Scilab Textbook Companion for Fluid Mechanics- Fundamentals And Applications
by Yunus A. Cengel, John M. Cimbala[1]

Created by
Keerthi Vasan M
Bachelor of technology
Chemical Engineering
Sastra Deemed To Be University
Cross-Checked by
Scilab TBC Team

February 17, 2020

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

Book Description

Title: Fluid Mechanics- Fundamentals And Applications

Author: Yunus A. Cengel, John M. Cimbala

Publisher: Mcgraw Hill & New York

Edition: 1

Year: 2006

ISBN: 0-07-247236-7

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

List of Figures

Chapter 1

INTRODUCTION

Scilab code Exa 1.3 Obtaining Formulas from Unit Considerations

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                //To clear the console screen
clear;              //To clear all the existing variables in the memory


//Given data
rho=850           //rho is the density of the oil in 'kg/m3'
V=2               //V is the volume of the tank in 'm3'

//Calulation
m=rho*V           //m is the mass of oil in the tank in 'kg'

//Display of results
printf("Mass of oil in the tank is %d kg.",m)
```

Scilab code Exa 1.4 `The Weight of One Pound Mass`

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                //To clear the console screen
clear;              //To clear all the existing
    variables in the memory


//Given data
m=1                 //m is mass in 'lbm'


//Assumption
g=32.174            //g is acceleration due to gravity
    in 'ft/s2'


//Calculation
W=m*g               //W is weight in '(lbm ft)/s2'
W=W/g               //Conversion from '(lbm ft)/s2' to '
    lbf'


//Display of result
printf("Weight is %.2f lbf.",W)
```

check Appendix AP 46 for dependency:

`fsolve1.sci`

Scilab code Exa 1.5 `Solving a System of Equations with EES`

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
```

```
clc;                                //To clear the console screen
clear;                              //To clear all the existing variables in the memory
exec('.\fsolve1.sci');
//Replace '.' present inside the 'exec('')' with the path to the folder location where the dependency fsolve1.sci file is saved.


//Given data
Difference=4                        //'Difference' is the difference between the numbers
Sum=20


//Calculation
Number0=[1 1]                       //Number0 is the matrix containing guess value for the actual numbers
Number=fsolve(Number0,fsolve1)      //Number is the matrix consisting 'x' and 'y'.


//Display of result
mprintf('\nNumbers are x=%d and y=%d.',Number(1),Number(2))
```

Scilab code Exa 1.6 Significant Digits and Volume Flow Rate

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                           //To clear the console screen
clear;                         //To clear all the
```

```
existing variables in the memory


//Given data
V1=0                          //V1 is the initial volume in 'gal'
V2=1.1                        //V2 is the final volume in 'gal'
delta_T=45.62                 //delta_T is the change in time in 's'


//Calculation
delta_V=V2-V1                 //delta_V is the change in volume in 'gal'
V_dot=delta_V/delta_T         //V_dot is the volume flow rate in 'gal/s'
V_dot=V_dot*3.785*60/1000     //Conversion 'gal/s' to 'm3/min'


//Display of result
printf("Volume flow rate is %f m3/min.",V_dot)
//The answers vary due to round off error
```

Chapter 2

PROPERTIES OF FLUIDS

Scilab code Exa 2.1 Density and Specific Gravity and Mass of Air in a Room

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                        //To clear the console
    screen
clear;                      //To clear all the existing
    variables in the memory


//Given data
T=25                        //T is temperature of the
    room in ' C '
P=100                       //P is pressure of the room
    in 'kPa'
L=4                         //L is the length of the
    room in 'm'
B=5                         //B is the breath of the
    room in 'm'
W=6                         //W is the width of the room
    in 'm'


```

```
//Assumption
R=0.287                    //R is the universal gas constant in '(kPa m3)/(kg K)'
rho_H2O=1000               //rho_H2O is the density of water in 'kg/m3'


//Calculation
rho=P/(R*(273+T))          //rho is air density in 'kg/m3'
SG=rho/rho_H2O             //SG is the dimensional specific gravity of air
V=L*B*W                    //V is the volume of the air in 'm3'
m=rho*V                    //m is the mass of the air in 'kg'


//Display of result
printf("\nDensity of the air in the room is %.2f kg/m3.\nSpecific gravity of the air is %.5f.\nMass of air in the room is %.d kg.",rho,SG,m)
```

Scilab code Exa 2.3 Variation of Density With Temperature and Pressure

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                       //To clear the console screen
clear;                     //To clear all the existing variables in the memory


//Given data
alpha=4.8E-5               //alpha is isothermal
```

```
compressibility of water in 'atm-1'
T1=20                  //T1 is initial temperature in ' C '
P1=1                   //P1 is initial pressure in 'atm'


//Assumption
rho1=998               //rho1 is density of the water at T1 and P1 in 'kg/m3'
Beta=0.337E-3          //Beta is volume expansion co-efficient in 'K-1'


//Calculation
//Part (a)
T2=50                  //T2 is final temperature ' C '
P2=1                   //P2 is final pressure in 'atm'
delta_T=T2-T1          //delta_T is change in temperature in ' C '
delta_P=P2-P1          //delta_P is change in pressure in 'atm'
delta_rho=rho1*((alpha*delta_P)-(Beta*delta_T))//delta_rho is change in density in 'kg/m3'
rho2=rho1+delta_rho    //rho2 is final density in 'kg/m3'


//Display of Part (a) result
printf("\n(a) Density when heated to %d C and at a constant pressure of %d atm is %.1f kg/m3.",T2,P2,rho2)
//The answers vary due to round off error


//Part (b)
```

```
T3=20                    //T3 is final temperature ' C '
P3=100                   //P3 is final pressure in 'atm'
delta_T=T3-T1            //delta_T is change in temperature in ' C '
delta_P=P3-P1            //delta_P is change in pressure in 'atm'
delta_rho=rho1*((alpha*delta_P)-(Beta*delta_T))//delta_rho is change in density in 'kg/m3'
rho2=rho1+delta_rho      //rho2 is final density in 'kg/m3'
//symbol 'rho2' is repeated in Part (b). It is already used in Part (a) of calculation.


//Display of Part (b) result
printf("\n(b) Density when compressed to %d atm and at a constant temperature of %d C is %.1f kg/m3.",P3,T3,rho2)
```

Scilab code Exa 2.4 Determining the Viscosity of a Fluid

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                  //To clear the console screen
clear;                //To clear all the existing variables in the memory


//Given data
T=1.8                 //T is torque in 'N m'
l=0.15                //l is the gap between the two cylinders in 'cm'
D=12                  //D is diameter of the inner
```

```
cylinder in 'cm'
L=40                //L is length of the concentric cylinders in 'cm'
n=300               //n is rotational speed of inner cylinder in 'rpm'


