J2-R3*[(Eb3-J2)/R5+(
    Eb2-J2)/R2]))/R1+(J2-(J2-R3*[(Eb3-J2)/R5+(Eb2-J2)
    /R2]))/R3+(Eb3-(J2-R3*[(Eb3-J2)/R5+(Eb2-J2)/R2]))
    /R4');
51 J2 = fsolve(1, f3); // [W/square meter]
52 J1 = (J2-R3*[(Eb3-J2)/R5+(Eb2-J2)/R2]); // [W/square
    meter]
53 // the total heat lost by plate 1 is
54 q1 = (Eb1-J1)/[(1-E1)/(E1*A1)]; // [W]
55 // and the heat lost by plate 2 is
56 q2 = (Eb2-J2)/[(1-E2)/(E2*A2)]; // [W]
57 // the total heat received by the room is
58 q3 = [(J1-J3)/(1/(A1*F13))]+[(J2-J3)/(1/(A2*F23))];
    // [W]
59 printf("the net heat transfer for plate 1 is %f kW",
    q1/1000)
60 printf("\n\n the net heat transfer for plate 2 is %f
    kW", q2/1000)
61 printf("\n\n the net heat transfer to the room is
    %f kW", q3/1000)

```

---

**Scilab code Exa 8.7** surface in radiant balance

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 8.7\n\n\n");
4 // surface in radiant balance
5 // Example 8.7 (page no.-404-405)
6 // solution
7
8 w = 0.5; // [m] width of plate
9 L = 0.5; // [m] length of plate

```

```

10 sigma = 5.669*10(-8); // [W/square meter K(4)]
11 // from the data of the problem
12 T1 = 1000; // [K] temperature of first surface
13 T2 = 27+273; // [K] temperature of room
14 A1 = w*L; // [square meter] area of rectangle
15 A2 = A1; // [square meter] area of rectangle
16 E1 = 0.6; // emissivity of surface 1
17 // although this problems involves two surfaces
    which exchange heat and one which is insulated or
    re-radiating , equation (8-41) may not be used
    for the calculation because one of the heat-
    exchanging surfaces(the room) is not convex. The
    radiation network is shown in figure example 8-7(
    page no.-404) where surface 3 is the room and
    surface 2 is the insulated surface. note that J3
    = Eb3 because the room is large and (1-E3)/(E3*A3
    ) approaches zero. Because surface 2 is insulated
    it has zero heat transfer and J2 = Eb2. J2 "
    floats" in the network and is determined from the
    overall radiant balance.
18 // from figure 8-14(page no.-387) the shape factors
    are
19 F12 = 0.2;
20 F21 = F12;
21 // because
22 F11 = 0;
23 F22 = 0;
24 F13 = 1-F12;
25 F23 = F13;
26 // the resistances are
27 R1 = (1-E1)/(E1*A1);
28 R2 = 1/(A1*F13);
29 R3 = 1/(A2*F23);
30 R4 = 1/(A1*F12);
31 // we also have
32 Eb1 = sigma*T1(4); // [W/square meter]
33 Eb3 = sigma*T2(4); // [W/square meter]
34 J3 = Eb3; // [W/square meter]

```

```

35 // the overall circuit is a series parallel
    arrangement and the heat transfer is
36 R_equiv = R1+(1/[(1/R2)+1/(R3+R4)]);
37 q = (Eb1-Eb3)/R_equiv;// [W]
38 // this heat transfer can also be written as q = (
    Eb1-J1)/((1-E1)/(E1*A1))
39 // inserting the values
40 J1 = Eb1-q*((1-E1)/(E1*A1));// [W/square meter]
41 // the value of J2 is determined from proportioning
    the resistances between J1 and J3, so that
42 // (J1-J2)/R4 = (J1-J3)/(R4+R2)
43 J2 = J1-((J1-J3)/(R4+R2))*R4;// [W/square meter]
44 Eb2 = J2;// [W/square meter]
45 // finally , we obtain the temperature of the
    insulated surface as
46 T2 = (Eb2/sigma)^(1/4);// [K]
47 printf("temperature of the insulated surface is %f K
    ",T2);
48 printf("\n\n heat lost by the surface at 1000K is %f
    kW",q/1000);

```

---

#### Scilab code Exa 8.8 open hemisphere in large room

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 8.8\n\n\n");
4 // open hemisphere in large room
5 // Example 8.8 (page no.-406-408)
6 // solution
7
8 d = 0.3;// [m] diameter of hemisphere
9 T1 = 500+273;// [degree celsius] temperature of
    hemisphere
10 T2 = 30+273;// [degree celsius] temperature of
    enclosure

```

```

11 E = 0.4; // surface emissivity of hemisphere
12 sigma = 5.669*10^(-8); // [W/square meter K^(4)]
    constant
13 // the object is completely surrounded by a large
    enclosure but the inside surface of the sphere is
    not convex.
14 // in the given figure example 8-8(page no.-407) we
    take the inside of the sphere as surface 1 and
    the enclosure as surface 2.
15 // we also create an imaginary surface 3 covering
    the opening.
16 // then the heat transfer is given by
17 Eb1 = sigma*T1^(4); // [W/square meter]
18 Eb2 = sigma*T2^(4); // [W/square meter]
19 A1 = 2*%pi*(d/2)^(2); // [square meter] area of
    surface 1
20 // calculating the surface resistance
21 R1 = (1-E)/(E*A1);
22 // since A2 tends to 0 so R2 also tends to 0
23 R2 = 0;
24 // now at this point we recognize that all of the
    radiation leaving surface 1 which will eventually
    arrive at enclosure 2 will also hit the
    imaginary surface 3(F12 = F13). we also
    recognize that A1*F13 = A3*F31. but
25 F31 = 1.0;
26 A3 = %pi*(d/2)^(2); // [square meter]
27 F13 = (A3/A1)*F31;
28 F12 = F13;
29 // then calculating space resistance
30 R3 = 1/(A1*F12);
31 // we can calculate heat transfer by inserting the
    quantities in equation (8-40):
32 q = (Eb1-Eb2)/(R1+R2+R3); // [W]
33 printf("net radiant exchange is %f W",q);

```

---



**Scilab code Exa 8.9** effective emissivity of finned surface

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 8.9\n\n\n");
4 // effective emissivity of finned surface
5 // Example 8.9 (page no.-409-410)
6 // solution
7
8 // for unit depth in the z-dimension we have
9 A1 = 10; // [square meter]
10 A2 = 5; // [square meter]
11 A3 = 60; // [square meter]
12 // the apparent emissivity of the open cavity area
13 // A1 is given by equation(8-47) as
14 //  $E_{a1} = E \cdot A3 / [A1 + E \cdot (A3 - A1)]$ 
15 // for constant surface emissivity the emitted
16 // energy from the total area A1+A2 is
17 //  $e_t = [(E_{a1} \cdot A1 + E \cdot A2) / (A1 + A2)] \cdot E_b$ 
18 // and the energy emitted per unit area for that
19 // total area is
20 // the coefficient of  $E_b$  is the effective emissivity
21 // ,  $E_{eff}$  of the combination of the surface and
22 // open cavity. inserting
23 // above equations gives the following values
24 // for  $E = 0.2$ 
25 E = 0.2;
26  $E_{a1} = E \cdot A3 / [A1 + E \cdot (A3 - A1)];$ 
27  $E_{eff} = [(E_{a1} \cdot A1 + E \cdot A2) / (A1 + A2)];$ 
28 printf("For emissivity of 0.2 the value of effective
29 emissivity is %f ", E_eff);
```

```

27
28 // for E = 0.5
29
30 E = 0.5;
31 Ea1 = E*A3/[A1+E*(A3-A1)];
32 E_eff = [(Ea1*A1+E*A2)/(A1+A2)];
33 printf("\n\n For emissivity of 0.5 the value of
    effective emissivity is %f ",E_eff);
34
35 // for E = 0.8
36
37 E = 0.8;
38 Ea1 = E*A3/[A1+E*(A3-A1)];
39 E_eff = [(Ea1*A1+E*A2)/(A1+A2)];
40 printf("\n\n For emissivity of 0.8 the value of
    effective emissivity is %f ",E_eff);

```

---

**Scilab code Exa 8.10** heat transfer reduction with parallel plate shield

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 8.10\n\n\n");
4 // heat transfer reduction with parallel plate
  shield
5 // Example 8.10 (page no.-413)
6 // solution
7
8 E1 = 0.3;// emissivity of first plane
9 E2 = 0.8;// emissivity of second plane
10 E3 = 0.04;// emissivity of shield
11 sigma = 5.669*10^(-8);// [W/square meter K^(4)]
12 // the heat transfer without the shield is given by
13 // q_by_A = sigma*(T1^4-T2^4)/((1/E1)+(1/E2)-1) =
    0.279*sigma*(T1^4-T2^4)
14 // where T1 is temperature of first plane and T2 is

```

```

    temperature of second plane
15 // the radiation network for the problem with the
    shield in place is shown in figure (8-32) (page
    no.-410).
16 // the resistances are
17 R1 = (1-E1)/E1;
18 R2 = (1-E2)/E2;
19 R3 = (1-E3)/E3;
20 // the total resistance with the shield is
21 R = R1+R2+R3;
22 // and the heat transfer is
23 //  $q_{by\_A} = \sigma \cdot (T1^4 - T2^4) / R = 0.01902 \cdot \sigma \cdot (T1$ 
     $^4 - T2^4)$ 
24 printf("so the heat tranfer is reduced by %f percent
    ", ((0.279-0.01902)/0.279)*100);

```

---

**Scilab code Exa 8.11** open cylindrical shield in large room

```

1 clear;
2 clc;
3 printf(" \t \t \t Example Number 8.11 \n \n \n");
4 // open cylindrical shield in large room
5 // Example 8.11 (page no.-413-415)
6 // solution
7
8 // two concentric cylinders of example(8.3) have
9 T1 = 1000; // [K]
10 E1 = 0.8;
11 E2 = 0.2;
12 T3 = 300; // [K] room temperature
13 sigma = 5.669*10(-8); // [W/square meter K(4)]
14 // please refer to figure example 8-11(page no.-413)
    for radiation network
15 // the room is designed as surface 3 and  $J3 = Eb3$ ,
    because the room is very large ,(i.e. its surface

```

```

    is very small)
16 // in this problem we must consider the inside and
    outside of surface 2 and thus have subscripts i
    and o to designate the respective quantities.
17 // the shape factor can be obtained from example 8-3
    as
18 F12 = 0.8253;
19 F13 = 0.1747;
20 F23i = 0.2588;
21 F23o = 1.0;
22 // also
23 A1 = %pi*0.1*0.2; // [square meter] area of first
    cylinder
24 A2 = %pi*0.2*0.2; // [square meter] area of second
    cylinder
25 Eb1 = sigma*T1^4; // [W/square meter]
26 Eb3 = sigma*T3^4; // [W/square meter]
27 // the resistances may be calculated as
28 R1 = (1-E1)/(E1*A1);
29 R2 = (1-E2)/(E2*A2);
30 R3 = 1/(A1*F12);
31 R4 = 1/(A2*F23i);
32 R5 = 1/(A2*F23o);
33 R6 = 1/(A1*F13);
34 // the network could be solved as a series-parallel
    circuit to obtain the heat transfer, but we will
    need the radiosities anyway, so we setup three
    nodal equations to solve for J1, J2i, and J2o.
35 // we sum the currents into each node and set them
    equal to zero:
36
37 // node J1: (Eb1-J1)/R1+(Eb3-J3)/R6+(J2i-J1)/R3 = 0
38 // node J2i: (J1-J2i)/R3+(Eb3-J2i)/R4+(J2o-J2i)/(2*
    R2) = 0
39 // node J2o: (Eb3-J2o)/R5+(J2i-J2o)/(2*R2) = 0
40 // these equations can be solved by matrix method
    and the solution is
41 J1 = 49732; // [W/square meter]

```

```

42 J2i = 26444; // [W/square meter]
43 J2o = 3346; // [W/square meter]
44 // the heat transfer is then calculated from
45 q = (Eb1-J1)/((1-E1)/(E1*A1)); // [W]
46 // from the network we see that
47 Eb2 = (J2i+J2o)/2; // [W/square meter]
48 // and
49 T2 = (Eb2/sigma)^(1/4); // [K]
50 // if the outer cylinder had not been in place
    acting as a "shield" the heat loss from cylinder
    1 could have been calculated from equation(8-43a)
    as
51 q1 = E1*A1*(Eb1-Eb3); // [W]
52 printf("temperature of the outer cylinder is %f K",
    T2);
53 printf("\n\ntotal heat lost by inner cylinder is %f
    W",q1);

```

---

**Scilab code Exa 8.12** network for gas radiation between parallel plates

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 8.12\n\n\n");
4 // network for gas radiation between parallel plates
5 // Example 8.12 (page no.-422-423)
6 // solution
7
8 T1 = 800; // [K] temperature of first plate
9 E1 = 0.3; // emissivity
10 T2 = 400; // [K] temperature of second plate
11 E2 = 0.7; // emissivity
12 Eg = 0.2; // emissivity of gray gas
13 tg = 0.8; // transmissivity of gray gas
14 sigma = 5.669*10^(-8); // [W/square meter K^(4)]
15 // the network shown in figure 8-39(page no.-419)

```

applies to this problem. all the shape factors are unity for large planes and the various resistors can be computed on a unit area basis as

```

16 F12 = 1;
17 F1g = 1;
18 F2g = F1g;
19 R1 = (1-E1)/E1;
20 R2 = (1-E2)/E2;
21 R3 = 1/(F12*(1-Eg));
22 R4 = 1/(F1g*Eg);
23 R5 = 1/(F2g*Eg);
24 Eb1 = sigma*T1^(4); // [W/square meter]
25 Eb2 = sigma*T2^(4); // [W/square meter]
26 // the equivalent resistance of the center "triangle
   " is
27 R = 1/[(1/R3)+(1/(R4+R5))];
28 // the total heat transfer is then
29 q_by_A = (Eb1-Eb2)/(R1+R2+R); // [W/square meter]
30 // if there were no gas present the heat transfer
   would be given by equation (8-42):
31 q_by_A1 = (Eb1-Eb2)/[(1/E1)+(1/E2)-1]; // [W/square
   meter]
32 // the radiosities may be computed from q_by_A = (
   Eb1-J1)*(E1/(1-E1)) = (J2-Eb2)*(E2/(1-E2))
33 J1 = Eb1-q_by_A*((1-E1)/E1); // [W/square meter]
34 J2 = Eb2+q_by_A*((1-E2)/E2); // [W/square meter]
35 // for the network Ebg is just the mean of these
   values
36 Ebg = (J1+J2)/2; // [W/square meter]
37 // so that the temperature of the gas is
38 Tg = (Ebg/sigma)^(1/4); // [K]
39 printf("the heat-transfer rate between the two
   planes is %f W/square meter",q_by_A);
40 printf("\n\n the temperature of the gas is %f K",Tg)
   ;
41 printf("\n\n the ratio of heat-transfer with
   presence of gas to without presence of gas is %f"
   ,q_by_A/q_by_A1);

