

Scilab Textbook Companion for
Engineering Heat Transfer
by W. S. Janna¹

Created by
Sintu Rongpipi
Bachelor of Technology
Chemical Engineering
Indian Institute of Technology Guwahati
College Teacher
Dr. Prakash Kotecha
Cross-Checked by
Mukul R. Kulkarni

May 24, 2016

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

Book Description

Title: Engineering Heat Transfer

Author: W. S. Janna

Publisher: CRC Press

Edition: 3

Year: 2009

ISBN: 978-1420072020

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 Fundamental Concepts	6
2 Steady State Conduction in One Dimension	13
3 Steady State Conduction in Multiple Dimensions	29
4 Unsteady State Heat Conduction	36
5 Introduction to Convection	56
6 Convection Heat Transfer in a Closed Conduit	58
7 Convection Heat Transfer in Flows Past Immersed Bodies	73
8 Natural Convection Systems	91
9 Heat Exchangers	103
10 Condensation and Vaporization Heat Transfer	120
11 Introduction to Radiation Heat Transfer	127
12 Radiation Heat Transfer between Surfaces	134

List of Scilab Codes

Exa 1.1	The Surface Temperature of firewall	6
Exa 1.2	Thermal Conductivity of Aluminium	7
Exa 1.3	Convective Heat Transfer	7
Exa 1.4	Average Film Conductance	8
Exa 1.5	Instantaneous Heat loss by radiation	9
Exa 1.6	Radiation thermal conductance	9
Exa 1.7	Thermal Circuit	10
Exa 1.8	Combined Heat Transfer Mechanisms	11
Exa 2.1	Materials in Series	13
Exa 2.2	Materials in Parallel	14
Exa 2.3	Overall Heat Transfer Coefficient	15
Exa 2.4	Pipe and Tube Specifications	17
Exa 2.5	Materials in Series in tubular arrangement	18
Exa 2.6	Overall Heat Transfer Coefficient in pipe	19
Exa 2.7	Thermal Contact Resistance	20
Exa 2.8	Analysis of a Pin Fin	21
Exa 2.9	Corrected Length Solution	24
Exa 2.10	Straight fins of triangular profile	25
Exa 2.11	Circular fin of rectangular profile	26
Exa 3.1	Flow Net method of solution	29
Exa 3.2	Conduction Shape Factor	30
Exa 3.3	Heat lost through shape factor method	30
Exa 3.4	Shape factor for given pipe	32
Exa 3.5	Numerical Method for temperature distribution	33
Exa 4.1	Response Time of thermocouple junction	36
Exa 4.2	Lumped Capacitance Approach	37
Exa 4.3	Temperature profile through Infinite plate chart	38
Exa 4.4	Dimensionless temperature history of a sphere	42

Exa 4.5	Estimation of the depth of Freeze line	44
Exa 4.6	Determination of temperature using kpc product	45
Exa 4.7	Combination of 1D Transient Systems for beer can . . .	46
Exa 4.8	Combination of 1D Transient Systems for rectangular bar	48
Exa 4.9	Numerical Method for transient conduction system . . .	50
Exa 4.10	Graphical Method for transient conduction system . . .	54
Exa 5.1	Heat transferred using specific heat	56
Exa 5.2	Volumetric thermal expansion coefficient	57
Exa 6.1	Constant heat flux at the wall	58
Exa 6.2	Constant wall temperature	60
Exa 6.3	Hydrodynamic and Thermal Entry Length	61
Exa 6.4	Variation of temperature with length of tube	64
Exa 6.5	Combined Entry Length Problem	69
Exa 6.6	Heat transfer to and from turbulent flow	71
Exa 7.1	Laminar Boundary Layer flow over flat plate	73
Exa 7.2	Flow over plate at constant temperature	76
Exa 7.3	Flow over Isothermal flat plate	79
Exa 7.4	Maximum heater surface temperature	81
Exa 7.5	Reynolds Colburn Analogy	81
Exa 7.6	Drag due to skin friction	83
Exa 7.7	Wattage requirement of heater	83
Exa 7.8	Flow past a telephone pole	85
Exa 7.9	Current through Hot Wire Anemometer	86
Exa 7.10	Pressure Drop for flow over tubes	88
Exa 8.1	Natural Convection on a vertical surface	91
Exa 8.2	1D heat flow through a window glass	92
Exa 8.3	The cooling unit of a refrigerator	94
Exa 8.4	Natural convection on an inclined flat plate	95
Exa 8.5	Natural convection on a horizontal flat surface	97
Exa 8.6	Natural convection on cylinders	98
Exa 8.7	Natural convection around blocks	100
Exa 8.8	Natural convection about an array of fins	101
Exa 9.1	Log Mean Temperature Difference	103
Exa 9.2	LMTD for equal outlet temperatures	104
Exa 9.3	Double Pipe Heat Exchanger	104
Exa 9.4	Fouling Factors in Double Pipe Heat Exchangers	107
Exa 9.5	Shell and Tube heat exchangers	111

Exa 9.6	Effectiveness NTU method of analysis	114
Exa 9.7	Crossflow heat exchangers	116
Exa 10.1	Laminar film condensation on a vertical flat surface . .	120
Exa 10.2	Film condensation on a horizontal tube	123
Exa 10.3	Nucleate pool boiling critical heat flux	124
Exa 11.1	Radiation Intensity	127
Exa 11.2	Irradiation and Radiosity	128
Exa 11.3	Emissivity and Rate of radiant emission	130
Exa 11.4	Plancks distribution law	131
Exa 11.5	Wiens displacement law	132
Exa 11.6	Transmission through glass windshield	132
Exa 12.3	View Factor algebra for pairs of surfaces	134
Exa 12.4	View Factor algebra for enclosures	135
Exa 12.5	Crossed String method	136
Exa 12.6	Radiation heat transfer within a broiler	137
Exa 12.7	Radiation heat transfer within an enclosure of diffuse gray surfaces	138
Exa 12.8	Radiation heat transfer within an enclosure of black sur- faces and diffuse gray surfaces	139

List of Figures

2.1	Analysis of a Pin Fin	22
3.1	Numerical Method for temperature distribution	34
4.1	Temperature profile through Infinite plate chart	39
4.2	Numerical Method for transient conduction system	51
6.1	Hydrodynamic and Thermal Entry Length	61
6.2	Variation of temperature with length of tube	65
7.1	Laminar Boundary Layer flow over flat plate	74
7.2	Flow over plate at constant temperature	77
10.1	Laminar film condensation on a vertical flat surface	121

Chapter 1

Fundamental Concepts

Scilab code Exa 1.1 The Surface Temperature of firewall

```
1 clear;
2 clc;
3 printf("\t\t\tChapter1_Example1\n\n\n");
4 // determination of surface temperature on one side
  of a firewall
5 k=9.4; // thermal conductivity in [BTU/hr.ft.
  Rankine]
6 q=6.3; // heat flux in [BTU/s. sq.ft]
7 T1=350; // the outside surface temperature of one
  aide of the wall [ F ]
8 // converting heat flux into BTU/hr sq.ft
9 Q=6.3*3600 // [BTU/hr.sq.ft]
10 printf("\nThe heat flux is %.2f BTU/hr. sq.ft",Q);
11 dx=0.5; // thickness in [inch]
12 //converting distance into ft
13 Dx=0.5/12 // thickness in [ft]
14 printf("\nThe thickness of firewall is %.2f ft",Dx);
15 // solving for temeprature T2
16 T2=T1-(Q*Dx/k); // [ F ]
```

```
17 printf("\nThe required temperature on the other side
    of the firewall is %.1f degree Fahrenheit",T2);
```

Scilab code Exa 1.2 Thermal Conductivity of Aluminium

```
1 clear;
2 clc;
3 printf("\t\t\tchapter1_example2\n\n\n");
4 // determination of thermal conductivity of
  aluminium
5 k_ss=14.4; // thermal conductivity of stainless
  steel in [W/m.K]
6 printf("\nThe thermal conductivity of stainless
  steel is %.1f W/m.K",k_ss);
7 dt_ss=40; // [K]
8 dt_al=8.65; // [K]
9 dz_ss=1; // [cm]
10 dz_al=3; // [cm]
11 k_al=k_ss*dt_ss*dz_al/(dt_al*dz_ss); // thermal
  conductivity of Al in [W/m.K]
12 printf("\nThe thermal conductivity of aluminium is
  %d W/m.K",k_al);
```

Scilab code Exa 1.3 Convective Heat Transfer

```
1 clear;
2 clc;
3 printf("\t\t\tchapter1_example3\n\n\n");
4 // determination of heat transferred by convection
```

```

5 h_c=3; // convective coefficient in [BTU/hr.ft^2
6 A=30*18; // Cross sectional area in ft^2
7 T_w=140; // Roof surface temperature in degree
  Fahrenheit
8 T_inf=85; // Ambient temperature in degree
  Fahrenheit
9 dT= (T_w-T_inf);
10 Q_c=h_c*A*dT; // Convective heat transfer in BTU/hr
11 printf("\nThe heat transferred by convection is %d
  BTU/hr",Q_c);

```

Scilab code Exa 1.4 Average Film Conductance

```

1 clear;
2 clc;
3 printf("\t\t\tchapter1_example4\n\n\n");
4 // determining average film conductance
5 D=2.43/100; // diameter in meter
6 L=20/100; // length in meter
7 A=3.14*D*L; // cross-sectional area in sq.m
8 cp=4200; // specific heat of water in J/kg.K
9 T_b2=21.4; // temperature of bulk fluid in degree
  celsius
10 T_in=20; // temperature of inlet water in degree
  celsius
11 T_w=75; // temperature of wall in degree celsius
12 Q=500; // volumetric flow rate in cc/s
13 density=1000; // density of water in kg/cu.m
14 m=Q*density/10^6; // mass flow rate in kg/s
15 printf("\nThe mass flow rate is %.1f kg/s",m);
16 // using definition of specific heat and Newton's
  law of cooling
17 hc=m*cp*(T_b2-T_in)/(A*(T_w-T_in));

```

```
18 printf("\n\nThe average film conductance is %d W/sq.m.  
    K",hc);
```

Scilab code Exa 1.5 Instantaneous Heat loss by radiation

```
1 clear;  
2 clc;  
3 printf("\t\t\tchapter1_example5\n\n\n");  
4 // determination of heat loss rate by radiation  
5 W=14; // width in ft  
6 L=30; // length in ft  
7 A=W*L; // area in ft^2  
8 F_12=1; // view factor assumed to be 1  
9 T1=120+460; // driveway surface temperature in  
    degree Rankine  
10 printf("\n\nThe driveway surface temperature is %d  
    degree Rankine",T1);  
11 T2=0; // space temperature assumed to be 0 degree  
    Rankine  
12 sigma=0.1714*10^(-8); // value of Stefan-Boltzmann's  
    constant in BTU/(hr.ft^2.(degree Rankine)^4)  
13 e=0.9; // surface emissivity  
14 q=sigma*A*e*F_12*((T1)^4-(T2)^4);  
15 printf("\n\nThe heat loss rate by radiation is %d BTU/  
    hr",q);
```

Scilab code Exa 1.6 Radiation thermal conductance

```
1 clear;
```

```

2  clc;
3  printf("\t\t\tchapter1_example6\n\n\n");
4  // determination of radiation thermal conductance
5  A=14*30; // area in sq.ft
6  T1=120+460; // driveway surface temperature in
   degree Rankine
7  T2=0; // surface temperature assumed to be 0 degree
   Rankine
8  Qr=73320; // heat loss rate in BTU/hr
9  hr=Qr/(A*(T1-T2)); // radiation thermal conductance
   in BTU/(hr.ft^2.(degree Rankine)
10 printf("\nthe radiation thermal conductance is %.2f
   BTU/(hr. sq.ft.(degree Rankine))",hr);

```

Scilab code Exa 1.7 Thermal Circuit

```

1  clear;
2  clc;
3  printf("\t\t\tchapter1_example7\n\n\n");
4  // Identification of all resistances and their
   values
5  // Estimation of heat transfer per unit area
6  // Determination of the inside and outside wall
   temperatures
7  printf("\n\t\t\tSolution to part (b)\n");
8  A=1; // assuming A=1 m^2 for convenience
9  hc1_avg=(5+25)/2; // taking average of extreme
   values for hc [W/m^2.K]
10 Rc1=1/(hc1_avg*A); // resistance on left side of
   wall [K/W]
11 printf("\nThe resistance on left side of wall is %.3
   f K/W",Rc1);
12 k=(0.38+0.52)/2; // thermal conductivity of common

```

```

    brick in W/M.k
13 L=0.1; //10 cm converted into m
14 Rk=(L/(k*A)); // resistance of construction material,
    assume common brick
15 printf("\nThe resistance of construction material of
    wall is %.3f K/W",Rk);
16 Rc2=Rc1;
17 printf("\nThe resistance on right side of wall is %
    .3f K/W",Rc2);
18 printf("\n\n\t\t\tSolution to part (c)\n");
19 T_inf1=1000; // temperature of exhaust gases in K
20 T_inf2=283; // temperature of ambient air in K
21 q=(T_inf1-T_inf2)/(Rc1+Rk+Rc2); // heat transferred
    per unit area
22 printf("\nThe Heat transferred per unit area is %d W
    = %.3f kW",q,q/1000);
23 printf("\n\n\t\t\tSolution to part (d)\n");
24 T_in=T_inf1-Rc1*q; //
25 T_out=T_inf2+Rc2*q;
26 printf("\nThe inside wall temperature is %d K",T_in)
    ;
27 printf("\nThe outside wall temperature is %d K",
    T_out);

```

Scilab code Exa 1.8 Combined Heat Transfer Mechanisms

```

1 clear;
2 clc;
3 printf("\t\t\tchapter1_example8\n\n\n");
4 // determination of surface temperature
5 k=0.604; // [BTU/(hr.ft.degree Rankine)]
6 hc=3; // average value for natural convection in BTU
    /(hr.ft^2.degree Rankine)

```

```

7 ew=0.93;
8 f_wr=1; // shape factor
9 sigma= 0.1714*10^(-8) // BTU/(hr.ft^2.degree Rankine
    ).
10 L=4/12; // length in ft
11 T1=80+460; // temperature of side-walk in degree
    Rankine
12 T_inf=20+460; // temperature of ambient air in
    degree Rankine
13 T_r=0; // assuming space temperature to be 0 degree
    Rankine
14 // LHS of the form a*Tw+b*Tw^4=c
15 a=((k/L)+hc);
16 b=(sigma*ew*f_wr);
17 c=(k*T1/L)+(hc*T_inf)+(sigma*f_wr*ew*T_r^4);
18 printf("\nRHS=%d",c);
19     Tw=[470;480;490;485;484.5];
20 for i=1:5
21     LHS(i)=a*Tw(i)+b*Tw(i)^4;
22 end
23 printf("\nSolving by trial and error yields the
    following, where LHS is the left-hand side of the
    equation");
24 printf("\n\tTw\tLHS");
25 for i=1:5
26     printf("\n\t%.1f\t%d",Tw(i),LHS(i));
27 end
28 printf("\nThe Surface temperature is %.1f degree R =
    %.1f degree F",Tw(5),Tw(i)-460);

```

Chapter 2

Steady State Conduction in One Dimension

Scilab code Exa 2.1 Materials in Series

```
1  clc;
2  clear;
3  printf("\t\t\tChapter2_example1\n\n\n");
4  //  determination of the heat flow through a
   composite wall
5  T3=-10; // temperature of inside wall in degree
   Fahrenheit
6  T0=70; // temperature of outside wall in degree
   Fahrenheit
7  dT=T0-T3; // overall temperature difference
8  // values of thermal conductivity in BTU/(hr.ft.
   degree Rankine) from appendix table B3
9  k1=0.38; // brick masonry
10 k2=0.02; // glass fibre
11 k3=0.063; // plywood
12 dx1=4/12; // thickness of brick layer in ft
13 dx2=3.5/12; // thickness of glass fibre layer in ft
```



```

14 dx3=0.5/12; // thickness of plywood layer in ft
15 A=1; // cross sectional area taken as 1 ft^2
16 R1=dx1/(k1*A); // resistance of brick layer in (hr.
    degree Rankine)/BTU
17 R2=dx2/(k2*A); // resistance of glass fibre layer in
    (hr.degree Rankine)/BTU
18 R3=dx3/(k3*A); // resistance of plywood layer in (hr
    .degree Rankine)/BTU
19 printf("\nResistance of brick layer is %.3f (hr.
    degree Rankine)/BTU",R1);
20 printf("\nResistance of glass fibre layer is %.1f (
    hr.degree Rankine)/BTU",R2);
21 printf("\nResistance of plywood layer is %.3f (hr.
    degree Rankine)/BTU",R3);
22 qx=(T0-T3)/(R1+R2+R3);
23 printf("\nHeat transfer through the composite wall
    is %.2f BTU/hr",qx);

```

Scilab code Exa 2.2 Materials in Parallel

```

1 clc;
2 clear;
3 printf("\t\t\tChapter2_example2\n\n\n");
4 // determination of heat transfer through composite
    wall for materials in parallel
5 // values of thermal conductivities in W/(m.K) from
    appendix table B3
6 k1=0.45; // thermal conductivity of brick
7 k2a=0.15; // thermal conductivity of pine
8 k3=0.814; // thermal conductivity of plaster board
9 k2b=0.025; // thermal conductivity of air from
    appendix table D1
10 // Areas needed for evaluating heat transfer in sq.

```

```

    m
11 A1=0.41*3; // cross sectional area of brick layer
12 A2a=0.038*3; // cross sectional area of wall stud
13 A2b=(41-3.8)*0.01*3; // cross sectional area of air
    layer
14 A3=0.41*3; // cross sectional area of plastic layer
15 dx1=0.1; // thickness of brick layer in m
16 dx2=0.089; // thickness of wall stud and air layer
    in m
17 dx3=0.013; // thickness of plastic layer in m
18 R1=dx1/(k1*A1); // Resistance of brick layer in K/W
19 R2=dx2/(k2a*A2a+k2b*A2b); // Resistance of wall stud
    and air layer in K/W
20 R3=dx3/(k3*A3); // Resistance of plastic layer in K/
    W
21 printf("\nResistance of brick layer is %.3f K/W",R1)
    ;
22 printf("\nResistance of wall stud and air layer is %
    .2f K/W",R2);
23 printf("\nResistance of plastic layer is %.3f K/W",
    R3);
24 T1=25; // temperature of inside wall in degree
    celsius
25 T0=0; // temperature of outside wall in degree
    celsius
26 qx=(T1-T0)/(R1+R2+R3); // heat transfer through the
    composite wall in W
27 printf("\nHeat transfer through the composite wall
    is %.1f W",qx);

```

Scilab code Exa 2.3 Overall Heat Transfer Coefficient

```
1 clc;
```

```

2 clear;
3 printf("\t\t\tChapter2_example3\n\n");
4 // determination of heat transfer rate and overall
  heat transfer coefficient
5 k1=24.8; // thermal conductivity of 1C steel in BTU
  /(hr.ft.degree Rankine)from appendix table B2
6 k2=0.02; // thermal conductivity of styrofoam steel
  in BTU/(hr.ft.degree Rankine)
7 k3=0.09; // thermal conductivity of fibreglass in
  BTU/(hr.ft.degree Rankine)
8 hc1=0.79; // convection coefficient between the air
  and the vertical steel wall in BTU/(hr.ft^2.
  degree Rankine)
9 hc2=150; // the convection coefficient between the
  ice water and the fiberglass
10 A=1; // calculation based on per square foot
11 dx1=0.04/12; // thickness of steel in ft
12 dx2=0.75/12; // thickness of styrofoam in ft
13 dx3=0.25/12; // thickness of fiberglass in ft
14 // Resistances in (degree Fahrenheit.hr)/BTU
15 disp('Resistances in (degree Fahrenheit.hr)/BTU:');
16 Rc1=1/(hc1*A); // Resistance from air to sheet metal
17 printf("\nResistance from air to sheet metal: %.3f
  degree F.hr/BTU",Rc1);
18 Rk1=dx1/(k1*A); // Resistance of steel layer
19 printf("\nResistance of steel layer: %.4f degree F.
  hr/BTU",Rk1);
20 Rk2=dx2/(k2*A); // Resistance of styrofoam layer
21 printf("\nResistance of styrofoam layer: %.3f degree
  F.hr/BTU",Rk2);
22 Rk3=dx3/(k3*A); // Resistance of fiberglass layer
23 printf("\nResistance of fiberglass layer: %.3f
  degree F.hr/BTU",Rk3);
24 Rc2=1/(hc2*A); // Resistance from ice water to
  fiberglass
25 printf("\nResistance from ice water to fiberglass: %
  .4f degree F.hr/BTU",Rc2);
26 U=1/(Rc1+Rk1+Rk2+Rk3+Rc2); // overall heat transfer

```

```

    coefficient in BTU/(hr.ft^2.degree Rankine)
27 printf("\nThe overall heat transfer coefficient is %
    .3f BTU/(hr. sq.ft.degree Rankine)",U);
28 T_inf1=90; // temperature of air in degree F
29 T_inf2=32; // temperature of mixture of ice and
    water in degree F
30 q=U*A*(T_inf1-T_inf2);
31 printf("\nThe heat transfer rate is %.1f BTU/hr",q);

```

Scilab code Exa 2.4 Pipe and Tube Specifications

```

1 clc;
2 clear;
3 printf("\t\t\tChapter2_example4\n\n\n");
4 // determination of the heat transfer through the
    pipe wall per unit length of pipe.
5 k=14.4; // thermal conductivity of 304 stainless
    steel in W/(m.K) from appendix table B2
6 // dimensions of steel pipes in cm from appendix
    table F1
7 D2=32.39;
8 D1=29.53;
9 T1=40;
10 T2=38;
11 Qr_per_length=(2*3.14*k)*(T1-T2)/log(D2/D1);
12 format(6);
13 printf("\nThe heat transfer through the pipe wall
    per unit length of pipe is %.1f W/m = %.2f kW/m",
    Qr_per_length,Qr_per_length/1000);

```

Scilab code Exa 2.5 Materials in Series in tubular arrangement

```
1  clc;
2  clear;
3  printf("\t\t\tChapter2_example5\n\n\n");
4  // determination of the heat gain per unit length
5  k1=231; // thermal conductivity of copper in BTU/(hr
   .ft.degree Rankine)from appendix table B1
6  k2=0.02; // thermal conductivity of insulation in BTU
   /(hr.ft.degree Rankine)
7  // Specifications of 1 standard type M copper tubing
   from appendix table F2 are as follows
8  D2=1.125/12; // outer diameter in ft
9  D1=0.08792; // inner diameter in ft
10 R2=D2/2; // outer radius
11 printf("\nOuter radius is %.4f ft",R2);
12 R1=D1/2; // inner radius
13 printf("\nOuter radius is %.3f ft",R1);
14 t=0.5/12; // wall thickness of insulation in ft
15 R3=R2+t;
16 printf("\nRadius including thickness is %.4f ft",R3)
   ;
17 LRk1=(log(R2/R1))/(2*3.14*k1); // product of length
   and copper layer resistance
18 printf("\nProduct of length and copper layer
   resistance is: %.1e",LRk1);
19 LRk2=(log(R3/R2))/(2*3.14*k2); // product of length
   and insulation layer resistance
20 printf("\nProduct of length and insulation layer
   resistance is: %.2f",LRk2);
21 T1=40; // temperature of inside wall of tubing in
   degree fahrenheit
```

```

22 T3=70; // temperature of surface temperature of
    insulation degree fahrenheit
23 q_per_L=(T1-T3)/(LRk1+LRk2); // heat transferred per
    unit length in BTU/(hr.ft)
24 printf("\nThe heat transferred per unit length is %
    .2f BTU/(hr.ft)",q_per_L);

```

Scilab code Exa 2.6 Overall Heat Transfer Coefficient in pipe

```

1  clc;
2  clear;
3  printf("\t\t\tChapter2_example6\n\n\n");
4  // Determination of the overall heat transfer
    coefficient
5  k12=24.8; // thermal conductivity of 1C steel in BTU
    /(hr.ft.degree Rankine)from appendix table B2
6  k23=.023; // // thermal conductivity of glass wool
    insulation in BTU/(hr.ft.degree Rankine)from
    appendix table B3
7  // Specifications of 6 nominal, schedule 40 pipe (no
    schedule was specified , so the standard is
    assumed) from appendix table F1 are as follows
8  D2=6.625/12; // outer diameter in ft
9  D1=0.5054; // inner diameter in ft
10 printf("\nOuter diameter is %.3f ft",D2);
11 printf("\nInner diameter is %.4f ft",D1);
12 t=2/12; // wall thickness of insulation in ft
13 D3=D2+t;
14 printf("\nDiameter including thickness is %.5f ft",
    D3);
15 hc1=12; // convection coefficient between the air
    and the pipe wall in BTU/(hr. sq.ft.degree
    Rankine).