//Unit conversion
L=L/100             //Conversion from 'cm' to 'm'
D=D/100             //Conversion from 'cm' to 'm'
l=l/100             //Conversion from 'cm' to 'm'
n=n/60              //Conversion from 'rpm' to 'rps'


//Calculation
R=D/2                            //R is radius of the inner cylinder in 'm'
Mu=T*l/(4*((%pi)^2)*(R^3)*n*L)   //Mu is viscosity of the fluid in '(N s)/m2'


//Display of result
printf("Viscosity of the fluid is %.3f (N s)/m2.",Mu)
```

Scilab code Exa 2.5 The Capiliary Rise of Water in a Tube

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                    //To clear the console screen
clear;                  //To clear all the existing variables in the memory


```

```
//Given data
D=0.6                        //D is diameter of glass tube in 'mm'
T=20                         //T is temperature of water in ' c '


//Assumption
Phi=0                        //Phi is contact angle of water with glass in ' (degree)'
rho=1000                     //rho is density of water in 'kg/m3'
Sigma_s=0.073                //Sigma_s is surface tension of water in 'N/m'
g=9.81                       //g is the acceleration due to gravity in 'm/s2'


//Unit conversion
D=D/1000                     //Conversion from 'mm' to 'm'
Phi=Phi*%pi/180              //Conversion from ' (degree)' to 'radian'


//Calculation
R=D/2                                  //R is radius of glass tube in 'm'
h=2*Sigma_s*cos(Phi)/(rho*g*R)         //h is the capillary rise in 'm'
h=h*100                                //Conversion from 'm' to 'cm'


//Display of result
printf("The capillary rise of water is %.1f cm.",h)
```

Chapter 3

PRESSURE AND FLUID STATICS

Scilab code Exa 3.1 Absolute Pressure of a Vacuum Chamber

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                        //To clear the console
   screen
clear;                      //To clear all the existing
   variables in the memory


//Given data
P_atm=14.5                  //P_atm is atmospheric
   pressure in 'psi'
P_vac=5.8                   //P_vac is vacuum pressure
   in 'psi'


//Calculation
P_abs=P_atm-P_vac           //P_abs is absolute pressure
    in 'psi'
//Absolute Pressure + Vacuum Pressure=Atmospheric
```

```
Pressure


//Display of result
mprintf("Absolute pressure (P_abs) is %.1f psi.",P_abs)
```

Scilab code Exa 3.2 `Measuring Pressure with a Manometer`

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                        //To clear the console screen
clear;                      //To clear all the existing variables in the memory


//Given data
SG=0.85                     //SG is specific gravity of the fluid
h=55                        //h is the manometer column height in 'cm'
P_atm=96                    //P_atm is atmospheric pressure in 'kPa'


//Unit conversion
h=h/100                     //Conversion form 'cm' to 'm'


//Assumption
g=9.81                      //g is acceleration due to gravity in 'm/s2'
rho_H2O=1000                //rho_H2O is water density
```

```
    in 'kg/m3'


//Calculation
rho=SG*rho_H2O            //rho is density of the manometer fluid in 'kg/m3'
P=P_atm+(rho*h*g/1000)    //P is absolute pressure within the tank in 'kPa'
//Division by 1000 on the second term of above equation R.H.S is to convert 'Pa' present in the second term to 'kPa'


//Display of result
mprintf("The absolute pressure within the tank is %.1f kPa.",P)
```

Scilab code Exa 3.3 Measuring Pressure with a Multifluid Manometer

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                        //To clear the console screen
clear;                      //To clear all the existing variables in the memory


//Given data
h1=0.1                      //h1 is height of oil in 'm'
h2=0.2                      //h2 is height of water in 'm'
h3=0.35                     //h3 is height of mercury in 'm'
rho_water=1000              //rho_water is water density in 'kg/m3'
```

```
rho_oil=850                //rho_oil is oil density in
    'kg/m3'
rho_mercury=13600          //rho_mercury is mercury
    density in 'kg/m3'
P_atm=85.6                 //P_atm is atmospheric
    pressure in 'kPa'


//Assumption
g=9.81                     //g is acceleration due to
    gravity in 'm/s2'


//Calculation
P1=P_atm+(((g*rho_mercury*h3)-(g*rho_oil*h2)-(g*
    rho_water*h1))/1000)//P1 is air pressure in the
    tank in 'kPa'
//Division by 1000 on the second term of above
    equation is to convert 'Pa' present in the second
     term to 'kPa'


//Display of result
mprintf('Air pressure in the tank is %.1f kPa.',P1)
//The answer vary due to round off error
```

Scilab code Exa 3.4 Analysing a Multifluid Manometer with EES

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                       //To clear the console
    screen
clear;                     //To clear all the existing
    variables in the memory

```

```
//Given data
h1=0.1                    //h1 is height of oil in 'm'
h2=0.2                    //h2 is height of water in 'm'
h3=0.35                   //h3 is height of mercury in 'm'
rho_water=1000            //rho_water is water density in 'kg/m3'
rho_oil=850               //rho_oil is oil density in 'kg/m3'
rho_mercury=13600         //rho_mercury is mercury density in 'kg/m3'
rho_seawater=1030         //rho_seawater is seawater density in 'kg/m3'
P_atm=85.6                //P_atm is atmospheric pressure in 'kPa'


//Unit conversion
P_atm=P_atm*1000          //Conversion from 'kPa' to 'Pa'


//Assumption
g=9.81                    //g is acceleration due to gravity in 'm/s2'


//Calculation
P1=P_atm+(((g*rho_mercury*h3)-(g*rho_oil*h2)-(g*rho_water*h1)))      //P1 is air pressure in the tank in 'Pa'
new_h3=((P1-P_atm)+(g*rho_oil*h2)+(g*rho_water*h1))/(g*rho_seawater)//new_h3 is differential fluid height in 'm'
P1=P1/1000                //Conversion from 'Pa' to 'kPa'
```

```


//Display of result
mprintf('\nAir pressure in the tank is %.1f kPa.',P1)
//The answer vary due to round off error
mprintf('\nDifferential fluid height h3 is %.2f m.',new_h3)
```

Scilab code Exa 3.5 Measuring Atmospheric Pressure with a Barometer

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                //To clear the console screen
clear;              //To clear all the existing variables in the memory


//Given data
g=9.81              //g is acceleration due to gravity in 'm/s2'
h=740               //h is barometric reading in 'mm Hg'
rho=13570           //rho is mercury density in 'kg/m3'


//Unit conversion
h=h/1000            //Conversion form 'mm' to 'm'


//Calculation
P_atm=rho*g*h       //P_atm is atmospheric pressure in 'Pa'
P_atm=P_atm/1000//Conversion from 'Pa' to 'kPa'


```