```

---

**Scilab code Exa 8.13** cavity with transparent cover

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 8.13\n\n\n");
4 // cavity with transparent cover
5 // Example 8.13 (page no.-433-434)
6 // solution
7
8 E1 = 0.5; // emissivity of rectangular cavity
9 t2 = 0.5; // transmissivity
10 rho2 = 0.1; // reflectivity
11 E2 = 0.4; // emissivity
12 // from example 8-9 we have
13 // per unit depth in the z direction we have
14 A1 = 25+25+10;
15 A2 = 10;
16 // we may evaluate K from equation(8-96a)
17 K = E1/(t2+(E2/2));
18 // the value of Ea is then computed from equation
19 // (8-96) as
20 Ea = (t2+(E2/2))*K/[(A2/A1)*(1-E1)+K];
21 printf("apparent emissivity of covered opening is %f\n",Ea);
22 // if there were no cover present, the value of Ea
23 // would be given by equation (8-47) as
24 Ea1 = E1*A1/[A2+E1*(A1-A2)];
25 printf("\n\n if there were no cover present, the
26 value of Ea(apparent emissivity) would be %f",Ea1
27 );
```

---

**Scilab code Exa 8.14** Transmitting and reflecting system for furnace opening

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 8.14\n\n\n");
4 // Transmitting and reflecting system for furnace
   opening
5 // Example 8.14 (page no.-434-435)
6 // solution
7
8 T1 = 1000+273; // [K] temperature of furnace
9 lambda = 4.0; // [micro meter]
10 //for 0 < lambda < 4 micro meter
11 t1 = 0.9;
12 E1 = 0.1;
13 rho1 = 0;
14 //for 4 micro meter < lambda < infinity
15 t2 = 0;
16 E2 = 0.8;
17 rho2 = 0.2;
18 sigma = 5.669*10^(-8); // [W/square meter K^(4)]
19 T3 = 30+273; // [K] room temperature
20 // the diagram of this problem is shown in figure
   example 8-14(page no.-434). because the room is
   large it may be treated as a blackbody also.
21 // we shall analyze the problem by calculating the
   heat transfer for each wavelength band and then
   adding them together to obtain the total. the
   network for each band is a modification of figure
   8-57(page no.-430), as shown here for black
   furnace and room. we shall make the calculation
   for unit area; then
22 A1 = 1.0; // [square meter]
23 A2 = 1.0; // [square meter]
24 A3 = 1.0; // [square meter]
25 F12 = 1.0;
26 F13 = 1.0;
```



```

27 F32 = 1.0;
28 // the total emissive powers are
29 Eb1 = sigma*T1^(4); // [W/square meter]
30 Eb3 = sigma*T3^(4); // [W/square meter]
31 // to determine the fraction of radiation in each
    wavelength band, we calculate
32 lambda_into_T1 = lambda*T1; // [micro meter K]
33 lambda_into_T3 = lambda*T3; // [micro meter K]
34 // consulting table 8-1(page no.-379-380), we find
35 Eb1_0_to_4 = 0.6450*Eb1; // [W/square meter]
36 Eb3_0_to_4 = 0.00235*Eb3; // [W/square meter]
37 Eb1_4_to_inf = (1-0.6450)*Eb1; // [W/square meter]
38 Eb3_4_to_inf = (1-0.00235)*Eb3; // [W/square meter]
39 // we now apply these numbers to the network for the
    two wavelengths bands, with unit areas.
40
41 // 0 < lambda < 4 micro meter band:
42 R1 = 1/(F13*t1);
43 R2 = 1/(F32*(1-t1));
44 R3 = 1/(F12*(1-t1));
45 R4 = rho1/(E1*(1-t1));
46 // the net heat transfer from the network is then
47 R_equiv_1 = 1/(1/R1+1/(R2+R3+R4));
48 q1 = (Eb1_0_to_4-Eb3_0_to_4)/R_equiv_1; // [W/square
    meter]
49
50 // 4 micro meter < lambda < infinity band:
51 R2 = 1/(F32*(1-t2));
52 R3 = 1/(F12*(1-t2));
53 R4 = rho2/(E2*(1-t2));
54 // the net heat transfer from the network is then
55 // R1 is infinity
56 R_equiv_2 = R2+R3+R4*2;
57 q2 = (Eb1_4_to_inf-Eb3_4_to_inf)/R_equiv_2; // [W/
    square meter]
58
59 // the total heat loss is then
60 q_total = q1+q2; // [W/square meter]

```

```

61 // with no windows at all , the heat transfer would
    have been the difference in blackbody emissive
        powers ,
62 Q = Eb1-Eb3;// [W/square meter]
63 printf("radiation lost through the quartz window to
    a room temperature of 30 degree celsius is %f W/
    square meter",q_total);
64 printf("\n\n with no windows at all , the heat
    transfer would be %e W/square meter",Q);

```

---

**Scilab code Exa 8.15** numerical solution for enclosure

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 8.15\n\n\n");
4 // numerical solution for enclosure
5 // Example 8.15 (page no.-440)
6 // solution
7
8 // the geometry of example 8-5 is used
9 d = 0.6;// [m] diameter of long half-circular
    cylinder
10 L = 0.2;// [m] length of square rod
11 E2 = 0.5;
12 T2 = 1000;// [K] temperature of body 2
13 T3 = 300;// [K] temperature of body 3
14 sigma = 5.669*10^(-8);// [W/square meter K^(4)]
15 // for unit length we have:
16 Eb2 = sigma*T2^(4);// [W/square meter]
17 Eb3 = sigma*T3^(4);
18 A1 = 4*L;// [square meter]
19 A2 = %pi*d/2;// [square meter/meter]
20 // we will use the numerical formulation. we find
    from example 8-5, using the nomenclature of the
    figure

```

```

21 F11 = 0.314;
22 F12 = 0.425;
23 F13 = 0.261;
24 F21 = 0.5;
25 F22 = 0;
26 F23 = 0.5;
27 // F31, F32 tends to zero so
28 F33 = 1;
29 // we now write the equations.
30 // surface 1 is insulated so we use equation(8-107a)
   :
31 //  $J_1(1-F_{11})-F_{12}J_2-F_{13}J_3 = 0$ 
32 // surface 2 is constant temperature so we use
   equation (8-106a):
33 //  $J_2(1-F_{22}(1-E_2))-(1-E_2)[F_{21}J_1+F_{23}J_3] = E_2Eb_2$ 
34 // because surface 3 is so large
35  $J_3 = Eb_3$ ; // [W/square meter]
36 // rearranging the equation gives
37 //  $J_1(1-F_{11})-J_2F_{12} = F_{13}J_3$ 
38 //  $J_1(-1)(1-E_2)F_{21}+J_2(1-F_{22}(1-E_2)) = E_2Eb_2+(1-
   E_2)(F_{23}J_3)$ 
39 // solving the above two equations using matrix
40  $X = [(1-F_{11}) \ -F_{12}; (-1)(1-E_2)F_{21} \ (1-F_{22}(1-E_2))]$ ;
41  $Y = [F_{13}J_3; E_2Eb_2+(1-E_2)(F_{23}J_3)]$ ;
42  $J = X^{(-1)}*Y$ ;
43  $J_1 = J(1)$ ; // [W/square meter]
44  $J_2 = J(2)$ ; // [W/square meter]
45 // the heat transfer is thus
46  $q = (Eb_2-J_2)/((1-E_2)/(E_2A_1))$ ; // [W/m] length
47 // because surface 1 is insulated
48  $Eb_1 = J_1$ ; // [W/square meter]
49 // we could calculate the temperature as
50  $T_1 = (Eb_1/\sigma)^{(1/4)}$ ; // [K]
51 printf("heat lost to the large room per unit length
   of surface 2 is %f W/m",q);
52 printf("\n\n temperature of the insulated surface is
   %f K",T1);

```

Scilab code Exa 8.16 numerical solution for parallel plates

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 8.16\n\n\n");
4 // numerical solution for parallel plates
5 // Example 8.16 (page no.-440-443)
6 // solution
7
8 T1 = 1000; // [K]
9 T2 = 400; // [K]
10 E1 = 0.8; //
11 E2 = 0.5; //
12 // consulting figure 8-12, we obtain
13 F12 = 0.2;
14 F21 = 0.2;
15 F11 = 0;
16 F22 = 0;
17 F13 = 0.8;
18 F23 = 0.8;
19 A1 = 1; // [square meter]
20 A2 = 1; // [square meter]
21 // surface 3 is the surrounding or insulated surface
   . For part A(the plates are surrounded by a large
     room at 300K)
22 T3 = 300; // [K]
23 sigma = 5.669*10^(-8); // [W/square meter K^(4)]
24 Eb1 = sigma*T1^(4); // [W/square meter]
25 Eb2 = sigma*T2^(4); // [W/square meter]
26 Eb3 = sigma*T3^(4); // [W/square meter]
27 // because A3 tends to infinity , F31 and F32 must
   approach zero since A1*F13 = A3*F31 and A2*F23 =
   A3*F32. the nodal equations are written in the
   form of equations (8-107):
```

```

28 // surface 1      J1-(1-E1)*(F11*J1+F12*J2+F13*J3)
    = E1*Eb1
29 // surface 2      J2-(1-E2)*(F21*J1+F22*J2+F23*J3)
    = E2*Eb2
30 // surface 3      J3-(1-E3)*(F31*J1+F32*J2+F33*J3)
    = E3*Eb3
31
32 // because F31 and F32 approach zero , F33 must be
    1.0.
33 F33 = 1;
34 // inserting the various numerical values for the
    various terms and solving the third equation we
    get
35 // the third equation as:  J3*E3 = E3*Eb3  so we get
    the value of J3  as
36 J3 = Eb3;// [W/square meter]
37 // finally the equations are written in compact form
    after getting the value of J3 we solve for J2
    and J1 by matrix method
38 Z = [1-(1-E1)*F11  -(1-E1)*F12;-(1-E2)*F21  1-(1-E2)*
    F22];
39 C = [E1*Eb1+(1-E1)*F13*J3;E2*Eb2+(1-E2)*F23*J3];
40 J = Z^(-1)*C;
41 J1 = J(1);// [W/square meter]
42 J2 = J(2);// [W/square meter]
43 // the heat transfers are obtained from equation
    (8-104):
44 q1 = A1*E1*(Eb1-J1)/(1-E1);// [W]
45 q2 = A2*E2*(Eb2-J2)/(1-E2);// [W]
46 // the net heat absorbed by the room is algebraic sum
    of q1 and q2
47 q3_absorbed = q1+q2;// [W]
48 printf("\t\t CASE(A)");
49 printf("\n\n the heat transfers are \n\n\t\t q1 = %f
    kW",q1/1000);
50 printf("\n\t\t q1 = %f kW",q2/1000)
51 printf("\n\n the net heat absorbed by the room in
    part (a) is %f kW",q3_absorbed/1000);

```

```

52
53 // for part(b), A3 for the enclosing wall is 4.0
    square meter
54
55 A3 = 4; // [square meter]
56 // and we set
57 J3 = Eb3; // [W/square meter], because surface 3 is
    insulated.
58 // from reciprocity we have
59 F31 = A1*F13/A3;
60 F32 = A2*F23/A3;
61 // then, we have
62 F33 = 1-F31-F32;
63 // the set of equations are same with J3 = Eb3
64 // surface 1      J1-(1-E1)*(F11*J1+F12*J2+F13*J3)
    = E1*Eb1
65 // surface 2      J2-(1-E2)*(F21*J1+F22*J2+F23*J3)
    = E2*Eb2
66 // surface 3      J3-(1-E3)*(F31*J1+F32*J2+F33*J3)
    = E3*J3
67 // the third equation of set can be written as
68 // J3(1-E3)-(1-E3)*(F31*J1+F32*J2+F33*J3) = 0
69 // so that (1-E3) term drops out, and we obtain
    three equation in three variable which can be
    solved by matrix method
70 Z = [1-(1-E1)*F11  -(1-E1)*F12  -(1-E1)*F13; -(1-E2)*
    F21  1-(1-E2)*F22  -(1-E2)*F23; -F31  -F32  1-F33];
71 C = [E1*Eb1; E2*Eb2; 0];
72 J = Z^(-1)*C;
73 J1n = J(1); // [W/square meter]
74 J2n = J(2); // [W/square meter]
75 J3n = J(3); // [W/square meter]
76 // the heat transfers are
77 q1n = A1*E1*(Eb1-J1n)/(1-E1); // [W]
78 q2n = A2*E2*(Eb2-J2n)/(1-E2); // [W]
79 // of course these heat transfers should be equal in
    magnitude with opposite sign because the
    insulated wall exchanges no heat.