```

```

16 hc2=1.5; // convection coefficient between the glass
    wool and the ambient air in BTU/(hr. sq.ft.
    degree Rankine).
17 U=1/((1/hc1)+(D1*log(D2/D1)/k12)+(D1*log(D3/D2)/k23)
    +(D1/(hc2*D3)));
18 printf("\nOverall heat transfer coefficient is %.3f
    BTU/(hr. sq.ft.degree Fahrenheit)",U);

```

Scilab code Exa 2.7 Thermal Contact Resistance

```

1 clc;
2 clear;
3 printf("\t\t\tChapter2_example7\n\n\n");
4 // Determination of the thermal contact resistance
5 k=14.4; // thermal conductivity of 304 stainless
    steel in W/(m.K)from appendix table B2
6 T1=543; // temperature in K at point 1
7 T2=460; // temperature in K at point 2
8 dT=T1-T2; // temperature difference between point 1
    and 2
9 dz12=0.035; // distance between thermocouple 1 and 2
    in cm
10 qz_per_A=k*dT/dz12; // heat flow calculated in W/m^2
    calculated using Fourier's law
11 printf("\nHeat flow calculated is %.2f kW/sq.m",
    qz_per_A/1000);
12 dz56=4.45; // distance between thermocouple 5 and 6
    in cm
13 dz6i=3.81; // distance between thermocouple 6 and
    interface in cm
14 dz5i=dz56+dz6i; // distance between thermocouple 5
    and interface in cm
15 T5=374; // temperature in K at point 5

```

```

16 T6=366; // temperature in K at point 6
17 T_ial=T5-(dz5i*(T5-T6)/dz56); // temperature of
    aluminium interface in K
18 printf("\nTemperature of aluminium interface is %.1f
    K",T_ial);
19 dzi7=2.45; // distance between thermocouple 7 and
    interface in cm
20 dz78=4.45; // distance between thermocouple 7 and 8
    in cm
21 dzi8=dzi7+dz78; // distance between thermocouple 8
    and interface in cm
22 T7=349; // temperature in K at point 7
23 T8=337; // temperature in K at point 8
24 T_img=dzi8*(T7-T8)/dz78+T8; // temperature of
    magnesium interface in K
25 printf("\nTemperature of magnesium interface is %.1f
    K",T_img);
26 Rtc=(T_ial-T_img)/qz_per_A;
27 printf("\nThe required thermal contact resistance is
    %.2e K. sq.m/W",Rtc);

```

Scilab code Exa 2.8 Analysis of a Pin Fin

```

1 clc;
2 clear;
3 printf("\t\t\tChapter2_example8\n\n\n");
4 // determination of temperature profile , heat
    transferred , efficiency , effectiveness .
5 printf("\n\t\t\tSolution to part (a)");
6 k=24.8; // thermal conductivity of 1C steel in BTU/(
    hr.ft.degree Rankine)from appendix table B2

```

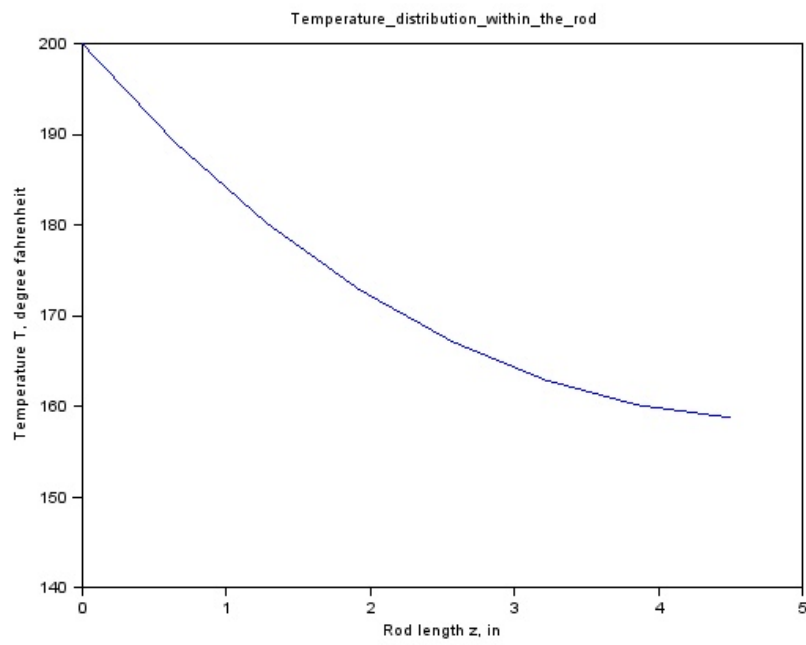



Figure 2.1: Analysis of a Pin Fin

```

7 D=(5/16)/12; // diameter of the rod in ft
8 P=(3.14*D); // Circumference of the rod in ft
9 printf("\nThe perimeter is %.4f ft",P);
10 A=(3.14/4)*D^2; // Cross sectional area of the rod
    in sq.ft
11 printf("\nThe Cross sectional area is %.6f sq.ft",A)
    ;
12 hc=1; // assuming the convective heat transfer
    coefficient as 1 BTU/(hr. sq.ft. degree Rankine)
13 m=sqrt(hc*P/(k*A));
14 printf("\nThe value of parameter m is: %.3f/ft",m);
15 L=(9/2)/12; // length of rod in ft
16 // using the equation  $(T-T_{inf})/(T_w-T_{inf})=(\cosh[m(L-z)])/(\cosh(mL))$  for temperature profile
17 T_inf=70;
18 T_w=200;
19 dT=T_w-T_inf;
20 const=dT/cosh(m*L);
21 printf("\nThe temperature profile is:\t");
22 printf("T=%d+%.2f cosh [%.3f (%.3f-z)]",T_inf,const,m,L
    );
23 z=0:.05:L;
24 T=T_inf+const*cosh(m*(L-z));
25 x=linspace(0,4.5,8);
26 plot(x,T);
27 a=gca();
28 a.data_bounds=[0,140;5,200];
29 newticks=a.x_ticks;
30 newticks(2)=[0;1;2;3;4;5];
31 newticks(3)=[ '0' ; '1' ; '2' ; '3' ; '4' ; '5' ];
32 a.x_ticks=newticks;
33 newticks1=a.y_ticks;
34 newticks1(2)=[140;150;160;170;180;190;200];
35 newticks1(3)=[ '140' ; '150' ; '160' ; '170' ; '180' ; '190' ; '
    200' ];
36 a.y_ticks=newticks1;
37 xlabel('Rod length z, in');
38 ylabel('Temperature T, degree fahrenheit');

```

```

39 title('Temperature_distribution_within_the_rod');
40 printf("\n\n\t\t\tSolution to part (b)\n");
41 // the heat transferred can be calculated using the
    equation qz=k*A*m*(T_w-T_inf)*tanh(m*L)
42 qz=k*A*m*dT*tanh(m*L);
43 printf("\nThe heat transferred is %.2f BTU/hr",qz);
44 printf("\n\n\t\t\tSolution to part (c)\n");
45 mL=m*L;
46 printf("\nThe value of mL is: %.3f",mL);
47 efficiency=0.78;
48 printf("\nThe efficiency found from the graph in
    figure 2.30 is: %.2f",efficiency);
49 printf('\n\n\t\t\tSolution to part (d)\n');
50 // the effectiveness can be found using the equation
    effectiveness=sqrt(k*P/h*A)*tanh(mL)
51 effectiveness=sqrt(k*P/(hc*A))*tanh(mL);
52 printf("\nThe effectiveness is found to be: %.1f",
    effectiveness);

```

Scilab code Exa 2.9 Corrected Length Solution

```

1  clc;
2  clear;
3  printf("\t\t\tChapter2_example9\n\n\n");
4  // determination of heat transferred
5  k=136; // thermal conductivity of aluminium in BTU/(
    hr.ft.degree Rankine)from appendix table B1
6  L=9/(8*12);
7  W=9/(4*12);
8  delta=1/(32*12);
9  printf("\nLength=%.5f ft , Width=%.4f ft , Delta=%.6f
    ft",L,W,delta);
10 hc=0.8; // the convective heat transfer coefficient

```

```

        estimated as 1 BTU/(hr.ft^2. degree Rankine)
11 T_w=1000; // the root temperature in degree
    fahrenheit
12 T_inf=90; // the ambient temperature in degree
    fahrenheit
13 m=sqrt(hc/(k*delta));
14 printf("\nThe value of m is %.3f",m);
15 P=2*W;
16 A=2*delta*W;
17 printf("\n\t\t\tSolution to part (a)\n");
18 qz1=sqrt(hc*P*k*A)*(T_w-T_inf)*(sinh(m*L)+(hc/(m*k)*
    cosh(m*L)))/(cosh(m*L)+(hc/(m*k)*sinh(m*L)));
19 printf("\nThe heat transferred is %.2f BTU/hr",qz1);
20 printf("\n\n\t\t\tSolution to part (b)\n");
21 qz2=sqrt(k*A*hc*P)*(T_w-T_inf)*tanh(m*L);
22 printf("\nThe heat transferred is %.2f BTU/hr\n",qz2
    );
23 printf("\n\t\t\tSolution to part (c)\n");
24 Lc=L+delta;
25 qz3=k*A*m*(T_w-T_inf)*tanh(m*L*(1+delta/Lc));
26 printf("\nThe heat transferred is %.2f BTU/hr\n",qz3
    );

```

Scilab code Exa 2.10 Straight fins of triangular profile

```

1 clc;
2 clear;
3 printf("\t\t\tChapter2_example10\n\n\n");
4 // determination of optimum fin length and heat
    transferred by fin
5 k=8.32; // thermal conductivity of Type 304
    stainless steel in BTU/(hr.ft.degree Rankine)from
    appendix table B2

```

```

6 hc=400; // the convective heat transfer coefficient
   given in BTU/(hr.ft^2. degree Rankine)
7 printf("\n\t\t\tSolution to part (a)\n");
8 delta_opt=0.55/(12*2);
9 // determination of dimension of one fin using the
   equation delta_opt=0.583*hc*Lc^2/k
10 Lc=sqrt(delta_opt*k/(0.583*hc));
11 printf("\nThe optimum length is %.3f in",Lc*12);
12 printf("\n\n\t\t\tSolution to part (b)\n");
13 A=Lc*delta_opt;
14 // determination of parameter for finding out
   efficiency from graph
15 parameter=Lc^1.5*sqrt(hc/(k*A));
16 printf("\nThe parameter value for finding the
   efficiency is: %.2f",parameter);
17 efficiency=0.6;
18 printf("\nThe efficiency found from the graph in
   figure 2.36 is %.1f", efficiency);
19 W=1/(2*12); // width in ft
20 T_w=190; // wall temperature in degree fahrenheit
21 T_inf=58; // ambient temperature in degree
   fahrenheit
22 L=1; // length in ft
23 delta=W/2;
24 q_ac=efficiency*hc*2*W*sqrt(L^2+delta^2)*(T_w-T_inf)
   ;
25 printf("\nThe actual heat transferred is %d BTU/hr",
   q_ac);

```

Scilab code Exa 2.11 Circular fin of rectangular profile

```

1 clc;
2 clear;

```

```

3 printf("\t\t\tChapter2_example11\n\n\n");
4 // determination of heat transferred and fin
  effectiveness
5 printf("\t\t\tSolution to part (a)\n");
6 //parameters of the problem are
7 N=9; // number of fins
8 delta=0.003/2;
9 L=0.025;
10 Lc=L+delta;
11 R=0.219/2;
12 R2c=R+delta;
13 R1=R-L;
14 T_w=260; // root wall temperature in degree celsius
15 T_inf=27; // ambient temperature in degree celsius
16 hc=15;
17 k=52; // thermal conductivity of cast iron in W/(m.K
  )from appendix table B2
18 Ap=2*delta*Lc;
19 As=2*3.14*(R2c^2-R1^2);
20 radius_ratio=R2c/R1; // for finding efficiency from
  figure 2.38
21 variable=Lc^1.5*sqrt(hc/(k*Ap));
22 printf("\n\nThe value of R2c/R1 is %.2f",
  radius_ratio);
23 printf("\n\nThe value of Lc^(3/2)(hc/kAp)^(1/2) is %
  .2f",variable);
24 efficiency=0.93; // efficiency from figure 2.38
25 printf("\n\nThe efficiency of the fin from figure
  2.38 is %.2f",efficiency);
26 qf=N*efficiency*As*hc*(T_w-T_inf);
27 printf("\n\nThe heat transferred by the nine fins is
  %.1f w",qf);
28 Sp=0.0127; // fin spacing
29 Asw=2*3.14*R1*Sp*N; // exposed surface area
30 qw=hc*Asw*(T_w-T_inf);
31 printf("\n\nThe heat transferred by exposed surface
  of the cylinder is %d W",qw);
32 q=qf+qw;

```

```

33 printf("\n\nThe total heat transferred from the
    cylinder is %d W",q)
34 printf("\n\n\t\t\tSolution to part (b)\n");
35 H=N*(Sp+2*delta); // height of cylinder
36 Aso=2*3.14*R1*H; // surface area without fins
37 qo=hc*Aso*(T_w-T_inf);
38 printf("\n\nThe Heat transferred without fins is %d
    W",qo)
39 printf("\n\n\t\t\tSolution to part (c)\n");
40 effectiveness=q/qo; // effectiveness defined as
    ratio of heat transferred with fins to heat
    transferred without fins
41 printf("\nThe fin effectiveness is %.2f",
    effectiveness);

```

Chapter 3

Steady State Conduction in Multiple Dimensions

Scilab code Exa 3.1 Flow Net method of solution

```
1  clc;
2  clear;
3  printf("\t\t\tChapter3_example1\n\n\n");
4  // Determination of the heat-flow rate from one tube
5  // specifications of 1 standard type K from table F2
6  OD=0.02858; // outer diameter in m
7  // from figure 3.11
8  M=8; // total number of heat-flow lanes
9  N=6; // number of squares per lane
10 S_L=M/N; // conduction shape factor
11 printf("\nThe Conduction shape factor is %.3f",S_L);
12 k=0.128; // thermal conductivity in W/(m.K) for
    concrete from appendix table B3
13 T1=85; // temperature of tube surface
14 T2=0; // temperature of ground beneath the slab
15 q_half=k*S_L*(T1-T2);
16 printf("\nThe heat flow per unit length from one
```



```

    half of one tube is %.1f W/m", q_half);
17 q=2*q_half;
18 printf("\nThe total heat flow per tube is %.1f W/m",
    q);

```

Scilab code Exa 3.2 Conduction Shape Factor

```

1  clc;
2  clear;
3  printf("\t\t\tChapter3_example2\n\n\n");
4  // Determination of the heat transferred from the
    buried pipe per unit length
5  // shape factor number 8 is selected from table 3.1
6  // specifications of 10 nominal, schedule 80 pipe
    from table F1
7  OD=10.74/12; // diameter in ft
8  R=OD/2;
9  T1=140;
10 T2=65;
11 k=0.072; // thermal conductivity in BTU/(hr-ft.
    degree R)
12 d=18/12; // distance from centre-line
13 S_L=(2*pi)/(acosh(d/R));
14 q_L=k*S_L*(T1-T2);
15 printf("\nThe heat transferred from the buried pipe
    per unit length is %.1f BTU/(hr.ft)", q_L);

```

Scilab code Exa 3.3 Heat lost through shape factor method

```

1  clc;
2  clear;
3  printf("\\t\\t\\tChapter3_example3\\n\\n\\n");
4  // Determination of the heat lost through the walls,
   using the shape-factor method. (b) Repeat the
   calculations but neglect the effects of the
   corners; that is, assume only one-dimensional
   effects through all the walls.
5  k = 1.07; // thermal conductivity of silica brick
   from appendix table B3 in W/(m.K)
6  // Calculation of total shape factor
7  // From figure 3.12, for component A
8  S1_A=0.138*0.138/0.006;
9  nA=2;
10 St_A=nA*S1_A; // Total shape factor of component A
11 printf("\\nThe Total shape factor of component A is %
   .3 f ",St_A);
12 // For component B
13 S1_B=0.138*0.188/0.006;
14 nB=4;
15 St_B=nB*S1_B; // Total shape factor of component B
16 printf("\\nThe Total shape factor of component B is %
   .3 f ",St_B);
17 // For component C
18 S3_C=0.15*0.006;
19 nC=8;
20 St_C=nC*S3_C; // Total shape factor of component C
21 printf("\\nThe Total shape factor of component C is %
   .4 f ",St_C);
22 // For component D
23 S2_D=0.54*0.188;
24 nD=4;
25 St_D=nD*S2_D; // Total shape factor of component D
26 printf("\\nThe Total shape factor of component D is %
   .5 f ",St_D);
27 // For component E
28 S2_E=0.138*0.54;
29 nE=8;

```

```

30 St_E=nE*S2_E; // Total shape factor of component E
31 printf("\nThe Total shape factor of component E is %
    .5f ",St_E);
32 S=St_A+St_B+St_C+St_D+St_E;
33 printf("\nThe Total shape factor is %.2f",S);
34 printf("\n\t\t\tSolution to part (a)\n");
35 T1=550;
36 T2=30;
37 q=k*S*(T1-T2);
38 printf("\nThe heat transferred through the walls of
    the furnace is %d W = %.1f kW",q,q/1000);
39 printf("\n\n\t\t\tSolution to part (b)\n");
40 // Neglecting the effects of the edges and corners ,
    the shape factor for all walls is found as
41 S=St_A+St_B;
42 printf("\nNeglecting the effects of the edges and
    corners , the shape factor for all walls is %.2f",
    S);
43 q_1=k*S*(T1-T2);
44 printf("\nNeglecting the effects of the edges and
    corners , the heat transferred is %d W = %.1f kW",
    q_1,q_1/1000);
45 Error=(q-q_1)/q;
46 printf("\nThe error introduced by neglecting heat
    flow through the edges and corners is %.1f
    percent",Error*100);

```

Scilab code Exa 3.4 Shape factor for given pipe

```

1 clc;
2 clear;
3 printf("\t\t\tChapter3_example4\n\n\n");
4 // Determination of the conduction shape factor for

```

```

    the underground portion of the configuration
5 // specifications of 4 nominal, schedule 40 pipe
    from table F1
6 OD=4.5/12; // diameter in ft
7 R=OD/2;
8 // For pipe A
9 L_A=4.5; // length in ft
10 // shape factor number 9 is selected from table 3.1
11 S_A=(2*pi*L_A)/(log(2*(L_A)/R));
12 printf("\nThe Shape Factor of pipe A is %.1f",S_A);
13 // For pipe B
14 L_B=18; // length in ft
15 // shape factor number 9 is selected from table 3.1
16 S_B=(2*pi*L_B)/(acosh(L_A/R));
17 printf("\nThe Shape Factor of pipe B is %.1f",S_B);
18 S=2*S_A+S_B;
19 printf("\nThe total conduction shape factor for the
    system is %.1f",S);

```

Scilab code Exa 3.5 Numerical Method for temperature distribution

```

1 clc;
2 clear;
3 printf("\t\t\tChapter3_example5\n\n\n");
4 // (a) Using the pin-fin equations for the case
    where the exposed tip is assumed insulated, graph
    the temperature distribution existing within the
    rod. (b) Use the numerical formulation of this
    section to obtain the temperature distribution. (
    c) Compare the two models to determine how well
    the numerical results approximate the exact

```

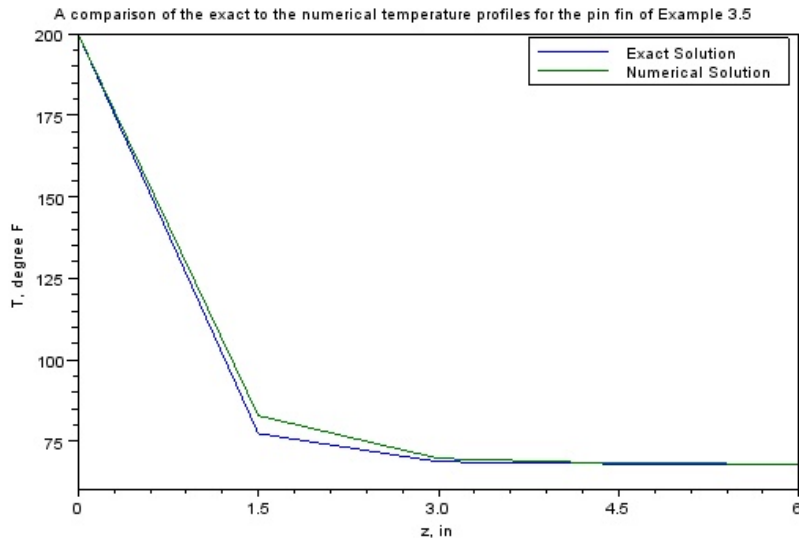


Figure 3.1: Numerical Method for temperature distribution

```

    results
5  h=1.1; // convective coefficient in BTU/(hr.ft^2.
    degree R)
6  Tw=200;
7  T_inf=68; // ambient temperature
8  printf("\n\t\t\tSolution to part (a)\n");
9  k=0.47; // thermal conductivity in BTU/(hr.ft.degree
    R) from table B3
10 D=0.25/12; // diameter in ft
11 A=%pi*D^2/4; // cross sectional area in ft^2
12 P=%pi*D; // perimeter in ft
13 printf("\nThe cross sectional area is %.3e sq.ft and
    Perimeter is %.3e ft",A,P);
14 L=6/12; // length in ft
15 mL=L*((h*P)/(k*A))^0.5;
16 printf("\nThe value of Product mL is %.2f",mL);
17 z=0:1.5:6;
18 [n m]=size(z);
19 for i=1:m

```

```

20     T(i)=T_inf+(Tw-T_inf)*(cosh(mL*(1-(z(i)/6)))/(
        cosh(mL)));
21 end
22 printf("\n\n\t\t\tSolution to part (b)\n");
23 d_zeta=1/4;
24 K=2+(mL*d_zeta)^2;
25 printf("\nThe value of K is %.4f",K);
26 T_(5)=T_inf+(Tw-T_inf)*(2/(K^4-4*K^2+2));
27 T_(4)=T_inf+(Tw-T_inf)*(K/(K^4-4*K^2+2));
28 T_(3)=T_inf+(Tw-T_inf)*((K^2-1)/(K^4-4*K^2+2));
29 T_(2)=T_inf+(Tw-T_inf)*((K^3-3*K)/(K^4-4*K^2+2));
30 T_(1)=200;
31 printf("\n\nA Comparison of Exact to Numerical
    Results for the Data of Example 3.5");
32 printf("\nz, in\tExact (e) T\tNumerical (n) T\t
    Percent error (e - n)/e");
33 for i=1:m
34 err(i)=(T(i)-T_(i))/T(i);
35 printf("\n%.1f\t%.2f\t\t%.2f\t\t%.2f\n",z(i),T(i),T_(
    i),err(i)*100);
36 end
37 plot(z,T,z,T_);
38 a=gca();
39 newticks1=a.x_ticks;
40 newticks1(2)=[0;1.5;3.0;4.5;6];
41 newticks1(3)=[ '0 ' ; ' 1.5 ' ; ' 3.0 ' ; ' 4.5 ' ; ' 6 ' ];
42 a.x_ticks=newticks1;
43 newticks2=a.y_ticks;
44 newticks2(2)=[75;100;125;150;175;200];
45 newticks2(3)=[ ' 75 ' ; ' 100 ' ; ' 125 ' ; ' 150 ' ; ' 175 ' ; ' 200 ' ];
46 a.y_ticks=newticks2;
47 title('A comparison of the exact to the numerical
    temperature profiles for the pin fin of Example
    3.5 ');
48 xlabel("z, in");
49 ylabel("T, degree F");
50 hl=legend(['Exact Solution';'Numerical Solution']);

```

Chapter 4

Unsteady State Heat Conduction

Scilab code Exa 4.1 Response Time of thermocouple junction

```
1  clc;
2  clear;
3  printf("\t\t\tChapter4_example1\n\n\n");
4  // determination of response time
5  k=12; // thermal conductivity in BTU/(hr.ft.degree
        Rankine)
6  c=0.1; // specific heat in BTU/(lbm.degree Rankine)
7  D=0.025/12; // diameter in ft
8  density=525; // density in lbm/cu.ft
9  hc=80; // convective coefficient in BTU/(hr. sq.ft.
        degree Rankine)
10 T_i=65; // intial temperature in degree fahrenheit
11 T_inf=140; // fluid temperature in degree fahrenheit
12 As=3.14*D^2; // surface area in sq.ft
13 Vs=3.14*D^3/6; // volume in cu.ft
14 reciprocal_timeconstant=(hc*As)/(density*Vs*c);
15 printf("\nThe reciprocal of time constant is %.1f /
```

```

    hr",reciprocal_timeconstant);
16 // selecting T=139 degree fahrenheit as T=140 gives
    an infinite time through the equation  $(T-T_{inf})/(T_i-T_{inf})=exp(-hc*As/density*Vs*c)t$ 
17 T=139;
18 t=log((T-T_inf)/(T_i-T_inf))/(-
    reciprocal_timeconstant);
19 printf('\n\nThe response time of the junction is %.1
    f s",t*3600);

```

Scilab code Exa 4.2 Lumped Capacitance Approach

```

1 clc;
2 clear;
3 printf("\t\t\tChapter4_example2\n\n\n");
4 // Determination of temperature of metal and
    cumulative heat rate
5 // properties of aluminium from appendix table B1
6 k=236; // thermal conductivity in W/(m.K)
7 Cp=896; // specific heat in J/(kg.K)
8 sp_gr=2.702; // specific gravity
9 density=2702; // density in kg/cu.m
10 D=0.05; // Diameter in m
11 L=0.60; // length in m
12 hc=550; // unit surface conductance between the
    metal and the bath in W/(K.sq.m)
13 Vs=(3.14*D^2*L)/4; // Volume in cu.m
14 As=(2*3.14*D^2/4)+(3.14*D*L); // surface area in sq.
    m
15 printf("\n\nThe volume of cylinder is %.5f cu.m",Vs)
    ;
16 printf("\n\nThe surface area of cylinder is %.3f sq.
    m",As);
17 Bi=(hc*Vs)/(k*As); // Biot Number
18 printf("\n\nThe Biot number is %.3f",Bi);