```
//Display of result
mprintf('The atmospheric pressure is %.1f kPa.',
    P_atm)
```

Scilab code Exa 3.6 Effect of Piston Weight on Pressure in a Cylinder

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                        //To clear the console
    screen
clear;                      //To clear all the existing
    variables in the memory


//Given data
m=60                        //m is mass of gas in the
    cylinder in 'kg'
g=9.81                      //g is accelaration due to
    gravity in 'm/s2'
A=0.04                      //A is cross sectional area
    in 'm2'
P_atm=0.97                  //P_atm is atmospheric
    pressure in 'bar'


//Calculation
W=m*g                       //W is weight of gas in
    cylinder in 'N'
P=P_atm+(W/(A*(10^5)))      //P is pressure inside the
    cylinder in 'bar'
//Division by 10^5 on second term of above equation
    R.H.S is to convert the 'Pa' into 'bar'


//Display of result
```

```
mprintf('Pressure inside the cylinder is %.2f bars.',P)
```

Scilab code Exa 3.7 Hydrostatic Pressure in a Solar Pond with Variable Density

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                //To clear the console screen
clear;              //To clear all the existing variables in the memory


//Given data
rho=1040            //rho is density of surface water in 'kg/m3'
H=4                 //H is thickness of the gradient zone in 'm'
h1=0.8              //h1 is surface zone thickness in 'm'


//Assumption
g=9.81              //g is acceleration due to gravity in 'm/s2'


//Calculation
P1=rho*g*h1         //P1 is gage pressure at the bottom of the surface zone in 'Pa'
P=P1+(rho*g*4*H*asinh(tan((%pi*H)/(4*H)))/%pi)//P is gage pressure at the bottom of the gradient zone in 'Pa'
P=P/1000            //Conversion from 'Pa' to 'kPa'


```

```
//Display of result
mprintf('\nGage pressure at the bottom of the gradient zone is %.1f kPa (gage).',P)
```

Scilab code Exa 3.8 Hydrostatic Force Acting on the Door of a Submerged Car

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                    //To clear the console screen
clear;                  //To clear all the existing variables in the memory


//Given data
s=8                     //s is distance between top edge of the door and water surface in 'm'
b=1.2                   //b is height of the door in 'm'
W=1                     //W is width of the door in 'm'


//Assumption
rho=1000                //rho is density of water in 'kg/m3'
g=9.81                  //g is the acceleration due to gravity in 'm/s2'


//Calculation
P_ave=rho*g*(s+(b/2))/1000      //P_ave is average pressure on the door in 'kN/m2'
A=b*W                           //A is area of door in 'm2'
F_R=P_ave*A                     //F_R is resultant hydrostatic force on the door in 'kN'
y_P=s+(b/2)+(b^2/(12*(s+(b/2))))//y_P is distance of
```

```
    pressure center from the lake surface in 'm'


//Display of result
mprintf('\nThe resultant hydrostatic on the door is
   %.1f kN.\nPressure center distance from the
   surface of the lake is %.2f m.',F_R,y_P)
//The answer vary due to round off error
```

Scilab code Exa 3.9 `A Gravity Controlled Cylindrical Gate`

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                                          //To clear
   the console screen
clear;                                        //To clear
   all the existing variables in the memory


//Given data
R=0.8                                         //R is
   radius of long solid cylinder in 'm'
s=4.2                                         //m
h_bottom=5                                    //h_bottom
   is water level in 'm'


//Assumption
rho=1000                                      //rho is
   density of water in 'kg/m3'
L=1                                           //L is
   length of cylinder in 'm'
g=9.81                                        //g is the
   acceleration due to gravity in 'm/s2'

```

```
//Part (a)
//Calculation
A=R*L                                    //Area is area of cylinder in 'm2'
F_H=rho*g*(s+(R/2))*A/1000               //F_H is horizontal force on vertical surface in 'kN'
F_y=rho*g*h_bottom*A/1000                //F_y is vertical force on horizontal surface in 'kN'
W=rho*g*((R^2)-(%pi*R^2/4))*L/1000       //W is weight of fluid block in 'kN'
//Division by 1000 on above three equation is to convert 'N' to 'kN'
F_V=F_y-W                                //F_V is net upward vertical force in 'kN'
F_R=sqrt((F_H^2)+(F_V^2))                //F_R is magnitude of the hydrostatic force in 'kN'
theta=atan(F_V/F_H)                      //theta is direction of the hydrostatic force in 'radian'
theta=theta*180/%pi                      //Conversion of theta from 'radian' to ' (degree)'


//Display of result
mprintf('\n\n(a) Hydrostatic force acting on the cylinder is %.1f kN.\n    Direction of the force is %.1f  .',F_R,theta)


//Part (b)
//Calculation
theta=theta*%pi/180                      //Conversion of theta from ' (degree)' to 'radian'
W_cyl=F_R*sin(theta)                     //W_cyl is weight of the cylinder per lenght in 'kN'


//Display of result
```

```
mprintf('\n\n(b) The weight of the cylinder per meter length of the cylinder is %.1f kN.',W_cyl)
```

Scilab code Exa 3.10 Measuring Specific Gravity by a Hydrometer

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                        //To clear the console screen
clear;                      //To clear all the existing variables in the memory


//Given data
D=1                         //D is diameter of the hydrometer in 'cm'
L=20                        //L is length of the hydrometer in 'cm'
h_sub=10                    //h_sub is lead height in hydrometer in 'cm'


//Unit conversion
D=D/100                     //Conversion from 'cm' to 'm'
L=L/100                     //Conversion from 'cm' to 'm'
h_sub=h_sub/100             //Conversion from 'cm' to 'm'


//Assumption
rho_W=1000                  //rho_W is density of water in 'kg/m3'

```

```

//Calculation
R=D/2                   //R is radius of hydrometer in 'm'
V_sub=%pi*(R^2)*h_sub   //V_sub is volume of lead submerged in 'm3'
m=rho_W*V_sub           //m is required mass of lead in 'kg'


//Diplay of result
mprintf("Mass of lead is %.5f kg.",m)
```

Scilab code Exa 3.11 Weight Loss of an Object in Seawater

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                    //To clear the console screen
clear;                  //To clear all the existing variables in the memory


//Given data
rho_f=1025              //rho_f is density of sea water in kg/m3
rho_concrete=2300       //rho_concrete is concrete block density in 'kg/m3'
L=0.4                   //L is length of the block in 'm'
B=0.4                   //B is breath of the block in 'm'
W=3                     //W is width of the block in 'm'

```

```

//Assumption
g=9.81                     //g is the accleration due to gravity in 'm/s2'


//Part (a)
//Calculation
V=L*B*W                    //V is block volume in 'm3'
W=rho_concrete*g*V         //W is weight of the block in 'N'
W=W/1000                   //Conversion from 'N' to 'kN'
F_T_air=W                  //F_T_air is the tension in the rope before the block is in water in 'kN'


//Display of result
mprintf('\n(a) Tension in the rope of the crane due to concrete block in air is %.1f kN',F_T_air)