```

```

80 // the temperature of the insulated wall is obtained
    from
81 T3 = (J3n/sigma)^(1/4); // [degree celsius]
82 printf("\n\n \t\tCASE(B)");
83 printf("\n\n the heat transfers are \n\n\t\t q1 = %f
    kW",q1n/1000);
84 printf("\n\t\t q2 = %f kW",q2n/1000);
85 printf("\n\n the temperature of the insulated wall
    is %f K",T3);

```

---

**Scilab code Exa 8.17** radiation from a hole with variable radiosity

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 8.17\n\n\n");
4 // radiation from a hole with variable radiosity
5 // Example 8.17 (page no.-443-446)
6 // solution
7
8 T1 = 1273; // [K]
9 T5 = 293; // [K]
10 E1 = 0.6;
11 // all the shape factors can be obtained with the
    aid of figure 8-13(page no.-387) and the
    imaginary disk surfaces 6 and 7. we have
12 sigma = 5.669*10^(-8); // [W/square meter K^(4)]
13 Eb1 = sigma*T1^(4); // [W/square meter]
14 Eb2 = Eb1; // [W/square meter]
15 Eb3 = Eb2; // [W/square meter]
16 Eb4 = Eb3; // [W/square meter]
17 Eb5 = sigma*T5^(4); // [W/square meter]
18 E2 = E1;
19 E3 = E2;
20 E4 = E3;
21 E5 = 1.0;

```

```

22 r = 0.01; // [m]
23 A1 = %pi*r^(2); // [square m]
24 A5 = A1; // [square m]
25 A6 = A1; // [square m]
26 A7 = A1; // [square m]
27 A2 = %pi*2*r*0.01; // [square m]
28 A3 = A2; // [square m]
29 A4 = A2; // [square m]
30 F11 = 0;
31 F55 = 0;
32 F16 = 0.37;
33 F17 = 0.175;
34 F15 = 0.1;
35 F12 = 1-F16;
36 F54 = F12;
37 F13 = F16-F17;
38 F53 = F13;
39 F14 = F17-F15;
40 F52 = F14;
41 F21 = F16*A1/A2;
42 F26 = F21;
43 F45 = F21;
44 F36 = F45;
45 F37 = F36;
46 F22 = 1-F21-F26;
47 F33 = F22;
48 F44 = F22;
49 F31 = F13*A1/A3;
50 F32 = F36-F31;
51 F34 = F32;
52 F43 = F34;
53 F23 = F34;
54 F27 = F26-F23;
55 F46 = F27;
56 F41 = F14*A1/A4;
57 F25 = F41;
58 F42 = F46-F41;
59 F24 = F42;

```



```

60 // the equations for the radiosities are now written
    in the form of equation 8-106, noting that
61 F11 = 0;
62 J5 = Eb5; // [W/square meter]
63 // J1 = (1-E1)*(F12*J2+F13*J3+F14*J4+F15*Eb5)+E1*
    Eb1
64 // J2 = [(1-E2)*(F21*J1+F23*J3+F24*J4+F25*Eb5)+E2*
    Eb2]/(1-F22*(1-E2))
65 // J3 = [(1-E3)*(F31*J1+F32*J2+F34*J4+F35*Eb5)+E3*
    Eb3]/(1-F33*(1-E3))
66 // J4 = [(1-E4)*(F41*J1+F42*J2+F43*J3+F45*Eb5)+E4*
    Eb4]/(1-F44*(1-E4))
67 // we have 4 equations with 4 variables which can be
    solved by matrix method
68 Z = [1 -(1-E1)*F12 -(1-E1)*F13 -(1-E1)*F14; -F21*(1-
    E2)/(1-F22*(1-E2)) 1 -F23*(1-E2)/(1-F22*(1-E2)) -
    F24*(1-E2)/(1-F22*(1-E2)); -F31*(1-E3)/(1-F33*(1-
    E3)) -F32*(1-E3)/(1-F33*(1-E3)) 1 -F34*(1-E3)/(1-
    F33*(1-E3)); -F41*(1-E4)/(1-F44*(1-E4)) -F42*(1-E4
    )/(1-F44*(1-E4)) -F43*(1-E4)/(1-F44*(1-E4)) 1];
69 C = [E1*Eb1+(1-E1)*F15*Eb5; (E2*Eb2+F25*Eb5*(1-E2))
    /(1-F22*(1-E2)); 104859; (E4*Eb4+F45*Eb5*(1-E4))
    /(1-F44*(1-E4))];
70 J = Z^(-1)*C;
71 J1 = J(1); // [W/square meter]
72 J2 = J(2); // [W/square meter]
73 J3 = J(3); // [W/square meter]
74 J4 = J(4); // [W/square meter]
75 // the heat transfer can be calculated from equation
    (8-104):
76 q1 = E1*A1*(Eb1-J1)/(1-E1); // [W]
77 q2 = E2*A2*(Eb2-J2)/(1-E2); // [W]
78 q3 = E3*A3*(Eb3-J3)/(1-E3); // [W]
79 q4 = E4*A4*(Eb4-J4)/(1-E4); // [W]
80 // THE TOTAL HEAT TRANSFER
81 q = q1+q2+q3+q4; // [W]
82 printf("the heat transfer rate is %f W",q);
83 // It is of interest to compare this heat transfer

```

```

    with the value we would obtain by assuming
    uniform radiosity on the hot surface. we would
    then have a two-body problem with
84 A1 = %pi+3*(2*%pi);// [square cm]
85 A5 = %pi*10^(-4);// [square cm]
86 F51 = 1.0;
87 E1 = 0.6;
88 E5 = 1.0;
89 // the heat transfer is then calculated from
    equation(8-43), with appropriate change of
    nomenclature:
90 q_new = (Eb1-Eb5)*A5/((1/E5)+(A5/A1)*((1/E1)-1));//
    [w]
91 printf("\n\nthus the assumption of uniform radiosity
    gives a heat transfer that is %f percent below
    the value obtained by breaking the hot surface
    into four parts for the calculations", (q-q_new)
    *100/q);
92 // let us now consider the case where surface 1 is
    still radiating at 1000 degree celsius
93 E = 0.6;
94 // the nodal equations for J1 is the same as before
    but now the equations for J2, J3, and J4 must be
    written in the form of equation(8-107). when that
    is done and the numerical values are inserted,
    we obtain
95 // J1 = 0.252*J2+0.078*J3+0.03*J4+89341
96 // J2 = 0.5*J1+0.3452*J3+0.09524*J4+24.869
97 // J3 = 0.1548*J1+0.3452*J2+0.3452*J4+64.66
98 // J4 = 0.05952*J1+0.0952*J2+0.3452*J3+208.9
99 // when these equations are solved, we obtain
100 J1 = 1.1532*10^(5);// [W/square meter]
101 J2 = 0.81019*10^(5);// [W/square meter]
102 J3 = 0.57885*10^(5);// [W/square meter]
103 J4 = 0.34767*10^(5);// [W/square meter]
104 // the heat transfer at surface 1 is
105 A1 = %pi*10^(-4);// [square cm]
106 A5 = %pi*10^(-4);// [square cm]

```

```

107 A2 = %pi*10^(-4); // [square cm]
108 q1 = (E1*A1)*(Eb1-J1)/(1-E1); // [W]
109 // the temperatures of the insulated surface
    elements are obtained from
110 T2 = 820; // [degree celsius]
111 T3 = 732; // [degree celsius]
112 T4 = 612; // [degree celsius]
113 // it is of interest to compare the heat transfer
    calculated above with that obtained by assuming
    surfaces 2,3 and 4 uniform in temperature and
    radiosity. equation(8-41) applies for this case:
114 q2 = A1*(Eb1-Eb5)/[((A1+A2+2*A1*F15)/(A5-A1*F15^2))
    +(1/E1-1)+(A1/A5)*(1/E5-1)]; // [w]

```

---

**Scilab code Exa 8.18** heater with constant heat flux and surrounding shields

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 8.18\n\n\n");
4 // heater with constant heat flux and surrounding
    shields
5 // Example 8.18 (page no.-446-449)
6 // solution
7
8 sigma = 5.669*10^(-8); // [W/square meter K^(4)]
9 T6 = 293; // [K] temperature of room
10 E1 = 0.8;
11 E2 = 0.4;
12 E3 = 0.4;
13 E4 = 0.4;
14 E5 = 0.4;
15 // in reality , surfaces 2,3,4, and 5 have two
    surfaces each; an inside and an outside surface.
    we thus have nine surfaces plus the room, so a 10
    body problem is involved. of course , from

```

```

    symmetry we can see that  $T_2 = T_4$  and  $T_3 = T_5$ .
16 // we designate the large room as surface 6 and it
    behaves as  $E_6 = 1.0$ .
17 // the shape factors of the inside surfaces are
    obtained from figure 8-12 and 8-14:
18 F16 = 0.285;
19 F61 = F16;
20 F13 = 0.24;
21 F15 = 0.24;
22 F31 = 0.24;
23 F51 = 0.24;
24 F12 = 0.115;
25 F14 = 0.115;
26 F24 = 0.068;
27 F42 = 0.068;
28 F35 = 0.285;
29 F53 = 0.285;
30 F32 = 0.115;
31 F52 = 0.115;
32 F34 = 0.115;
33 F25 = 0.23;
34 F23 = 0.23;
35 F45 = 0.23;
36 F43 = 0.23;
37 F21 = 0.23;
38 F41 = 0.23;
39 F26 = 0.23;
40 F46 = 0.23;
41 F11 = 0;
42 F22 = 0;
43 F33 = 0;
44 F44 = 0;
45 F55 = 0;
46 // for the outside surfaces ,
47 F_26 = 1;
48 F_36 = 1;
49 F_46 = 1;
50 F_56 = 1;

```

```

51 // Where the underscore indicate the outside
    surfaces .
52 // for the room
53 Eb6 = sigma*T6^(4); // [W/square meter]
54 // for surface 1 with constant heat flux , we use
    equation (8-108a) and write
55 //  $J_1 - (F_{12} * J_2 + F_{13} * J_3 + F_{14} * J_4 + F_{15} * J_5 + F_{16} * J_6) =$ 
    1.0*10^(5) GIVEN (a)
56 // because of the radiant balance condition we have
57 //  $(J_2 - Eb_2) * E_2 * A_2 / (1 - E_2) = (Eb_2 - J_{_2}) * E_2 * A_2 / (1 - E_2)$ 
58 // and  $Eb_2 = (J_2 + J_{_2}) / 2$ 

    (b)
59 // Where underscore indicates the outside radiosity .
    a similar relation applies for surfaces 3,4, and
    5. thus we can use equation (8-106a) for inside
    surface 2
60 //  $J_2 - (1 - E_2) * (F_{21} * J_1 + F_{23} * J_3 + F_{24} * J_4 + F_{25} * J_5 + F_{26} * J_6) =$ 
    E2*(J2+J_2)/2 (c)
61 // and for outside surface 2
62 //  $J_{_2} - (1 - E_2) * (F_{_26} * J_6) = E_2 * (J_2 + J_{_2}) / 2$ 

    (
    d)
63 // Equations like (c) and (d) are written for
    surfaces 3,4, and 5 also , and with the shape
    factors and emmissivities inserted the following
    set of equations is obtained
64 //  $J_1 - 0.115 * J_2 - 0.24 * J_3 - 0.115 * J_4 - 0.24 * J_5 =$ 
    1.0012*10^(5) 1
65 //  $-0.138 * J_1 + 0.8 * J_2 - 0.2 * J_{_2} - 0.138 * J_3 - 0.0408 * J_4$ 
    -0.138 * J_5 = 57.66 2
66 //  $0.2 * J_2 - 0.8 * J_{_2} = -250.68$ 

    3
67 //  $-0.144 * J_1 - 0.069 * J_2 + 0.8 * J_3 - 0.2 * J_{_3} - 0.069 * J_4 - 0.05 *$ 
    J_5 = 60.16 4
68 //  $0.2 * J_3 - 0.8 * J_{_3} = -250.68$ 

```

```

5
69 // -0.138*J1-0.0408*J2-0.138*J3+0.8*J4-0.2*J_4
    -0.138*J5 = 57.66                                6
70 // 0.2*J4-0.8*J_4 = -250.68

7
71 // -0.144*J1-0.069*J2-0.057*J3-0.069*J4+0.8*J5-0.2*
    J_5 = 60.16                                        8
72 // 0.2*J5-0.8*J_5 = -250.68

9
73 // We thus have nine equations and nine unknowns,
    which may be solved by matrix method
74 Z = [1 -0.115 -0.24 -0.115 -0.24 0 0 0 0;-0.138 0.8
    -0.138 -0.0408 -0.138 -0.2 0 0 0 0;0 0.2 0 0 0 -0.8
    0 0 0;-0.144 -0.069 0.8 -0.069 -0.05 0 -0.2 0
    0;0 0 0.2 0 0 0 -0.8 0 0;-0.138 -0.0408 -0.138
    0.8 -0.138 0 0 -0.2 0;0 0 0 0.2 0 0 0 -0.8
    0;-0.144 -0.069 -0.057 -0.069 0.8 0 0 0 -0.2;0 0
    0 0.2 0 0 0 -0.8];
75 C = [1.0012*10^(5)
    ;57.66;-250.68;60.16;-250.68;57.66;-250.68;60.16;-250.68];

76 J = Z^(-1)*C;
77 J1 = J(1); // [W/square meter]
78 J2 = J(2); // [W/square meter]
79 J3 = J(3); // [W/square meter]
80 J4 = J(4); // [W/square meter]
81 J5 = J(5); // [W/square meter]
82 J_2 = J(6); // [W/square meter]
83 J_3 = J(7); // [W/square meter]
84 J_4 = J(8); // [W/square meter]
85 J_5 = J(9); // [W/square meter]
86 // the temperatures are thus computed from equation
    (b):
87 Eb2 = (J2+J_2)/2; // [W/square meter]
88 T2 = (Eb2/sigma)^(1/4); // [K]
89 T4 = T2; // [K]

```

```

90 Eb3 = (J3+J_3)/2; // [W/square meter]
91 T3 = (Eb3/sigma)^(1/4); // [K]
92 T5 = T3; // [K]
93 // for surface 1 we observed that
94 q = 1*10^(5); // [W/square meter]
95 Eb1 = q*(1-E1)/E1+J1; // [W/square meter]
96 // and
97 T1 = (Eb1/sigma)^(1/4); // [K]
98 printf("temperature of all surfaces are following ")
    ;
99 printf("\n\n\t T1 = %f K",T1);
100 printf("\n\n\t T2 = %f K",T2);
101 printf("\n\n\t T3 = %f K",T3);
102 printf("\n\n\t T4 = %f K",T4);
103 printf("\n\n\t T5 = %f K",T5);
104
105 // surfaces 2,3,4, and 5 as one surface
106 // we now go back and take surfaces 2,3,4, and 5 as
    one surface , which we choose to call surface 7.
    the shape factors are then
107 F16 = 0.285;
108 F61 = 0.285;
109 F17 = 1-0.285;
110 A1 = 2.0;
111 A7 = 6.0;
112 // THUS
113 F71 = A1*F17/A7;
114 F77 = 1-2*F71;
115 F76 = F71;
116 F_76 = 1.0;
117 // then for surface 1 we use equation(8-109a) to
    obtain
118 //  $J_1 - (F_{17} * J_7 + F_{16} * J_6) = 1.0 * 10^5$ 
119 // using  $Eb_7 = (J_7 + J_{-7}) / 2$ , we have for the inside of
    surface 7
120 //  $J_7 * [1 - F_{77} * (1 - E_7)] - (1 - E_7) * (F_{71} * J_1 + F_{76} * J_6) = (J_7 +$ 
     $J_{-7}) * E_7 / 2$ 
121 // while for the outside we have

```

```

122 // J_7-(1-E7)*F_76*J6 = (J7+J_7)*E7/2
123 // when the numerical values are inserted , we obtain
    the set of three equations:
124 // J1-0.715J7 = 1.0012*10^(5)
125 // -0.143*J1+0.486*J7-0.2*J_7 = 59.74
126 // 0.2*J7-0.8*J_7 = -250.68
127 // Solving above three equations by matrix method
128 Z = [1 -0.715 0;-0.143 0.486 -0.2;0 0.2 -0.8];
129 C = [1.0012*10^(5);59.74;-250.68];
130 J = Z^(-1)*C;
131 J1 = J(1);// [W/square meter]
132 J7 = J(2);// [W/square meter]
133 J_7 = J(3);// [W/square meter]
134 // the temperatures are thus computed as before
135 Eb7 = (J7+J_7)/2;// [W/square meter]
136 T7 = (Eb7/sigma)^(1/4);// [K]
137 Eb1 = q*(1-E1)/E1+J1;// [W/square meter]
138 T11 = (Eb1/sigma)^(1/4);// [K]
139 printf("\n\n from second method T1 = %f K",T11);
140 printf("\n\n so there is a difference of %f K
    between the two methods",T11-T1);