```



```

19 // Biot number is less than 1 hence lump capacitance
    equations apply
20 printf("\n\n\t\t\tSolution to part (a)\n");
21 T_i=50; // initial temperature in degree celsius
22 T_inf=2; // temperature of ice water bath in degree
    celsius
23 t=60; // time=1 minute=60 s
24 T=T_inf+(T_i-T_inf)*exp(-(hc*As)/(density*Vs*Cp)*t);
25 printf("\nThe temperature of aluminium is %.1f
    degree celsius",T);
26 printf("\n\n\t\t\tSolution to part (b)\n");
27 Q=density*Vs*Cp*(T_inf-T_i)*[1-exp(-(hc*As)/(density
    *Vs*Cp)*t)];
28 printf("\nThe cumulative heat transferred is %d J =%
    .1f kJ",abs(Q),abs(-Q/1000));

```

Scilab code Exa 4.3 Temperature profile through Infinite plate chart

```

1 clc;
2 clear;
3 printf("\t\t\tChapter4_example3\n\n\n");
4 hc=30;
5 L=0.24;
6 k=1.25;
7 c=890;
8 rou=550;
9 Bi=hc*L/k;
10 alpha=k/(rou*c);
11 printf("The value of diffusivity is %.2e sq.m/s",
    alpha);
12 Tc=150;

```

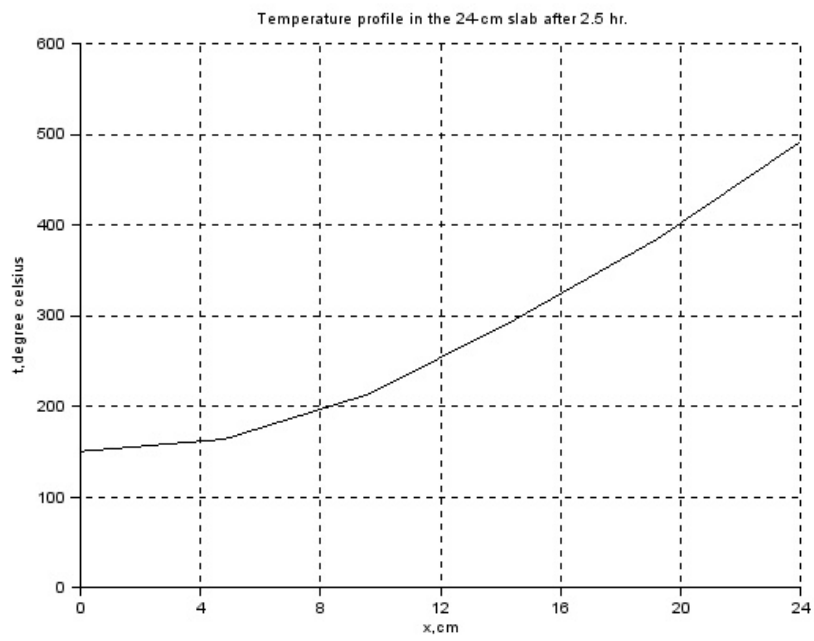


Figure 4.1: Temperature profile through Infinite plate chart

```

13 T_inf=600;
14 T_i=50;
15 printf("\nThe Biot number is %.2f,",Bi);
16 if Bi<0.1 then
17     n=0;
18 else if Bi>0.1 then
19     n=1;
20 end
21 end
22 select n
23 case 0 then
24     disp('The Lumped capacity approach is applicable
25         ');
26 case 1 then
27     disp('Since value of Biot number is greater than
28         0.1, Lumped capacity approach would not give
29         accurate results, so figure 4.6 is to be
30         used');
31     reciprocal_Bi=1/Bi;
32     dimensionless_temp=(Tc-T_inf)/(T_i-T_inf);
33     Fo=0.4; //the value of Fourier Number from
34             figure 4.6(a)
35     t=L^2*Fo/alpha;
36     printf("The required time is %d s = %.1f hr",t,t
37         /3600);
38 end
39 // reading values of dimensionless temperature from
40 // figure 4.6(b) using reciprocal of Biot number
41 x_per_L=[0 0.2 0.4 0.6 0.8 0.9 1.0];
42 [n,m]=size(x_per_L);
43 printf("\nThe choosen values of x/L are: \n");
44 disp(x_per_L);
45 printf("\n Values for dimensionless temperature for
46     corresponding values of x/L:")
47 dim_T=[1.0 .97 .86 .68 .48 .36 .24]; // value for
48     dimensionless temperature for corresponding value
49     of x/L
50 disp(dim_T);

```

```

41 printf("the temperature profile with distance is\n")
    ;
42 printf("\tx/L\t\t");
43 for j=1:m
44     printf("%.2f\t", x_per_L(1,j));
45
46 end
47 printf("\n");
48 printf("(T-T_inf)/T_i-T_inf)\t");
49 for i=1:m
50     printf("%.2f\t", dim_T(i));
51 end
52 T=zeros(1,m);
53 x=zeros(1,m);
54 for i=1:m
55     T(1,i)=dim_T(1,i)*dimensionless_temp*(T_i-T_inf)
        +T_inf;
56     x(1,i)=x_per_L(1,i)*L;
57 end
58 printf("\n\tx ,cm\t\t");
59 for i=1:m
60     X(1,i)=x(1,i)*100;
61     printf("%.1f\t", X(1,i));
62 end
63 printf("\nT, degree celsius\t");
64 for i=1:m
65     printf("%d\t", T(1,i));
66 end
67 plot2d(X,T,rect=[0,0,24,600]);
68 a=gca();
69 newticks=a.x_ticks;
70 newticks(2)=[0;4;8;12;16;20;24];
71 newticks(3)=[ '0' ; '4' ; '8' ; '12' ; '16' ; '20' ; '24' ];
72 a.x_ticks=newticks;
73 newticks1=a.y_ticks;
74 newticks1(2)=[0;100;200;300;400;500;600];
75 newticks1(3)=[ '0' ; '100' ; '200' ; '300' ; '400' ; '500' ; '600
    '];

```

```

76 a.y_ticks=newticks1;
77 xlabel('x,cm');
78 ylabel('t,degree celsius');
79 title('Temperature profile in the 24-cm slab after
      2.5 hr. ');
80 filename='Temperature profile in the 24-cm slab
      after 2.5 hr. ';
81 xgrid(1);
82 xs2jpg(0,filename);

```

Scilab code Exa 4.4 Dimensionless temperature history of a sphere

```

1  clc;
2  clear;
3  printf("\t\t\tChapter4_example4\n\n\n");
4  hc=6;
5  D=0.105;
6  k=0.431;
7  c=2000;
8  rou=998;
9  Vs=%pi*D^3/6;
10 As=%pi*D^2;
11 // calculating Biot Number for lumped capacitance
    approach
12 Bi_lumped=hc*Vs/(k*As);
13 printf("\nThe Biot number is %.3f,",Bi_lumped);
14 alpha=k/(rou*c);
15 printf("\nThe value of diffusivity is %.2e sq.m/s",
    alpha);
16 Tc=20;
17 T_inf=23;
18 T_i=4;
19 if Bi_lumped<0.1 then

```

```

20     n=0;
21 else if Bi_lumped>0.1 then
22     n=1;
23 end
24 end
25 select n
26 case 0 then
27     disp('The Lumped capacity approach is applicable
28         ');
29 case 1 then
30     printf("\n\nSince value of Biot number is
31         greater than 0.1,\nLumped capacity approach
32         would not give accurate results , so figure
33         4.8 is to be used\n");
34     // calculating Biot Number for using figure 4.8
35     Bi_figure=hc*D/(2*k);
36     printf("\nThe Biot Number for using figure 4.8
37         is %.3f",Bi_figure);
38     reciprocal_Bi=1/Bi_figure;
39     dimensionless_temp=(Tc-T_inf)/(T_i-T_inf);
40     printf("\nThe dimensionless temperature is %.3f"
41         ,dimensionless_temp);
42     Fo=1.05;//The corresponding value of Fourier
43         Number from figure 4.8a
44     t=(D/2)^2*Fo/alpha;
45     printf("\nThe required time is %.2e s = %.1f hr"
46         ,t,t/3600);
47 end
48 Bi2Fo=Bi_figure^2*Fo;
49 printf("\nBi^2Fo=%.1e",Bi2Fo);
50 Dimensionless_HeatFlow=0.7; // The corresponding
51     dimensionless heat flow ratio from figure 4.8c
52 Q=Dimensionless_HeatFlow*rou*c*Vs*(T_i-T_inf);
53 printf("\nThe heat transferred is %.3e J",Q);

```

Scilab code Exa 4.5 Estimation of the depth of Freeze line

```
1  clc;
2  clear;
3  printf("\t\t\tChapter4_example5\n\n\n");
4  hc=6;
5  D=0.105;
6  k=0.3;
7  c=0.41;
8  sp_gr=2.1;
9  rou_water=62.4;
10 alpha=k/(sp_gr*rou_water*c);
11 printf("\nThe diffusivity of the soil is %.2e sq.ft/
    hr",alpha);
12 t=3*30*24;
13 printf("\nTime in hours is %d hr",t);
14 // Bi_sqrt(Fo) is infinite
15 T_inf=10;
16 Ts=10;
17 T=32;
18 T_i=70;
19 dimensionless_temp=(T-T_i)/(T_inf-T_i);
20 printf("\nThe dimensionless temperature is %.4f",
    dimensionless_temp);
21 variable_fig4_12=0.38; //The value of x/(2*(alpha*t)
    ^0.5) from figure 4.12
22 x=2*sqrt(alpha*t)*variable_fig4_12;
23 printf("\nThe depth of the freeze line in soil is %
    .2f ft",x);
```

Scilab code Exa 4.6 Determination of temperature using kpc product

```
1  clc;
2  clear;
3  printf("\t\t\tChapter4_example6\n\n\n");
4  // properties of aluminium from appendix table B1
5  k_al=236;
6  p_al=2.7*1000;
7  c_al=896;
8  // properties of oak from appendix table B3
9  k_oak=0.19;
10 p_oak=0.705*1000;
11 c_oak=2390;
12 sqrt_kpc_al=sqrt(k_al*p_al*c_al);
13 printf("\nThe square root of kpc product of
    aluminium is %.2e sq.W.s/(m^4.sq.K)",sqrt_kpc_al)
    ;
14 kpc_R=4;
15 T_Li=20;
16 T_Ri=37.3;
17 T_al=(T_Li*(sqrt_kpc_al)+T_Ri*sqrt(kpc_R))/(
    sqrt_kpc_al+sqrt(kpc_R));
18 printf("\nThe temperature of aluminium is felt as %
    .1f degree celsius",T_al);
19 sqrt_kpc_oak=sqrt(k_oak*p_oak*c_oak);
20 printf("\nThe square root of kpc product of oak is %
    .2e sq.W.s/(m^4.sq.K)",sqrt_kpc_oak);
21 T_oak=(T_Li*(sqrt_kpc_oak)+T_Ri*sqrt(kpc_R))/(
    sqrt_kpc_oak+sqrt(kpc_R));
22 printf("\nThe temperature of oak is felt as %.1f
    degree celsius",T_oak);
23 if (T_al>T_oak) then
```



```

24     printf("\nThe aluminium will feel warmer.");
25 elseif (T_al<T_oak) then
26         printf("\nThe oak will feel warmer.");
27 else
28     printf("\nBoth will be felt equally warm.")
29 end

```

Scilab code Exa 4.7 Combination of 1D Transient Systems for beer can

```

1  clc;
2  clear;
3  printf("\t\t\tChapter4_example7\n\n\n");
4  // properties of water at 68 degree fahrenheit from
   appendix table C11
5  rou=62.46;
6  cp=0.9988;
7  k=0.345;
8  alpha=k/(rou*cp);
9  printf("\nThe diffusivity at 68 degree fahrenheit is
   %.2e sq.ft/hr",alpha);
10 D=2.5/12;
11 L=4.75/12;
12 Vs=%pi*D^2*L/4;
13 As=(%pi*D*L)+(%pi*D^2)/2;
14 Lc=Vs/As;
15 printf("\nThe volume of the can is %.4f cu.ft",Vs);
16 printf("\nThe surface area of the can is %.3f sq.ft"
   ,As);
17 printf("\nThe characteristic length is %.3f ft",Lc);
18 hc=1.7;
19 Bi=hc*Lc/k;
20 printf("\nThe Biot number is %.3f",Bi);
21 t=4;

```

```

22 // for the cylinder solution
23 Fo_cylinder=alpha*t/(D/2)^2;
24 Bi_cylinder=hc*(D/2)/k;
25 printf("\nFor the cylinder, The Fourier number is %
    .2f and Biot Number is %.3f",Fo_cylinder,
    Bi_cylinder);
26 reciprocal_Bi_cylinder=1/Bi_cylinder;
27 printf("\nThe reciprocal for Biot number for
    cylinder is %.2f",reciprocal_Bi_cylinder);
28 dim_T_cylinder=0.175; //The value of dimensionless
    temperature of cylinder from figure 4.7a at
    corresponding values of Fo and 1/Bi
29 // for the infinite plate solution
30 Fo_plate=alpha*t/(L/2)^2;
31 Bi_plate=hc*L/(2*k);
32 printf("\nFor the infinite plate, The Fourier number
    is %.3f and Biot Number is %.2f",Fo_plate,
    Bi_plate);
33 reciprocal_Bi_plate=1/Bi_plate;
34 printf("\nThe reciprocal for Biot number for
    infinite plate is %.2f",reciprocal_Bi_plate);
35 dim_T_plate=0.55; //The value of dimensionless
    temperature of infinite plate from figure 4.7a at
    corresponding values of Fo and 1/Bi
36 // Table 4. I, for the short-cylinder problem,
    indicates that the solution is the product of the
    infinite-cylinder problem (Figure 4.7) and the
    infinite-plate problem (Figure 4.6).
37 // For short cylinder problem
38 dim_T_shortcylinder=dim_T_cylinder*dim_T_plate;
39 printf("\nThe value of dimensionless temperature for
    short cylinder is %.3f ",dim_T_shortcylinder);
40 T_inf=30;
41 T_i=72;
42 Tc=dim_T_shortcylinder*(T_i-T_inf)+T_inf;
43 printf("\nThe temperature at centre of can is %.1f
    degree celsius",Tc);
44 dim_Tw_cylinder=0.77; //The dimensionless

```

```

    temperature from figure 4.7b corresponding to the
    value of 1/Bi and r/R=1
45 dim_Tw_plate=0.65; //The dimensionless temperature
    from figure 4.6b corresponding to the value of 1/
    Bi and x/L=1
46 dim_Tw_shortcylinder=dim_Tw_cylinder*dim_Tw_plate;
47 printf("\nThe value of dimensionless temperature at
    the wall for short cylinder is %.2f ",
    dim_Tw_shortcylinder);
48 Tw=dim_Tw_shortcylinder*(Tc-T_inf)+T_inf;
49 printf("\nThe wall temperature is %.1f degree F",Tw)
    ;

```

Scilab code Exa 4.8 Combination of 1D Transient Systems for rectangular bar

```

1  clc;
2  clear;
3  printf("\t\t\tChapter4_example8\n\n\n");
4  rou=7817;
5  c=461;
6  k=14.4;
7  alpha=.387e-5;
8  L1=.03;
9  L2=0.03;
10 L3=0.04;
11 x=0.04;
12 T_i=95;
13 T_inf=17;
14 // for infinite plate
15 L=L1/2;
16 hc=50;
17 reciprocal_Bi_plate=k/(hc*L);

```

```

18 printf("\nThe value of 1/Bi for infinite plate is %
    .1 f",reciprocal_Bi_plate);
19 T=50;
20 n=1;
21 t=[3000 1500 700 400 200 300 350];
22 [n m]=size(t);
23 // parameter for infinite plate Fourier Number,Fo is
    named as parameter1
24 for i=1:m
25     parameter1(i)=alpha*t(i)/L^2;
26 // parameters for semi-infinite solid Bi(Fo)^0.5 and
    x/(2*(alpha*t)^0.5) are named as parameter2 and
    parameter3
27 parameter2(i)=hc*((alpha*t(i))^0.5)/k;
28 parameter3(i)=x/(2*(alpha*t(i))^0.5);
29 dim_T_plate=[0.085 0.34 0.55 0.7 0.8 0.8 0.7]; //the
    corresponding values of dimensionless
    temperature for infinite plate from figure 4.6a
30 dim_T_solid=[0.225 0.14 0.075 0.046 0.02 0.035
    0.042]; // the corresponding values of
    dimensionless temperature for semi-infinite solid
    from figure 4.12
31 dim_T_bar(i)=dim_T_plate(i)*dim_T_plate(i)*(1-
    dim_T_solid(i));
32 T(i)=dim_T_plate(i)*dim_T_plate(i)*(1-dim_T_solid(i)
    )*(T_i-T_inf)+T_inf;
33 end
34 printf("\nThe Results for different time instances:\
    n");
35 printf("\n\tInfinite Plate\t\t\t\t\tSemi-Infinite
    Solid\t\t\t\t\tDimensionless Temperature\
    tTemperature");
36 printf("\ntime t, s\t1/Bi\tFo\t(T-Tinf)/(Ti-Tinf)\
    tBi(Fo)^0.5\tx/(2*(at)^0.5)\t(T-Tinf)/(Ti-Tinf)\t
    (T-Tinf)/(Ti-Tinf)\t\tT");
37 for i=1:m
38     printf("\n%d\t\t%.1 f\t%.2 f\t\t%.2 f\t\t%.3 f\t\t%
        .3 f\t\t%.3 f\t\t\t%.3 f\t\t\t\t%.1 f",t(i),

```

```

        reciprocal_Bi_plate,parameter1(i),dim_T_plate
        (i),parameter2(i),parameter3(i),dim_T_solid(i)
        ),dim_T_bar(i),T(i));
39 end

```

Scilab code Exa 4.9 Numerical Method for transient conduction system

```

1  clc;
2  clear;
3  printf("\t\t\tChapter4_example9\n\n\n");
4  rou=.5*1000;
5  cp=837;
6  k=0.128;
7  alpha=0.049e-5;
8  // let Fo=0.5 and dx=0.05
9  dt=0.5*(0.05)^2/alpha;
10 printf("\nThe time increment is %.3f hr",dt/3600);
11 p=1;
12 m=6;
13 A=2*eye(6,6);
14 n=1;
15 N=1;
16 for j=1:n
17     for i=1:6
18         T(i,j)=20;
19     end
20 end
21 for n=1:7
22     for i=1:4
23         B(i+1,n)=T(i+2,n)+T(i,n);
24         B(1,n)=T(i+1,n)+200;

```

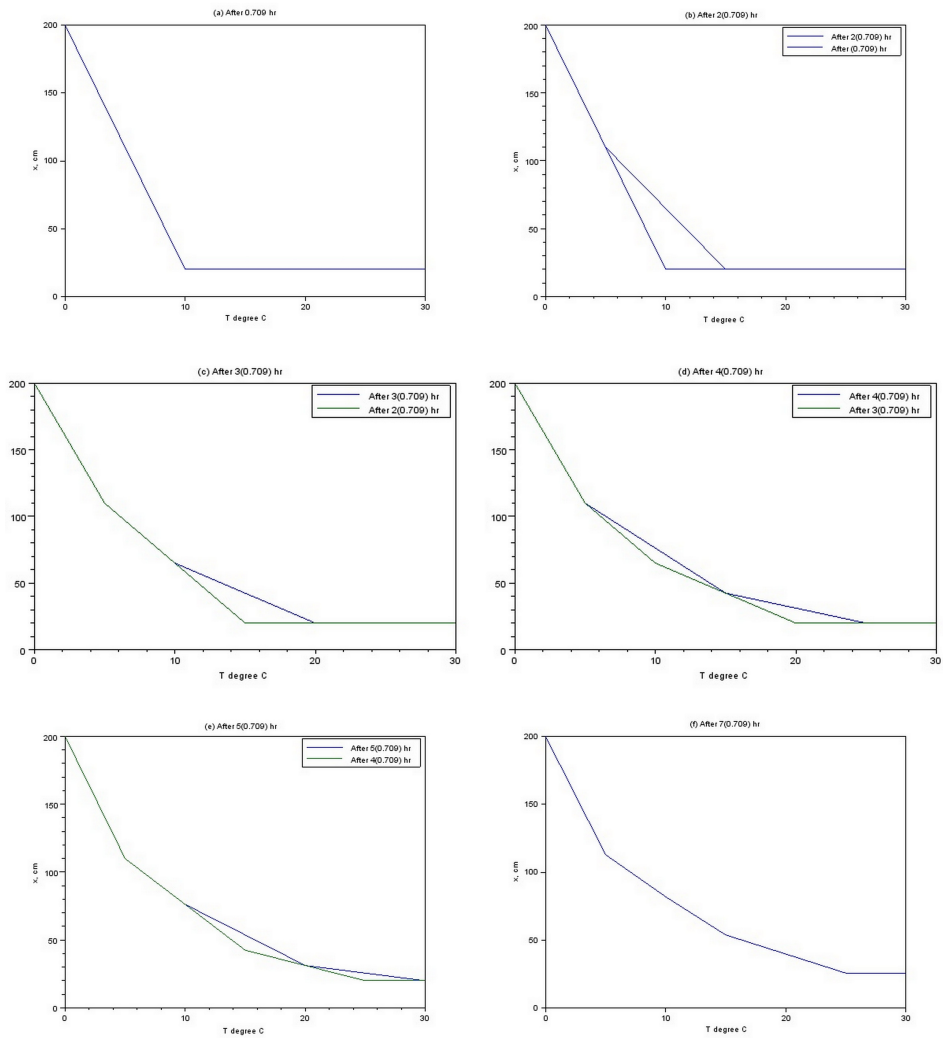


Figure 4.2: Numerical Method for transient conduction system

```

25         B(6,n)=2*T(i+1,n);
26     end
27     Temp=inv(A)*B(:,n); // temperature at the different
        points
28     printf("\nThe temperature at different points after
        %d time interval are:",n);
29     T(:,n+1)=Temp;
30     disp(T(:,n+1));
31 end
32 time=n*dt;
33 printf("\nThe required time is %.2f hr",time/3600);
34 x=0:5:30;
35 plot(x,[200;T(:,2)]);
36 a1=gca();
37 a1.data_bounds=[0,0;30,200];
38 xtitle('(a) After 0.709 hr','T degree C','x, cm');
39 newticks=a1.x_ticks;
40 newticks(2)=[0;10;20;30];
41 newticks(3)=[ '0'; '10'; '20'; '30' ];
42 a1.x_ticks=newticks;
43 newticks1=a1.y_ticks;
44 newticks1(2)=[0;50;100;150;200];
45 newticks1(3)=[ '0'; '50'; '100'; '150'; '200' ];
46 a1.y_ticks=newticks1;
47 plot(x,[200;T(:,3)]);
48 a2=gca();
49 hl=legend(['After 2(0.709) hr ','After (0.709) hr '
        ]);
50 a2.data_bounds=[0,0;30,200];
51 xtitle('(b) After 2(0.709) hr','T degree C','x, cm'
        );
52 newticks=a2.x_ticks;
53 newticks(2)=[0;10;20;30];
54 newticks(3)=[ '0'; '10'; '20'; '30' ];
55 a2.x_ticks=newticks;
56 newticks1=a2.y_ticks;
57 newticks1(2)=[0;50;100;150;200];
58 newticks1(3)=[ '0'; '50'; '100'; '150'; '200' ];

```

```

59 a2.y_ticks=newticks1;
60 filename='(b) After 2(0.709) hr ';
61 clf();
62 plot(x,[200;T(:,4)],x,[200;T(:,3)]);
63 a3=gca();
64 hl=legend(['After 3(0.709) hr ','After 2(0.709) hr '
           ]);
65 a3.data_bounds=[0,0;30,200];
66 xtitle('(c) After 3(0.709) hr ','T degree C','x, cm'
        );
67 newticks=a3.x_ticks;
68 newticks(2)=[0;10;20;30];
69 newticks(3)=[ '0 ','10 ','20 ','30 '];
70 a3.x_ticks=newticks;
71 newticks1=a3.y_ticks;
72 newticks1(2)=[0;50;100;150;200];
73 newticks1(3)=[ '0 ','50 ','100 ','150 ','200 '];
74 a3.y_ticks=newticks1;
75 clf();
76 plot(x,[200;T(:,5)],x,[200;T(:,4)]);
77 a4=gca();
78 hl=legend(['After 4(0.709) hr ','After 3(0.709) hr '
           ]);
79 a4.data_bounds=[0,0;30,200];
80 xtitle('(d) After 4(0.709) hr ','T degree C','x, cm'
        );
81 newticks=a4.x_ticks;
82 newticks(2)=[0;10;20;30];
83 newticks(3)=[ '0 ','10 ','20 ','30 '];
84 a4.x_ticks=newticks;
85 newticks1=a4.y_ticks;
86 newticks1(2)=[0;50;100;150;200];
87 newticks1(3)=[ '0 ','50 ','100 ','150 ','200 '];
88 a4.y_ticks=newticks1;
89 clf();
90 plot(x,[200;T(:,6)],x,[200;T(:,5)]);
91 a5=gca();
92 hl=legend(['After 5(0.709) hr ','After 4(0.709) hr '