//Part (b)
//Calculation
F_B=rho_f*g*V              //F_B is the upward buoyancy force in 'N'
F_B=F_B/1000               //Conversion from 'N' to 'kN'
F_T_water=W-F_B            //F_T_water is the tension in the rope after the block is in water in 'kN'


//Display of result
printf('\n(b) Tension in the rope of the crane due to concrete block in water is %.1f kN',F_T_water)
```

Scilab code Exa 3.12 Overflow from a Water Tank During Acceleration

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                        //To clear the console
    screen
clear;                      //To clear all the existing
    variables in the memory


//Given data
h=80                        //h is height of fish tank
    in 'cm'
b1=2                        //m
b2=0.6                      //m
V1=0                        //V1 is initial velocity of
    the truck in 'km/h'
V2=90                       //V2 is final velocity of
    the truck in 'km/h'
t1=0                        //t1 is initial time in 's'
t2=10                       //t2 is final time in 's'


//Assumption
a_z=0                       //a_z is the vertical
    acceleration component of truck in 'm/s2'
g=9.81                      //g is acceleration due to
    gravity in 'm/s2'


//Unit conversion
V1=V1*1000/3600             //Conversion from 'km/h' to
    'm/s'
V2=V2*1000/3600             //Conversion from 'km/h' to
```

```
'm/s'


//Calculation
a_X=(V2-V1)/(t2-t1)                    //a_X is the horizontal acceleration component of truck in 'm/s2'
theta=atan(a_X/(g+a_z))                //theta is angle that free surface makes with the horizontal in 'radian'
theta=theta*180/%pi                    //Conversion of theta from 'radian' to ' (degree)'
delta_z_s1=(b1/2)*tan(theta*%pi/180)//delta_z_s1 is vertical rise incase the long side is aligned parallel to the direction of motion in 'm'
delta_z_s2=(b2/2)*tan(theta*%pi/180)//delta_z_s2 is vertical rise incase the short side is aligned parallel to the direction of motion in 'm'


//Display of result
mprintf('\nHorizontal acceleration component magnitude is %.1f m/s2.\ntheta that free surface make with the horizontal is %.1f .\nVertical rise incase the long side is aligned parallel to the direction of motion is %.1f cm.\nVertical rise incase the short side is aligned parallel to the direction of motion is %.1f cm.',a_X,theta,delta_z_s1*100,delta_z_s2*100)
if(delta_z_s1<delta_z_s2)
    mprintf("\n\nLong side must be aligned parallel to the direction of motion.")
else
    mprintf("\n\nShort side must be aligned parallel to the direction of motion.")
end
```

Scilab code Exa 3.13 Rising of a Liquid During Rotation

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                //To clear the console screen
clear;              //To clear all the existing
   variables in the memory


//Given Data
D=20                //D is diameter of the vertical
   cylindrical container in 'cm'
h=60                //h is height of the vertical
   cylindrical container in 'cm'
h0=50               //h0 is the original height of the
   liquid before rotation in 'cm'
rho=850             //rho is density of liquid in 'kg/m3
   '


//Unit conversion
D=D/100             //Conversion from 'cm' to 'm'
h=h/100             //Conversion from 'cm' to 'm'
h0=h0/100           //Conversion from 'cm' to 'm'


//Assumption
g=9.81              //g is the acceleration due to
   gravity in 'm/s2'


//Calculation
R=D/2                                             //R is
   radius of vertical cylindrical container in 'm'
```

```
Z_s_R=h                                        //Z_s_R is the height at the time of spilling in 'm'
Omega=sqrt((4*g*(Z_s_R-h0))/(R^2))             //Omega is angular velocity of the container in 'rad/s'
n=Omega/(2*%pi)                                //n is rotational speed of the container in 'rps'
n=n*60                                         //Conversion from 'rps' to 'rpm'


//Display of the result
mprintf("Rotational speed at the start of spilling from the edges of the container is %d rpm.",n)
```

Chapter 4

FLUID KINEMATICS

Scilab code Exa 4.2 Acceleration of a Fluid Particle through a Nozzle

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                                            //To clear
   the console screen
clear;                                          //To clear
   all the existing variables in the memory


//Given Data
delta_x=3.90                                  //delta_x is
   length of the nozzle in 'in'
D_inlet=0.420                                 //D_inlet is
   inlet diameter of the nozzle in 'in'
D_outlet=0.182                                //D_outlet is
   outlet diameter of the nozzle in 'in'
V_dot=0.841                                   //V_dot is
   volume flow rate through the hose in 'gal/min'


//Unit conversion
delta_x=delta_x/12                            //Conversion
```

```
from 'in' to 'ft'
D_inlet=D_inlet/12                       //Conversion from 'in' to 'ft'
D_outlet=D_outlet/12                     //Conversion from 'in' to 'ft'
V_dot=V_dot*0.133681/60                  //Conversion form 'gal/min' to 'ft3/s'


//Calculation
u_inlet=4*V_dot/(%pi*D_inlet^2)          //u_inlet is inlet velocity in 'ft/s'
u_outlet=4*V_dot/(%pi*D_outlet^2)        //u_outlet is outlet velocity in 'ft/s'
a_x=(u_outlet^2-u_inlet^2)/(2*delta_x)//a_x is axial acceleration in 'ft/s2'


//Display of result
mprintf('\nInlet speed is %.2f ft/s.\nOutlet speed is %.1f ft/s.\nAxial acceleration is %d ft/s2.',u_inlet,u_outlet,a_x)
//The answers vary due to round off error
```

Chapter 5

MASS AND BERNOULI AND ENERGY EQUATIONS

Scilab code Exa 5.1 Water Flow through a Gardan Hose Nozzle

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                    //To clear the console screen
clear;                  //To clear all the existing
   variables in the memory


//Given data
V=10                    //V is volume of bucket in 'gal'
ID=2                    //ID is inner diameter of the
   hose in 'cm'
d_e=0.8                 //d_e is nozzle exit diameter in
    'cm'
Delta_t=50              //Delta_t is time taken to fill
   the bucket with water in 's'


//Assumption
rho=1000                //rho is density of water in 'kg
```

```
/m3'


//Unit conversion
rho=rho/1000          //Conversion from 'kg/m3' to 'kg/L'
V=V*3.7854            //Conversion from 'gal' to 'L'
ID=ID/100             //Conversion from 'cm' to 'm'
d_e=d_e/100           //Conversion from 'cm' to 'm'


//Part (a)
//Calculation
V_dot=V/Delta_t       //V_dot is volume flow rate of water in 'L/s'
m=rho*V_dot           //m is mass flow rate of water in 'kg/s'


//Display of result
mprintf('\n(a) Volume flow rate of water is %.3f L/s.\n    Mass flow rate of water is %.3f kg/s.',V_dot,m)