```

---

**Scilab code Exa 8.19** numerical solution for combined convection and radiation non linear system

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 8.19\n\n\n");
4 // numerical solution for combined convection and
    radiation(non-linear system)
5 // Example 8.19 (page no.-449-452)
6 // solution
7
8 l = 0.5;// [m] length of plate
9 b = 0.5;// [m] breadth of plate

```



```

10 T1 = 1300; // [K] temperature of plate
11 Tinf = 300; // [K] temperature of surrounding
12 T4 = Tinf; // [degree celsius]
13 h = 50; // [W/square meter] convection heat transfer
    coefficient
14 E1 = 0.8;
15 E2 = 0.3;
16 E3 = 0.3;
17 // using figures 8-12(page no.-386) and 8-14(page no
    .-387), we can evaluate the shape factors as
18 F12 = 0.2;
19 F13 = 0.2;
20 F23 = 0.2;
21 F32 = 0.2;
22 F14 = 1-0.2-0.2;
23 F24_L = 1;
24 F34_R = 1;
25 F21 = F12;
26 F31 = F12;
27 F24_R = 0.6;
28 F34_L = 0.6;
29 F11 = 0;
30 F22 = 0;
31 F33 = 0;
32 // J2R = J3L
33 // J2L = J3R          From symmetry
34 sigma = 5.669*10^(-8); // [W/square meter K^(4)]
35 Eb4 = sigma*T4^(4); // [W/square meter]
36 Eb1 = sigma*T1^(4); // [W/square meter]
37 J4 = Eb4; // [W/square meter]
38 // we now use equation(8-107) to obtain a relation
    for J1:
39 // J1 = (1-E1)*[F12*J2R+F13*J3L+F14*J4]+E1*Eb1
40 // but J2R = J3L and F12 = F13 so that
41 // J1 = (1-E1)*[2*F13*J2R+F14*J4]+E1*Eb1
    (a)
42 // we use equation (8-108) for the overall energy
    balance on surface 2:

```

```

43 // 2*h*(Tinf-T2) = E2*(Eb2-J2R)/(1-E2)+E2*(Eb2-J2L)
    /(1-E2)
44 // 2*h*(Tinf-T2) = E2*(2*Eb2-J2R-J2L)/(1-E2)
    (b)
45 // equation (8-105) is used for surface J2R.
46 // J2R = (1-E2)*(F21*J1+F23*J3L+F24_R*J4)+E2*Eb2
47 // But J2R = J3L so that
48 // J2R = [(1-E2)*(F21*J1+F24_R*J4)+E2*Eb2]/(1-(1-E2)
    *F23) (c)
49 // for surface J2L the equation is
50 // J2L = (1-E2)*(F24_L*J4)+E2*Eb2
    (d)
51 // we now have four equations with four unknowns, J1
    ,J2R,J2L,Eb2, with T2 = (Eb2/sigma)^(1/4).
52 // however equation (b) is nonlinear in Eb so we
    must use a special procedure to solve the set.
53 for T2 = 300:0.1:400
54     Z = [1 -(1-E1)*2*F13 0 0;0 -E2/(1-E2) -E2/(1-E2)
        2*E2/(1-E2);(1-E2)*F21/(1-(1-E2)*F23) -1 0
        E2/(1-(1-E2)*F23);0 0 1 -E2];
55     C = [E1*Eb1;2*h*(Tinf-T2);-F24_R/(1-(1-E2)*F23)
        ;(1-E2)*F24_L];
56     S = Z^(-1)*C;
57     Eb2_E = S(4);
58     Eb2_T = sigma*T2^(4);
59     dEb2 = Eb2_E-Eb2_T;
60     if (dEb2>0 & dEb2<5) then
61         J1 = S(1);// [W/square meter]
62         J2R = S(2);// [W/square meter]
63         J2L = S(3);// [W/square meter]
64         Eb2 = S(4);// [W/square meter]
65         T2new = T2;// [K]
66     end
67 end
68 // the total heat flux lost by surface 1 is
69 q1_by_A1 = h*(T1-Tinf)+(Eb1-J1)*E1/(1-E1);// [W/
    square meter]
70 // for a 0.5 by 0.5 m surface the heat lost is thus

```

```

71 q1 = q1_by_A1*l*b; // [W]
72 printf("\n\n the heat lost by plate is %f W",q1);

```

---

**Scilab code Exa 8.20** solar environment equilibrium temperatures

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 8.20\n\n\n");
4 // solar-environment equilibrium temperatures
5 // Example 8.20 (page no.-454)
6 // solution
7
8 q_by_A_sun = 700; // [W/m^(2)] solar flux
9 T_surr = 25+273; // [K] surrounding temperature
10 sigma = 5.669*10^(-8); // [W/square meter K^(4)]
11 // at radiation equilibrium the netenergy absorbed
    from sun must equal the long-wavelength radiation
    exchange with the surroundings, or
12 // (q_by_A_sun)*alpha_sun = alpha_low_temp*sigma*(T
    ^4-T_surr^4) (a)
13
14 // case (a) for white paint
15
16 // for white paint we obtain from table 8-4
17 alpha_sun = 0.12;
18 alpha_low_temp = 0.9;
19 // so that equation (a) becomes
20 T = [(q_by_A_sun)*alpha_sun/(alpha_low_temp*sigma)+
    T_surr^4]^(1/4); // [K]
21 printf("radiation equilibrium temperature for the
    plate exposed to solar flux if the surface is
    coated with white paint is %f degree celsius",T
    -273);
22
23 // case (b) for flat black lacquer we obtain

```

```

24
25 alpha_sun = 0.96;
26 alpha_low_temp = 0.95;
27 // so that equation (a) becomes
28 T = [(q_by_A_sun)*alpha_sun/(alpha_low_temp*sigma)+
      T_surr^(4)]^(1/4); // [K]
29 printf("\n\nradiation equilibrium temperature for
      the plate exposed to solar flux if the surface is
      coated with flat black lacquer is %f degree
      celsius",T-273);

```

---

**Scilab code Exa 8.21** influence of convection on solar equilibrium temperature

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 8.21\n\n\n");
4 // influence of convection on solar equilibrium
   temperature
5 // Example 8.21 (page no.-455)
6 // solution
7
8 T_surr = 25+273; // [K] surrounding temperature
9 sigma = 5.669*10^(-8); // [W/square meter K^(4)]
10 h = 10; // [W/square meter] heat transfer coefficient
11 // in this case the solar energy absorbed must equal
   the sum of the radiation and convection
   transfers to the surroundings
12 // (q_by_A_sun)*alpha_sun = alpha_low_temp*sigma*(T
   ^4-T_surr^4)+h*(T-T_surr) (a)
13 q_by_A_sun = 700; // [W/m^(2)] solar flux
14
15 // for the white paint, using the same surface
   properties as in example 8-20 gives
16

```

```

17 alpha_sun = 0.12;
18 alpha_low_temp = 0.9;
19 // so that equation (a) becomes
20 deff(' [y] = f(T)', 'y = (q_by_A_sun)*alpha_sun -
      alpha_low_temp*sigma*(T^4-T_surr^4)-h*(T-T_surr)');
21 T = fsolve(1,f);
22 printf("the radiation-convection equilibrium
      temperatures for case (a) is %f degree celsius",T
      -273);
23
24 //for flat black lacquer we obtain
25
26 alpha_sun = 0.96;
27 alpha_low_temp = 0.95;
28 // so that equation (a) becomes
29 deff(' [y] = f2(T1)', 'y = (q_by_A_sun)*alpha_sun -
      alpha_low_temp*sigma*(T1^4-T_surr^4)-h*(T1-T_surr
      )');
30 T1 = fsolve(1,f2);
31 printf("\n\n the radiation-convection equilibrium
      temperatures for case (b) is %f degree celsius",
      T1-273);
32 printf("\n\n where case (a)      surface is coated
      with white paint");
33 printf("\n\n      case (b)      surface is coated
      with flat black lacquer");

```

---

**Scilab code Exa 8.23** temperature measurement error caused by radiation

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 8.23\n\n\n");
4 // temperature measurement error caused by radiation
5 // Example 8.23 (page no.-460)

```

```

6 // solution
7
8 E = 0.9; // emissivity of mercury-in-glass
    thermometer
9 Tt = 20+273; // [K] temperature indicated by
    thermometer
10 Ts = 5+273; // [K] temperature of walls
11 sigma = 5.669*10^(-8); // [W/square meter K^(4)]
12 h = 8.3; // [W/square meter] heat transfer
    coefficient for thermometer
13 // we employ equation(8-113) for the solution: h*(
    Tinf-Tt) = sigma*E*(Tt^4-Ts^4)
14 // inserting the values in above equation
15 Tinf = sigma*E*(Tt^4-Ts^4)/h+Tt; // [K]
16 printf("the true air temperature is %f degree
    celsius",Tinf-273);

```

---

# Chapter 9

## Condensation and Boiling Heat Transfer

Scilab code Exa 9.1 condensation on vertical plate

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 9.1\n\n\n");
4 // condensation on vertical plate
5 // Example 9.1(page no.-492)
6 // solution
7
8 // we have to check the reynolds no. to that film is
   laminar or turbulent
9 Tf = (100+98)/2;// [degree celsius]
10 Tw = 98;// [degree celsius]
11 RHOf=960;// [kg/cubic meter]
12 MUf=2.82*10(-4);// [kg/m s]
13 Kf=0.68;// [W/m degree celsius]
14 g=9.81;// [m/s(2)]
15 L=0.3;// [m]
16 // RHOf(RHOf-RHOv)~RHOf(2)
17 // let us assume laminar film condensate
18 Tsat=100;// [degree celsius]
```

```

19 Tg=100; // [degree celsius]
20 Hfg=2255*10^(3); // [J/kg]
21 hbar=0.943*((RHOf^(2)*g*Hfg*Kf^(3)/(L*MUf*(Tg-Tw)))
    ^(0.25)); // [W/square meter degree celsius]
22 h=hbar; // [W/square meter degree celsius]
23 // checking reynolds no. with equation(9-17)
24 Ref=4*h*L*(Tsat-Tw)/(Hfg*MUf);
25 printf("value of reynolds no. is %f \n\n so the
    laminar assumption was correct ",Ref);
26 // the heat transfer is now calculated from
27 A=0.3*0.3; // [square meter]
28 q=hbar*A*(Tsat-Tw); // [W]
29 mdot=q/Hfg; // [kg/h]
30 printf("\n\n the heat transfer is %f w",q);
31 mdot=mdot*3600; // [kg/h]
32 printf("\n\n total mass flow condensate is %f kg/h",
    mdot);

```

---

### Scilab code Exa 9.2 condensation on tube tank

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 9.2\n\n\n");
4 // condensation on tube tank
5 // Example 9.2(page no.-493)
6 // solution
7
8 // the condensate properties are obtained from
    previous example
9 // replacing L by n*d
10 Tw=98; // [degree celsius]
11 RHOf=960; // [kg/cubic meter]
12 MUf=2.82*10^(-4); // [kg/m s]
13 Kf=0.68; // [W/m degree celsius]
14 g=9.81; // [m/s^(2)]

```



```

15 Tsat=100; // [degree celsius]
16 Tg=100; // [degree celsius]
17 Hfg=2255*10^(3); // [J/kg]
18 d=0.0127; // [m]
19 n=10;
20 hbar=0.725*((RHOf^(2)*g*Hfg*Kf^(3)/(n*d*MUf*(Tg-Tw))
    )^(0.25)); // [W/square meter degree celsius]
21 // total surface area is
22 n=100;
23 A1=n*22*d/7; // [square meter]
24 printf("total surface area is %f square meter/m",A1)
    ;
25 // so the heat transfer is
26 Q1=hbar*A1*(Tg-Tw); // [W]
27 printf("\n\n heat transfer is %f kW/m",Q1/1000);
28 // total mass flow of condensate is then
29 mdot1=Q1/Hfg; // [kg/h]
30 mdot1=mdot1*3600; // [kg/h]
31 printf("\n\n total mass flow of condensate is %f kg/
    h",mdot1);

```

---

### Scilab code Exa 9.3 boiling on brass plate

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 9.3\n\n\n");
4 // boiling on brass plate
5 // Example 9.3(page no.-501-502)
6 // solution
7
8 Qawater_platinum=946.1; // [kw/square meter] from
    figure (9-8) heat flux for water platinum
    combination
9 Tw=117; // [degree celsius]
10 Tsat=100; // [degree celsius]

```

```

11 // from table (9-2)
12 Csfwater_platinum=0.013; // for water platinum
13 Csfwater_brass=0.006; // for water brass
14 def('y = G(Qawater_brass)', 'y = (((Qawater_brass)
    /((Qawater_platinum))) - ((Csfwater_platinum/
    Csfwater_brass)^(3)))');
15 Qawater_brass = fsolve(0,G);
16 printf("heat transfer for water brass system is %f W
    /square meter",Qawater_brass);

```

---

#### Scilab code Exa 9.4 Flow boiling

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 9.4\n\n\n");
4 // Flow boiling
5 // Example 9.4(page no.-506)
6 // solution
7
8 p = 5*101325/10^(6); // [MPa] pressure of water
9 d = 0.0254; // [m] diameter of tube
10 Tw = 10; // [degree celsius]
11 // for calculation we use equation (9-45), noting
    that
12 dT = 10; // [degree celsius]
13 // the heat transfer coefficient is calculated as
14 h = 2.54*Tw^(3)*exp(p/1.551); // [W/square meter
    degree celsius]
15 // the surface area for a 1-m length of tube is
16 L = 1; // [m]
17 A = %pi*d*L; // [square meter]
18 // so the heat transfer is
19 q = h*A*dT; // [W/m]
20 printf("the heat transfer in a 1.0 m length of tube
    is %f W/m",q);