```



```

    ]);
93 a5.data_bounds=[0,0;30,200];
94 xtitle('(e) After 5(0.709) hr ', 'T degree C', 'x, cm'
    );
95 newticks=a5.x_ticks;
96 newticks(2)=[0;10;20;30];
97 newticks(3)=[ '0 ' ; '10 ' ; '20 ' ; '30 ' ];
98 a5.x_ticks=newticks;
99 newticks1=a5.y_ticks;
100 newticks1(2)=[0;50;100;150;200];
101 newticks1(3)=[ '0 ' ; '50 ' ; '100 ' ; '150 ' ; '200 ' ];
102 a5.y_ticks=newticks1;
103 clf();
104 plot(x,[200;T(:,7)]);
105 a6=gca();
106 a6.data_bounds=[0,0;30,200];
107 xtitle('(f) After 7(0.709) hr ', 'T degree C', 'x, cm'
    );
108 newticks=a6.x_ticks;
109 newticks(2)=[0;10;20;30];
110 newticks(3)=[ '0 ' ; '10 ' ; '20 ' ; '30 ' ];
111 a6.x_ticks=newticks;
112 newticks1=a6.y_ticks;
113 newticks1(2)=[0;50;100;150;200];
114 newticks1(3)=[ '0 ' ; '50 ' ; '100 ' ; '150 ' ; '200 ' ];
115 a6.y_ticks=newticks1;

```

Scilab code Exa 4.10 Graphical Method for transient conduction system

```

1 clc;
2 clear;
3 printf("\t\t\tChapter4_example10\n\n\n");
4 // determination of time required to cool to a

```

```
        certain temperature
5 rou=7.817*62.4;
6 c=.110;
7 k=8.32;
8 alpha=0.417e-4;
9 dx=1/12;
10 // taking Fo=1
11 Fo=1;
12 dt=Fo*dx^2/alpha;
13 printf("\nThe time increments is %.1f s",dt);
14 // We have to draw the Saul'ev plot to determine the
    number of time intervals
15 n=8; //Enter the number of time intervals from
    Saulev plot
16 time=n*dt;
17 printf("\nThe required time is %.2f hr",time/3600);
```

Chapter 5

Introduction to Convection

Scilab code Exa 5.1 Heat transferred using specific heat

```
1  clc;
2  clear;
3  printf("\t\t\tChapter5_example1\n\n\n");
4  // properties of CO at 300K from appendix table D2
5  Cp=871;
6  Gamma=1.3;
7  Cv=Cp/Gamma;
8  printf("\nThe specific heat at constant volume is %d
          J/(kg.K)",Cv);
9  dT=20;
10 m=5;
11 Qp=m*Cp*dT;
12 Qv=m*Cv*dT;
13 printf("\n The heat required at constant pressure is
          %.1f kJ",Qp/1000);
14 printf("\nThe heat required at constant volume is %d
          kJ",Qv/1000);
```

Scilab code Exa 5.2 Volumetric thermal expansion coefficient

```
1  clc;
2  clear;
3  printf("\t\t\tChapter5_example2\n\n\n");
4  // properties of Freon-12 from appendix table C3
5  T1_Fr=-50;
6  T2_Fr=-40;
7  rou1_Fr=1.546*1000;
8  rou2_Fr=1.518*1000;
9  beta_Fr=-(rou1_Fr-rou2_Fr)/(rou1_Fr*(T1_Fr-T2_Fr));
10 printf("\nThe volumetric thermal expansion
        coefficient calculated for Freon-12 is %.3e /K",
        beta_Fr);
11 beta_acc_Fr=2.63e-3; // the accurate value of
        volumetric thermal expansion coefficient for
        Freon-12
12 error_Fr=(beta_acc_Fr-beta_Fr)*100/beta_acc_Fr;
13 printf("\nThe error introduced in the case of Freon
        -12 is %d percent",error_Fr);
14 // properties of helium from appendix table D3
15 T1_He=366;
16 T2_He=477;
17 rou1_He=0.13280;
18 rou2_He=0.10204;
19 beta_He=-(rou1_He-rou2_He)/(rou1_He*(T1_He-T2_He));
20 printf("\nThe volumetric thermal expansion
        coefficient calculated for Freon-12 is %.3e /K",
        beta_He);
```

Chapter 6

Convection Heat Transfer in a Closed Conduit

Scilab code Exa 6.1 Constant heat flux at the wall

```
1  clc;
2  clear;
3  printf("\t\t\tChapter6_example1\n\n\n");
4  // Determination of the fluid outlet tetnperature
   and the tube-wall temperature at the outlet.
5  // properties of ethylene glycol at 20 degree
   celsius from appendix table C5
6  Cp_20=2382;
7  rou_20=1.116*1000;
8  v_20=19.18e-6;
9  kf_20=.249;
10 a_20=.939e-7;
11 Pr_20=204;
12 // specifications of 1/2 standard type M seamless
   copper water tubing from appendix table F2
13 OD=1.588/100;
14 ID=1.446/100;
```

```

15 A=1.642e-4;
16 Q=3.25e-6;
17 V=Q/A;
18 printf("\nThe average flow velocity is %.1f m/s",V
    *100);
19 // calculation of Reynold's Number to check flow
    regime
20 Re=V*ID/v_20;
21 printf("\nThe Reynolds Number is %.1f",Re);
22 // since Re>he 2100, the flow regime is laminar and
    the hydrodynamic length can be calculated as
23 Z_h=0.05*ID*Re;
24 printf("\nThe hydrodynamic length is %.1f cm",Z_h
    *100);
25 Tbi=20; // bulk-fluid inlet temperature in degree
    celsius
26 qw=2200; // incident heat flux in W/m^2
27 L=3; // Length of copper tube in m
28 R=ID/2; // inner radius in m
29 Tbo=Tbi+(2*qw*a_20*L)/(V*kf_20*R);
30 printf("\nThe bulk-fluid outlet temperature is %.1f
    degree celsius",Tbo);
31 // This result is based on fluid properties
    evaluated at 20 C . taken as a first
    approximation
32 Z_t=0.05*ID*Re*Pr_20;
33 printf("\nThe thermal entry length is %.1f m",Z_t);
34 Two=Tbo+(11*qw*ID)/(48*kf_20); // The wall
    temperature at outlet in degree celsius
35 printf("\nThe wall temperature at outlet is %.1f
    degree celsius",Two);
36 //The result is based on first approximation based
    on flow properties evaluated at the fluid inlet
    temperature.

```

Scilab code Exa 6.2 Constant wall temperature

```
1 clc;
2 clear;
3 printf("\t\t\tChapter6_example2\n\n\n");
4 // determination of average convection coefficient
5 T_avg=(140+70)/2;
6 printf("\nThe average bulk temperature is %d degree
   celsius",T_avg);
7 // properties of water at average bulk temperature
   from appendix table C11
8 rou=.994*62.4;
9 kf=.363;
10 cp=.9980;
11 a=5.86e-3;
12 v=0.708e-5;
13 Pr=4.34;
14 // specifications of 1 standard type M copper tube
   from appendix table F2
15 OD=1.125/12; // outer diameter in ft
16 ID=0.8792; // inner diameter in ft
17 A=0.006071 // cross sectional area in sq.ft
18 m_flow=1.5; // mass flow rate in lbm/s
19 V=m_flow*3600/(rou*A); // velocity in ft/hr
20 printf("\nThe velocity is %d ft/hr",V);
21 L=20;
22 Tw=240;
23 Tbo=140;
24 Tbi=70;
25 hL=- (rou*V*ID*cp*log((Tw-Tbo)/(Tw-Tbi)))/(4*L);
26 printf("\nThe average convective coefficient is %d
   BTU/(hr. sq.ft.degree Rankine)",hL);
```

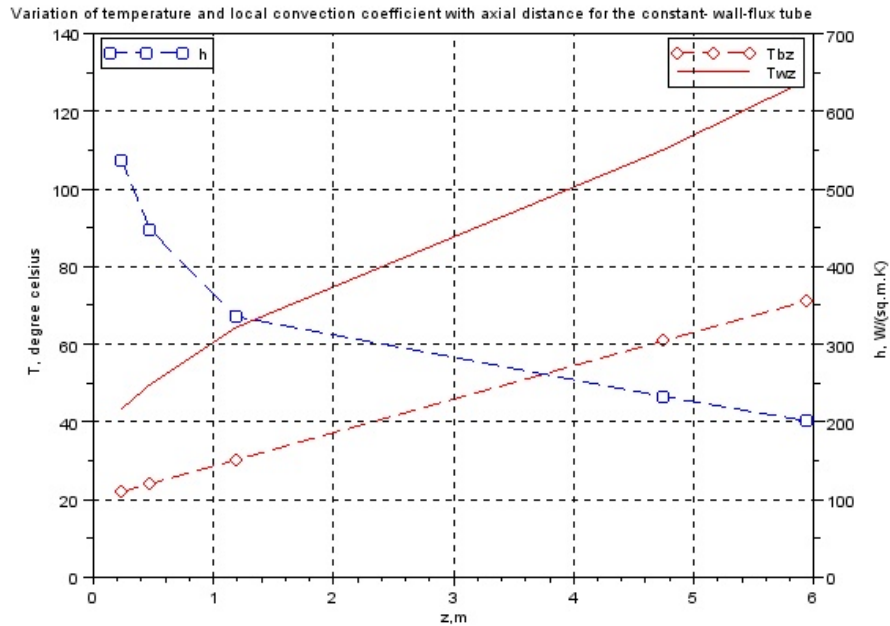


Figure 6.1: Hydrodynamic and Thermal Entry Length

Scilab code Exa 6.3 Hydrodynamic and Thermal Entry Length

```

1  clc;
2  clear;
3  printf("\t\t\tChapter6_example3\n\n\n");
4  // Determination of the variation of wall
   temperature with length up to the point where the
   flow becomes fully developed.

```



```

5 // properties of milk
6 kf=0.6; // thermal conductivity in W/(m-K)
7 cp=3.85*1000; // specific heat in J/(kg*K)
8 rou=1030; // density in kg/m^3
9 mu=2.12e3; // viscosity in N s/m^2
10 // specifications of 1/2 standard type K tubing from
    appendix table F2
11 OD=1.588/100; // outer diameter in m
12 ID=1.340/100; // inner diameter in m
13 A=1.410e-4 // cross sectional area in m^2
14 rou=1030;
15 V=0.1;
16 mu=2.12e-3
17 // determination of flow regime
18 Re=rou*V*ID/(mu);
19 printf("\nThe Reynolds Number is %d",Re);
20 // The flow being laminar, the hydrodynamic entry
    length is calculated as follows
21 ze=0.05*ID*Re;
22 printf("\nThe hydrodynamic entry length is %.1f cm",
    ze*100);
23 Tbo=71.7; // final temperature in degree celsius
24 Tbi=20; // initial temperature in degree celsius
25 L=6; // heating length in m
26 qw=rou*V*ID*cp*(Tbo-Tbi)/(4*L);
27 printf("\nThe heat flux is %d W/sq.m",qw);
28 q=qw*pi*ID*L;
29 printf("\nThe power required is %.1f W",q);
30 printf("\nA 3000 W heater would suffice");
31 Pr=(cp*mu)/kf; // Prandtl Number
32 printf("\nThe Prandtl Number is %.1f",Pr);
33 zf=0.05*ID*Re*Pr;
34 printf("\nThe length required for flow to be
    thermally developed is %.1f m",zf);
35 // calculations of wall temperature of the tube
36 reciprocal_Gz=[0.002 0.004 0.01 0.04 0.05]; // values
    of 1/Gz taken
37 [n m]=size(reciprocal_Gz);

```

```

38 Nu=[12 10 7.5 5.2 4.5]; //Enter the corresponding
    value of Nusselts Number from figure 6.8
39 for i=1:m
40     z(i)=ID*Re*Pr*reciprocal_Gz(i);
41     h(i)=kf*Nu(i)/ID;
42     Tbz(i)=20+(8.617*z(i));
43     Twz(i)=Tbz(i)+(11447/h(i));
44 end
45 printf("\nSummary of Calculations to Find the Wall
    Temperature of the Tube");
46 printf("\n\tl/Gz\t\tNu\t\tz (m)\t\tth W/(sq.m.K)\t\t
    tTbz (degree celsius)\t\tTzw (degree celsius)");
47 for i=1:m
48 printf("\n\t%.3f\t\t%.1f\t\t%.3f\t\t%d\t\t\t%.1f\t\t
    \t\t%.1f",reciprocal_Gz(i),Nu(i),z(i),h(i),Tbz(i)
    ,Tzw(i));
49 end
50 subplot(211);
51 plot(z,Tbz,'r—d',z,Tzw,'r—'); // our first figure
52 a1 = gca();
53 h1=legend(["Tbz";"Tzw"]);
54 subplot(212)
55 plot(z,h,'o—'); // our second figure
56 h1=legend(['h'],2);
57 title('Variation of temperature and local convection
    coefficient with axial distance for the constant
    – wall–flux tube');
58 a2 = gca();
59 a2.axes_visible = ["off", "on", "on"];
60 a2.y_location = "right";
61
62 a1.axes_bounds=[0 0 1 1]; // modify the first
    figure to occupy the whole area
63 a2.axes_bounds=[0 0 1 1]; // modify the second
    figure to occupy the whole area too
64
65 a1.data_bounds=[0,0;6,140];
66 a2.data_bounds=[0,0;6,700];

```

```

67
68 a1.x_ticks = tlist(["ticks", "locations", "labels"],
    (0:6)', ["0";"1";"2";"3";"4";"5";"6"]);
69 a1.x_label
70 a1.y_label
71 x_label=a1.x_label;
72 x_label.text=" z,m"
73 a2.x_label
74 a2.y_label
75 y_label=a1.y_label;
76 y_label.text="T, degree celsius"
77 y_label=a2.y_label;
78 y_label.text="h, W/(sq.m.K)"
79 xgrid(1);
80 a2.filled = "off";

```

Scilab code Exa 6.4 Variation of temperature with length of tube

```

1 clc;
2 clear;
3 printf("\t\t\tChapter6_example4\n\n\n");
4 // The average bulk temperature of the Freon-12 is
   [-40 +(-4)]/2 = -22 F
5 // properties of Freon-12 at average bulk
   temperature
6 kf=0.04; // thermal conductivity in BTU/(hr.ft. R)
7 cp=0.2139; // specific heat in BTU/(lbm- R)
8 rou= 1.489*(62.4); // density in lbm/cu.ft
9 v=0.272e-5; // viscosity in sq.ft/s
10 a=2.04e-3; // diffusivity in sq.ft/hr
11 Pr=4.8; // Prandtl Number

```

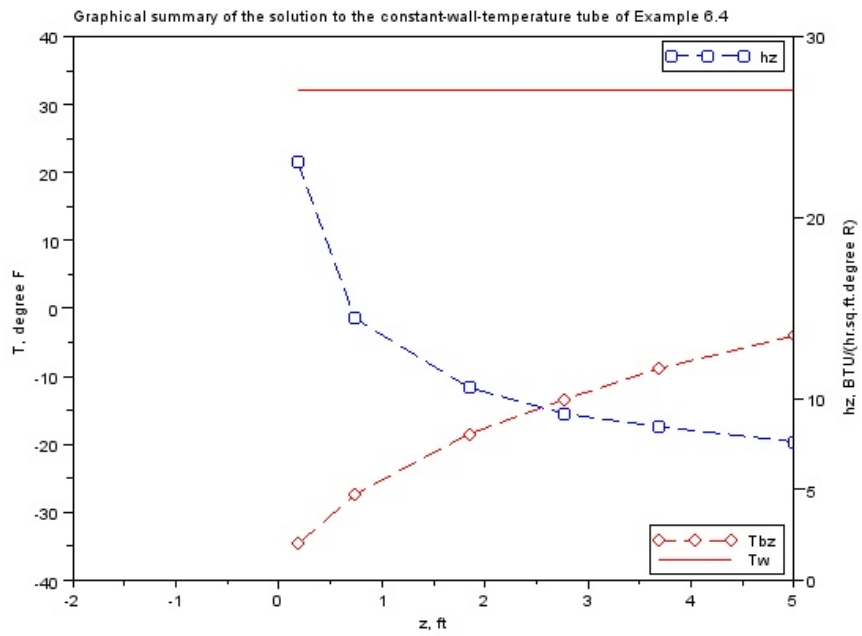


Figure 6.2: Variation of temperature with length of tube

```

12 // specifications of 3/8 standard type K copper
    tubing from appendix table F2
13 OD=0.5/12; // outer diameter in ft
14 ID=0.03350; // inner diameter in ft
15 A=0.0008814 // cross sectional area in sq.ft
16 // Laminar conditions are assumed
17 z=5;
18 Tw=32;
19 Tbo=-4;
20 Tbi=-40;
21 L=5;
22 i=1;
23 V_assumed(i)=100; //assumed value for velocity
24 for i=1:6
25     inv_Gz(i)=(z*a)/(V_assumed(i)*ID^2);
26     Nu=[4.7 5.8 6.2 6.3 6.4 6.4]; // corresponding
        Nusselt numbers from fig. 8.8:
27     hL(i)=Nu(i)*kf/ID;
28     V(i)=-((2*a*L*hL(i))/(kf*ID/2)*log((Tw-Tbo)/(Tw-
        Tbi)));
29     V_assumed(i+1)=V(i);
30 end
31 printf("\nSummary of Results\n");
32 printf("Assumed V (ft/hr)\t1/Gz\tNu (fig 8.8)\thL BTU
    /(hr. sq.ft. degree R)\tV (ft/hr)\n");
33 for j=1:6
34     printf("\t%d\t\t%.4f\t%.1f\t\t%.2f\t\t\t\t\t\t\t\t\t\t",
        V_assumed(j),inv_Gz(j),Nu(j),hL(j),V(j));
35 end
36 V_final=V(i-1);
37 hL_final=hL(i-1);
38 printf("\nThe final velocity is %d ft/hr = %.4f ft/s
    ",V_final,V_final/3600);
39 printf("\nThe final convective coefficient is %.2f
    BTU/(hr. sq.ft. degree R)",hL_final);
40 // checking the laminar-flow assumption by
    calculating the Reynolds number
41 Re=(V_final/3600)*ID/v;

```

```

42 printf("\nThe Reynolds number is %d",Re);
43 // The flow is laminar
44 m_Fr=rou*A*V_final/3600;
45 printf("\nThe mass flow rate of Freon-12 is %.2e lbm
/s = %.2f lbm/hr",m_Fr,m_Fr*3600);
46 As=%pi*ID*L;
47 q=hL_final*As*[(Tw-Tbo)-(Tw-Tbi)]/(log((Tw-Tbo)/(Tw-
Tbi)));
48 printf("\nThe heat gained by Freon-12 is %.1f BTU/hr
",q);
49 q_check=m_Fr*cp*(Tbo-Tbi);
50 printf("\nOn checking the heat transferred we find
almost equal to the heat gained by Freon-12");
51 rou_water=1.002*62.4; // density of water in lbm/ft
^3 from appendix table C11
52 m_water=rou_water*L*(2/12)*(3/12);
53 printf("\nThe mass of water in the prescribed volume
is %.1f lbm",m_water);
54 // to remove 144 BTU/lbm of water, the time required
is caalculated as below
55 t=144*m_water/q;
56 printf("\nThe required time is %.1f hr",t);
57 inv_Gz1=[0.001 0.004 0.01 0.015 0.02 0.0271]; //
guess values of 1/Gz
58 Nu_D=[19.3 12.1 8.9 7.7 7.1 6.4]; //corresponding
Nusselt number from fig. 6.8
59 [n m]=size(inv_Gz1);
60 for j=1:m
61     Z(j)=ID*Re*Pr*(inv_Gz1(j));
62     hz(j)=Nu_D(j)*kf/ID;
63     Tbz(j)=32-72*exp(-0.01812*Z(j)*hz(j));
64 end
65 printf("\nSummary of Data for Example 6.4 ");
66 printf("\n\t1/Gz\tNu_D\tz (ft)\thz, BTU/(hr. sq.ft.
degree R)\tTbz, degree F\n");
67 for p=1:m
68     printf("\t%.4f\t%.1f\t%.2f\t%.2f\t\t\t%.1f\n",
inv_Gz1(p),Nu_D(p),Z(p),hz(p),Tbz(p));

```

```

69 end
70 subplot(211);
71 plot(Z,Tbz, 'r—d',Z,Tw, 'r-'); // your first figure
72 a1 = gca();
73 hl=legend(['Tbz'; 'Tw'],4);
74 subplot(212)
75 plot(Z,hz, 'o—'); // your second figure
76 a2 = gca();
77 hl=legend(['hz'],1);
78 a2.axes_visible = ["off", "on", "on"];
79 a2.y_location = "right";
80
81 a1.axes_bounds=[0 0 1 1]; // modify the first
    figure to occupy the whole area
82 a2.axes_bounds=[0 0 1 1]; // modify the second
    figure to occupy the whole area too
83 a2.filled = "off";
84 a1.data_bounds=[-2,-40;5,40];
85 a2.data_bounds=[-2,0;5,30];
86 x_label1=a1.x_label;
87 x_label1.text="z, ft";
88 y_label2=a2.y_label;
89 y_label2.text="hz, BTU/(hr.sq.ft.degree R)";
90 y_label=a1.y_label;
91 y_label.text="T, degree F";
92 newticks1=a1.y_ticks;
93 newticks1(2)=[-40;-30;-20;-10;0;10;20;30;40];
94 newticks1(3)=[ '-40'; '-30'; '-20'; '-10'; '0'; '10'; '20';
    '30'; '40' ];
95 a1.y_ticks=newticks1;
96 newticks2=a2.y_ticks;
97 newticks2(2)=[0;5;10;20;30];
98 newticks2(3)=[ '0'; '5'; '10'; '20'; '30' ];
99 a2.y_ticks=newticks2;
100 newticks=a1.x_ticks;
101 newticks(2)=[-2;-1;0;1;2;3;4;5];
102 newticks(3)=[ '-2'; '-1'; '0'; '1'; '2'; '3'; '4'; '5' ];
103 a1.x_ticks=newticks;

```

```

104
105 title('Graphical summary of the solution to the
        constant-wall-temperature tube of Example 6.4');

```

Scilab code Exa 6.5 Combined Entry Length Problem

```

1  clc;
2  clear;
3  printf("\t\t\tChapter6_example5\n\n\n");
4  // Determination for the power required for heating
   and the wall temperature at the outlet.
5  // The liquid properties are evaluated at the mean
   temperature of  $(80 + 20)/2 = 50$  C .
6  // specifications of 1 standard type K copper water
   tubing from appendix table F2
7  OD = 2.858/100; // outer diameter in m
8  ID = 2.528/100; // inner diameter in m
9  A = 5.019e-4; // cross sectional area in sq.m
10 // 1 oz =  $2.957e-5$  m3
11 Q=80*2.957e-5/120; // The volume flow rate of water
   (at 20 C) in cu.m/s
12 printf("\nThe volume flow rate of water (at 20 C)
   is %.2e cu.m/s",Q);
13 p_20= 1.000*1000; // density of water at 20 C in kg
   /cu.m
14 // properties of water at 50 C from appendix table
   C11
15 p_50= 0.990*(1000); // density in kg/m3
16 cp= 4181; // specific heat in J/(kg*K)
17 v = 0.586e-6; // viscosity in sq.m/s
18 kf = 0.640; // thermal conductivity in W/(m.K)
19 a = 1.533e-7; // diffusivity in sq.m/s
20 Pr = 3.68; // Prandtl number

```



```

21 mass_flow=p_20*Q; // mass flow rate through the tube
    in kg/s
22 printf("\nmass flow rate through the tube is %.4f kg
    /s",mass_flow);
23 L=3; // length of tube in m
24 As=%pi*ID*L;
25 Tbo=80; // final temperature in C
26 Tbi=20; // initial temperature in C
27 qw=mass_flow*cp*(Tbo-Tbi)/(As);
28 q=qw*As;
29 A=%pi*(ID/2)^2;
30 printf("\nThe power required in %.3e W/sq.m = %d W",
    qw,q);
31 V=mass_flow/(p_50*A); // average velocity at 50 C
32 printf("\nThe average velocity at 50 C is %.2e m/s"
    ,V);
33 Re=(V*ID)/v; // Reynold's Number
34 printf("\nThe Reynolds Number for the flow is %d",Re
    );
35 // The flow is laminar so we can use Figure 6.12 to
    obtain the information needed on Nusselt number
    and to find hz
36 inv_Gz=L/(Re*ID*Pr); // The inverse Graetz number at
    tube end, based on 50 C conditions
37 printf("\nThe inverse Graetz number at tube end,
    based on 50 C conditions is %.4f",inv_Gz);
38 Nu=6.9; //value of corresponding Nusselts Number
    from figure 6.12
39 hz=(Nu*kf)/ID;
40 printf("\nThe local convection coefficient is %.1f W
    /(sq.m.K)",hz);
41 Two=(qw/hz)+Tbo; // The outlet wall temperature in
    C
42 printf("\nThe outlet wall temperature is %d C",Two
    );