//Part (b)
//Calculation
r_e=d_e/2             //r_e is radius of nozzle exit in 'm'
A_e=%pi*(r_e^2)       //A_e is area of nozzle exit in 'm2'
V_dot=V_dot/1000      //Conversion from 'L/s' to 'm3/s'
V_e=V_dot/(A_e)       //V_e is average velocity of water at the nozzle exit in 'm/s'


//Display of result
```

```
mprintf('\n\n(b) Average velocity of water at nozzle exit is %.1f m/s.',V_e)
```

Scilab code Exa 5.2 Discharge of Water from a Tank

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                        //To clear the console screen
clear;                      //To clear all the existing variables in the memory


//Given data
h0=4                        //h0 is initial height of water in the tank in 'ft'
D_tank=3                    //D_tank is diameter of the tank in 'ft'
h2=2                        //h2 is final height of water in the tank in 'ft'
D_jet=0.5                   //D_jet is diameter of the water jet in 'in'


//Unit conversion
h0=h0*0.3048                //Converiosn from 'ft' to 'm'
h2=h2*0.3048                //Converiosn from 'ft' to 'm'
D_tank=D_tank*0.3048        //Converiosn from 'ft' to 'm'
D_jet=D_jet*0.0254          //Converiosn from 'in' to 'm'


```

```
//Assumption
g=9.81                        //g is acceleration due to
    gravity in 'm/s2'


//Calculation
t=((sqrt(h0)-sqrt(h2))/sqrt(g/2))*((D_tank/D_jet)^2)
    /60//t is time taken for the water level to drop
    half of its initial level in 's'
h2=h2/0.3048                  //Conversion from 'ft' from
    'm'


//Display of result
mprintf('\nTime taken for the water level to drop to
     %d ft is %.1f min.',h2,t)
```

Scilab code Exa 5.3 Performance of a Hydraulic Turbine Generator

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                          //To clear the console
    screen
clear;                        //To clear all the existing
    variables in the memory


//Given data
h=50                          //h is depth of the water in
     the lake in 'm'
m=5000                        //m is mass flow rate of
    water in 'kg/s'
W_elect_out=1862              //W_elect_out is electric
    power generated in 'kW'
Eta_generator=0.95            //Eta_generator is
```

```
efficiency of the generator


//Assumption
rho=1000                    //rho is water density in 'kg/m3'
g=9.81                      //g is acceleration due to gravity in 'm/s2'


//Part (a)
//Calculation
MechEnergyChange=g*h                                //m2/s2
MechEnergyChange=MechEnergyChange/1000              //Conversion from 'm2/s2' to 'kJ/kg'
Delta_E_mech_fluid=abs(m*MechEnergyChange)          //Delta_E_mech_fluid is the rate at which mechanical energy is supplied to the turbine in 'kW'
Eta_turbine_gen=W_elect_out/Delta_E_mech_fluid      //Eta_turbine_gen is the overall efficiency


//Display of result
mprintf('\n\n(a) Overall efficiency is %.2f.',Eta_turbine_gen)


//Part (b)
//Calculation
Eta_turbine=Eta_turbine_gen/Eta_generator    //Eta_turbine is the mechanical efficiency of the turbine


//Display of result
mprintf('\n(b) Turbine efficiency is %.2f.',
```

```
Eta_turbine)


//Part (C)
//Calculation
W_shaft_out=Eta_turbine*Delta_E_mech_fluid  //W_shaft_out is the shaft power output in 'kW'


//Display of result
mprintf('\n(c) Shaft power output is %d kW.',W_shaft_out)
//The answers vary due to round off error
```

Scilab code Exa 5.5 Spraying Water into the Air

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                  //To clear the console screen
clear;                //To clear all the existing variables in the memory


//Given data
P1=400                //P1 is pressure water flowing from water hose in 'kPa gage'


//Assumption
rho=1000              //rho is water density in 'kg/m3'
V1=0                  //V1 is velocity inside the hose in 'm/s'
V2=0                  //V2 is velocity at the top of the water trajectory in 'm/s'
```

```
Z1=0                    //Z1 is the height of the hose outlet in 'm'
P2=101325               //P2 is pressure at the top of the water trajectory in 'Pa'
g=9.81                  //g is the acceleration due to gravity in 'm/s2'


//Unit conversion
P1=P1*1000              //Conversion from 'kPa gage' to 'Pa gage'
P1=P1+101325            //Conversion from 'Pa gage' to 'Pa absolute'


//Calculation
Z2=((P1-P2)/(rho*g))+((V1^2-V2^2)/(2*g))+Z1//Z2 is the maximum height that the jet could acheive in 'm'


//Display of result
mprintf('Maximum height that the jet could achieve is %.1f m.',Z2)
```

Scilab code Exa 5.6 `Water Discharge from a Large Tank`

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                    //To clear the console screen
clear;                  //To clear all the existing variables in the memory


//Given data
```

```
Z1=5                            //Z1 is height of water level in the tank in 'm'


//Assumption
rho=1000                        //rho is water density in 'kg/m3'
P1=101325                       //P1 is pressure at free surface of water in 'Pa'
P2=101325                       //P2 is pressure at the outlet in 'Pa'
Z2=0                            //Z2 is height of the outlet in 'm'
V1=0                            //V1 is velocity of water at the water surface in 'm/s'
g=9.81                          //g is the acceleration due to gravity in 'm/s2'


//Calculation
V2=sqrt(2*g*(((P1-P2)/(rho*g))+(V1^2/(2*g))+Z1-Z2))     //V2 is water velocity at the outlet in 'm/s'


//Display of result
mprintf('Water velocity at the outlet is %.1f m/s.',V2)
```

Scilab code Exa 5.7 Siphoning Out Gasoline from a Fuel Tank

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                            //To clear the console screen
clear;                          //To clear all the existing variables in the memory
```

```


//'1' deontes the gas tank, '2' denotes the gas can and '3' denotes the peak of tube as shown in FIGURE 5-20 page number 196
//Given data
rho=750                   //rho is the density of gasoline in 'kg/m3'
D=4                       //D is diameter of the siphon in 'mm'
V=4                       //V is volume of gasoline to be withdrawn in 'L'


//Assumption
P1=101.3                  //kPa
P2=101.3                  //kPa
Z2=0                      //m
Z1=0.75                   //m
Z3=2.75                   //m
V1=0                      //m/s
g=9.81                    //g is the acceleration due to gravity in 'm/s2'


//Unit conversion
P1=P1*1000                //Conversion from 'kPa' to 'Pa'
P2=P2*1000                //Conversion from 'kPa' to 'Pa'
D=D/1000                  //Conversion from 'mm' to 'm'
V=V/1000                  //Comversion from 'L' to 'm3'


//Part (a)
//Calculation
R=D/2                     //R is the radius of siphon in 'm'
A=%pi*R^2                 //A is the area of siphon in 'm2'
```

```
V2=sqrt(2*g*(((P1-P2)/(rho*g))+(V1^2/(2*g))+Z1-Z2))
    //m/s
Q=V2*A                  //Q is the volume flow rate of
    gasoline in 'm3/s'
Delta_t=V/Q             //Delta_t is the time needed to
    siphoning in 's'
V=V*1000                //Conversion from 'L' to 'm3'


//Display of result
mprintf('\n(a) Velocity at the gas can is %.2f m/s.\
   n     Cross sectional A of the tube is %f m2.\n
      V flow rate at the gas can is %f m3/s.\n
    Delta_t needed to siphon %d L is %.1f s.',V2,A,Q,
    V,Delta_t)
//The answer provided in the textbook is wrong


//Part (b)
//Calculation
V3=V2                   //m/s
P3=rho*g*((P2/(rho*g))+((V2^2-V3^2)/(2*g))+(Z2-Z3))
    //Pa
P3=P3/1000              //Conversion from 'Pa' to 'kPa'


//Display of result
mprintf('\n\n(b) Pressure at the position 3 is %.1f
    kPa.',P3)
```