```

---

**Scilab code Exa 9.5** water boiling in a pan

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 9.5\n\n\n");
4 // water boiling in a pan
5 // Example 9.5(page no.-506-507)
6 // solution
7
8 p = 101325/10^(6); // [MPa] pressure of water
9 dT_x = 8; // [degree celsius]
10 p1 = 0.17; // [MPa] given operating pressure
11 // we will use the simplified relation of table
    // 9-13(page no.-506) for the estimates.we do not
    // know the value of q_by_A and so must choose one
    // of the two relation for a horizontal surface from
    // the table
12 // we anticipate nucleate boiling , so choose
13 h = 5.56*dT_x^(3); // [W/square meter degree celsius]
14 // and the heat flux is
15 q_by_A = h*dT_x; // [W/square meter]
16 // for operation as a pressure cooker we obtain the
    // value of h from equation(9-44)
17 hp = h*(p1/p)^(0.4); // [W/square meter degree
    // celsius]
18 // the corresponding heat flux is
19 q_by_A1 = hp*dT_x; // [W/square meter]
20 printf("heat flux obtained is %f kW/square meter",
    q_by_A/1000);
21 per_inc = 100*(q_by_A1-q_by_A)/q_by_A;
22 printf("\n\n if the pan operates as a pressure
    // cooker at 0.17 MPa the increase in heat flux is
    // %f percent",per_inc);
```

---

### Scilab code Exa 9.6 heat flux comparisons

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 9.6\n\n\n");
4 // heat-flux comparisons
5 // Example 9.6(page no.-509)
6 // solution
7
8 Tw = 200; // [degree celsius] water temperature
9 L = 0.08; // [m] length of solid copper bar
10 dT = 100; // [degree celsius] temperature
    differential in copper bar
11 //using the data of table 9-4(page no.-508)
12 // the heat flux per unit area is expressed as
    q_by_A = -k*del_T/dx
13 // from table A-2(page no.-) the thermal
    conductivity of copper is
14 k = 374; // [W/m degree celsius]
15 q_by_A = -k*(-dT)/L; // [W/square meter]
16 // from table 9-4(page no.-508) the typical axial
    heat flux for a water heat flux for a water heat
    pipe is
17 q_by_A_axial = 0.67; // [kW/csquare meter]
18 q_by_A = q_by_A/(1000*10^(4)); // [kW/csquare meter]
19 printf("thus the heat transfers more than %f times
    the heat of a pure copper rod with a substantial
    temperature gradient.",q_by_A_axial/q_by_A);
```

---

# Chapter 10

## Heat Exchangers

Scilab code Exa 10.1 overall heat transfer coefficient for pipe in air

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 10.1\n\n");
4 // overall heat transfer coefficient for pipe in air
5 // Example 10.1 (page no.-520-522)
6 // solution
7
8 Tw = 98; // [degree celsius] temperature of hot water
9 k_p = 54; // [W/m degree celsius] heat transfer
   coefficient of pipe
10 Ta = 20; // [degree celsius] atmospheric air
   temperature
11 u = 0.25; // [m/s] water velocity
12 // from appendix A the dimensions of 2-in schedule
   40 pipe are
13 ID = 0.0525; // [m]
14 OD = 0.06033; // [m]
15 // the properties of water at 98 degree celsius are
16 rho = 960; // [kg/cubic meter]
17 mu = 2.82*10^(-4); // [kg/m s]
18 k_w = 0.68; // [W/m degree celsius]
```

```

19 Pr = 1.76; // prandtl number
20 // the reynolds number is
21 Re = rho*u*ID/mu;
22 // and since turbulent flow is encountered, we may
    use equation(6-4):
23 Nu = 0.023*Re^(0.8)*Pr^(0.4);
24 hi = Nu*k_w/ID; // [W/square meter degree celsius]
25 // for unit length of pipe the thermal resistance of
    the steel is
26 Rs = log(OD/ID)/(2*pi*k_p);
27 // again, on a unit length basis the thermal
    resistance on the inside is
28 Ai = pi*ID; // [square meter]
29 Ri = 1/(hi*Ai);
30 Ao = pi*OD; // [square meter]
31 // the thermal resistance for outer surface is as
    yet unknown but is written, for unit lengths, is
    Ro = 1/(ho*Ao) (a)
32 // from table 7-2(page no.-339), for laminar flow,
    the simplified relation for ho is
33 // ho = 1.32*(dT/d)^(1/4) = 1.32*((To-Ta)/OD)^(1/4)

    (b)
34 // where To is the unknown outside pipe surface
    temperature. we designate the inner pipe surface
    as Ti and the water temperature as Tw; then the
    energy balance requires
35 // (Tw-Ti)/Ri = (Ti-To)/Rs = (To-Ta)/Ro

    (c)
36 // combining equations (a) and (b) gives
37 // (To-Ta)/Ro = pi*OD*1.32*(To-Ta)^(5/4)/OD^(1/4)
38 // this relation may be introduced into equation (c)
    to yield two equations with the two unknowns Ti
    and To:
39
40 // (Tw-Ti)/Ri = (Ti-To)/Rs (1)
41 // (Ti-To)/Rs = pi*OD*1.32*(To-Ta)^(5/4)/OD^(1/4)

```

```

                                (2)
42 // this is a non-linear equation which can be solved
    as
43 for Ti = 50:0.001:100
44     Q = ((Ti-(Ti-(Tw-Ti)*(Rs/Ri)))/Rs)-(%pi*OD
          *1.32*((Ti-(Tw-Ti)*(Rs/Ri))-Ta)^(5/4)/OD
          ^(1/4));
45     if Q>0 & Q<6 then
46         Tinew = Ti;
47     else
48         Ti = Ti;
49     end
50 end
51 Ti = Tinew;// [degree celsius]
52 To = (Ti-(Tw-Ti)*(Rs/Ri));// [Degree celsius]
53 // as a result , the outside heat transfer
    coefficient and thermal resistance are
54 ho = 1.32*((To-Ta)/OD)^(1/4);// [W/square meter
    degree celsius]
55 Ro = 1/(OD*7.91*%pi);//
56 // the overall heat transfer coefficient based on
    the outer area is written in terms of these
    resistances as
57 Uo = 1/(Ao*(Ri+Ro+Rs));// [W/area degree celsius]
58 // in this calculation we used the outside area for
    1.0 m length as Ao
59 // so
60 Uo = Uo;// [W/square meter degree celsius]
61 printf("overall heat transfer coefficient is %f W/
    square meter degree celsius",Uo);

```

---

**Scilab code Exa 10.2** overall heat transfer coefficient for pipe exposed to steam

```
1 clear;
```

```

2  clc;
3  printf("\\t\\t\\tExample Number 10.2\\n\\n\\n");
4  // overall heat transfer coefficient for pipe
   exposed to steam
5  // Example 10.2 (page no.-523-524)
6  // solution
7
8  p = 101325; // [Pa] pressure of steam
9  Tg = 100; // [degree celsius] temperature of steam
10 // we have already determined the inside convection
   heat-transfer coefficient in example(10.1) as
11 hi = 1961; // [W/square meter]
12 // the water film properties are
13 rho = 960; // [kg/cubic meter] density
14 mu_f = 2.82*10(-4); // [kg/m s]
15 kf = 0.68; // [W/m degree celsius]
16 hfg = 2255*10(3); // [J/kg]
17 g = 9.8; // [m/s(2)] acceleration due to gravity
18 d = 0.06033; // [m] diameter of the pipe
19 // the convection coefficient for condensation on
   the outside of the pipe is obtained by using
   equation(9-12)
20 // h_o = 0.725*[(rho(2)*g*hfg*kf(3))/(mu_f*d*(Tg-
   To))]^(1/4) (a)
21 Ao = %pi*d; // [square meter] outside area
22 // outside thermal resistance per unit length is
23 // R_o = 1/(h_o*A_o)

   (b)
24 // the energy balance requires
25 // [Tg-To]/R_o = [To-Ti]/R_s = [Ti-Tw]/R_i

   (c)
26 // from example 10.1 we have
27 Ri = 3.092*10(-3);
28 Rs = 4.097*10(-4);
29 Tw = 98; // [degree celsius]
30 // equation (b) and (c) may be combined to give

```



```

31 // (Tg-To)^(3/4)/3403 = (To-Ti)/Rs          (1)
32 // (To-Ti)/Rs = (Ti-Tw)/Ri                (2)
33 // this is a non-linear equation which can be solved
    as
34 for Ti = 98.1:0.01:99.75
35     P = ((Tg-(Ti+Rs*(Ti-Tw)/Ri))^(3/4))*3403-(((Ti+
        Rs*(Ti-Tw)/Ri)-Ti)/Rs);
36     if P>(-10) & P<0 then
37         Tnew = Ti;
38     else
39         Ti = Ti;
40     end
41
42 end
43 Ti = Tnew; // [degree celsius]
44 To = (Ti+Rs*(Ti-Tw)/Ri); // [degree celsius]
45 // the exterior heat-transfer coefficient and
    thermal resistance then become
46 ho = 0.725*[(rho^(2)*g*hfg*kf^(3))/(mu_f*d*(Tg-To))
    ]^(1/4); // [W/square meter degree celsius]
47 Ro = 1/(ho*Ao);
48 // based on unit length of pipe, the overall heat
    transfer coefficient is
49 Uo = 1/(Ao*(Ri+Ro+Rs)); // [W/area degree celsius]
50 // since Ao and the R's were per unit length
51 // so
52 Uo = Uo; // [W/square meter degree celsius]
53 printf("overall heat transfer coefficient is %f W/
    square meter degree celsius",Uo);

```

---

### Scilab code Exa 10.3 influence of fouling factor

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 10.3\n\n");

```

```

4 // influence of fouling factor
5 // Example 10.2 (page no.-524-525)
6 // solution
7
8 // the fouling factor influences the heat transfer
  coefficient on the inside of the pipe. we have
9 Rf = 0.0002;
10 // using
11 h_clean = 1961; // [W/square meter degree celsius]
12 // we obtain
13 hi = 1/[Rf+(1/h_clean)]; // [W/square meter degree
  celsius]
14 printf("the percent reduction because of fouling
  factor is %f ",(h_clean-hi)*100/h_clean);

```

---

**Scilab code Exa 10.4** calculation of heat exchanger size from known temperatures

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 10.4\n\n\n");
4 // calculation of heat exchanger size from known
  temperatures
5 // Example 10.4 (page no.-532-533)
6 // solution
7
8 m_dot = 68; // [kg/min] water flow rate
9 U = 320; // [W/square meter degree celsius] overall
  heat transfer coefficient
10 T1 = 35; // [degree celsius] initial temperature
11 T2 = 75; // [degree celsius] final temperature
12 Toe = 110; // [degree celsius] oil entering
  temperature
13 Tol = 75; // [degree celsius] oil leaving temperature
14 Cw = 4180; // [J/kg degree celsius] water specific

```

```

    heat capacity
15 // the total heat transfer is determined from the
    energy absorbed by the water:
16 q = m_dot*Cw*(T2-T1); // [J/min]
17 q = q/60; // [W]
18 // since all the fluid temperatures are known, the
    LMTD can be calculated by using the temperature
    scheme in figure 10-7b(page no.-530)
19 dT_m = ((Toe-To1)-(T2-T1))/log((Toe-To1)/(T2-T1)); //
    [degree celsius]
20 // then, since q = U*A*dT_m
21 A = q/(U*dT_m); // [square meter] area of heat-
    exchanger
22 printf("area of heat-exchanger is %f square meter ",
    A);

```

---

#### Scilab code Exa 10.5 shell and tube heat exchanger

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 10.5\n\n\n");
4 // shell-and-tube heat exchanger
5 // Example 10.5 (page no.-533-534)
6 // solution
7
8 // to solve this problem, we determine a correction
    factor from figure 10-8 to be used with the LMTD
    calculated on the basis of counterflow exchanger.
9 // the parameters according to the nomenclature of
    figure 10-8(page no.-532) are
10 T1 = 35; // [degree celsius]
11 T2 = 75; // [degree celsius]
12 t1 = 110; // [degree celsius]
13 t2 = 75; // [degree celsius]
14 P = (t2-t1)/(T1-t1);

```

```

15 R = (T1-T2)/(t2-t1);
16 // so the correction factor is
17 F = 0.81;// from figure 10-10(page no.-534)
18 // and the heat transfer is q = U*A*F*dT_m
19 // so that. from example 10-4 we have
20 U = 320;// [W/square meter degree celsius] overall
    heat transfer coefficient
21 q = 189493.33;// [W]
22 dT_m = 37.44;// [degree celsius]
23 A = q/(U*F*dT_m);// [square meter]
24 printf("area required for this exchanger is %f
    square meter",A)

```

---

#### Scilab code Exa 10.6 design of shell and tube heat exchanger

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 10.6\n\n\n");
4 // design of shell-and-tube heat exchanger
5 // Example 10.5 (page no.-534-536)
6 // solution
7
8 m_dot_c = 3.8;// [kg/s] water flow rate
9 Ti = 38;// [degree celsius] initial temperature of
    water
10 Tf = 55;// [degree celsius] final temperature of
    water
11 m_dot_h = 1.9;// [kg/s] water flow rate entering the
    exchanger
12 Te = 93;// [degree celsius] entering water
    temperature
13 U = 1419;// [W/square meter degree celsius] overall
    heat transfer coefficient
14 d = 0.019;// [m] diameter of tube
15 v_avg = 0.366;// [m/s] average water velocity in

```

```

    exchanger
16 Cc = 4180; // [] specific heat of water
17 Ch = Cc; // [] specific heat
18 rho = 1000; // [kg/cubic meter] density of water
19 // we first assume one tube pass and check to see if
    it satisfies the conditions of this problem. the
    exit temperature of the hot water is calculated
    from
20 dTh = m_dot_c*Cc*(Tf-Ti)/(m_dot_h*Ch); // [degree
    celsius]
21 Th_exit = Te-dTh; // [degree celsius]
22 // the total required heat transfer is obtained for
    the cold fluid is
23 q = m_dot_c*Cc*(Tf-Ti); // [W]
24 // for a counterflow exchanger, with the required
    temperature
25 LMTD = ((Te-Tf)-(Th_exit-Ti))/log((Te-Tf)/(Th_exit-
    Ti)); // [degree celsius]
26 dTm = LMTD; // [degree celsius]
27 A = q/(U*dTm); // [square meter]
28 // using the average water velocity in the tubes and
    the flow rate, we calculate the total area with
29 A1 = m_dot_c/(rho*v_avg); // [square meter]
30 // this area is the product of number of tubes and
    the flow area per tube:
31 n = A1*4/(%pi*d^(2)); // no. of tubes
32 n = ceil(n); // rounding of value of n because no. of
    pipe is an integer value
33 // the surface area per tube per meter of length is
34 S = %pi*d; // [square meter/tube meter]
35 // we recall that the total surface area required
    for a one tube pass exchanger was calculated
    above .
36 // we may thus compute the length of tube for this
    type of exchanger from
37 L = A/(S*n); // [m]
38 // this length is greater than the allowable 2.438 m
    , so we must use more than one tube pass. when we

```

```

    increase the number of passes , we
    correspondingly increase the total surface area
    required because of the reduction in LMTD caused
    by the correction factor F.
39 // we next try two tube passes. from figure 10-8(
    page no.-532)
40 F = 0.88;
41 A_total = q/(U*F*dTm); // [square meter]
42 // the number of tubes per pass is still 37 because
    of the velocity requirement. for the two pass
    exchanger the total surface area is now related
    to the length by
43 L1 = A_total/(2*S*n); // [m]
44 // this length is within the 2.438 m requirement , so
    the final design choice is
45 printf("number of tubes per pass = %f",n);
46 printf("\n\n number of passes = 2");
47 printf("\n\n length of tube per pass = %f m",L1);

```

---

**Scilab code Exa 10.7** cross flow exchanger with one fluid mixed

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 10.7\n\n\n");
4 // cross flow exchanger with one fluid mixed
5 // Example 10.7 (page no.-537)
6 // solution
7
8 m_dot = 5.2; // [kg/s] mass flow rate
9 T1 = 130; // [degree celsius] temperature of entering
    steam
10 T2 = 110; // [degree celsius] temperature of leaving
    steam
11 t1 = 15; // [degree celsius] temperature of entering
    oil