```

Scilab code Exa 6.6 Heat transfer to and from turbulent flow

```
1  clc;
2  clear;
3  printf("\t\t\tChapter6_example6\n\n\n");
4  // determibation of heat gained
5  // air properties to be calculated at T=(72+45)
   //      /2=58.5 degree Fahrenheit
6  // properties at T=58.5 degree fahrenheit from
   //      appendix table D1
7  p = 0.077; // density in lbm/ft^3
8  cp = 0.240; // specific heat in BTU/(lbm.degree
   //      Rankine)
9  v = 15.28e-5; // viscosity in ft^2/s
10 kf = 0.0146; // thermal conductivity in BTU/(hr.ft."
   //      R)
11 a = 0.776; // diffusivity in ft^2/hr
12 Pr = 0.711; // prandtl number
13 D=7/12; // diameter in ft
14 L=40; // length in ft
15 Tbo=72; // outlet temperature in degree Fahrenheit
16 Tbi=45; // inlet temperature in degree Fahrenheit
17 A=%pi*(D^2)/4; // cross sectional area of duct in ft
   //      ^2
18 // density at outlet temperature in lbm/ft^3
19 rou_o=.0748;
20 V=10; // average velocity in ft/s
21 mass_flow=rou_o*A*V;
22 printf("\nThe mass flow rate is %.1f lbm/s",
   //      mass_flow);
23 // average velocity evaluated by using the average
   //      bulk temperature
```

```

24 V_avg=mass_flow/(p*A);
25 printf("\nThe average velocity evaluated by using
    the average bulk temperature is %.2f ft/s",V_avg)
    ;
26 Re=(V_avg*D)/v;
27 printf("\nThe Reynolds number for the flow is %.3e "
    ,Re);
28 // the flow is in turbulent regime
29 q=mass_flow*cp*(Tbo-Tbi);
30 printf("\nThe heat gained by air is %.3f BTU",q);
31 hc=1; // convection coefficient between the outside
    duct wall and the attic air in BTU/(hr. sq.ft.
    degree Rankine).
32 T_inf=105; // The temperature of attic air
    surrounding the duct in degree Fahrenheit
33 hz=(0.023*Re^(4/5)*Pr^0.4)*kf/D; // The local
    coefficient at the duct end is %.2f BTU/(hr. sq.
    ft.degree Rankine)
34 printf("\nThe local coefficient at the duct end is %
    .2f BTU/(hr. sq.ft.degree Rankine)",hz);
35 qw=(T_inf-Tbo)/((1/hc)+(1/hz)); // wall flux in BTU
    /(hr. sq.ft.degree Rankine)
36 printf("\nThe wall flux is %.1f BTU/(hr. sq.ft.
    degree Rankine)",qw);
37 Two=qw*(1/hz)+Tbo; // The wall temperature at exit
    in degree Fahrenheit
38 printf("\nThe wall temperature at exit is %.1f
    degree Fahrenheit",Two);

```

Chapter 7

Convection Heat Transfer in Flows Past Immersed Bodies

Scilab code Exa 7.1 Laminar Boundary Layer flow over flat plate

```
1 clc;
2 clear;
3 printf("\t\t\tChapter7_example1\n\n\n");
4 printf("\t\t\tSolution to part (a)\n");
5 // determination of boundary layer growth with
   length
6 // properties of air at 27 degree celsius from
   appendix table D.1
7 rou=1.177; // density in kg/cu.m
8 v=15.68e-6; // viscosity in sq.m/s
9 L=0.5; // length in m
10 V_inf=1; // air velocity in m/s
11 Re= (V_inf*L)/v; // Reynolds Number
12 printf("The Reynolds Number is %.2e ",Re);
13 // Reynolds Number is less than 5e5 hence the flow
```

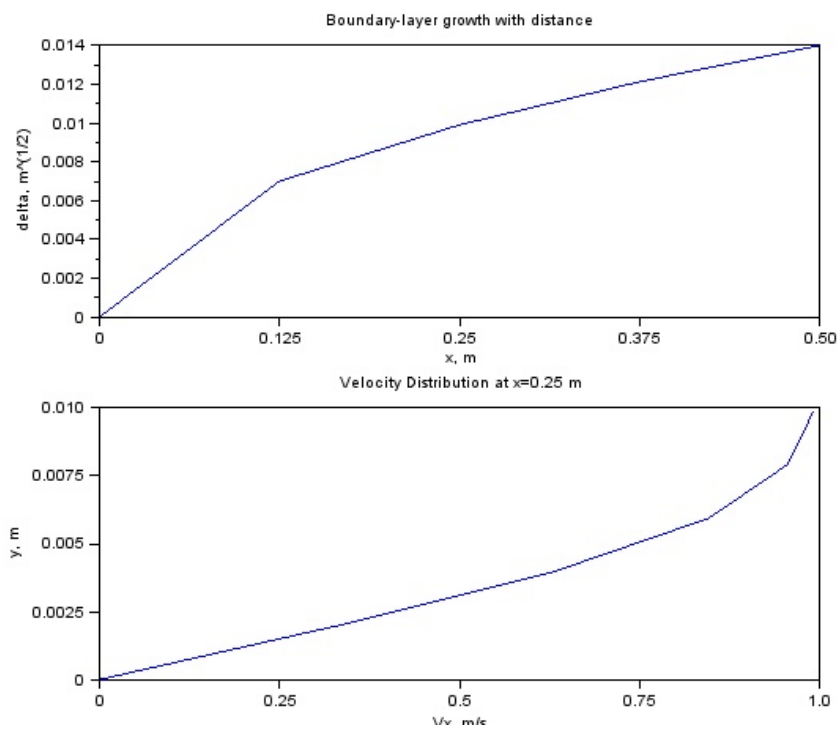


Figure 7.1: Laminar Boundary Layer flow over flat plate


```

46 newTicks1(2)=[0;0.25;0.5;0.75;1.0];
47 newTicks1(3)=[ '0 ' ; '0.25 ' ; '0.5 ' ; '0.75 ' ; '1.0 ' ];
48 b.x_ticks=newTicks1;
49 newTicks2=b.y_ticks;
50 newTicks2(2)=[0;0.0025;0.005;0.0075;0.010];
51 newTicks2(3)=[ '0 ' ; '0.0025 ' ; '0.005 ' ; '0.0075 ' ; '0.010 '
    ];
52 b.y_ticks=newTicks2;
53 title('Velocity Distribution at x=0.25 m');
54 xlabel('Vx, m/s');
55 ylabel('y, m');
56 printf("\t\t\tSolution to part (c)\n");
57 // calculation of absolute viscosity
58 gc=1;
59 mu=rou*v/gc;
60 printf("\n\nThe absolute viscosity is %.3e N.s/sq.m",
    mu);
61 b=1; // width in m
62 Df=0.664*V_inf*mu*b*(Re)^0.5;
63 printf("\n\nThe skin-drag is %.2e N",Df);
64 printf("\n\nThe skin-drag including both sides of
    plate is %.2e N",2*Df);

```

Scilab code Exa 7.2 Flow over plate at constant temperature

```

1 clc;
2 clear;
3 printf("\t\t\tChapter7_example2\n\n\n");
4 // determination of temperature profile
5 // properties of water at (40 + 100)/2 = 70 F = 68
    F from appendix table C11

```

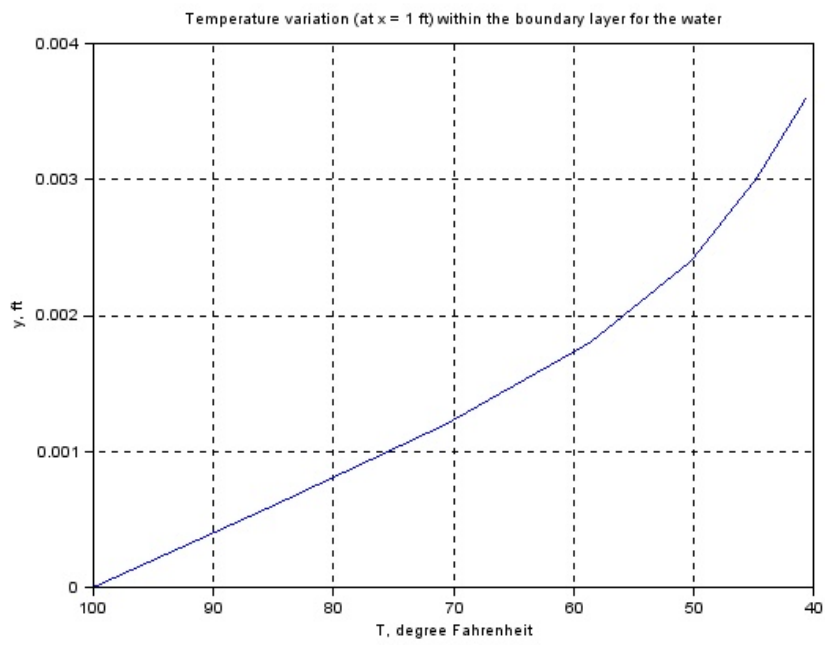


Figure 7.2: Flow over plate at constant temperature


```

6 rou= 62.4; // density in lbm/ft^3
7 cp=0.9988; // specific heat BTU/(lbm-degree Rankine)
8 v= 1.083e-5; // viscosity in sq.ft/s
9 kf = 0.345 ; // thermal conductivity in BTU/(hr.ft.
    degree Rankine)
10 a = 5.54e-3; // diffusivity in sq.ft/hr
11 Pr = 7.02; // Prandtl Number
12 V=1.2; // velocity in ft/s
13 x=[1 2]; // distances from plate entry in ft
14 for i=1:2
15 Re(i)=(V*x(i))/v; // Reynolds Number at x=1 ft
16 printf("\nThe Reynolds Number at x=%d ft is %.3e",i,
    Re(i));
17 // since Reynolds Number is less than 5*10^5, the
    flow is laminar
18 hL(i)=0.664*Pr^(1/3)*Re(i)^0.5*kf/x(i);
19 printf("\nThe average convection coefficient at x=%d
    is %.1f BTU/(hr. sq.ft. degree Rankine)",i,hL(i)
    );
20 Tw=100; // temperature of metal plate in degree
    fahrenheit
21 T_inf=40; // temperature of water in degree
    fahrenheit
22 A(i)=x(i)*18/12; // cross sectional area for 1 ft
    length
23 q(i)=hL(i)*A(i)*(Tw-T_inf);
24 printf("\nThe heat transferred to water over the x=
    %d ft is %.3e BTU/hr",i,q(i));
25 end
26 eta=0:0.2:1.2;
27 [n m]=size(eta);
28 theta=[1 .75 .51 .31 .17 .08 0.01]; // values of
    dimensionless temperature from figure 7.7
    corresponding to eta value taken
29 for i=1:m
30 y(i)=eta(i)*(v*x(1)/V(1))^0.5;
31 T(i)=theta(i)*(Tw-T_inf)+T_inf;
32 end

```

```

33 printf("\\nSolution Chart for example 7.2\\n");
34 printf("\\teta\\t\\ttheta\\t\\ty, ft\\t\\t\\tT, degree F\\n")
    ;
35 for i=1:m
36 printf("\\t%.1 f\\t\\t%.2 f\\t\\t%.1 e\\t\\t\\t%.1 f\\n",eta(i),
    theta(i),y(i),T(i));
37 end
38 plot(T,y);
39 a=gca();
40 newTicks=a.x_ticks;
41 newTicks(2)=[100; 90; 80; 70; 60;50; 40];
42 newTicks(3)=[ '100 ' ; ' 90 ' ; ' 80 ' ; ' 70 ' ; ' 60 ' ; ' 50 ' ; ' 40
    ' ] ;
43 a.x_ticks=newTicks;
44 newTicks1=a.y_ticks;
45 newTicks1(2)=[0; 0.001; 0.002; 0.003; 0.004];
46 newTicks1(3)=[ '0 ' ; ' 0.001 ' ; ' 0.002 ' ; ' 0.003 ' ; ' 0.004
    ' ] ;
47 a.y_ticks=newTicks1;
48 a.axes_reverse=["on", "off"];
49 xgrid(1);
50 title('Temperature variation (at x = 1 ft) within
    the boundary layer for the water');
51 xlabel('T, degree Fahrenheit');
52 ylabel('y, ft');

```

Scilab code Exa 7.3 Flow over Isothermal flat plate

```

1 clc;
2 clear;
3 printf("\\t\\t\\tChapter7_example3\\n\\n\\n");
4 // Determination of the average convection
    coefficient and the total drag force exerted on

```

```

the plate.
5 // properties of air at (300 + 400)/2 = 350 K from
  appendix table D1
6 rou= 0.998; // density in kg/cu.m
7 cp= 1009; // specific heat in J/(kg*K)
8 v= 20.76e-6; // viscosity in sq.m/s
9 Pr = 0.697; // Prandtl Number
10 k= 0.03003; // thermal conductivity in W/(m.K)
11 a = 0.2983e-4; // diffusivity in sq.m/s
12 L=1; // Length of plate in m
13 V=5; // velocity of air in m/s
14 b=0.5; // width in m
15 Re=V*L/v; // Reynolds number at plate end
16 printf("\nThe Reynolds number is %.2e",Re);
17 // Since the flow is laminar at plate end, The
  average convection coefficient is calculated with
  Equation  $Nu=h*L/k= 0.664 Re^{(1/2)}Pr^{(1/3)}$ 
18 h=k*0.664*Re^(1/2)*Pr^(1/3)/L; // The average
  convection coefficient in W/(sq.m.K)
19 printf("\nThe average convection coefficient is %.2f
  W/(sq.m.K)",h);
20 Df=0.664*V*rou*v*b*(Re)^0.5; // drag force in N
21 printf("\nThe drag force is %.2e N",Df);
22 hx=(1/2)*h; // local convective coefficient
23 printf("\nThe local convective coefficient is %.2f
  W/(sq.m.K)",hx);
24 delta=5*L/(Re)^0.5; // The boundary-layer thickness
  at plate end
25 printf("\nThe boundary-layer thickness at plate end
  is %.2f cm",delta*100);
26 delta_t=delta/(Pr)^(1/3);
27 printf("\nThe thermal-boundary-layer thickness is %
  .2f cm",delta_t*100);

```

Scilab code Exa 7.4 Maximum heater surface temperature

```
1  clc;
2  clear;
3  printf("\t\t\tChapter7_example4\n\n\n");
4  // Determination of the maximum heater-surface
   temperature for given conditions
5  // fluid properties at (300 degree R + 800 degree R)
   /2 = 550 degree R=540degree R from Appendix Table
   D.6
6  rou= 0.0812; // density in lbm/ft^3
7  cp=0.2918; // specific heat BTU/(lbm-degree Rankine)
8  v= 17.07e-5; // viscosity in ft^2/s
9  kf = 0.01546 ; // thermal conductivity in BTU/(hr.ft
   .degree Rankine)
10 a = 0.8862; // diffusivity in ft^2/hr
11 Pr = 0.709; // Prandtl Number
12 qw=10/(1.5*10.125)*(1/.2918)*144; // The wall flux
13 printf("\nThe wall flux is %d BTU/(hr. sq.ft)",qw);
14 V_inf=20; // velocity in ft/s
15 L=1.5/12; // length in ft
16 Re_L=V_inf*10*L/v; // Reynolds number at plate end
17 printf("\nThe Reynolds number at plate end is %.2e",
   Re_L);
18 // So the flow is laminar and we can find the wall
   temperature at plate end as follows
19 T_inf=300; // free stream temperature in degree
   Rankine
20 Tw=T_inf+(qw*L*10/(kf*0.453*(Re_L)^0.5*(Pr)^(1/3)));
21 printf("\nThe maximum heater surface temperature is
   %d degree Rankine",Tw);
```

Scilab code Exa 7.5 Reynolds Colburn Analogy

```

1  clc;
2  clear;
3  printf("\t\t\tChapter7_example5\n\n\n");
4  // validation of the equation  $st.(Pr)^{(2/3)}=Cd/2$ 
   where St: Stanton Number Pr:Prandtl Number Cd:
   Drag Coefficient
5  // values of parameters from example 7.4
6  rou= 0.0812; // density in lbm/ft^3
7  cp=0.2918; // specific heat BTU/(lbm-degree Rankine)
8  v= 17.07e-5; // viscosity in ft^2/s
9  kf = 0.01546 ; // thermal conductivity in BTU/(hr.ft
   .degree Rankine)
10 a = 0.8862; // diffusivity in ft^2/hr
11 Pr = 0.709; // Prandtl Number
12 Tw=469; // maximum heater temperature in degree
   Rankine
13 T_inf=300; // free-stream temperature in degree
   Rankine
14 qw=324; // The wall flux in BTU/(hr.ft^2)
15 V_inf=20; // velocity in ft/s
16 hx=qw/(Tw-T_inf); // The convection coefficient
17 printf("\nThe convection coefficient is %.2f BTU/(hr
   .sq.ft.degree R)",hx);
18 LHS=(hx/3600)*(Pr)^(2/3)/(rou*cp*V_inf);
19 printf("\nThe value of left hand side of the
   equation is %.2e",LHS);
20 Re_L=1.46e+005; // Reynolds number at plate end
21 RHS=0.332*(Re_L)^(-0.5);
22 printf("\nThe value of left hand side of the
   equation is %.2e",RHS);
23 err=(LHS-RHS)*100/LHS;
24 printf("\nThe error is %d percent",err);
25 printf("\nSince the error is only %d percent , the
   agreement is quite good",err);

```

Scilab code Exa 7.6 Drag due to skin friction

```
1  clc;
2  clear;
3  printf("\t\t\tChapter7_example6\n\n");
4  // Estimation of the drag due to skin friction
5  // properties of water at 68 F from Appendix Table
   C.11
6  rou= 62.4; // density in lbm/cu.ft
7  v= 1.083e-5; // viscosity in sq.ft/s
8  V_inf=5*.5144/.3048; // barge velocity in ft/s using
   conversion factors from appendix table A1
9  printf("\nThe barge velocity is %.2f ft/s",V_inf);
10 L=20; // Length of barge in ft
11 Re_L=V_inf*L/v; // Reynolds number at plate end
12 printf("\nThe Reynolds number at plate end is %.2e",
   Re_L);
13 Cd=0.003; //value of Cd corresponding to the
   Reynolds number from figure 7.11
14 gc=32.2;
15 b=12; // width in ft
16 Df=(Cd*rou*V_inf^2*b*L)/(2*gc);
17 printf("\nThe drag force is %d lbf",Df);
```

Scilab code Exa 7.7 Wattage requirement of heater

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

```

3 printf("\t\t\tChapter7_example7\n\n");
4 // Determination of wattage requirement
5 // properties of carbon dioxide at a film
   temperature of (400+600)/2 = 500 K from appendix
   table D2
6 rou= 1.0732; // density in kg/m^3
7 cp= 1013; // specific heat in J/(kg*K)
8 v= 21.67e-6; // viscosity in m^2/s
9 Pr = 0.702; // Prandtl Number
10 k= 0.03352; // thermal conductivity in W/(m.K)
11 a = 0.3084e-4; // diffusivity in m^2/s
12 V_inf=60; // carbon dioxide velocity in m/s
13 x_cr=(5e5)*v/V_inf; // The transition length in m
14 printf("\nThe transition length is %.1f cm",x_cr
   *100);
15 w=4; // width of each heater in cm
16 b=.16; // effective heating length in m
17 Tw=600; // temperature of heater surface in K
18 T_inf=400; // temperature of carbon dioxide in K
19 r=pmodulo(x_cr*100,w);
20 n=(x_cr*100+r)/w; // number of heater where
   transition occurs
21 printf("\nThe transition thus occur at %dth heater",
   n);
22 m=6; // number of heater strips
23 q=zeros(m+1,1);
24 x=[0.04 0.08 0.12 0.16 0.20 0.24];
25 for i=1:n-1 // transition occurs at 5th heater, so
   laminar zone equation is followed till then
26   h(i)=(0.664*k)*(V_inf/v)^0.5*(Pr)^(1/3)/x(i)
   ^0.5;
27   printf("\n\nThe convective coefficient for
   heater no. %d is %d W/(sq.m.K)",i,h(i));
28   q(i+1)=h(i)*x(i)*b*(Tw-T_inf);
29   dq(i)=q(i+1)-q(i);
30   printf("\nThe heat transferred by heater no. %d
   is %d W",i,dq(i));
31 end

```

```

32 // Turbulent zone exists from 5th heater onwards so
    the following equation is followed  $Nu=h*x/kf$ 
     $=[0.0359*(Re_L)^{(4/5)} - 830]*(Pr)^{(1/3)}$ 
33 for i=5:6
34     Re_L(i)=V_inf*x(i)/v;
35     h(i)=(k/x(i))*[0.0359*(Re_L(i))^(4/5)-830]*(Pr)
        ^(1/3)
36     printf("\n\nThe Reynolds number for heater no.
        %d is %.2e",i,Re_L(i));
37     printf("\nThe convective coefficient for heater
        no. %d is %.1f W/(sq.m.K)",i,h(i));
38     q(i+1)=h(i)*x(i)*b*(Tw-T_inf);
39     dq(i)=q(i+1)-q(i);
40     printf("\nThe heat transferred by heater no. %d
        is %d W",i,dq(i));
41 end

```

Scilab code Exa 7.8 Flow past a telephone pole

```

1  clc;
2  clear;
3  printf("\t\t\tChapter7_example8\n\n\n");
4  // Estimation of force exerted on the pole
5  // properties of air at given conditions from
    appendix table D1
6  rou= 0.0735; // density in lbm/ft^3
7  v= 16.88e-5; // viscosity in ft^2/s
8  V=20*5280/3600; // flow velocity in ft/s
9  printf("\nThe flow velocity is %.1f ft/s",V);
10 D=12/12; // diameter of pole in ft
11 L=30; // length of pole in ft
12 gc=32.2;
13 Re_D=V*D/v; // Reynolds Number for flow past the

```



```

    pole
14 printf("\nThe Reynolds Number for flow past the pole
    is %.2e ",Re_D);
15 Cd_cylinder=1.1; // value of Cd for smooth cylinder
    from figure 7.22
16 A_cylinder=D*L; // frontal area of pole
17 printf("\nThe frontal area of pole is %d sq.ft",
    A_cylinder);
18 Df_cylinder=Cd_cylinder*(1/2)*rou*V^2*A_cylinder/gc;
19 printf("\nThe Drag force exerted on only the pole is
    %.1f lbf",Df_cylinder);
20 D_square=2/12; // length of square part of pole
21 L_square=4;
22 Re_square=V*D_square/v; // Reynolds Number for
    square part of pole
23 printf("\nThe Reynolds Number for square part of
    pole is %.1e",Re_square);
24 Cd_square=2; // Corresponding value of Cd for square
    part from figure 7.23
25 A_square=D_square*L_square; // projected frontal
    area of square part
26 printf("\nThe frontal area of square part of pole is
    %.3f sq.ft",A_square);
27 Df_square=Cd_square*(1/2)*rou*V^2*A_square/gc;
28 printf("\nThe Drag force exerted on cross piece of
    the pole is %.2f lbf",Df_square);
29 Df_total=Df_cylinder+Df_square;
30 printf("\nThe total drag force on the pole is %.1f
    lbf",Df_total);

```

Scilab code Exa 7.9 Current through Hot Wire Anemometer

```

1 clc;

```

```

2 clear;
3 printf("\t\t\tChapter7_example9\n\n");
4 // determination of required current
5 // properties of air at film temperature (300 + 500)
   /2 = 400 K from appendix table D1
6 rou= 0.883; // density in kg/cu.m
7 cp= 1014; // specific heat in J/(kg*K)
8 v= 25.90e-6; // viscosity in sq.m/s
9 Pr = 0.689; // Prandtl Number
10 kf= 0.03365; // thermal conductivity in W/(m.K)
11 a = 0.376e-4; // diffusivity in sq.m/s
12 V_inf=1; // velocity in m/s
13 D=0.00013; // diameter in m
14 L=1/100; // length of wire in cm
15 Re_D=V_inf*D/v; // The Reynolds number of flow past
   the wire
16 printf("\nThe Reynolds number of flow past the wire
   is %.3f",Re_D);
17 C=0.911; //value of C for cylinder from table 7.4
18 m=0.385; //value of m for cylinder from table 7.4
19 hc=kf*C*(Re_D)^m*(Pr)^(1/3)/D; // the convection
   coefficient in W/(m^2.K)
20 printf("\nThe convection coefficient is %d W/(sq.m.K
   )",hc);
21 Tw=500; // air stream temperature in K
22 T_inf=300; // wire surface temperature in K
23 As=%pi*D*L; // cross sectional area in sq.m
24 qw=hc*As*(Tw-T_inf); // The heat transferred to the
   air from the wire
25 printf("\nThe heat transferred to the air from the
   wire is %.3f W",qw);
26 resistivity=17e-6; // resistivity in ohm cm
27 Resistance=resistivity*(L/(%pi*D^2)); // resistance
   in ohm
28 printf("\nThe resistance is %.3f ohm",Resistance
   /100);
29 i=(qw*100/Resistance)^0.5; // current in ampere
30 printf("\nThe current is %.1f A",i);