Scilab code Exa 5.8 Velocity Measurement by a Pitot Tube

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;              //To clear the console screen
```

```
clear;            //To clear all the existing variables in the memory


//Let '1' and '2' denotes the point under the piezometer and tip point of the pitot tube as shown in FIGURE 5-41.
//Given data
h1=3              //cm
h2=7              //cm
h3=12             //cm


//Unit conversion
h1=h1/100         //Conversion from 'cm' to 'm'
h2=h2/100         //Conversion from 'cm' to 'm'
h3=h3/100         //Conversion from 'cm' to 'm'


//Assumption
Z1=0                        //m
Z2=0                        //m
rho=1000                    //rho is density of water in 'kg/m3'
V2=0                        //V2 is velocity at the tip of the pitot tube in 'm/s'
g=9.81                      //g is the acceleration due to gravity in 'm/s2'


//Calculation
P1=rho*g*(h1+h2)            //P1 is pressure at the centre of the pipe in 'Pa'
P2=rho*g*(h1+h2+h3)         //P2 is pressure at the tip of the pitot tube in 'Pa'
V1=sqrt(2*g*(((P2-P1)/(rho*g))+(V2^2/(2*g))+(Z2-Z1)))//V1 is velocity at the centre of the pipe in 'm/s'
```

```
//Display of result
mprintf('Velocity at the centre of the pipe is %.2f m/s.',V1)
```

Scilab code Exa 5.9 Rise of Ocean Due to a Hurricane

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                          //To clear the console screen
clear;                        //To clear all the existing variables in the memory


//Let '1', '2', '3', 'A', and 'B' be the same naming notation as shown in FIGURE 5-42
//Given data
h1Hg=30                       //h1Hg is atmospheric pressure at point1 in 'in Hg'
h2Hg=22                       //h2Hg is hurricane atmospheric pressure at the eye of the storm in 'in Hg'
rho_Hg=848                    //rho_Hg is density of mercury in 'lbm/ft3'
rho_atm_air=0.076             //rho_atm_air is density of air at normal conditions in 'lbm/ft3'
rho_sw=64                     //rho_sw is density of sea water in 'lbm/ft3'
V_A=155                       //V_A is wind velocity in 'mph'


//Unit conversion
```

```
h1Hg=h1Hg/12              //Conversion from 'in' to 'ft'
h2Hg=h2Hg/12              //Conversion from 'in' to 'ft'
V_A=V_A*1.46667           //Conversion from 'mph' to 'ft/s'


//Part (a)
//Calculation
h_Hg=h1Hg-h2Hg            //'in of Hg'
h1=rho_Hg*h_Hg/rho_sw     //h1 is pressure difference between points 1 and 3 in terms of seawater column height in 'ft'


//Display of result
mprintf('\n(a) Storm surge at the eye of the hurricane is %.2f ft.',h1)


//Part (b)
//Assumption
H_A=0                     //ft
H_B=0                     //ft
V_B=0                     //V_B is velocity at point B in 'ft/s'
g=32.185                  //g is the acceleration due to gravity in 'm/s2'


//Calculation
h_air=(H_A-H_B)+((V_A^2-V_B^2)/(2*g))     //ft
rho_air=(h2Hg/h1Hg)*rho_atm_air           //rho_air is density of air in the hurricane in 'lbm/ft3'
h_dynamic=(rho_air/rho_sw)*h_air          //h_dynamic is seawater column height in 'ft'
h2=h1+h_dynamic                           //h2 is
```

```
total storm surge at point 2 in 'ft'


//Display of result
mprintf('\n(b) Storm surge at point 2 is %.2f ft.',h2)
```

Scilab code Exa 5.12 `Pumping Power and Frictional Heating in a Pump`

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                        //To clear the console screen
clear;                      //To clear all the existing variables in the memory


//Given data
W_electric=15               //W_electric is electric motor power in 'kW'
Eta_motor=0.90              //Eta_motor is efficiency of the motor
V_dot=50                    //V_dot is water flow rate through the pump in 'L/s'
P1=100                      //P1 is pressure at the inlet of the pump in 'kPa'
P2=300                      //P2 is pressure at the outlet of the pump in 'kPa'


//Unit conversion
V_dot=V_dot/1000            //Conversion from 'L/s' to 'm3/s'


```

```
//Assumption
C=4.18                        //C is the specific heat of the water in 'kJ/(kg  C )'
rho=1000                      //rho is water density in 'kg/m3'
Z1=0                          //Z1 is the elevation of the inlet of the pump in 'm'
Z2=0                          //Z2 is the elevation of the outlet of the pump in 'm'
g=9.81                        //g is the acceleration due to gravity in 'm/s2'


//Part (a)
//Calculation
W_pump_shaft=Eta_motor*W_electric        //W_pump_shaft is the mechanical shaft power in 'kW'
m=rho*V_dot                                           //m is the mass flow rate of water in 'kg/s'
Delta_E_mech_fluid=(m*((P2/rho)+(g*Z2)))-(m*((P1/rho)+(g*Z1)))//Delta_E_mech_fluid is the mechanical energy of the fluid in 'kW'
Eta_pump=Delta_E_mech_fluid/W_pump_shaft//Eta_pump is the efficiency of the pump


//Display of result
mprintf("\n(a) Mechanical efficiency of the pump is %.3f or %.1f Percentage.",Eta_pump,100*Eta_pump)


//Part (b)
//Calculation
E_mech_loss=W_pump_shaft-Delta_E_mech_fluid     //E_mech_loss is the lost mechanical energy in 'kW'
Delta_T=E_mech_loss/m/C                         //Delta_T is the temperature rise of water in ' C '
```