```

```

12 t2 = 85; // [degree celsius] temperature of leaving
    oil
13 c_oil = 1900; // [J/kg degree celsius] heat capacity
    of oil
14 c_steam = 1860; // [J/kg degree celsius] heat
    capacity of steam
15 U = 275; // [W/square meter degree celsius] overall
    heat transfer coefficient
16 //the total heat transfer may be obtained from an
    energy balance on the steam
17 q = m_dot*c_steam*(T1-T2); // [W]
18 // we can solve for the area from equation (10-13).
    the value of dT_m is calculated as if the
    exchanger were counterflow double pipe, thus
19 dT_m = ((T1-t2)-(T2-t1))/log((T1-t2)/(T2-t1)); // [
    degree celsius]
20 // t1,t2 is representing the unmixed fluid(oil) and
    T1,T2 is representing the mixed fluid(steam) so
    that:
21 // we calculate
22 R = (T1-T2)/(t2-t1);
23 P = (t2-t1)/(T1-t1);
24 // consulting figure 10-11(page no.-534) we find
25 F = 0.97;
26 // so the area is calculated from
27 A = q/(U*F*dT_m); // [square meter]
28 printf("surface area of heat exchanger is %f square
    meter",A);

```

---

**Scilab code Exa 10.8** effects of off design flow rates for exchanger in previous example

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 10.8\n\n\n");

```

```

4 // effects of off-design flow rates for exchanger in
   example 10-7
5 // Example 10.8 (page no.-537-538)
6 // solution
7
8 // we did not calculate the oil flow in example 10-7
   but can do so now from
9 q = 193; // [kW]
10 c_oil = 1.9; // [J/kg degree celsius] heat capacity
   of oil
11 t1 = 15; // [degree celsius] temperature of entering
   oil
12 t2 = 85; // [degree celsius] temperature of leaving
   oil
13 m_dot_o = q/(c_oil*(t2-t1)); // [kg/s]
14 // the new flow rate will be half this value
15 m_dot_o = m_dot_o/2; // [kg/s]
16 // we are assuming the inlet temperatures remain the
   same at 130 degree celsius for the steam and 15
   degree celsius for the oil.
17 // the new relation for the heat transfer is  $q =$ 
    $m\_dot\_o * c\_oil * (T_{e,o} - 15) = m\_dot\_s * c_p * (130 - T_{e,s})$ 
   (a)
18 // but the exit temperatures,  $T_{e,o}$  and  $T_{e,s}$  are
   unknown. furthermore,  $dT_m$  is unknown without
   these temperatures, as are the values of R and P
   from figure 10-11(page no.-535). this means we
   must use an iterative procedure to solve for the
   exit temperatures using equation (a) and  $q = U *$ 
    $A * F * dT_m$  (b)
19 // the general procedure is to assume values of the
   exit temperatures until the q's agree between
   equations(a) and (b).
20 printf("the objective of this example is to show
   that an iterative procedure is required when the
   inlet and outlet temperatures are not known or
   easily calculated");
21 printf("\n\n there is no need to go through this

```



iteration because it can be avoided by using the techniques described in section 10-6”);

---

**Scilab code Exa 10.9** off design calculation using E NTU method

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 10.9\n\n\n");
4 // off-design calculation using E-NTU method
5 // Example 10.9 (page no.-542-544)
6 // solution
7
8 m_dot_o = 0.725; // [kg/s] oil flow rate
9 m_dot_s = 5.2; // [kg/s] steam flow rate
10 t1 = 15; // [degree celsius] temperature of entering
    oil
11 T1 = 130; // [degree celsius] temperature of entering
    steam
12 c_oil = 1900; // [J/kg degree celsius] heat capacity
    of oil
13 c_steam = 1860; // [J/kg degree celsius] heat
    capacity of steam
14 // for the steam
15 Cs = m_dot_s*c_steam; // [W/degree celsius]
16 // for the oil
17 Co = m_dot_o*c_oil; // [W/degree celsius]
18 // so the oil is minium fluid. we thus have
19 C_min_by_C_max = Co/Cs;
20 U = 275; // [W/square meter degree celsius] overall
    heat transfer coefficient
21 A = 10.83; // [square meter] surface area of heat
    exchanger
22 NTU = U*A/Co;
23 // we choose to use the table and note that Co(
    minimum) is unmixed and Cs(maximum) is mixed so
```

```

        that the first relation in the table 10-3(page no
        .-543) applies.
24 // we therefore calculate E(effectiveness) as
25 E = (1/C_min_by_C_max)*{1-exp(-C_min_by_C_max*(1-exp
        (-NTU)))};
26 // if we were using figure 10-14(page no.-544) we
        would have to evaluate
27 C_mixed_by_C_unmixed = Cs/Co;
28 // and would still determine
29 E = 0.8;// approximately
30 // now, using the effectiveness we can determine the
        temperature difference of the minimum fluid(oil
        as)
31 dT_o = E*(T1-t1);// [degree celsius]
32 // so that heat transfer is
33 q = m_dot_o*c_oil*(dT_o);// [W]
34 q_initial = 193440;// [W] heat transfer when oil
        flow rate is 100 %
35 printf("we find a reduction in the oil flow rate of
        50 %% causes a reduction in heat transfer of only
        %f %%", (q_initial-q)*100/q_initial);

```

---

**Scilab code Exa 10.10** off design calculation of exchanger in example 10  
4

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 10.10\n\n");
4 // off-design calculation of exchanger in example
        10-4
5 // Example 10.10 (page no.-544-546)
6 // solution
7
8 m_dot_c = 68;// [kg/min] water flow rate
9 T1 = 35;// [degree celsius] initial temperature

```

```

10 T2 = 75; // [degree celsius] final temperature
11 Toe = 110; // [degree celsius] oil entering
    temperature
12 Tol = 75; // [degree celsius] oil leaving temperature
13 Cc = 4180; // [J/kg degree celsius] water specific
    heat capacity
14 Ch = 1900; // [J/kg degree celsius] heat capacity of
    oil
15 U = 320; // [W/square meter degree celsius] overall
    heat transfer coefficient
16 A = 15.814568; // [square meter] area of heat
    exchanger (from example 10-4)
17 // the flow rate of oil is calculated from the
    energy balance for the original problem:
18 m_dot_h = m_dot_c*Cc*(T2-T1)/(Ch*(Toe-Tol)); // [kg/
    min]
19 // the capacity rates for the new conditions are
    calculated as
20 C_h = m_dot_h*Ch/60; // [W/degree celsius]
21 C_c = m_dot_c*Cc/60; // [W/degree celsius]
22 // so that the water (cold fluid) is the minimum
    fluid , and
23 C_min_by_C_max = C_c/C_h;
24 NTU_max = U*A/C_c;
25 // from figure 10-13(page no.-542) or table 10-3(
    page no.-543) the effectiveness is
26 E = 0.744;
27 // and because the cold fluid is the minimum, we can
    write
28 dT_cold = E*(Toe-T1); // [degree celsius]
29 // and the exit water temperature is
30 Tw_exit = T1+dT_cold; // [degree celsius]
31 // the total heat transfer under the new flow
    conditions is calculated as
32 m_dot_c = 40; // [kg/min]
33 q = m_dot_c*Cc*dT_cold/60; // [W]
34 printf("exit water temperature is %f degree celcius"
    ,Tw_exit);

```

```
35 printf("\n\n the total heat transfer under the new
    flow conditions is %f kW",q/1000);
```

---

**Scilab code Exa 10.11** cross flow exchanger with both fluid unmixed

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 10.11\n\n");
4 // cross-flow exchanger with both fluid unmixed
5 // Example 10.11 (page no.-547-549)
6 // solution
7
8 pa = 101325; // [Pa] pressure of air
9 Ti = 15.55; // [degree celsius] initial temperature
    of air
10 Tf = 29.44; // [degree celsius] final temperature of
    air
11 Thw = 82.22; // [degree celsius] hot water
    temperature
12 U = 227; // [W/square meter degree celsius] overall
    heat transfer coefficient
13 S = 9.29; // [square meter] total surface area of
    heat exchanger
14 R = 287; // [] universal gas constant
15 Cc = 1006; // [J/kg degree celsius] specific heat of
    air
16 Ch = 4180; // [J/kg degree celsius] specific heat of
    water
17 // the heat transfer is calculated from the energy
    balance on the air. first, the inlet air density
    is
18 rho = pa/(R*(Ti+273.15)); // [kg/cubic meter]
19 // so the mass flow of air (the cold fluid) is
20 mdot_c = 2.36*rho; // [kg/s]
21 // the heat transfer is then
```

```

22 q = mdot_c*Cc*(Tf-Ti); // [W]
23 // from the statement of the problem we do not know
    whether the air or water is the minimum fluid. a
    trial and error procedur must be used with figure
    10-15(page no.-545) or table 10-3(page no.-543)
    .
24 // we assume that the air is the minimum fluid and
    then check out our assumption. then
25 Cmin = mdot_c*Cc; // [W/degree celsius]
26 NTU_max = U*S/Cmin;
27 // and the effectiveness based on the air as the
    minimum fluid is
28 E = (Tf-Ti)/(Thw-Ti);
29 // entering figure 10-15, we are unable to match
    these quantities with the curves. this require
    that the hot fluid be the minimum. we must
    therefore assume values for the water flow rate
    until we are able to match the performance as
    given by figure 10-15 or table 10-3. we first
    note that
30 Cmax = mdot_c*Cc; // [W/degree celsius] (a)
31 // NTU_max = U*S/Cmin; (b)
    )
32 // E = dT_h/(Thw-Ti) (c)
33 // dT_h = q/Cmin (d)
34
35 // now we assume different values for Cmin abd
    calculate different-different values for NTU_max,
    dT_h, and E
36
37 // for
38 Cmin_by_Cmax1 = 0.5;
39 Cmin1 = Cmin_by_Cmax1*Cmax; // [W/degree celsius]
40 NTU_max1 = U*S/Cmin1;
41 dT_h1 = q/Cmin1; // [degree celsius]
42 E1_c1 = dT_h1/(Thw-Ti); // calculated
43 E1_t1 = 0.65; // from table
44

```

```

45 // for
46 Cmin_by_Cmax2 = 0.25;
47 Cmin2 = Cmin_by_Cmax2*Cmax; // [W/degree celsius]
48 NTU_max2 = U*S/Cmin2;
49 dT_h2 = q/Cmin2; // [degree celsius]
50 E1_c2 = dT_h2/(Thw-Ti); // calculated
51 E1_t2 = 0.89; // from table
52
53 // for
54 Cmin_by_Cmax3 = 0.22;
55 Cmin3 = Cmin_by_Cmax3*Cmax; // [W/degree celsius]
56 NTU_max3 = U*S/Cmin3;
57 dT_h3 = q/Cmin3; // [degree celsius]
58 E1_c3 = dT_h3/(Thw-Ti); // calculated
59 E1_t3 = 0.92; // from table
60
61 // we estimate the water-flow rate as about
62 Cmin = 660; // [W/degree celsius]
63 mdot_h = Cmin/Ch; // [kg/s]
64 // the exit water temperature is accordingly
65 Tw_exit = Thw-q/Cmin; // [degree celsius]
66 printf("the exit water temperature is %f degree
        celsius", Tw_exit);
67 printf("\n\n the heat transfer is %f kW", q/1000);

```

---

**Scilab code Exa 10.12** comparison of single or two exchanger options

```

1 clear;
2 clc;
3 printf("\t\t\t\tExample Number 10.12\n\n\n");
4 // comparison of single- or two-exchanger options
5 // Example 10.12 (page no.-549-551)
6 // solution
7
8 mdot_c = 1.25; // [kg/s] water flow rate

```

```

9  Ti = 35; // [degree celsius] initial temperature of
    water
10 Tf = 80; // [degree celsius] final temperature of
    water
11 Toi = 150; // [degree celsius] initial temperature of
    oil
12 Tof = 85; // [degree celsius] final temperature of
    oil
13 U = 850; // [W/square meter degree celsius] overall
    heat transfer coefficient
14 Cp_water = 4180; // [] specific heat of water
15 Cp_oil = 2000; // [J/kg degree celsius]
16 // we calculate the surface area required for both
    alternatives and then compare costs. for the one
    large exchanger
17 q = mdot_c*Cp_water*(Tf-Ti); // [W]
18 mdot_c_into_Cp_water = mdot_c*Cp_water; // [W/degree
    celsius]
19 mdot_h_into_Cp_oil = q/(Toi-Tof); // [W/degree
    celsius]
20 Cmin = mdot_h_into_Cp_oil; // [W/degree celsius]
21 Cmax = mdot_c_into_Cp_water; // [W/degree celsius]
22 // so that oil is the minimum fluid:
23 Eh = (Toi-Tof)/(Toi-Ti);
24 Cmin_by_Cmax = Cmin/Cmax;
25 // from figure 10-13(page no.-542),
26 NTU_max = 1.09;
27 A = NTU_max*Cmin/U; // [square meter]
28 // we now wish to calculate the surface-area
    requirement for the two small exchanger because U
    *A and Cmin are the same for each exchanger.
29 // this requires that the effectiveness be the same
    for each exchanger. thus,
30 // E1 = (Toi-Toe_1)/(Toi-Ti) = E2 = (Toi-Toe_2)/(Toi
    -Tw2)
    (a)
31 // where the nomenclature for the temperatures is
    indicated in the sketch. because the oil flow is

```

```

    the same in each exchanger and the average exit
    oil temperature must be 85 degree celsius , we may
    write
32 // (Toe_1+Toe_2)/2 = 85

    (b)
33 // an energy balance on the second heat exchanger
    gives
34 //  $\dot{m}_c c_{p,water} (T_f - T_{w2}) =$ 
     $\dot{m}_h c_{p,oil} (T_{oi} - T_{oe_2}) / 2$ 
    (c)
35 // we now have three equations (a), (b), and (c)
    which may be solved for the three unknowns Toe_1,
    Toe_2, and Tw2.
36 // eliminating Tw2, and Toe_1 from equation (a) by
    the help of equation (b) and (c)
37 deff(' [y] = H(Toe_2)', 'y = (T_{oi} - (170 - Toe_2)) / (T_{oi} - T_{i}) - (T_{oi} - Toe_2) / (T_{oi} - (T_f - (\dot{m}_h c_{p,oil} (T_{oi} - Toe_2) / (\dot{m}_c c_{p,water} * 2))))');
38 Toe_2 = fsolve(1,H); // [degree celsius]
39 Toe_1 = (170 - Toe_2); // [degree celsius]
40 Tw2 = (T_f - (\dot{m}_h c_{p,oil} (T_{oi} - Toe_2) / (\dot{m}_c c_{p,water} * 2))); // [degree celsius]
41 // the effectiveness can then be calculated as
42 E1 = (T_{oi} - Toe_1) / (T_{oi} - T_{i});
43 E2 = E1;
44 // from figure 10-13 (page no.-542), we obtain
45 NTU_max = 1.16;
46 // so that
47 A1 = NTU_max * C_min / (U * 2); // [square meter]
48 printf("we have find that %f square meter of area is
    required for each of small exchangers , or a
    total of %f square meter", A1, 2 * A1);
49 printf("\n\n the area required in the one larger
    exchanger is %f square meter", A);
50 printf("\n\n the cost per unit area is greater so
    that the most economical choice would be the
    single larger exchanger ");