```

Scilab code Exa 7.10 Pressure Drop for flow over tubes

```
1  clc;
2  clear;
3  printf("\t\t\tChapter7_example10\n\n\n");
4  // Calculation of the pressure drop for the air
   passing over the tubes and the heat transferred
   to the air.
5  // properties of air at 70 + 460 = 530 degree R =
   540 degree R from appendix table D1
6  rou= 0.0735; // density in lbm/cu.ft
7  cp=0.240; // specific heat BTU/(lbm-degree Rankine)
8  v= 16.88e-5; // viscosity in sq.ft/s
9  kf = 0.01516 ; // thermal conductivity in BTU/(hr.ft
   .degree Rankine)
10 a = 0.859; // diffusivity in sq.ft/hr
11 Pr = 0.708; // Prandtl Number
12 // specifications of 3/4 standard type K copper
   tubing from appendix table F2
13 OD=0.875/12; // outer diameter in ft
14 ID=0.06208; // inner diameter in ft
15 A=0.003027; // cross sectional area in sq.ft
16 L=2;
17 sL=1.5/12;
18 sT=1.3/12;
19 V_inf=12; // velocity of air in ft/s
20 V1=(sT*V_inf)/(sT-OD); // velocity at area A1 in ft/
   s
21 printf("\nVelocity at area A1 is %.1f ft/s",V1);
22 sD=((sL)^2+(sT/2)^2)^0.5; // diagonal pitch in inch
23 printf("\nThe diagonal pitch is %.2f in",sD*12);
24 V2=(sT*V_inf)/(2*(sD-OD));
```

```

25 printf("\nVelocity at area A2 is %.1f ft/s",V2);
26 if V1>V2 then
27     Vmax=V1;
28     else Vmax=V2;
29 end
30 Re_D=Vmax*OD/v; // Reynolds Number
31 printf("\nThe Reynolds number is %.2e ",Re_D);
32 sT_OD=1.3/0.875;
33 sT_sL=1.3/1.5;
34 printf("\nThe values of parameters are sT/Do=%.2f
        and sT/sL=%.2f",sT_OD,sT_sL);
35 f1=0.35; //value of f1 for above values of sT/Do and
        Re
36 f2=1.05; //Corresponding value of f2 for above
        values of sT/sL and Re
37 gc=32.2;
38 N=7;
39 dP=N*f1*f2*(rou*Vmax^2/(2*gc));
40 printf("\nThe pressure drop is %.2f lbf/ft^2 = %.4f
        psi",dP, dP/147);
41 sL_Do=sL/OD;
42 C1=0.438; //value of C1 for above values of sT/Do
        and sL/Do
43 C2=0.97; //value of C2 for above values of sT/Do and
        sL/Do
44 m=0.565; //value of m for above values of sT/Do and
        sL/Do
45 hc=kf*1.13*C1*C2*(Re_D)^m*(Pr)^(1/3)/OD; // The
        convection coefficient
46 printf("\nThe convection coefficient is %.1f BTU/(hr
        .sq.ft.degree Rankine)",hc);
47 As=70*%pi*OD*L; // outside surface area of 70 tubes
48 printf("\nThe outside surface area of 70 tubes is %
        .1f sq.ft",As);
49 Tw=200; // outside surface temeprature in degree F
50 T_inf=70; // air temperature in degree F
51 q=hc*As*(Tw-T_inf);// heat transferred
52 printf("\nThe heat transferred is %.2e BTU/hr",q);

```


Chapter 8

Natural Convection Systems

Scilab code Exa 8.1 Natural Convection on a vertical surface

```
1  clc;
2  clear;
3  printf("\t\t\tChapter8_example1\n\n\n");
4  // Determination of the heat transferred to the wall
5  // air properties at (400+120)/2 =260 degree F = 720
   // degree R from Appendix Table D1
6  rou= 0.0551; // density in lbm/cu.ft
7  cp=0.2420; // specific heat BTU/(lbm-degree Rankine)
8  v= 27.88e-5; // viscosity in sq.ft/s
9  kf = 0.01944 ; // thermal conductivity in BTU/(hr.ft
   // .degree Rankine)
10 a = 1.457; // diffusivity in sq.ft/hr
11 Pr = 0.689; // Prandtl Number
12 T_inf=120+460; // wall temperature in degree R
13 Tw=400+460; // inside wall temperature in degree R
14 Beta=1/T_inf;
15 printf("\nThe volumetric thermal expansion
   // coefficient is %.5f/degree R",Beta);
```

```

16 gc=32.2;
17 L=1; // length of wall in ft
18 W=2; // width in ft
19 Gr=(gc*Beta*(Tw-T_inf)*L^3)/v^2; // Grashof Number
20 printf("\nThe Grashof number is %.2e",Gr);
21 temperature_slope=0.505; //value of temperature
    slope from table 8.1 corresponding to Pr=.72
22 hL=(kf/L)*(4/3)*(Gr/4)^(1/4)*temperature_slope; //
    The convection coefficient in BTU/(hr.ft^2.degree
    R)
23 printf("\nThe convection coefficient is %.2f BTU/(hr
    .sq.ft.degree R)",hL);
24 A=L*W; // cross sectional area in sq.ft
25 qw=hL*A*(Tw-T_inf);
26 printf("\nThe heat transferred is %d BTU/hr",qw);

```

Scilab code Exa 8.2 1D heat flow through a window glass

```

1 clc;
2 clear;
3 printf("\t\t\tChapter8_example2\n\n\n");
4 // Determination of heat lost through the glass per
    unit area
5 // properties of air at 22 + 273 = 295 K = 300 K(
    approx) and 273 K from appendix table D1
6 rou= [1.177 1.295]; // density in kg/cu.m
7 cp= [1005 1005.5]; // specific heat in J/(kg*K)
8 v= [15.68e-6 12.59e-6]; // viscosity in sq.m/s
9 Pr = [0.708 0.713]; // Prandtl Number
10 kf= [0.02624 0.02426]; // thermal conductivity in W
    /(m.K)
11 a = [0.22160e-4 0.17661e-4]; // diffusivity in sq.m/
    s

```

```

12 T_inf=[22 0]// inside and outside temperature in K
13 Beta=[1/(T_inf(1)+273) 1/(T_inf(2)+273)]; //
    volumetric thermal expansion coefficient at 295 K
    and 273 K
14 printf("\nThe volumetric thermal expansion
    coefficients at 295 K and 273 K are respectively
    %.5f and %.5f",Beta(1),Beta(2));
15 g=9.81;
16 t=0.005; // thickness of glass
17 L=0.60; // window length in m
18 k=0.81; // thermal conductivity of glass from
    appendix table B3
19 // for first guess
20 Tw=[18 4];
21 printf("\nFor first guess, the results are:\n");
22 for i=1:2
23     Ra(i)=(g*Beta(i)*(Tw(i)-T_inf(i))*L^3)/(v(i)*a(i)
        ));
24     hL(i)=(kf(i)/L)*(0.68+((0.67*(abs(Ra(i)))^(1/4))
        /(1+(0.492/Pr(i))^(9/16)))^(4/9)));
25 end
26 printf("\nThe Rayleigh Numbers are %.3e and %.3e",-
    Ra(1),Ra(2));
27 printf("\nThe convective coefficients are %.2f W/(sq
    .m.K) and %.2f W/(sq.m.K)",hL(1),hL(2));
28 q=(T_inf(1)-T_inf(2))/((1/hL(2))+(t/k)+(1/hL(1)));
29 printf("\nThe heat flux is %.1f W/sq.m",q);
30 for i=1:2
31     Tw_final(i)=T_inf(i)-q*(1/hL(i));
32     printf("\nThe wall temperature calculated is %.1
        f",abs(Tw_final(i)));
33     Tw(i)=abs(Tw_final(i)); // second guess
34 end
35 printf("\nFor second guess, the results are:\n");
36 for i=1:2
37     Ra(i)=(g*Beta(i)*(Tw(i)-T_inf(i))*L^3)/(v(i)*a(i)
        ));
38     hL(i)=(kf(i)/L)*(0.68+((0.67*(abs(Ra(i)))^(1/4))

```



```

        /((1+(0.492/Pr(i))^(9/16))^(4/9)));
39 end
40 printf("\nThe Rayleigh Numbers are %.3e and %.3e",-
        Ra(1),Ra(2));
41 printf("\nThe convective coefficients are %.2f W/(sq
        .m.K) and %.2f W/(sq.m.K)",hL(1),hL(2));
42 q=(T_inf(1)-T_inf(2))/((1/hL(2))+(t/k)+(1/hL(1)));
43 printf("\nThe heat flux is %.1f W/sq.m",q);
44 for i=1:2
45     Tw_final(i)=T_inf(i)-q*(1/hL(i));
46     printf("\nThe wall temperature calculated is %.1
        f degree celsius",abs(Tw_final(i)));
47 end

```

Scilab code Exa 8.3 The cooling unit of a refrigerator

```

1  clc;
2  clear;
3  printf("\t\t\tChapter8_example3\n\n\n");
4  // determination of heat loss through the side.
5  rou= 0.0735; // density in lbm/cu.ft
6  cp=0.240; // specific heat BTU/(lbm-degree Rankine)
7  v= 16.88e-5; // viscosity in sq.ft/s
8  kf = 0.01516 ; // thermal conductivity in BTU/(hr.ft
        .degree Rankine)
9  a = 0.859; // diffusivity in sq.ft/hr
10 Pr = 0.708; // Prandtl Number
11 Tw=90;
12 T_inf=70;
13 g=32.2;
14 L=5.5; // length in ft
15 W=2+(4/12); // width in ft
16 Beta=1/(Tw+460); // volumetric thermal expansion

```

```

    coefficient in per degree Rankine
17 printf("\nThe volumetric thermal expansion
    coefficient is %.5f /degree R",Beta);
18 Ra=(g*Beta*(Tw-T_inf)*L^3)/(v*a/3600);
19 printf("\nThe Rayleigh Number is %.2e ",Ra);
20 hc=(kf/L)*(0.825+((0.387*(Ra)^(1/6))/(1+(0.492/Pr)
    ^(9/16))^(8/27)))^2;
21 printf("\nThe value of convection coefficient is %.3
    f BTU/(hr.sq.ft.degree R)",hc);
22 q=hc*L*W*(Tw-T_inf);
23 printf("\nThe heat gained is %d BTU/hr",q);

```

Scilab code Exa 8.4 Natural convection on an inclined flat plate

```

1 clc;
2 clear;
3 printf("\t\t\tChapter8_example4\n\n\n");
4 // Determination of the variation of average
    convection coefficient with distance
5 // properties of air at (65 + 20)/2 = 42.5 degree C
    =315 K. from appendix table D1
6 rou= 1123; // density in kg/m^3
7 cp= 1006.7; // specific heat in J/(kg*K)
8 v= 17.204e-6; // viscosity in m^2/s
9 Pr =0.703; // Prandtl Number
10 kf= 0.02738; // thermal conductivity in W/(m.K)
11 a = 0.2446e-4; // diffusivity in m^2/s
12 g=9.81;
13 L=5;
14 theta=45;
15 T_inf=20; // ambient air temperature in degree C
16 Tw=65; // roof surface temperature in degree C
17 Beta=1/(T_inf+273); // volumetric thermal expansion

```

```

    coefficient in per K
18 printf("\nThe volumetric thermal expansion
    coefficient is %.5f /K",Beta);
19 // determination of Laminar-turbulent transition
    length by Vliet equation  $Ra=3 \times 10^5 \exp(0.1368 \cos(90-\theta))$ 
    (90-theta))
20 x=((3e5*exp(0.1368*cos(90-theta))*v*a)/(g*cos(theta)
    *Beta*(Tw-T_inf)))^(1/3);
21 printf("\nThe Laminar-turbulent transition length
    by Vliet equation is %.3f m",x);
22 i=1;
23 N=1;
24 n=0;
25 X=[0.02 0.04 0.05 0.051 0.1 1.0 3.0 5.0]; //
    entering values for length(m)
26 [n m]=size(X);
27 for i=1:m
28     if X(i)<=x then
29         // Laminar Flow regime exists
30         Ra(i)=(g*cos(%pi*45/180)*Beta*(Tw-T_inf)*X(i)
            )^3)/(v*a);
31         hc(i)=(kf/X(i))*(0.68+(0.670*Ra(i)^(1/4))
            /(1+(0.492/Pr)^(9/16))^(4/9));
32     else
33         // Turbulent Flow regime exists
34         Ra(i)=(g*Beta*(Tw-T_inf)*X(i)^3)/(v*a);
35         hc(i)=(0.02738/X(i))*(0.825+0.324*Ra(i)
            ^ (1/6))^2;
36     end
37 end
38 printf("\n\tx ,m\t\tRa\t\t hc ,W/(sq.m.K)\n");
39 for i=1:m
40     printf("\t%.2f\t\t%.2e\t%.2f\n",X(i),Ra(i),hc(i)
        );
41 end

```

Scilab code Exa 8.5 Natural convection on a horizontal flat surface

```
1  clc;
2  clear;
3  printf("\t\t\tChapter8_example5\n\n");
4  // determine if heat is lost lose more heat through
   // its upper surface or one of its vertical sides
5  // properties of air at (100 + 60)/2 = 80 F = 540
   // degree R from appendix table D1
6  rou= 0.0735; // density in lbm/cu.ft
7  cp=0.240; // specific heat BTU/(lbm-degree Rankine)
8  v= 16.88e-5; // viscosity in sq.ft/s
9  kf = 0.01516 ; // thermal conductivity in BTU/(hr.ft
   // .degree Rankine)
10 a = 0.859; // diffusivity in sq.ft/hr
11 Pr = 0.708; // Prandtl Number
12 Tw=100; // temperature of outside surface
   // temperature of oven in degree F
13 T_inf=60; // ambient temperature in degree F
14 g=32.2;
15 L=2; // length in ft
16 W=2; // width in ft
17 Beta=1/(T_inf+460); // volumetric thermal expansion
   // coefficient in per degree Rankine
18 printf("\nThe volumetric thermal expansion
   // coefficient is %.5f /degree R",Beta);
19 Ra=(g*Beta*(Tw-T_inf)*L^3)/(v*a/3600);
20 printf("\nThe Rayleigh Number is %.2e ",Ra);
21 hc=(kf/L)*(0.68+(0.670*Ra^(1/4))/(1+(0.492/Pr)
   // ^((9/16))^(4/9)));
22 printf("\nThe value of convection coefficient is %.3
   // f BTU/(hr.sq.ft.degree R)",hc);
```

```

23 q1side=hc*L*W*(Tw-T_inf);
24 printf("\nThe heat transferred from one side is %.1f
        BTU/hr",q1side);
25 // For the top, we have a heated surface facing
        upward, The characteristic length is determined
        as follows
26 Lc=(2*2)/(2+2+2+2);
27 Ra_L=(g*Beta*(Tw-T_inf)*Lc^3)/(v*a/3600); //
        Rayleigh number based on characteristic length
28 printf("\nThe Rayleigh Number based on
        characteristic length is %.2e ",Ra_L);
29 hc_L=(kf/Lc)*0.54*(Ra_L)^(1/4);
30 printf("\nThe convective coefficient based on
        characteristic length is %.3f BTU/(hr.sq.ft.
        degree R)",hc_L);
31 qtop=hc_L*L*W*(Tw-T_inf);
32 printf("\nThe heat transferred from top is %d BTU/hr
        ",qtop);
33 if qtop>q1side then
34     printf("\nThe top transfers heat at a higher
        rate");
35 elseif qtop<q1side
36     printf("\nThe side transfers heat at a higher
        rate");
37     else printf("\nThe top and side transfer heat at
        equal rates");
38 end

```

Scilab code Exa 8.6 Natural convection on cylinders

```

1 clc;
2 clear;
3 printf("\t\t\tChapter8_example6\n\n\n");

```

```

4 // determination of heat lost from the insulation by
   convection
5 // properties of air at  $(50 + 5)/2 = 27.5$  degree C =
   300 K from appendix table D1
6 rou= 1.177; // density in kg/cu.m
7 cp= 1005.7; // specific heat in J/(kg*K)
8 v= 15.68e-6; // viscosity in sq.m/s
9 Pr =0.708; // Prandtl Number
10 kf= 0.02624; // thermal conductivity in W/(m.K)
11 a = 0.22160e-4; // diffusivity in sq.m/s
12 g=9.81;
13 L=4; // length in m
14 D=15/100; // diameter in m
15 T_inf=5; // ambient air temperature in degree C
16 Tw=50; // outside surface temperature in degree C
17 Beta=1/(T_inf+273); // volumetric thermal expansion
   coefficient in per K
18 printf("\nThe volumetric thermal expansion
   coefficient is %.5f /K",Beta);
19 Ra=(g*Beta*(Tw-T_inf)*D^3)/(v*a);
20 printf("\nThe Rayleigh Number is %.2e ",Ra);
21 // for horizontal pipe, the convective coefficient
   is determined as follows
22 hc_h=(kf/D)*(0.60+(0.387*Ra^(1/6))/(1+(0.559/Pr)
   ^((9/16))^(8/27)))^2;
23 printf("\nThe convection coefficient for horizontal
   length is %.2f W/(sq.m.K)",hc_h);
24 As=%pi*D*L;
25 q_hor=hc_h*As*(Tw-T_inf);
26 printf("\nThe heat transferred from the horizontal
   length of 4 m is %d W",q_hor);
27 // for vertical pipe, the convective coefficient is
   determined as follows
28 hc_v=(kf/D)*0.6*(Ra*(D/L))^(1/4);
29 printf("\nThe convection coefficient for vertical
   length is %.2f W/(sq.m.K)",hc_v);
30 q_ver=hc_v*As*(Tw-T_inf);
31 printf("\nThe heat transferred from the vertical

```

```

    length of 4 m is %d W",q_ver);
32 q=q_ver+q_hor;
33 printf("\nThe total heat lost from the pipe is %d W"
    ,q);

```

Scilab code Exa 8.7 Natural convection around blocks

```

1  clc;
2  clear;
3  printf("\t\t\tChapter8_example7\n\n\n");
4  // Determinion of the convection coefficient about
   the ice cube
5  // properties of air at (0 + 70)/2 = 35 F == 495
   degree R from appendix table D1
6  rou= 0.0809; // density in lbm/cu.ft
7  cp=0.240; // specific heat BTU/(lbm-degree Rankine)
8  v= 13.54e-5; // viscosity in sq.ft/s
9  kf = 0.01402 ; // thermal conductivity in BTU/(hr.ft
   .degree Rankine)
10 a = 0.685; // diffusivity in sq.ft/hr
11 Pr = 0.712; // Prandtl Number
12 Tw=0; // temperature of outside surface temperature
   of oven in degree F
13 T_inf=70; // ambient temperature in degree F
14 g=32.2;
15 Beta=1/(T_inf+460); // volumetric thermal expansion
   coefficient in per degree Rankine
16 printf("\nThe volumetric thermal expansion
   coefficient is %.5f /degree R",Beta);
17 // The characteristic length is found by using King
   Equation
18 Lc=1/((1/1)+(1/1.2));
19 printf("\nThe characteristic length is %.3f ft",Lc);

```

```

20 Ra=(g*Beta*abs(Tw-T_inf)*Lc^3)/(v*a/3600);
21 printf("\nThe Rayleigh Number is %.2e ",Ra);
22 hc=(kf/Lc)*0.6*(Ra)^(1/4);
23 printf("\nThe value of convection coefficient is %.2
    f BTU/(hr.sq.ft.degree R)",hc);

```

Scilab code Exa 8.8 Natural convection about an array of fins

```

1  clc;
2  clear;
3  printf("\t\t\tChapter8_example8\n\n\n");
4  // determination of the maximum amount of heat that
    fins can transfer
5  // properties of air at (100 + 35)/2 = 67.5 degree C
    from appendix table D1
6  rou= 0.998; // density in kg/cu.m
7  cp= 1009.0; // specific heat in J/(kg*K)
8  v= 20.76e-6; // viscosity in sq.m/s
9  Pr =0.697; // Prandtl Number
10 kf= 0.03003; // thermal conductivity in W/(m.K)
11 a = 0.2983e-4; // diffusivity in sq.m/s
12 g=9.81;
13 T_inf=35; // ambient air temperature in degree C
14 Tw=100; // surface temperature in degree C
15 Beta=1/(T_inf+273); // volumetric thermal expansion
    coefficient in per K
16 printf("\nThe volumetric thermal expansion
    coefficient is %.5f /K",Beta);
17 // properties of aluminium from appendix table B1
18 rou_Al=2702; // density in kg/cu.m
19 k_Al=236; // thermal conductivity in W/(m.K)
20 cp_Al=896; // specific heat in J/(kg*K)
21 a_Al=97.5e-6; // diffusivity in sq.m/s

```



```
22 b=46/100;
23 w=24/100;
24 // Applying the Bar-Cohen Equations
25 zeta=((w*v^2)/(g*Beta*(Tw-T_inf)*Pr))^(1/4);
26 printf("\nThe value of zeta is %.2e ",zeta);
27 L=1.54*(k_A1/kf)^(1/2)*zeta;
28 printf("\nThe fin length is %.3f m",L);
29 S=2.89*zeta;
30 printf("\nThe fin spacing is %.5f m",S);
31 q=(b*w*(Tw-T_inf)*1.3*(k_A1*kf)^(1/2))/(6*zeta);
32 printf("\nThe heat transfer rate is %d W",q);
33 N=b/(2*S);
34 printf("\nThe number of fins can be atmost %d",N);
```

Chapter 9

Heat Exchangers

Scilab code Exa 9.1 Log Mean Temperature Difference

```
1  clc;
2  clear;
3  printf("\t\t\tChapter9_example1\n\n\n");
4  // determination of counterflow and parallel-flow
   configurations.
5  // temperatures of hot fluid in degree C
6  T1=100;
7  T2=75;
8  // temperatures of cold fluid in degree C
9  t1=5;
10 t2=50;
11 // for counterflow
12 LMTD_counter=((T1-t2)-(T2-t1))/(log((T1-t2)/(T2-t1))
   );
13 printf("\nThe LMTD for counter flow configuration is
   %.1f degree C",LMTD_counter);
14 // for parallel flow
15 LMTD_parallel=((T1-t1)-(T2-t2))/(log((T1-t1)/(T2-t2)
   ));
```

```
16 printf("\nThe LMTD for parallel flow configuration
    is %.1f degree C",LMTD_parallel);
```

Scilab code Exa 9.2 LMTD for equal outlet temperatures

```
1 clc;
2 clear;
3 printf("\t\t\tChapter9_example2\n\n\n");
4 // Determination of the LMTD for both counterflow
  and parallel-flow configurations.
5 // temperatures of hot fluid in degree F
6 T1=250;
7 T2=150;
8 // temperatures of cold fluid in degree F
9 t1=100;
10 t2=150;
11 // for counterflow
12 LMTD_counter=((T1-t2)-(T2-t1))/(log((T1-t2)/(T2-t1))
  );
13 printf("\nThe LMTD for counter flow configuration is
  %.1f degree C",LMTD_counter);
14 // for parallel flow
15 printf("\nFor a finite heat-transfer rate and a
  finite overall heat-transfer coefficient,\nif
  parallel flow is to give equal outlet
  temperatures,\nthen the area needed must be
  infinite which is not feasible economically.");
```

Scilab code Exa 9.3 Double Pipe Heat Exchanger

```

1  clc;
2  clear;
3  printf("\t\t\tChapter9_example3\n\n\n");
4  // Determination of the outlet temperature of the
   ethylene glycol for counterflow.
5  // properties of air at (195 + 85)/2 = 140 F. from
   appendix table CII
6  rou_1= 0.985*62.4; // density in lbm/ft^3
7  cp_1=0.9994; // specific heat BTU/(lbm-degree
   Rankine)
8  v_1= 0.514e-5; // viscosity in ft^2/s
9  kf_1 = 0.376 ; // thermal conductivity in BTU/(hr.ft
   .degree Rankine)
10 a_1 = 6.02e-3; // diffusivity in ft^2/hr
11 Pr_1 = 3.02; // Prandtl Number
12 m_1=5000; // mass flow rate in lbm/hr
13 T_1=195; // temperature in degree F
14 // properties of ethylene glycol at 140 degree F
   from Appendix Table C.5
15 rou_2= 1.087*62.4; // density in lbm/ft^3
16 cp_2=0.612; // specific heat BTU/(lbm-degree Rankine
   )
17 v_2= 5.11e-5; // viscosity in ft^2/s
18 kf_2 = 0.150 ; // thermal conductivity in BTU/(hr.ft
   .degree Rankine)
19 a_2 = 3.61e-3; // diffusivity in ft^2/hr
20 Pr_2 = 51; // Prandtl Number
21 m_2=12000; // mass flow rate in lbm/hr
22 T_2=85; // temperature in degree F
23 // specifications of seamless copper water tubing
   (subscripts: a = annulus, p = inner pipe or tube)
   from appendix table F2
24 ID_a=0.1674;
25 ID_p=0.1076;
26 OD_p=1.375/12;
27 // Flow Areas
28 A_p=%pi*ID_p^2/4;
29 A_a=%pi*((ID_a)^2-(OD_p)^2)/4;