```
//Display of result
mprintf('\n(b) Temperature rise of water as it flow
   through the pump due to mechanical inefficiency
   is %.3f  C .',Delta_T)
```

Scilab code Exa 5.13 Hydroelectric Power Generation from a Dam

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                        //To clear the console
   screen
clear;                      //To clear all the existing
   variables in the memory


//Let '1' and '2' be same numbering notations as
   shown in FIGURE 5-55 page number 211
//Given data
V_dot=100                   //V_dot is volume flow rate
   of water in 'm3/s'
Z1=120                      //m
Z2=0                        //m
h_L=35                      //h_L is total irreversible
   head loss in the piping system in 'm'
Eta_turbine_gen=0.80        //Eta_turbine_gen is overall
    efficiency of turbine-generator


//Assumptiom
rho=1000                    //rho is water density in '
   kg/m3'
P1=101325                   //Pa
P2=101325                   //Pa
```

```
V1=0                    //m/s
V2=0                    //m/s
h_pump=0                //h_pump is head loss due to pump in 'm'
alpha1=1.03             //Assuming flow to be turblent
alpha2=1.03             //Assuming flow to be turblent
g=9.81                  //g is acceleration due to gravity in 'm/s2'


//Calculation
m=rho*V_dot                                      //m is mass flow rate of water through the turbine in 'kg/s'
h_turbine_e=((P1-P2)/(rho*g))+((alpha1*V1^2/(2*g))-(alpha2*V2^2/(2*g)))+(Z1-Z2)+h_pump-h_L//h_turbine_e is extrscted turbine head in 'm'
W_turbine_e=m*g*h_turbine_e                      //W_turbine is turbine power in 'W'
W_electric=Eta_turbine_gen*W_turbine_e           //W_electric is electric power generated in 'W'
W_electric=W_electric/1E6                        //Conversion from 'W' to 'MW'


//Display of result
mprintf('Electric power output is %.1f MW.',W_electric)
```

Scilab code Exa 5.14 Fan Selection for Air Cooling of a Computer

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
```

```
clc;                                //To clear the console screen
clear;                              //To clear all the existing variables in the memory


//Let '1', '2', '3' and '4' be the same naming notations as shown in FIGURE 5-56 page number 212
//Given data
L=12                                //L is length of the computer case in 'cm'
B=40                                //B is breath of the computer case in 'cm'
W=40                                //W is width of the computer case in 'cm'
D=5                                 //D is diameter of hole available to install fan in 'cm'
rho=1.20                            //rho is the air density in 'kg/m3'
Eta_fan_motor=0.30                  //Eta_fan_motor is efficiency of fan motor
Void_Fraction=0.5                   //Void_Fraction is the void fraction of the case
t1=0                                //t1 is the initail time in 's'
t2=1                                //t2 is initial time in 's'


//Unit Conversion
L=L/100                             //Conversion from 'cm' to 'm'
B=B/100                             //Conversion from 'cm' to 'm'
W=W/100                             //Conversion from 'cm' to 'm'
D=D/100                             //Conversion from 'cm' to 'm'
```

```
//Assumption
P1=101325                      //P1 is pressure at
   the inlet section in 'Pa'
P2=101325                      //P2 is pressure at
   the outlet section in 'Pa'
Z1=0                           //Z1 is height of
   the inlet section in 'm'
Z2=0                           //Z2 is height of
   the outlet section in 'm'
V1=0                           //V1 is velocity at
   the inlet section in 'm/s'
alpha1=1.1                     //Assuming the flow
   to be turblent
alpha2=1.1                     //Assuming the flow
   to be turblent
W_turbine=0                    //W_turbine is
   mechanical work output of turbine in 'W'
Z3=0                           //Z3 is height of
   point 3 in 'm'
Z4=0                           //Z4 is height of
   point 4 in 'm'
g=9.81                         //g is acceleration
   due to gravity in 'm/s2'


//Part (a)
//Calculation
R=D/2                                //R is radius of
    the case in 'm'
Total_case_volume=L*B*W              //
   Total_case_volume is volume of case in 'm3'
V=Void_Fraction*Total_case_volume    //V is air
   volume in the case in 'm3'
V_dot=V/(t2-t1)                      //V_dot is
   volume flow rate of air through the case in 'm3/s
   '
```

```
m=rho*V_dot                                  //m is mass flow rate of air through the case in 'kg/s'
A=%pi*R^2                                    //A is the CSA of the case in 'm2'
V2=V_dot/A                                   //V2 is velocity at the outlet section in 'm/s'
W_fan_u=(m*((P2/rho)+(alpha2*V2^2/2)+(g*Z2)))-(m*((P1/rho)+(alpha1*V1^2/2)+(g*Z1)))+W_turbine//W
W_elect=W_fan_u/Eta_fan_motor        //W_elect is required electric power input to the fan in 'W'


//Display of result
mprintf('\n(a) Electric power input to the fan is %.3f W.',W_elect)
//The answers vary due to round off error


//Part (b)
//Calculation
PressureDrop=rho*W_fan_u/m           //PressureDrop is pressure rise across the fan in 'Pa'


//Display of result
mprintf('\n(b) Pressure difference across the fan is %.1f Pa.',PressureDrop)
```

Scilab code Exa 5.15 `Head and Power Loss During Water Pumping`

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                        //To clear the console screen
clear;                      //To clear all the existing
```

```
variables in the memory


//Let '1' and '2' be the same naming notations as shown in FIGURE 5-57 page number 5-15
//Given data
W_pump=20                    //W_pump is the mechanical power provided by the pump in 'kW'
Z1=0                         //Z1 is height of the lower reservoir surface in 'm'
Z2=45                        //Z2 is height of the upper reservoir surface in 'm'
V_dot=0.03                   //V_dot is the water volume flow rate in 'm3/s'


//Assumption
P1=101325                    //P1 is pressure at the surface of the lower reservoir in 'Pa'
P2=101325                    //P2 is pressure at the surface of the upper reservor in 'Pa'
V1=0                         //V1 is velocity of the water at the lower reservoir surface in 'm/s'
V2=0                         //V2 is velocity of the water at the upper reservoir surface in 'm/s'
alpha1=1.1                   //Assuming the flow to be turblent
alpha2=1.1                   //Assuming the flow to be turblent
rho=1000                     //rho is the water density in 'kg/m3'
W_turbine=0                  //W_turbine is mechanical work output of turbine in 'W'
g=9.81                       //g is acceleration due to gravity in 'm/s2'


//Unit conversion
```

```
W_pump=W_pump*1000       //Conversion from 'kW' to 'W'


//Calculation
m=rho*V_dot
    //m is mass flow rate of water through the system in 'kg/s'
E_mech_loss=(m*((P1/rho)+(alpha1*V1^2/2)+(Z1*g))-m
    *((P2/rho)+(alpha2*V2^2/2)+(Z2*g)))+W_pump-
    W_turbine//E_mech_loss is the mechanical power in 'W'
E_mech_loss_piping=E_mech_loss
    //E_mech_loss_piping is the mechanical losses due to friction in piping in 'W'
h_L=E_mech_loss_piping/(m*g)
    //h_L is the irreversible head loss in 'm'
E_mech_loss_piping=E_mech_loss_piping/1000
    //Conversion from 'w' to 'kW'


//Display of result
mprintf('\nLost mechanical power is %.2f kW.\nHead
    loss is %.1f m.',E_mech_loss_piping,h_L)
```