```



---

**Scilab code Exa 10.13** shell and tube exchanger as air heater

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 10.13\n\n");
4 // shell and tube exchanger as air heater
5 // Example 10.13 (page no.-551-552)
6 // solution
7
8 To = 100; // [degree celsius] temperature of hot oil
9 m_dot_a = 2; // [kg/s] flow rate of air
10 T1 = 20; // [degree celsius] initial temperature of
    air
11 T2 = 80; // [degree celsius] final temperature of air
12 Cp_o = 2100; // [J/kg degree celsius] specific heat
    of the oil
13 Cp_a = 1009; // [J/kg degree celsius] specific heat
    of the air
14 m_dot_o = 3; // [kg/s] flow rate of oil
15 U = 200; // [W/square meter] overall heat transfer
    coefficient
16 // the basic energy balance is m_dot_o*Cp_o*(To-Toe)
    = m_dot_a*Cp_a*(T2-T1)
17 Toe = To - m_dot_a*Cp_a*(T2-T1)/(m_dot_o*Cp_o); // [
    degree celsius]
18 // we have
19 m_dot_h_into_Ch = m_dot_o*Cp_o; // [W/degree celsius]
20 m_dot_c_into_Cc = m_dot_a*Cp_a; // [W/degree celsius]
21 // so the air is minimum fluid
22 C = m_dot_c_into_Cc/m_dot_h_into_Ch;
23 // the effectiveness is
24 E = (T2-T1)/(To-T1);
25 // now we may use either figure 10-16(page no.-546)
    or the analytical relation from table 10-4(page
```

```

    no.-543) to obtain NTU.
26 // for this problem we choose to use the table
27 NTU = -(1+C^(2))^(-1/2)*log((2/E-1-C-(1+C^2)^(1/2))
    /(2/E-1-C+(1+C^2)^(1/2)));
28 // now, we calculate the area as
29 A = NTU*m_dot_c_into_Cc/U;// [square meter]
30 printf("area required for the heat exchanger is %f
    square meter",A);

```

---

#### Scilab code Exa 10.14 ammonia condenser

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 10.14\n\n\n");
4 // ammonia condenser
5 // Example 10.14 (page no.-552-553)
6 // solution
7
8 Ta = 50;// [degree celsius] temperature of entering
    ammonia vapour
9 Tw1 = 20;// [degree celsius] temperature of entering
    water
10 q = 200;// [kW] total heat transfer required
11 U = 1;// [kW/square meter degree celsius] overall
    heat transfer coefficient
12 Tw2 = 40;// [degree celsius] temperature of exiting
    water
13 Cw = 4.18;// [kJ/kg degree celsius] specific heat of
    water
14 // the mass flow can be calculated from the heat
    transfer with
15 m_dot_w = q/(Cw*(Tw2-Tw1));// [kg/s]
16 // because this is the condenser the water is the
    minimum fluid and
17 C_min = m_dot_w*Cw;// [kW/degree celsius]

```

```

18 // the value of NTU is obtained from the last entry
    of table 10-4(page no.-543), with
19 E = 0.6; // effectiveness
20 NTU = -log(1-E);
21 // so that area is calculated as
22 A = C_min*NTU/U; // [square meter]
23 // when the flow rate is reduced in half the new
    value of NTU is
24 NTU1 = U*A/(C_min/2);
25 // and the effectiveness is computed from the last
    entry of table 10-3(page no.-543):
26 E1 = 1-exp(-NTU1);
27 // the new water temperature difference is computed
    as
28 dT_w = E1*(Ta-Tw1); // [degree celsius]
29 // so that the heat transfer is
30 q1 = C_min*dT_w/2; // [kW]
31 printf("the area to achieve a heat exchanger
    effectiveness of 60%% with an exit water
    temperature of 40 degree celsius is %f square
    meter",A);
32 printf("\n\n by reducing the flow rate we have
    lowered the heat transfer by %d percent",(q-q1)
    *100/q);

```

---

**Scilab code Exa 10.15** crossflow exchanger as energy conservation device

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 10.15\n\n\n");
4 // crossflow exchanger as energy conservation device
5 // Example 10.15 (page no.-553-555)
6 // solution
7
8 q = 210000; // [W] heat to be removed from

```

```

    atmospheric air
9  m_dot_h = 1200/60; // [kg/s] hot air flow rate
10 m_dot_c = m_dot_h; // [kg/s] cold air flow rate
11 Ta1 = 25; // [degree celsius] atmospheric air
    temperature
12 Ta2 = 0; // [degree celsius] temperature of air
    entering from out-door conditions
13 U = 30; // [W/m degree celsius] overall heat transfer
    coefficient
14 Cp = 1005; // [J/kg degree celsius] specific heat of
    air
15
16 //*****calculation 1. the design value for
    the area of the heat exchanger *****/
17
18 // the hot and cold fluids have the same flow rate
19 // and
20 Ch = m_dot_h*Cp; // [W/degree celsius]
21 Cc = m_dot_c*Cp; // [W/degree celsius]
22 Cmin_by_Cmax = 1; // for use in table 10-3(page no
    .-543)
23 // the energy balance gives  $q = Ch*dT_h = Cc*dT_c$ 
24 // and
25 dT_h = q/Ch; // [degree celsius]
26 dT_c = q/Cc; // [degree celsius]
27 // the heat exchanger effectiveness is
28 E = dT_h/(Ta1-Ta2);
29 // consulting table 10-3(page no.-543) for a cross
    flow exchanger with both fluids unmixed, and
    inserting the value
30 C = 1;
31 // we have
32 deff(' [y] = f(N)', 'y = E-1+exp(N^(0.22))*(exp(-N
    ^ (0.78))-1)');
33 N = fsolve(1,f);
34 // solving above to get the value of NTU
35 // area is
36 A = N*Ch/U; // [square meter]

```

```

37 printf("the design value for the area of heat
    exchanger is %f square meter",A);
38
39 //*****calculation 2. the percent reduction
    in heat transfer rate if the flow rate is reduced
    by 50% while keeping the inlet temperatures and
    value of U constant *****//
40
41 // we now examine the effect of reducing the flow
    rate by half, while keeping the inlet
    temperatures and value of U the same.
42 // note that the flow rate of both fluids is reduced
    because they are physically the same fluid. this
    means that the value of Cmin_by_Cmax will remain
    the same at a value of 1.0.
43 // the new value of Cmin is
44 Cmin = Cc/2;// [W/degree celsius]
45 // so that NTU is
46 N = U*A/Cmin;
47 // equation (b) may be used for the calculation of
    effectiveness
48 E = 1-exp(N^(0.22)*(exp(-N^(0.78))-1));
49 // the temperature difference for each fluid is then
50 dT = E*(Ta1-Ta2);// [degree celsius]
51 // the resulting heat transfer is then
52 q_dot = m_dot_c*Cp*dT/2;// [W]
53 printf("\n\nthe percent reduction in heat transfer
    rate if the flow rate is reduced by 50%% is %f "
    ,(q-q_dot)*100/q);
54
55 //*****calculation 3. the percent reduction
    in heat transfer rate if the flow rate is reduced
    by 50% and the value of U varies as mass flow to
    the 0.8 power, with the same inlet temperature
    conditions
56
57 // finally, we examine the effect of reducing the
    flow rate by 50 percent coupled with reduction in

```

```

    overall heat-transfer coefficient under the
    assumption that U varies as  $m_{\dot{}}^{(0.8)}$  or,
    correspondingly, as  $C_{\min}^{(0.8)}$ 
58 // still keeping the area constant, we would find
    that NTU varies as  $N = U \cdot A / C_{\min} \sim C^{(0.8)} \cdot C^{(-1)}$ 
    =  $C^{(-0.2)}$ 
59 // our new value of N under these conditions would
    be
60 N1 = 0.8*(Cmin/Cc)^(-0.2);
61 // inserting this value in equation (b) above for
    the effectiveness
62 E1 = 1-exp(N1^(0.22)*(exp(-N1^(0.78))-1));
63 // the corresponding temperature difference in each
    fluid is
64 dT = E1*(Ta1-Ta2); // [degree celsius]
65 // the heat transfer is calculated as
66 q1 = Cmin*dT; // [W]
67 printf("\n\n the percent reduction in heat transfer
    is %f ",(q-q1)*100/q);

```

---

**Scilab code Exa 10.16** heat transfer coefficient in compact exchanger

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 10.16\n\n\n");
4 // heat-transfer coefficient in compact exchanger
5 // Example 10.16 (page no.-556-557)
6 // solution
7
8 p = 101325; // [Pa] pressure of air
9 T = 300; // [K] temperature of entering air
10 u = 15; // [m/s] velocity of air
11 // we obtain the air properties from table A-5(page
    no.-607)
12 rho = 1.1774; // [kg/cubic meter] density of air

```

```

13 Cp = 1005.7; // [J/kg degree celsius] specific heat
    of air
14 mu = 1.983*10^(-5); // [kg/m s] viscosity of air
15 Pr = 0.708; // prandtl number
16 // from figure 10-19(page no.-557) we have
17 Ac_by_A = 0.697;
18 sigma = Ac_by_A;
19 Dh = 3.597*10^(-3); // [m]
20 // the mass velocity is thus
21 G = rho*u/sigma; // [kg/square meter s]
22 // and the reynolds number is
23 Re = Dh*G/mu;
24 // from figure 10-19(page no.-557) we can read
25 St_into_Pr_exp_2_by_3 = 0.0036;
26 // and the heat transfer coefficient is
27 h = St_into_Pr_exp_2_by_3*G*Cp*(Pr)^(-2/3); // [W/
    square meter degree celsius]
28 printf("heat-transfer coefficient is %f W/square
    meter degree celsius",h);

```

---

**Scilab code Exa 10.17** transient response of thermal energy storage system

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 10.17\n\n\n");
4 // transient response of thermal-energy storage
    system
5 // Example 10.17 (page no.-559-562)
6 // solution
7
8 Rinf = 0.176; // [degree celsius square meter/W]
    overall R value of material
9 A = 2.25; // [square meter] inlet flow area
10 l = 3; // [m] rock bed length

```

```

11 // properties of the rock are:
12 rho_r = 1281.4; // [kg/cubic meter]
13 Cr = 0.880; // [kJ/kg degree celsius]
14 kr = 0.87; // [W/m degree celsius]
15 Ti = 5; // [degree celsius] initial temperature of
    rock bed
16 Ta = 40; // [degree celsius] air temperature
17 Tinf = Ta; // [degree celsius]
18 p = 101.325; // [kPa] pressure of air
19 Ts = 5; // [degree celsius] surrounding temperature
20 v1 = 0.3; // [m/s] inlet velocity 1
21 v2 = 0.9; // [m/s] inlet velocity 2
22 Cpa = 1.004; // [kJ/kg degree celsius]
23 R = 0.287; // [kJ/kg K] universal gas constant
24 // it can be seen that the axial energy conduction
    is small compared to the mass energy transport.
25 // for a 35 degree celsius temperature difference
    over a 0.6 length
26 dx = 1/5; // [m]
27 q_cond = kr*A*(Ta-Ti)/dx; // [W]
    )
    (a)
28 // the density of air at 40 degree celsius
29 rho_a = p/(R*(Ta+273)); // [kg/cubic meter]
    (b)
30 // and the mass flow rate at 0.3 m/s is
31 mdot_a = rho_a*A*v1; // [kg/s]
    (c)
32 // the corresponding energy transport for a
    temperature difference of 35 degree celsius is
33 q = mdot_a*Cpa*(Ta-Ti); // [kW]
    (d)
34 // and this is much larger than the value in
    equation (a).
35 // we now write an energy balance for one of the
    axial nodes as

```



```

36 // energy transported in - energy transported out -
    // energy lost to surroundings = rate of energy
    // accumulation of node
37 // or  $\dot{m} \cdot c_p (T_{m,o}(t) - T_m(t)) - (T_m(t) - T_{inf}) \cdot P$ 
    //  $\cdot dx / R_{inf} = \rho_r \cdot C_r \cdot dV_r \cdot (T_m(t+1) - T_m(t)) / dt$ 
    // (e)
38 // where the exit temperature from node m is assumed
    // to be the rock temperature of that node ( $T_m(t)$ ).
    // equation (e) may be solved to give
39 //  $T_m(t+1) = F \cdot \dot{m} \cdot c_p \cdot T_{m,o}(t) + [1 - F \cdot (\dot{m} \cdot c_p$ 
    //  $\cdot P \cdot dx / R_{inf})] \cdot T_m(t) + F \cdot P \cdot dx \cdot T_{inf} / R_{inf}$ 
    // (f)
40 // where
41 //  $F = dt / (\rho_r \cdot C_r \cdot dV_r)$ 
42 // here P is perimeter and dx is the increment.
43 P = 4*1.5; // [m]
44 // the stability requirement is such that the
    // coefficient on the  $T_m(t)$  terms cannot be
    // negative. using dx = 0.6m, we find that the
    // maximum value of
45 dx = 0.6; // [m]
46 Fmax = 6.4495*10(-4);
47 // which yields a maximum time increment of
48 tmax = 0.54176; // [h]
49 // with a velocity of 0.9 m/s the maximum time
    // increment for stability is
50 tmax_v2 = 0.1922; // [h]
51 // for the calculations we select the following
    // values of dt with the resultant values of F:
52
53 // for v1
54 dt1 = 0.2; // [h]
55 F1 = 2.38095*10(-4);
56 // for v2
57 dt2 = 0.1; // [h]
58 F2 = 1.190476*10(-4);
59
60 // with the appropriate properties and these values

```