```

```

30 printf("\nThe area of annulus is %.5f sq.ft",A_a);
31 printf("\nThe area of inner pipe is %.5f sq.ft",A_p)
    ;
32 if A_a>A_p then
33     printf("\nAir flows through annulus");
34     else printf("\ncarbon dioxide flows through
        annulus");
35 end
36 // Annulus Equivalent Diameters
37 D_h=ID_a-OD_p;
38 D_e=(ID_a^2-OD_p^2)/(OD_p);
39 printf("\nThe Annulus Equivalent Diameter for
    friction is %.4f ft",D_h);
40 printf("\nThe Annulus Equivalent Diameter for heat
    transfer is %.4f ft",D_e);
41 // Reynolds Numbers
42 Re_1=(m_1/3600)*(ID_p)/(v_1*rou_1*A_p);
43 printf("\nThe Reynolds Number for water is %.1e",
    Re_1);
44 Re_2=(m_2/3600)*(D_e)/(v_2*rou_2*A_a);
45 printf("\nThe Reynolds Number for ethylene glycol is
    %.2e",Re_2);
46 // Nusselt numbers
47 Nu_1=0.023*(Re_1)^(4/5)*(Pr_1)^0.3;
48 Nu_2=0.023*(Re_2)^(4/5)*(Pr_2)^0.4;
49 printf("\nThe Nusselt number for water is %d",Nu_1);
50 printf("\nThe Nusselt number for ethylene glycol is
    %d",Nu_2);
51 // Convection Coefficients
52 h_1i=Nu_1*kf_1/ID_p;
53 h_1o=h_1i*ID_p/OD_p;
54 h_2=Nu_2*kf_2/D_e;
55 printf("\nThe convective coefficient for water based
    on inner diameter is %d BTU/(hr.ft^2.degree R)",
    h_1i);
56 printf("\nThe convective coefficient for water based
    on outer diameter is %d BTU/(hr.sq.ft.degree R)"
    ,h_1o);

```

```

57 printf("\nThe convective coefficient for ethylene
    glycol is %d BTU/(hr.sq.ft.degree R)",h_2);
58 // Exchanger Coefficient
59 Uo=1/((1/h_1o)+(1/h_2));
60 printf("\nThe overall exchanger coefficient is %d
    BTU/(hr.sq.ft.degree R)",Uo);
61 R=(m_2*cp_2)/(m_1*cp_1);
62 L=20;
63 A=%pi*OD_p*L;
64 printf("\nThe ratio is %.2f and area is %.1f sq.ft",
    R,A);
65 T1=195;
66 t1=85;
67 T2=((T1*(R-1))-(R*t1*(1-exp((Uo*A*(R-1))/(m_2*cp_2))
    )))/(R*exp(Uo*A*(R-1)/(m_2*cp_2))-1);
68 printf("\nThe temperature T2=%d degree F",T2);
69 t2=t1+(T1-T2)/R;
70 printf("\nThe outlet temperature of Ethylene glycol
    is %.1f degree F",t2);

```

Scilab code Exa 9.4 Fouling Factors in Double Pipe Heat Exchangers

```

1 clc;
2 clear;
3 printf("\t\t\tChapter9_example4\n\n\n");
4 // Determination of (a) no. of exchangers required ,
    (b) the overall coefficient of (all) the
    exchanger(s), and (c) the pressure drop for each
    stream.
5 // assuming counterflow arrangement
6 // properties of air at 323 K. from appendix table
    D1
7 rou_1= 1.088; // density in kg/m^3

```

```

8 cp_1= 1007; // specific heat in J/(kg*K)
9 v_1= 18.2e-6; // viscosity in m^2/s
10 Pr_1 =0.703; // Prandtl Number
11 kf_1= 0.02814; // thermal conductivity in W/(m.K)
12 a_1 = 0.26e-4; // diffusivity in m^2/s
13 m_1=100; // mass flow rate in kg/hr
14 // temperatures in K
15 t1_air=20+273;
16 t2_air=80+273;
17 // properties of carbon dioxide at [600 + (20 + 273)
    ]/2 = 480 = 500 K. from appendix table D2
18 rou_2= 1.0732; // density in kg/m^3
19 cp_2= 1013; // specific heat in J/(kg*K)
20 v_2= 21.67e-6; // viscosity in m^2/s
21 Pr_2 =0.702; // Prandtl Number
22 kf_2= 0.03352; // thermal conductivity in W/(m.K)
23 a_2 = 0.3084e-4; // diffusivity in m^2/s
24 m_2=90; // mass flow rate in kg/hr
25 // temperatures in K
26 T1_C02=600;
27 // specifications of seamless copper tubing from
    appendix table F2
28 ID_a=.098;
29 ID_p=.07384;
30 OD_p=.07938;
31 // Flow Areas
32 A_p=%pi*ID_p^2/4;
33 A_a=%pi*((ID_a)^2-(OD_p)^2)/4;
34 printf("\nThe area of annulus is %.2e sq.m",A_a);
35 printf("\nThe area of inner pipe is %.2e sq.m",A_p);
36 if A_a>A_p then
37     printf("\nAir flows through annulus");
38     else printf("\nair flows through inner pipe");
39 end
40 // Heat Balance
41 q_air=(m_1/3600)*(cp_1)*(t2_air-t1_air);
42 printf("\nThe heat transferred is %.2e W",q_air);
43 T2_C02=T1_C02-(q_air/(m_2*cp_2/3600));

```

```

44 printf("\nThe low temperature of carbon dioxide is
      %d K",T2_CO2);
45 // Log-Mean Temperature Difference
46 LMTD_counter=((T1_CO2-t2_air)-(T2_CO2-t1_air))/(log
      ((T1_CO2-t2_air)/(T2_CO2-t1_air)));
47 printf("\nThe LMTD for counter flow configuration is
      %d degree C",LMTD_counter);
48 // Annulus Equivalent Diameters
49 D_h=ID_a-OD_p;
50 D_e=(ID_a^2-OD_p^2)/(OD_p);
51 printf("\nThe Annulus Equivalent Diameter for
      friction is %.5f m",D_h);
52 printf("\nThe Annulus Equivalent Diameter for heat
      transfer is %.4f m",D_e);
53 // Reynolds Numbers
54 Re_1=(m_1/3600)*(ID_p)/(v_1*rou_1*A_p);
55 printf("\nThe Reynolds Number for air is %.2e",Re_1)
      ;
56 Re_2=(m_2/3600)*(D_e)/(v_2*rou_2*A_a);
57 printf("\nThe Reynolds Number for carbon dioxide is
      %.2e",Re_2);
58 // Nusselt numbers
59 Nu_1=0.023*(Re_1)^(4/5)*(Pr_1)^0.3;
60 Nu_2=0.023*(Re_2)^(4/5)*(Pr_2)^0.4;
61 printf("\nThe Nusselt number for air is %.1f",Nu_1);
62 printf("\nThe Nusselt number for carbon dioxide is %
      .1f",Nu_2);
63 // Convection Coefficients
64 h_1i=Nu_1*kf_1/ID_p;
65 h_1o=h_1i*ID_p/OD_p;
66 h_2=Nu_2*kf_2/D_e;
67 printf("\nThe convective coefficient for air based
      on inner diameter is %.1f W/(sq.m.K)",h_1i);
68 printf("\nThe convective coefficient for air based
      on outer diameter is %.1f W/(sq.m.K)",h_1o);
69 printf("\nThe convective coefficient for carbon
      dioxide is %.1f W/(sq.m.K)",h_2);
70 // Fouling Factors in (m^2.K)/W

```



```

71 Rd_air=.0004;
72 Rd_CO2=0.002;
73 // exchanger coefficients
74 Uo=1/((1/h_1o)+(1/h_2));
75 Uo=1/((1/Uo)+Rd_air+Rd_CO2);
76 printf("\nThe overall exchanger coefficient is %.1f
      W/(sq.m.K)",Uo);
77 // area required
78 A=q_air/(Uo*LMTD_counter);
79 printf("\nThe area required is %.2f sq.m",A);
80 // surface area of one exchanger is A=%pi*OD*L, so
81 L=(A/(%pi*OD_p)); // length of each exchanger
82 L_available=2; // available exchanger length
83 N=L_available/L; // no. of exchangers
84 printf("\nThe number of exchangers is %d",N);
85 //friction factors
86 fp=0.0245; //friction factor for air fom figure 6.14
      corresponding to Reynolds Number calculated
      above
87 fa=0.033; //friction factor for carbon dioxide fom
      figure 6.14 corresponding to Reynolds Number
      calculated above
88 // Velocities
89 V_air=(m_1/3600)/(rou_1*A_p);
90 V_CO2=(m_2/3600)/(rou_2*A_a);
91 printf("\nThe velocity of air is %.2f m/s",V_air);
92 printf("\nThe velocity of carbon dioxide is %.2f m/s
      ",V_CO2);
93 // pressure drops
94 dP_p=(fp*L_available*rou_1*V_air^2)/(ID_p*2);
95 dP_a=((rou_2*V_CO2^2)/2)*((fa*L_available/D_h)+1);
96 printf("\nThe pressure drop for tube side is %.2f Pa
      ",dP_p);
97 printf("\nThe pressure drop for shell side is %d Pa"
      ,dP_a);
98 printf("\n\t\t\tSummary of Requested Information\n")
      ;
99 printf("(a) Exchanger required: %d\n(b) Overall

```

```

exchanger coefficient = %.1f W/(sq.m.K)\n(c) Air
pressure drop = %.2f Pa\nDiesel exhaust pressure
drop = %d Pa",N,Uo,dP_p,dP_a);

```

Scilab code Exa 9.5 Shell and Tube heat exchangers

```

1  clc;
2  clear;
3  printf("\t\t\tChapter9_example5\n\n\n");
4  // Determination of the outlet temperature of the
   // distilled water and the pressure drop for each
   // stream.
5  // properties of (distilled) water at 104 F from
   // appendix table CII
6  rou_1= 0.994*62.4; // density in lbm/ft^3
7  cp_1=0.998; // specific heat BTU/(lbm-degree Rankine
   // )
8  v_1= 0.708e-5; // viscosity in ft^2/s
9  kf_1 = 0.363 ; // thermal conductivity in BTU/(hr.ft
   // .degree Rankine)
10 a_1 = 5.86e-3; // diffusivity in ft^2/hr
11 Pr_1 = 4.34; // Prandtl Number
12 m_1=170000; // mass flow rate in lbm/hr
13 T1=110; // temperature in degree F
14 // properties of (raw) water at 68 F from Appendix
   // Table C11
15 rou_2= 62.4; // density in lbm/ft^3
16 cp_2=0.9988; // specific heat BTU/(lbm-degree
   // Rankine)
17 v_2= 1.083e-5; // viscosity in ft^2/s
18 kf_2 = 0.345 ; // thermal conductivity in BTU/(hr.ft
   // .degree Rankine)
19 a_2 = 5.54e-3; // diffusivity in ft^2/hr

```

```

20 Pr_2 = 7.02; // Prandtl Number
21 m_2=150000; // mass flow rate in lbm/hr
22 t1=65; // temperature in degree F
23 // specifications of 3/4-in-OD, 18-BWG tubes, from
    table 9.2
24 OD=3/(4*12);
25 ID=0.652/12;
26 OD_p=1.375/12;
27 Nt=224; // from table 9.3
28 Np=2; // no. of tube passes
29 // Shell dimensions and other miscellaneous data
30 Ds=17.25/12;
31 Nb=15; // no. of baffles
32 B=1;
33 sT=15/(16*12);
34 C=sT-OD;
35 // flow areas
36 At=(Nt*pi*ID^2)/(4*Np);
37 As=(Ds*C*B)/sT;
38 printf("\nThe areas are %.3f sq.ft and %.3f sq.ft",
    At,As);
39 if At>As then
40     printf("\nThe distilled water flows through the
        tubes");
41     else printf("\nThe raw water flows through the
        tubes");
42 end
43 // Shell Equivalent Diameter
44 De=4*[(sT/2)*(0.86*sT)-(pi*OD^2/8)]/(pi*OD/2);
45 printf("\nThe equivalent diameter is %.4f ft",De);
46 // Reynolds Numbers
47 Re_s=(m_1/3600)*(De)/(v_1*rou_1*As);
48 printf("\nThe Reynolds Number for raw water is %.2e"
    ,Re_s);
49 Re_t=(m_2/3600)*(ID)/(v_2*rou_2*At);
50 printf("\nThe Reynolds Number for distilled water is
    %.2e",Re_t);
51 // Nusselt numbers

```

```

52 Nu_t=0.023*(Re_t)^(4/5)*(Pr_2)^0.4;
53 Nu_s=0.36*(Re_s)^(.55)*(Pr_1)^(1/3);
54 printf("\nThe Nusselt number for raw water is %.1f",
    Nu_t);
55 printf("\nThe Nusselt number for distilled water is
    %.1f",Nu_s);
56 h_ti=Nu_t*kf_2/ID;
57 h_to=h_ti*ID/OD;
58 h_s=Nu_s*kf_1/De;
59 printf("\nThe convective coefficient for raw water
    based on inner diameter is %d BTU/(hr.sq.ft.
    degree R)",h_ti);
60 printf("\nThe convective coefficient for raw water
    based on outer diameter is %d BTU/(hr.sq.ft.
    degree R)",h_to);
61 printf("\nThe convective coefficient for distilled
    water is %d BTU/(hr.sq.ft.degree R)",h_s);
62 // Exchanger Coefficient
63 Uo=1/((1/h_to)+(1/h_s));
64 printf("\nThe overall exchanger coefficient is %d
    BTU/(hr.sq.ft.degree R)",Uo);
65 R=(m_2*cp_2)/(m_1*cp_1);
66 L=16;
67 Ao=Nt*%pi*OD*L;
68 printf("\nThe ratio is %.3f and area is %.1f sq.ft",
    R,Ao);
69 UoAo_mccp=(Uo*Ao)/(m_2*cp_2);
70 printf("\n(UoAo)/(McCpc)=%.2f",UoAo_mccp);
71 S=0.58; //value of S from fig. 9.13 Ten Broeck graph
    corresponding to the value of (UoAo)/(McCpc)
72 t2=S*(T1-t1)+t1;
73 T2=T1-R*(t2-t1);
74 printf("\nt2=%.1f degree F\nT2=%.1f degree F",t2,T2)
    ;
75 //friction factors
76 ft=0.029; //friction factor for raw water fom figure
    6.14 corresponding to Reynolds Number calculated
    above

```

```

77 printf("\nFriction factor for raw water fom figure
    6.14 corresponding to Reynolds Number calculated
    above is %.3f",ft);
78 fs=0.281; //friction factor for distilled water fom
    figure 6.14 corresponding to Reynolds Number
    calculated above
79 printf("\nFriction factor for distilled water fom
    figure 6.14 corresponding to Reynolds Number
    calculated above is %.3f",fs);
80 // Velocities
81 V_t=(m_2/3600)/(rou_2*At);
82 V_s=(m_1/3600)/(rou_1*As);
83 printf("\nThe velocity of raw water is %.2f ft/s",
    V_t);
84 printf("\nThe velocity of distilled water is %.2f ft
    /s",V_s);
85 // pressure drops
86 gc=32.2;
87 dP_t=(rou_2*V_t^2)*((ft*L*Np/ID)+4*Np)/(2*gc);
88 dP_s=((rou_1*V_s^2)*(fs*Ds*(Nb+1)))/(2*gc*De);
89 printf("\nThe pressure drop for tube side is %.1f
    lbf/sq.ft = %.1f psi",dP_t,dP_t/147);
90 printf("\nThe pressure drop for shell side is %.1f
    lbf/sq.ft = %.1f psi",dP_s,dP_s/147);
91 printf("\n\t\t\tSummary of Requested Information\n")
    ;
92 printf("\nOutlet Temperatures:\n\tRaw Water: %.1f
    degree F\n\tDistilled Water: %.1f degree F\n",t2,
    T2);
93 printf("\nPressure Drops:\n\tRaw Water: %.1f ddegree
    F\n\tDistilled Water: %.1f degree F\n",dP_t/147,
    dP_s/147);

```

Scilab code Exa 9.6 Effectiveness NTU method of analysis

```
1  clc;
2  clear;
3  printf("\t\t\tChapter9_example6\n\n\n");
4  // Using the effectiveness-NTU method to calculate
   the outlet temperatures of the fluids
5  // Data from Example 9.5
6  // properties of (distilled) water at 104 F
7  m_1=170000; // mass flow rate in lbm/hr
8  T1=110; // temperature in degree F
9  cp_1=0.998; // specific heat BTU/(lbm-degree Rankine
   )
10 // properties of (raw) water at 68 F
11 m_2=150000; // mass flow rate in lbm/hr
12 t1=65; // temperature in degree F
13 cp_2=0.9988; // specific heat BTU/(lbm-degree
   Rankine)
14 Uo=350; // exchanger coefficient
15 Ao=703.7;
16 // The effectiveness-NTU approach is used when the
   overall heat transfer coefficient is known
17 // determining the capacitances
18 mcp_raw=m_2*cp_2;
19 mcp_distilled=m_1*cp_1;
20 printf("\nThe capacitance value of raw water is %d
   BTU/(hr. degree R)",mcp_raw);
21 printf("\nThe capacitance value of distilled water
   is %d BTU/(hr. degree R)",mcp_distilled);
22 if mcp_raw>mcp_distilled then
23     mcp_max=mcp_raw;
24     mcp_min=mcp_distilled;
25     printf("\nDistilled water has minimum
   capacitance");
26 else mcp_max=mcp_distilled;
27     mcp_min=mcp_raw;
28     printf("\nRaw water has minimum capacitance");
29 end
```

```

30 // determination of parameters for determining
    effectiveness
31 mcp_min_max=mcp_min/mcp_max;
32 UA_mcpmin=(Uo*Ao)/(mcp_min);
33 printf("\nThe required parameters are mcp_min/
    mcp_max=%0.3f and (UoAo/mcp_min)=%0.2f",mcp_min_max
    ,UA_mcpmin);
34 effectiveness=0.58; //value of effectiveness from
    figure 9.15 corresponding to the above calculated
    values of capacitance ratio and (UoAo/mcp_min)
35 qmax=mcp_min*(T1-t1);
36 printf("\nThe maximum heat transfer is %0.2e BTU/hr",
    qmax);
37 q=effectiveness*qmax; // actual heat transfer
38 printf("\nThe actual heat transfer is %0.2e BTU/hr",q
    );
39 t2=(q/mcp_raw)+t1;
40 T2=T1-(q/mcp_distilled);
41 printf("\nThe Outlet temperatures are:\n\tRaw Water:
    %0.1f degree F\n\tDistilled Water:%0.1f degree F\n"
    ,t2,T2);

```

Scilab code Exa 9.7 Crossflow heat exchangers

```

1 clc;
2 clear;
3 printf("\t\t\tChapter9_example7\n\n\n");
4 // (a) Determine the UA product for the exchanger
    . (b) Calculate the exit temperatures for the
    exchanger, assuming that only the inlet
    temperatures are known
5 // properties of engine oil at (190 + 158)/2 = 174
    F = 176 degree F from appendix table C4

```

```

6 rou_1= 0.852*62.4; // density in lbm/ft^3
7 cp_1=0.509; // specific heat BTU/(lbm-degree Rankine
)
8 v_1= 0.404e-3; // viscosity in ft^2/s
9 kf_1 = 0.08; // thermal conductivity in BTU/(hr.ft.
degree Rankine)
10 a_1 = 2.98e-3; // diffusivity in ft^2/hr
11 Pr_1 = 490; // Prandtl Number
12 m_1=39.8; // mass flow rate in lbm/min
13 // temperatures in degree F
14 T1=190;
15 T2=158;
16 // properties of air at (126 + 166)/2 = 146 F = 606
degree R from appendix table D1
17 rou_2= 0.0653; // density in lbm/ft^3
18 cp_2=0.241; // specific heat BTU/(lbm-degree Rankine
)
19 v_2= 20.98e-5; // viscosity in ft^2/s
20 kf_2 = 0.01677 ; // thermal conductivity in BTU/(hr.
ft.degree Rankine)
21 a_2 = 1.066; // diffusivity in ft^2/hr
22 Pr_2 = 0.706; // Prandtl Number
23 m_2=67; // mass flow rate in lbm/min
24 // temperatures in degree F
25 t1=126;
26 t2=166;
27 // Heat Balance
28 q_air=m_2*cp_2*60*(t2-t1);
29 q_oil=m_1*cp_1*60*(T1-T2);
30 printf("\nThe heat gained by air is %.2e BTU/hr",
q_air);
31 printf("\nThe heat lost by oil is %.2e BTU/hr",q_oil
);
32 // for counterflow
33 LMTD=((T1-t2)-(T2-t1))/(log((T1-t2)/(T2-t1)));
34 printf("\nThe LMTD for counter flow configuration is
%.1f degree F",LMTD);
35 // Frontal Areas for Each Fluid Stream

```



```

36 Area_air=(9.82*8)/144;
37 Area_oil=(3.25*9.82)/144;
38 printf("\nThe Core frontal area on the air side is %
    .3f sq.ft\nThe Core frontal area on the oil side
    is %.3f sq.ft ",Area_air,Area_oil);
39 // Correction Factors (parameters calculated first)
40 S=(t2-t1)/(T1-t1);
41 R=(T1-T2)/(t2-t1);
42 F=0.87; //value of correction factor from figure
    9.21a corresponding to above calculated values of
    S and R
43 // Overall Coefficient (q = U*A*F*LMTD)
44 UA=q_air/(F*LMTD);
45 printf("\nThe Overall Coefficient is %.2e BTU/(hr.
    degree R)",UA);
46 // determining the capacitances
47 mcp_air=m_2*cp_2*60;
48 mcp_oil=m_1*cp_1*60;
49 printf("\nThe capacitance value of air is %d BTU/(hr
    . degree R)",mcp_air);
50 printf("\nThe capacitance value of engine oil is %d
    BTU/(hr. degree R)",mcp_oil);
51 if mcp_air>mcp_oil then
52     mcp_max=mcp_air;
53     mcp_min=mcp_oil;
54     printf("\nEngine Oil has minimum capacitance");
55 else mcp_max=mcp_oil;
56     mcp_min=mcp_air;
57     printf("\nAir has minimum capacitance");
58 end
59 // determination of parameters for determining
    effectiveness
60 mcp_min_max=mcp_min/mcp_max;
61 NTU=(UA/mcp_min);
62 printf("\nThe required parameters are mcp_min/
    mcp_max=%.3f and (UoAo/mcp_min)=%.2f",mcp_min_max
    ,NTU);
63 effectiveness=0.62; //value of effectiveness from

```

```

        figure 9.21b corresponding to the above
        calculated values of capacitance ratio and (UoAo/
        mcp_min): ');
64  t2_c=(T1-t1)*effectiveness+t1;
65  T2_c=T1-(mcp_min_max)*(t2_c-t1);
66  printf("\n\t\t\tSummary of Requested Information\n")
    ;
67  printf("\n(a) UA = %.2e BTU/(hr. degree R)",UA);
68  printf("\n(b) The Outlet temperatures (degree F)");
69  printf("\n\t\tCalculated\t\tGiven in Problem Statement")
    ;
70  printf("\nAir\t\t\t%d\t\t%d",t2_c,t2);
71  printf("\nEngine Oil\t\t\t%d\t\t%d",T2_c,T2);

```

Chapter 10

Condensation and Vaporization Heat Transfer

Scilab code Exa 10.1 Laminar film condensation on a vertical flat surface

```
1 clc;
2 clear;
3 printf("\t\t\tChapter10_example1\n\n\n");
4 // Calculation of the heat-transfer rate and the
   amount of steam condensed
5 // properties of engine oil at (328 + 325)/2 = 326.5
   degree F = 320 F from appendix table C11
6 rou_f= 0.909*62.4; // density in lbm/ft^3
7 cp=1.037; // specific heat BTU/(lbm-degree Rankine)
8 v_f= 0.204e-5; // viscosity in ft^2/s
9 kf = 0.393; // thermal conductivity in BTU/(lbm.ft.
   degree Rankine)
10 a = 6.70e-3; // diffusivity in ft^2/hr
11 Pr = 1.099; // Prandtl Number
12 V_v=4.937; // specific volume in ft^3/lbm from
```

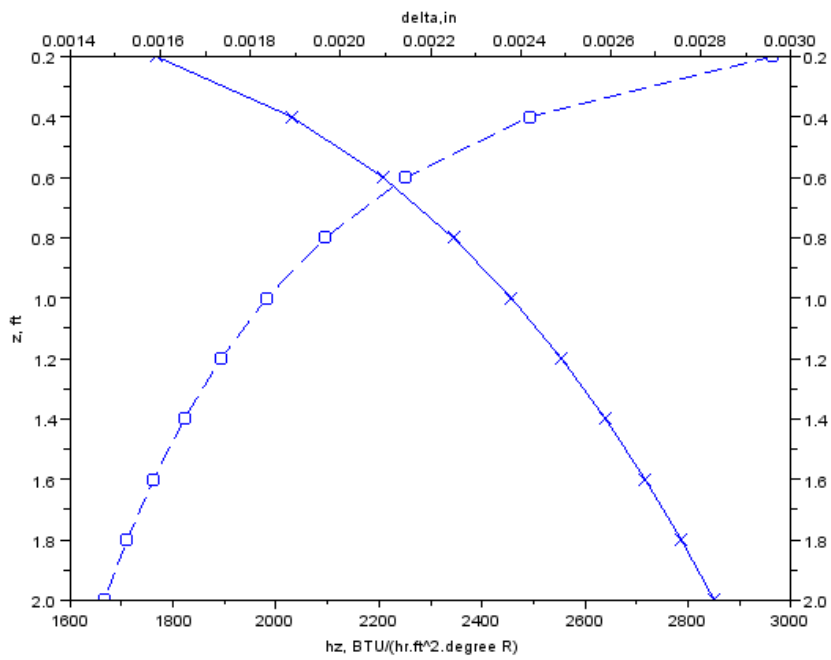


Figure 10.1: Laminar film condensation on a vertical flat surface

```

    superheated steam tables
13 rou_v=1/V_v; // vapour density
14 g=32.2;
15 hfg=888.8; // from saturated steam tables
16 Tg=327.81;
17 Tw=325;
18 L=2; // length in ft
19 W=3; // width in ft
20 z=0.2:0.2:2; // distance from entry of plate in ft
21 [n m]=size(z);
22 // film thickness is given as follows
23 for i=1:m
24 delta(i)=[(4*kf*v_f*z(i)*(Tg-Tw)/3600)/(rou_f*g*hfg
    *(1-(rou_v/rou_f)))]^(1/4);
25 hz(i)=(kf/delta(i));
26 end
27 printf("\nGrowth of and Heat-Transfer Coefficient
    for the Condensate Film of Example 10.1 ");
28 printf("\nz, ft\tdelta, ft\tdelta, in\thz, BTU/(hr.
    sq.ft.degree Rankine)");
29 for i=1:m
30 printf("\n%.1f\t%.2e\t%.4f\t\t%d\n",z(i),delta(i)
    ,12*delta(i),hz(i));
31 end
32 hL=(4/3)*hz(m); // at plate end
33 mf=(hL*L*W*(Tg-Tw))/hfg;
34 printf("\nThe convective coefficient at the plate
    end is %d BTU/(hr.sq.ft. degree Rankine)",hL);
35 printf("\nThe amount of steam condensed is %.1f lbm/
    hr",mf);
36 q=mf*hfg;
37 printf("\nThe heat transfer rate is %.2e BTU/hr",q);
38 Re=(4*mf/3600)/(W*rou_f*v_f);
39 printf("\nThe Reynolds Number is %d",Re);
40 if Re<1800 then
41     printf("\nThe film is laminar and above
        equations apply");
42     else printf("\nThe film is not laminar and above

```

```

        assumption is wrong");
43 end
44 subplot(211);
45 plot(delta*12,z,'x-'); // our first figure
46 a1 = gca();
47 a1.x_location="top";
48 a1.axes_reverse=["off","on"];
49 subplot(212)
50 plot(hz,z,'o—'); // our second figure
51 a2 = gca();
52 a2.axes_reverse=["off","on"];
53 a2.x_location="bottom";
54 a2.axes_visible = ["on","on","on"];
55 a2.y_location = "right";
56 x_label1=a1.x_label;
57 x_label1.text="delta ,in";
58 x_label2=a2.x_label;
59 x_label2.text="hz , BTU/(hr .sq .ft .degree R)";
60 y_label=a1.y_label;
61 y_label.text="z , ft";
62 a1.axes_bounds=[0 0 1 1]; // modify the first
    figure to occupy the whole area
63 a2.axes_bounds=[0 0 1 1]; // modify the second
    figure to occupy the whole area too
64 a2.filled="off";

```

Scilab code Exa 10.2 Film condensation on a horizontal tube

```

1 clc;
2 clear;
3 printf("\t\t\tChapter10_example2\n\n\n");
4 // Determination of both the heat that the cooling
    fluid must remove and the condensation rate.