```

        inserted into equation(f) there results
61 // for v1
62 //  $Tm^{(t+1)} = F1*mdot\_a*Cpa*Tm\_o^{(t)} + [1-F1*(mdot\_a$ 
         $*Cpa+P*dx/Rinf)]*Tm^{(t)} + F1*P*dx*Tinf/Rinf$ 
        (g)
63 // for v2
64 //  $Tm^{(t+1)} = F2*mdot\_a*Cpa*Tm\_o^{(t)} + [1-F2*(mdot\_a$ 
         $*Cpa+P*dx/Rinf)]*Tm^{(t)} + F2*P*dx*Tinf/Rinf$ 
        (h)
65
66 // the energy storage relative to 5 degree celsius
        can then be calculated from
67 E_t = 0;
68 i = 1;
69 T1 = 40;
70 T2 = 5;
71 T3 = 5;
72 T4 = 5;
73 T5 = 5;
74     for i = 1:100
75         T2 = (F2*mdot_a*Cpa*1000*T1 + [1-F2*(mdot_a*Cpa
            *1000-P*dx/Rinf)]*T2 + F2*P*dx*Tinf/Rinf);
76         T3 = (F2*mdot_a*Cpa*1000*T2 + [1-F2*(mdot_a*Cpa
            *1000-P*dx/Rinf)]*T3 + F2*P*dx*Tinf/Rinf);
77         T4 = (F2*mdot_a*Cpa*1000*T3 + [1-F2*(mdot_a*Cpa
            *1000-P*dx/Rinf)]*T4 + F2*P*dx*Tinf/Rinf);
78         T5 = (F2*mdot_a*Cpa*1000*T4 + [1-F2*(mdot_a*Cpa
            *1000-P*dx/Rinf)]*T5 + F2*P*dx*Tinf/Rinf);
79         Temp(i,:) = [T1 T2 T3 T4 T5];
80         E_t = (dt1/F1)*[(T1-5)+(T2-5)+(T3-5)+(T4-5)+(T5
            -5)];
81         val(i) = i;
82         val1(i) = E_t;
83     end
84
85 E_t = 0;
86 i = 1;
87 T1 = 40;

```

```

88 T2 = 5;
89 T3 = 5;
90 T4 = 5;
91 T5 = 5;
92     for i = 1:100
93         T2 = (F1*mdot_a*Cpa*1000*T1 + [1-F1*(mdot_a*Cpa
          *1000-P*dx/Rinf)]*T2 + F1*P*dx*Tinf/Rinf);
94         T3 = (F1*mdot_a*Cpa*1000*T2 + [1-F1*(mdot_a*Cpa
          *1000-P*dx/Rinf)]*T3 + F1*P*dx*Tinf/Rinf);
95         T4 = (F1*mdot_a*Cpa*1000*T3 + [1-F1*(mdot_a*Cpa
          *1000-P*dx/Rinf)]*T4 + F1*P*dx*Tinf/Rinf);
96         T5 = (F1*mdot_a*Cpa*1000*T4 + [1-F1*(mdot_a*Cpa
          *1000-P*dx/Rinf)]*T5 + F1*P*dx*Tinf/Rinf);
97         Temp(i,:) = [T1 T2 T3 T4 T5];
98         E_t = (dt1/F1)*[(T1-5)+(T2-5)+(T3-5)+(T4-5)+(T5
          -5)];
99         val2(i) = i;
100        val3(i) = E_t;
101    end
102    plot(val, val1, val2, val3);
103    legend("v = 0.3m/s", "v = 0.9m/s");
104    xlabel("time(h)");
105    ylabel("E(t) kJ ");
106    printf("the result of the calculations are shown in
          the accompanying figure");

```

---

**Scilab code Exa 10.18** variable properties analysis of a duct heater

```

1 clear;
2 clc;
3 printf("\t\t\tExample Number 10.18\n\n");
4 // variable-properties analysis of a duct heater
5 // Example 10.18 (page no.-562-564)
6 // solution
7

```

```

8 d = 0.3; // [m] diameter of duct
9 Tma = 700; // [K] temperature of hot air
10 E = 0.6; // emissivity of outside duct surface
11 Tinf = 20+273; // [K] room temperature
12 // air properties at 700 K
13 rho = 0.5030; // [kg/cubic meter] density of air
14 mu = 3.332*10^(-5); // [kg/m s] viscosity of air
15 k = 0.05230; // [W/m degree celsius] heat transfer
    coefficient
16 Pr = 0.684; // prandtl no. of air
17 A = %pi*d^(2)/4; // [square meter] area of duct
18 sigma = 5.669*10^(-8); // [W/square meter K^(4)]
19 P = %pi*d; // [m]
20 Cp = 1083.5; // [J/kg degree celsius]
21 // this is a problem where a numerical solution must
    be employed.
22 // we choose a typical section of the duct with
    length dx and perimeter P as shown inn figure
    example 10-18A(page no.-562) and make the energy
    balances.
23 // we assume that resistance of the duct wall is
    negligible.
24 // inside the duct the energy balance is
25 // mdot_a*Cp*Tma = hi*P*dx*(Tma-Tmw)+mdot_a*Cp*
    Tm-po-a
    (a)
26 // where hi is the convection heat transfer
    coefficient on the inside which may be calculated
    from(the flow is turbulent)
27 // Nu = hi*d/k = 0.023*Re_d^(0.8)*Pr^(0.3)
    (b)
28 // with the properties evaluated at the bulk
    temperature of air(Tma). the energy balance for
    the heat flow through the wall is
29 // qconv_i = qconv_o+grad_o
30 // or, by using convection coefficients and
    radiation terms per unit area,
31 // hi*(Tma-Tmw) = hc*(Tmw-Tinf)+sigma*E*(Tmw^(4)-
    Tinf^(4))
    (c)

```

```

32 // where the outside convection coefficient can be
    calculated from the free convection relation
33 //  $hc = 0.27 * ((Tmw - Tinf) / d)^{1/4}$ 
    (d)
34 // inserting this relation in equation (c) gives
35 //  $hi * (Tma - Tmw) = 0.27 * (Tmw - Tinf)^{5/4} / d^{1/4} +$ 
     $sigma * E * (Tmw^4 - Tinf^4)$ 
    (e)
36 // equation (a) may be solved for Tm_po_a to give
37 //  $Tm\_po\_a = (1 - hi * P * dx / (mdot\_a * Cp)) * Tma + (hi * P * dx / (mdot\_a * Cp)) * Tmw$ 
    (f)
38
39 // for
40 x=180;
41 mdot_a = [0.14 0.45 0.68]; // [kg/s]
42 for i = 1:3
43
44 v = mdot_a(i) / (A * rho); // [m/s]
45 Re_d = d * v * rho / mu;
46 hi = k * 0.023 * Re_d^(0.8) * Pr^(0.3) / d; // [W/square
    meter degree celsius]
47
48
49 for dx = 1:1:179
50     for Tmw = 295:1:715
51         Z = (hi/dx) * (Tma - Tmw) - 0.27 * (Tmw - Tinf)^(5/4) /
            d^(1/4) - sigma * E * (Tmw^4 - Tinf^4);
52         if (Z > 0 & Z < 40) then
53             Tmw_new = Tmw;
54         end
55     end
56     for Tm_po_a = 275:1:715
57         X = Tm_po_a - (1 - (hi/dx) * P * dx / (mdot_a(i) * Cp)) *
            Tmw_new + ((hi/dx) * P * dx / (mdot_a(i) * Cp)) *
            Tmw_new;
58         if (X > 0 & X < 5) then
59             Tm_po_a_new = Tm_po_a;
60         end
61     end

```

```

62     q_by_A = (hi/dx)*(Tma-Tmw_new); // [W/square
        meter]
63     val1(dx,i) = q_by_A;
64     val(dx) = dx;
65     val2(dx,i) = Tmw_new;
66     val3(dx,i) = Tm_po_a_new;
67 end
68 end
69 scf(1);
70 plot(val, val1(:,1), val, val1(:,2), val, val1(:,3));
71 legend("mdot_a=0.14", "mdot_a=0.45", "mdot_a=0.68");
72 xlabel("Duct Length x,m");
73 ylabel("Local Heat Flux q / A,W / m^2");
74 xgrid();
75 title("Heat Flux");
76
77 scf(2);
78 plot(val, val2(:,1), val, val2(:,2), val, val2(:,3));
79 legend("Tw=0.14", "Tw=0.45", "Tw=0.68");
80 xlabel("Duct Length x,m");
81 ylabel("Local Wall Temperature Tw K");
82 xgrid();
83 title("Temperature Profile");
84
85 scf(3);
86 plot(val, val3(:,1), val, val3(:,2), val, val3(:,3));
87 legend("Ta=0.14", "Ta=0.45", "Ta=0.68");
88 xlabel("Duct Length x,m");
89 ylabel("Local Air Temperature Ta K");
90 xgrid();
91 title("Temperature Profile");
92 printf("plots are shown as :");

```

---

# Chapter 11

## Mass Transfer

Scilab code Exa 11.1 diffusion coefficient for co2

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 11.1\n\n\n");
4 // diffusion coefficient for co2
5 // Example 11.1(page no.-583)
6 // solution
7
8 T = 25+273.15; // [K] temperature of air
9 Vco2 = 34.0; // molecular volume of co2
10 Vair = 29.0; // molecular volume of air
11 Mco2 = 44; // molecular weight of co2
12 Mair = 28.9; // molecular weight of air
13 P = 1.01325*10^(5); // [Pa] atmospheric pressure
14 // using equation (11-2)
15 D = 435.7*T^(1.5)*(((1/Mco2)+(1/Mair))^(1/2))/(P*(
    Vco2^(1/3)+Vair^(1/3))^(2));
16 printf("value of diffusion coefficient for co2 in
    air is %f square centimeter/s",D);
```

---

### Scilab code Exa 11.2 diffusion coefficient for co2

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 11.2\n\n\n");
4 // diffusion coefficient for co2
5 // Example 11.2(page no.-586-587)
6 // solution
7
8 T = 25+273.15; // [K] temperature of air
9 p = 1.01325*10^(5); // [Pa] atmospheric pressure
10 pw1 = 3166.7618901; // [Pa] partial pressure at the
    bottom of test tube
11 pw2 = 0; // [Pa] partial pressure at the top of test
    tube
12 pa1 = p-pw1; // [Pa]
13 pa2 = p-pw2; // [Pa]
14 D = .256*10^(-4); // [square meter/s] diffusion
    coefficient
15 Mw = 18; // [g] molecular weight of water
16 A = 22*((5*10^(-3))^2)/7; // [square meter] area of
    test tube
17 R = 8314; // [J/mol K] gas constant
18 // using equation(11-16)
19 mw = (D*p*Mw*A/(R*T*0.15))*log(pa2/pa1); // mass flow
    rate
20 printf(" mass flow rate is %e kg/s",mw);
```

---

### Scilab code Exa 11.3 Wet bulb temperature

```
1 clear;
2 clc;
3 printf("\t\t\tExample Number 11.3\n\n\n");
4 // Wet-bulb temperature
5 // Example 11.3(page no.-590-591)
```



```

6 // solution
7
8 Pg = 2107; // [Pa] from steam table at 18.3 degree
   celcius
9 Pw = Pg*18; // [Pa]
10 Rw = 8315; // [J/mol K] gas constant
11 Tw = 273.15+18.3; // [K]
12 RHOw = Pw/(Rw*Tw); // [kg/cubic meter]
13 Cw = RHOw; // [kg/cubic meter]
14 RHOinf = 0; // since the free stream is dry air
15 Cinf = 0;
16 P = 1.01325*10^(5); // [Pa]
17 R = 287; // [ J /kg K ]
18 T = Tw; // [K]
19 RHO = P/(R*T); // [kg/cubic meter]
20 Cp = 1004; // [J/kg degree celsius]
21 Le = 0.845;
22 Hfg = 2.456*10^(6); // [J/kg]
23 // now using equation(11-31)
24 Tinf = (((Cw-Cinf)*Hfg)/(RHO*Cp*(Le^(2/3))))+Tw; // [
   K]
25 Tin = Tinf-273.15; // [degree celsius]
26 printf("temperature of dry air is %f degree celsius"
   ,Tin);
27 printf("\n\n these calculations are now recalculated
   the density at the arithmetic-average
   temperature between wall and free-stream
   conditions");
28 printf("\n\n with this adjustments these results are
   RHO = 1.143 kg/m^(3) and Tinf = 55.8 degree
   celcius");

```

---

Scilab code Exa 11.4 relative humidity of air stream

```
1 clear;
```

```

2  clc;
3  printf("\t\t\tExample Number 11.4\n\n\n");
4  // relative humidity of air stream
5  // Example 11.4(page no.-591)
6  // solution
7
8  // these data were taken from previous example
9  Rho = 1.212; // [kg/cubic meter]
10 Cp = 1004; // [J/kg]
11 Le = 0.845;
12 Tw = 18.3; // [degree celsius]
13 Tinf = 32.2; // [degree celsius]
14 Rhow = 0.015666; // [kg/cubic meter]
15 Cw = Rhow; // [kg/cubic meter]
16 Hfg = 2.456*10^(6); // [J/kg]
17 // we use eqn 11-31
18 Cinf = Cw-(Rho*Cp*Le^(2/3)*(Tinf-Tw)/Hfg); // [kg/
    cubic meter]
19 Rhoinf = Cinf; // [kg/cubic meter]
20 Rhog = 0.0342; // [kg/cubic meter]
21 RH = (Rhoinf/Rhog)*100;
22 printf(" relative humidity is therefore %f
    percentage",RH);

```

---

### Scilab code Exa 11.5 water evaporation rate

```

1  clear;
2  clc;
3  printf("\t\t\tExample Number 11.5\n\n\n");
4  // water evaporation rate
5  // Example 11.5(page no.-593-594)
6  // solution
7
8  Ta = 38+273; // [K] temperature of atmospheric air
9  RH = 0.30; // relative humidity

```

```

10 u = 10; // [mi/h] mean wind speed
11 R = 0.287; // universal gas constant
12 Dw = 0.256*10^(-4); // [square meter/s] from table A
    -8(page no.-610)
13 rho_w = 1000; // [kg/cubic meter]
14 // for this calculation we can make use of equation
    (11-36). from thermodynamic steam tables
15 p_g = 6.545; // [kPa] at 38 degree celsius
16 p_s = p_g; // [kPa]
17 p_w = RH*p_s; // [kPa]
18 p_s = 1.933; // [in Hg]
19 p_w = 0.580; // [in Hg]
20 // also
21 u_bar = u*24; // [mi/day]
22 // equation(11-36) yields , with the application of
    the 0.7 factor
23 E_lp = 0.7*(0.37+0.0041*u_bar)*(p_s-p_w)^(0.88); // [
    in/day]
24 E_lp = E_lp*2.54/100; // [m/day]
25 // noting that standard pan has the diameter of 1.2m
    , we can use the figure to calculate the mass
    evaporation rate per unit area as
26 m_dot_w_by_A = E_lp*rho_w/24; // [kg/h square meter]
27 // as a matter of interest , we might calculate the
    molecular-diffusion rate of water vapour from
    equation(11-35), taking z1 as the 1.5m dimension
    above the standard pan.
28 z1 = 1.5; // [m]
29 // since rho = p/(R*T)
30 // equation(11-35) can be written as
31 m_dot_w_by_A1 = 0.622*Dw*p_g*3600/(R*Ta*z1); // [kg/h
    square meter]
32 printf("evaporation rate on the land under these
    conditions is %e kg/h square meter",m_dot_w_by_A1
    );

```

---