```

```

5 // properties of water at (100 + 60)/2 = 80 C from
  appendix table C11
6 rou_f= 947; // density in kg/m^3
7 cp_1= 4196; // specific heat in J/(kg*K)
8 v_1= 0.364e-6; // viscosity in m^2/s
9 Pr_1 =2.22; // Prandtl Number
10 kf= 0.668; // thermal conductivity in W/(m.K)
11 a_1 =1.636e-7; // diffusivity in m^2/s
12 Vv=1.9364; // specific volume in m^3/kg
13 rou_v=1/Vv; // vapor density;
14 g=9.81;
15 hfg=2257.06*1000;
16 Tg=100;
17 Tw=60;
18 L=1;
19 printf("\nThe vapor density is %.3f kg/cu.m",rou_v);
20 // specifications of 1 nominal schedule 40 pipe from
  appendix F1
21 OD=.03340;
22 hD=0.782*[(g*rou_f*(1-(rou_v/rou_f))*(kf^3)*hfg)/(
  v_1*OD*(Tg-Tw))]^(1/4);
23 printf("\nThe average heat transfer coefficient is %
  .3e W/(sq.m.K)",hD);
24 q=hD*%pi*OD*L*(Tg-Tw);
25 printf("\nThe heat flow rate is %.1e W",q);
26 mf=q/hfg;
27 printf("\nThe rate at which steam condenses is %.2f
  kg/s = %d kg/hr",mf,.02*3600);

```

Scilab code Exa 10.3 Nucleate pool boiling critical heat flux

```

1 clc;
2 clear;

```

```

3 printf("\t\t\tChapter10_example3\n\n\n");
4 // Calculation of (a) the power input to the water
   for boiling to occur, (b) the evaporation rate of
   water, and (c) the critical heat flux.
5 // properties of water at 100 C = 373 K from
   appendix table 10.3
6 rou_f=958; // density in kg/m^3
7 cp_f= 4217; // specific heat in J/(kg*K)
8 v_f= 2.91e-7; // viscosity in m^2/s
9 Pr_f =1.76; // Prandtl Number
10 rou_g=0.596;
11 sigma=0.0589; // surface tension in N/m
12 hfg=2257000;
13 Tw=120
14 Tg=100;
15 D=.141; // diameter of pan in m
16 g=9.81;
17 gc=1;
18 // nucleate boiling regime
19 Cw=0.0132; // formechanically polished stainless
   steel from table 10.2
20 q_A=(rou_f*v_f*hfg)*[(g*rou_f*(1-(rou_g/rou_f)))/(
   sigma*gc)]^(1/2)*[(cp_f*(Tw-Tg))/(Cw*hfg*Pr_f
   ^1.7)]^3;
21 printf("\nThe heat transferred per unit area is %.2e
   W/sq.m",q_A);
22 A=%pi*D^2/4;
23 printf("\nThe area of the pan inside-bottom surface
   in contact with liquid is %.2e sq.m",A);
24 printf("\n\n\t\t\tSolution to part (a)");
25 q=q_A*A; // power delivered to the water in W
26 printf("\nThe power delivered to the water is %.2f
   kW",q/1000);
27 printf("\n\n\t\t\tSolution to part (b)");
28 mf=q/hfg; // water evaporation rate
29 printf("\nThe water evaporation rate is %.2e kg/s =
   %.2f kg/hr",mf,mf*3600);
30 printf("\n\n\t\t\tSolution to part (c)");

```



```
31 q_cr=0.18*hfg*[sigma*g*gc*rou_f*rou_g^2]^(1/4);
32 printf("\nThe critical heat flux is %.2e W/sq.m",
    q_cr);
```

Chapter 11

Introduction to Radiation Heat Transfer

Scilab code Exa 11.1 Radiation Intensity

```
1 clc ;
2 clear ;
3 printf("\t\t\tChapter11_example1\n\n\n");
4 // Calculation of the value of the solid angle
   subtended by surfaces dA2 and dA3 with respect to
   dA1 (b) the intensity of emission from dA, in
   the direction of the other areas (c) the rate at
   which radiation emitted by dA, is intercepted by
   the other areas
5 printf("\t\t\tSolution to Part (a)\n");
6 // solid angle is calculate using the equation  $d\omega = dA$ 
    $\cdot \cos(\text{Beta}) / r^2$ 
7 // Beta is the angle between the surface normal of a
   receiver surface and the line connecting the two
   surfaces
8 // For area A2
9 // dimensions are 1X1 in , so
```

```

10 dA2=(1*1)/144;
11 Beta1=40*%pi/180;
12 r=4;
13 dw2_1=dA2*cos(Beta1)/r^2;
14 printf("\nThe solid angle subtended by area dA2 with
        respect to dA1 is %.2e sr",dw2_1);
15 dA3=dA2;
16 Beta2=0;
17 dw3_1=dA3*cos(Beta2)/r^2;
18 printf("\nThe solid angle subtended by area dA3 with
        respect to dA1 is %.2e sr",dw3_1);
19 printf("\n\n\t\t\tSolution to Part (b)\n");
20 theta2=%pi*50/180;
21 theta3=%pi*60/180;
22 I_theta2=2000*(1-0.4*(sin(theta2))^2);
23 I_theta3=2000*(1-0.4*(sin(theta3))^2);
24 printf("\n The intensity of radiation emitted from
        dA1 in the direction of dA2 is %d BTU/(hr.sq.ft.
        sr)",I_theta2);
25 printf("\n The intensity of radiation emitted from
        dA1 in the direction of dA3 is %d BTU/(hr.sq.ft.
        sr)",I_theta3);
26 printf("\n\n\t\t\tSolution to Part (c)\n");
27 dA1=1/144;
28 dq1_2=I_theta2*dA1*cos(theta2)*dw2_1;
29 dq1_3=I_theta3*dA1*cos(theta3)*dw3_1;
30 printf("\nThe rate at which radiation emitted by dA1
        is intercepted by dA2 is %.2e BTU/hr",dq1_2);
31 printf("\nThe rate at which radiation emitted by dA1
        is intercepted by dA3 is %.2e BTU/hr",dq1_3);

```

Scilab code Exa 11.2 Irradiation and Radiosity

```

1  clc;
2  clear;
3  printf("\\t\\t\\tChapter11_example2\\n\\n\\n");
4  // Calculation of the value of the solid angle
   subtended by surfaces dA2 with respect to dA1 (b)
   the rate at which radiation emitted by dA1 is
   intercepted by dA2 (c) the irradiation
   associated with dA2
5  printf("\\t\\t\\tSolution to Part (a)\\n");
6  // solid angle is calculate using the equation  $dw=dA$ 
    $*\cos(\text{Beta})/r^2$ 
7  // The angle Beta is 0 because the surface normal of
   dA2 is directed at dA1
8  dA2=0.02*0.02;
9  Beta=0;
10 r=1;
11 dw2_1=dA2*cos(Beta)/r^2;
12 printf("\\n\\n\\n\\t\\t\\tThe solid angle subtended by area dA2 with
   respect to dA1 is %.2e sr",dw2_1);
13 printf("\\n\\n\\n\\t\\t\\tSolution to Part (b)\\n");
14 dA1=dA2;
15 theta=%pi*30/180;
16 I_theta=1000;// The intensity of radiation leaving
   dA1 in any direction is 1 000 W/(m^2.sr
17 dq1_2=I_theta*dA1*cos(theta)*dw2_1;
18 printf("\\n\\n\\n\\t\\t\\tThe rate at which radiation emitted by dA1
   is intercepted by dA2 is %.2e W",dq1_2);
19 printf("\\n\\n\\n\\t\\t\\tSolution to Part (c)\\n");
20 // The irradiation associated with dA2 can be found
   by dividing the incident radiation by the
   receiver area
21 dQ1_2=dq1_2/dA2;
22 printf("\\n\\n\\n\\t\\t\\tThe irradiation associated with dA2 is %.2
   e W/sq.m",dQ1_2);

```

Scilab code Exa 11.3 Emissivity and Rate of radiant emission

```
1  clc;
2  clear;
3  printf("\t\t\tChapter11_example3\n\n\n");
4  // (a) Calculation of the emissivity of the hole.(b)
   the rate of radiant emission from the hole
5  D=2.5/12; // diameter in ft
6  L=4.5/12; // length in ft
7  A=(2*%pi*D^2/4)+(%pi*D*L);
8  printf("\nThe inside surface area is %.3f sq.ft ",A)
   ;
9  A_hole=%pi*(1/(8*12))^2/4;
10 printf("\nThe area of a 1/8 inch hole is %.3e sq.ft"
   ,A_hole);
11 f=A_hole/A; // fraction of area removed
12 printf("\nThe fraction of area removed is %.3e ",f);
13 printf("\n\n\t\t\tSolution to Part (a)\n");
14 // for rolled and polished aluminum, that emissivity
   = 0.039 from appendix table E1
15 emissivity=0.039;
16 emissivity_hole=emissivity/(emissivity+(1-emissivity
   )*f);
17 printf("\nThe emissivity of the hole is %.4f",
   emissivity_hole);
18 printf("\n\n\t\t\tSolution to Part (b)\n");
19 sigma=0.1714e-8; // stefan Boltzmann constant in BTU
   /(hr~ft^2 degree R)
20 T=150+460; // temperature in degree R
21 qe=emissivity_hole*sigma*T^4;
22 printf("\nThe heat lost per unit area of the hole is
   %d BTU/hr",qe);
```

```
23 Qe=A_hole*qe;
24 printf("\nThe heat lost by the hole is %.2e BTU/hr",
    Qe);
```

Scilab code Exa 11.4 Plancks distribution law

```
1  clc;
2  clear;
3  printf("\t\t\tChapter11_example4\n\n\n");
4  // Determination of the percentage of total emitted
   energy that lies in the visible range.
5  T=2800;
6  lambda1=4e-7;
7  lambda2=7e-7;
8  hT=lambda1*T;
9  lambdaT=lambda2*T;
10 printf("\nhT=%.2e m.K and lambda2_T=%.2e m.K",hT,
    lambdaT);
11 I1=0.0051; //Fraction of Total Radiation Emitted for
   lower Wavelength-Temperature Product from Table
   11.1
12 I2=0.065; //Fraction of Total Radiation Emitted for
   upper Wavelength-Temperature Product from Table
   11.1
13 dI=I2-I1;
14 printf("\nThe percentage of total emitted energy
   that lies in the visible range is %.1f percent",
    dI*100);
```

Scilab code Exa 11.5 Wiens displacement law

```
1  clc;
2  clear;
3  printf("\t\t\tChapter11_example5\n\n\n");
4  // Estimation of the surface temperature of the sun
   and the emitted heat flux
5  lambda_max=0.5e-6; // maximum wavelength in m
6  // From Wien s Displacement Law we can write
   lambda_max*T=2.898e-3 m.K
7  T=2.898e-3/lambda_max;
8  printf("\nThe Surface Temperature of the Sun is %d K
   ",T);
9  // The heat flux is given by the Stefan-Boltzmann
   Equation as q=sigma*T^4
10 sigma=5.675e-8; // value of Stefan-Boltzmann
   constant in W/(m^2.K^4)
11 q=sigma*T^4;
12 printf("\nThe heat flux emitted is %.3e W/sq.m",q);
```

Scilab code Exa 11.6 Transmission through glass windshield

```
1  clc;
2  clear;
3  printf("\t\t\tChapter11_example6\n\n\n");
4  // (a) Calculation of the rate at which the sun s
   radiant energy is transmitted through the glass
   windshield. The interior of the car is considered
   to be a black body that radiates at 100 F. (b)
   Calculation of the rate at which radiant energy
   from the car interior is transmitted through the
   glass windshield.
5  printf("\t\t\tSolution to Part (a)\n\n");
```

```

6 lambda1=300e-9; // lower limit of wavelength
7 lambda2=380e-9; // upper limit of wavelength
8 T=5800;
9 lambda1_T=lambda1*T;
10 lambda2_T=lambda2*T;
11 printf("\nThe Lower and Upper limits of Wavelength-
    Temperature Products are %.2e m.K and %.3e m.K
    respectively",lambda1_T,lambda2_T);
12 I1=0.101; //Fraction of Total Radiation Emitted for
    lower Wavelength-Temperature Product from Table
    11.1
13 I2=0.0334; //Fraction of Total Radiation Emitted for
    upper Wavelength-Temperature Product from Table
    11.1
14 dI=abs(I2-I1);
15 t=dI*0.68; // transmissivity
16 printf("\nThe Transmittivity is %.4f",t);
17 q=1100; // radiation received by car in W/sq.m
18 q_in=t*q; // energy transmitted from the sun through
    the glass
19 printf("\nThe energy transmitted from the sun
    through the glass is %.1f W/sq.m",q_in);
20 printf("\n\t\t\t\t\tSolution to Part (b)\n");
21 Tb=311; // temperature of black body source in K
22 lambda1_Tb=lambda1*Tb;
23 lambda2_Tb=lambda2*Tb;
24 printf("\nThe Lower and Upper limits of Wavelength-
    Temperature Products are %.2e m.K and %.2e m.K
    respectively",lambda1_Tb,lambda2_Tb);
25 dI_b=0; // Table 11.1 gives negligibly small values
    of the corresponding integrals.
26 t_b=dI_b*0.68; // transmissivity
27 q_out=t_b*q;
28 printf("\nthe rate at which radiant energy from the
    car interior is transmitted through the glass
    windshield is %d W/sq.m",q_out);

```

Chapter 12

Radiation Heat Transfer between Surfaces

Scilab code Exa 12.3 View Factor algebra for pairs of surfaces

```
1  clc ;
2  clear ;
3  printf("\t\t\tChapter12_example3\n\n\n");
4  //  Determination of the heat transferred by
    radiation from dA1 to A.
5  //  The view factor Fd1_2 can be calculated as Fd1_2=
    Fd1_3+Fd1_4+Fd1_5
6  //  For Fd1_3
7  a_13=100;
8  b_13=250;
9  c_13=100;
10 X_13=a_13/c_13;
11 Y_13=b_13/c_13;
12 printf("\nFor Fd1_3, the values of a/c=%0.1f and b/c=
    %0.1f",X_13,Y_13);
13 Fd1_3=0.17; // value for Fd1_3 corresponding to
    above calculated values of a/c and b/c
```

```

14 // For Fd1_4
15 a_14=300;
16 b_14=50;
17 c_14=100;
18 X_14=a_14/c_14;
19 Y_14=b_14/c_14;
20 printf("\nFor Fd1_4, the values of a/c=%0.1f and b/c=
    %0.1f",X_14,Y_14);
21 Fd1_4=0.11; //value for Fd1_4 corresponding to above
    calculated values of a/c and b/c
22 // For Fd1_5
23 a_15=100;
24 b_15=50;
25 c_15=100;
26 X_15=a_15/c_15;
27 Y_15=b_15/c_15;
28 printf("\nFor Fd1_5, the values of a/c=%0.1f and b/c=
    %0.1f",X_15,Y_15);
29 Fd1_5=0.09; //value for Fd1_3 corresponding to above
    calculated values of a/c and b/c
30 Fd1_2=Fd1_3+Fd1_4-Fd1_5;
31 printf("\nFd1_2=%0.2f",Fd1_2);
32 printf("\n%d percent of the energy leaving dA1
    reaches A",100*Fd1_2);
33 sigma=0.1714e-8; // Stefan-Boltzmann constant
34 T1=660;
35 T2=560;
36 q12_A1=sigma*Fd1_2*(T1^4-T2^4);
37 printf("\nThe net heat transferred is %0.1f BTU/(hr.
    sq.ft)",q12_A1);

```

Scilab code Exa 12.4 View Factor algebra for enclosures

```

1  clc;
2  clear;
3  printf("\t\t\tChapter12_example4\n\n\n");
4  // Determination of the heat transferred to the
   conveyed parts for the conditions given
5  L1=1;
6  angle=%pi*45/180;
7  L2=L1*sin(angle);
8  L3=L2;
9  printf("\nThe Widths are L1=%d m, L2=%.3f m and L3=%
   .3f m",L1,L2,L3);
10 T1=303;
11 T2=473;
12 sigma=5.67e-8; // Stefan-Boltzmann constant
13 q21_A2=sigma*(T2^4-T1^4)*((L1/L2)+1-(L3/L2))/2;
14 q31_A3=sigma*(T2^4-T1^4)*((L1/L2)-1+(L3/L2))/2;
15 printf("\nThe heat transferred from A2 to A1 is %.2e
   W/sq.m",q21_A2);
16 printf("\nThe heat transferred from A3 to A1 is %.2e
   W/sq.m",q31_A3);

```

Scilab code Exa 12.5 Crossed String method

```

1  clc;
2  clear;
3  printf("\t\t\tChapter12_example5\n\n\n");
4  // Determination of the heat exchanged between the
   two plates
5  // The view factor can be found with the crossed-
   string method
6  // from figure 12.13(b)
7  ac=1;
8  bd=1;

```

```

9  ad=(9+1)^0.5;
10 bc=ad;
11  crossed_strings=ad+bc;
12  uncrossed_strings=ac+bd;
13  L1_F12=(1/2)*(crossed_strings-uncrossed_strings);
14  printf("\nThe Product L1F12=%0.2f ft",L1_F12);
15  L1=3;
16  F12=L1_F12/L1;
17  printf("\nThe view factor F12=%0.2f",F12);
18  sigma=5.67e-8; // Stefan-Boltzmann constant
19  T1=560;
20  T2=460;
21  q12_A1=sigma*(T1^4-T2^4)*F12;
22  printf("\nThe heat transfer rate is %0.2e W/sq.m",
        q12_A1);

```

Scilab code Exa 12.6 Radiation heat transfer within a broiler

```

1  clc;
2  clear;
3  printf("\t\t\tChapter12_example6\n\n\n");
4  // Determination of the heat that must be supplied
   to each of the isothermal surfaces, and also the
   temperature of the insulated surface.
5  // we can apply the equations as follows
6  // q1=sigma*A1*[(T1^4-T2^4)F12+(T1^4-T3^4)F13].....
   (1)
7  // q2=sigma*A2*[(T2^4-T1^4)F21+(T2^4-T3^4)F23].....
   (2)
8  // q3=sigma*A3*[(T3^4-T1^4)F31+(T3^4-T2^4)F32].....
   (3)
9  // given data:
10 T1=1000;

```

```

11 T3=500;
12 q2=0;
13 F12=1/2;
14 F13=1/2;
15 F21=1/2;
16 F23=1/2;
17 F31=1/2;
18 F32=1/2;
19 T2=[(T1^4+T3^4)/2]^(1/4); // using equation (2)
20 printf("\nThe temperature T2=%.1f degree R",T2);
21 sigma=0.1714e-8; // Stefan-Boltzmann constant
22 q1_A1=sigma*[(T1^4-T2^4)*F12+(T1^4-T3^4)*F13]; //
    using equation (1)
23 printf("\nThe heat flux through area A1 is %d BTU/(
    hr.sq.ft)",q1_A1);
24 q3_A3=sigma*[(T3^4-T1^4)*F31+(T3^4-T2^4)*F32]; //
    using equation (3)
25 printf("\nThe heat flux through area A3 is %d BTU/(
    hr.sq.ft)",q3_A3);
26 printf("\nThe results are logical in that the heat
    entering the system (the oven itself) must equal
    that which leaves under steady-state conditions."
    );

```

Scilab code Exa 12.7 Radiation heat transfer within an enclosure of diffuse gray surfaces

```

1 clc;
2 clear;
3 printf("\t\t\tChapter12_example7\n\n\n");
4 // Determination of the heat lost by the oven
    through its top surface.
5 // all energy leaving A1 is intercepted by A2 and

```

```

        vice versa
6  F12=1;
7  F21=1;
8  F11=0; // the surfaces are flat
9  F22=0;
10 emissivity1=0.94; // for oxidized steel from
    appendix table E1
11 emissivity2=0.94
12 T1=533;
13 T2=323;
14 sigma=5.67e-8; // Stefan-Boltzmann constant
15 q1=(sigma*(T1^4-T2^4))/((1/emissivity1)+(1/
    emissivity2)-1);
16 printf("\nThe heat lost through bottom surface is %d
    W/sq.m",q1);
17 q2=-q1;
18 printf("\nThe heat lost through top surface is %d W/
    sq.m",q2);

```

Scilab code Exa 12.8 Radiation heat transfer within an enclosure of black surfaces and diffuse gray surfaces

```

1  clc;
2  clear;
3  printf("\t\t\tChapter12_example8\n\n\n");
4  // Determination of the net heat exchanged between
    the dish and the surroundings by radiation at the
    instant the dish is removed from the oven.
    Perform the calculations (a) if the dish and
    surroundings behave like black bodies, and again
    (b) if the dish has an emissivity of 0.82 and the
    surroundings have an emissivity of 0.93.
5  D=12/12; // diameter in ft

```

```

6 L=6/12; // length in ft
7 A=2*%pi*D^2/4+%pi*D*L;
8 printf("\nThe Surface area is %.2f sq.ft",A);
9 printf("\n\t\t\t\tSolution to part (a)\n");
10 F12=1; // the view factor between the dish and the
    surroundings is unity
11 T1=810;
12 T2=530;
13 sigma=0.1714e-8; // Stefan-Boltzmann constant
14 q1=sigma*A*(T1^4-T2^4)*F12;
15 printf("\nThe heat exchanged between the dish and
    the surroundings is %d BTU/hr",q1);
16 printf("\n\t\t\t\tSolution to part (b)\n");
17 // For gray-surface behavior, we can apply the
    following Equation
18 //  $q_1/(A_1e_1) - [F_{11}*(q_1/A_1)*(1-e_1)/e_1 + F_{12}*(q_2/A_2)*(1-e_2)/e_2] = \sigma*T_1^4 - (F_{11}*\sigma*T_1^4 + F_{12}*\sigma*T_2^4)$  ... equation (1)
19 F11=0;
20 e1=0.82;
21 e2=0.93;
22 // putting  $q_2/A_2=0$  in equation (1) as  $A_2$  tends to
    infinity
23 q1_ = A*e1*[sigma*T1^4-F12*sigma*T2^4];
24 printf("\nThe heat exchanged between the dish and
    the surroundings for the second case is %d BTU/hr
    ",q1_);

```