Chapter 6

MOMENTUM ANALYSIS OF FLOW SYSTEMS

Scilab code Exa 6.1 `Momentum Flux Correction Factor for Laminar Pipe Flow`

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                        //To clear the console
   screen
clear;                      //To clear all the existing
   variables in the memory


//Calculation
Beta=integrate('-4*y^2','y',1,0)    //Beta is the
   momentum flux correction factor


//Display of result
mprintf("\nMomentum flux correction factor is %f.",
   Beta)
```

Scilab code Exa 6.2 The Force to Hold a Deflector Elbow in Place

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                        //To clear the console screen
clear;                      //To clear all the existing variables in the memory


//Let '1' and '2' be the same naming notations as shown in FIGURE 6-20 page number 239.
//Given data
m=14                        //m is the mass flow rate of water in 'kg/s'
theta=30                    //theta is bend angle of the reducing elbow in ' (degree)'
A1=113                      //A1 is inlet CSA of the elbow in 'cm2'
A2=7                        //A2 is outlet CSA of the elbow in 'cm2'
Z1=0                        //Z1 is the elevation of the inlet in 'cm'
Z2=30                       //Z2 is the elevation of the outlet in 'cm'


//Unit conversion
A1=A1*10^-4                 //Conversion from 'cm2' to 'm2'
A2=A2*10^-4                 //Conversion from 'cm2' to 'm2'
Z1=Z1/100                   //Conversion from 'cm' to 'm'
Z2=Z2/100                   //Conversion from 'cm' to 'm'
theta=theta*%pi/180         //Conversion from ' (degree)' to 'radian'
```

```
//Assumption
rho=1000                  //rho is the water denisty in 'kg/m3'
P1=101325                 //P1 is pressure at the inlet in 'Pa'
g=9.81                    //g is acceleration due to gravity in 'm/s2'
Beta=1.03                 //Beta is momentum flux correction factor(Assuming flow to be turblent)


//Part (a)
//Calculation
V1=m/(rho*A1)             //V1 is the inlet water velocity in 'm/s'
V2=m/(rho*A2)             //V2 is the outlet water velocity in 'm/s'
P_1_gage=(((P1/(rho*g))+(V2^2/(2*g))+Z2-Z1-(V1^2/(2*g)))*rho*g)//P_1_gage is the pressure at the centre of the inlet of the elbow in 'Pa absolute'
P_1_gage=P_1_gage-101325//Conversion from 'Pa absolute' to 'Pa gage'
P_1_gage=P_1_gage/1000  //Conversion from 'Pa gage' to 'kPa gage'


//Display of result
mprintf('\n(a) The pressure at the centre of the inlet of the elbow is %.1f kPa gage.',P_1_gage)


//Part (b)
//Calculation
P_1_gage=P_1_gage*1000                                    //Conversion from 'kPa' to 'Pa'
```

```
F_Rx=(Beta*m*((V2*cos(theta)-V1)))-(P_1_gage*A1)          //F_Rx is the x-component of the anchoring force of the elbow in 'N'
F_Rz=Beta*m*sin(theta)*V2                                   //F_Rz is the z-component of the anchoring force of the elbow in 'N'


//Display of result
mprintf('\n\n(b) x-component of the anchoring force of the elbow is %d N.\n    z-component of the anchoring force of the elbow is %d N.',F_Rx,F_Rz)
//The answers vary due to round off error
```

Scilab code Exa 6.3 The Force to Hold a Reversing Elbow in Place

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                          //To clear the console screen
clear;                        //To clear all the existing variables in the memory


//Let '1' and '2' be the same naming notations as shown in FIGURE 6-21 page number 240.
//Given data
m=14                          //m is the mass flow rate of water in 'kg/s'
theta=180                     //theta is bend angle of the reducing elbow in ' (degree)'
A1=113                        //A1 is inlet CSA of the elbow in 'cm2'
A2=7                          //A2 is outlet CSA of the
```

```
elbow in 'cm2'
Z1=0                        //Z1 is the elevation of the inlet in 'cm'
Z2=30                       //Z2 is the elevation of the outlet in 'cm'


//Unit conversion
A1=A1*10^-4                 //Conversion from 'cm2' to 'm2'
A2=A2*10^-4                 //Conversion from 'cm2' to 'm2'
Z1=Z1/100                   //Conversion from 'cm' to 'm'
Z2=Z2/100                   //Conversion from 'cm' to 'm'
theta=theta*%pi/180         //Conversion from 'degree' to 'radian'


//Assumption
rho=1000                    //rho is the water denisty in 'kg/m3'
P1=101325                   //P1 is pressure at the inlet in 'Pa'
g=9.81                      //g is acceleration due to gravity in 'm/s2'
Beta=1.03                   //Beta is momentum flux correction factor(Assuming flow to be turblent)
F_Rz=0                      //F_Rz is the z-component of the anchoring force of the elbow in 'N'


//Part (a)
//Calculation
V1=m/(rho*A1)               //V1 is the inlet water velocity in 'm/s'
V2=m/(rho*A2)               //V2 is the outlet water
```

```
velocity in 'm/s'
P_1_gage=(((P1/(rho*g))+(V2^2/(2*g))+Z2-Z1-(V1^2/(2*g)))*rho*g)//P_1_gage is the pressure at the centre of the inlet of the elbow in 'Pa absolute'
P_1_gage=P_1_gage-101325//Conversion from 'Pa absolute' to 'Pa gage'
P_1_gage=P_1_gage/1000  //Conversion from 'Pa gage' to 'kPa gage'


//Display of result
mprintf('\n(a) The pressure at the centre of the inlet of the elbow is %.1f kPa gage.',P_1_gage)


//Part (b)
//Calculation
P_1_gage=P_1_gage*1000                                    //Conversion from 'kPa' to 'Pa'
F_Rx=(Beta*m*((V2*cos(theta)-V1)))-(P_1_gage*A1)         //F_Rx is the x-component of the anchoring force of the elbow in 'N'


//Display of result
mprintf('\n\n(b) x-component of the anchoring force of the elbow is %d N.',F_Rx)
//The answers vary due to round off error
```

Scilab code Exa 6.4 Water Jet Striking a Stationary Plate

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                    //To clear the console screen
```

```
clear;                    //To clear all the existing
   variables in the memory



//Given data
V1=20                     //V1 is velocity at which water
   strikes the vertical plate in 'm/s'
m=10                      //m is mass flow rate of water
   when it strikes the vertical plate in 'kg/s'


//Assumption
Beta=1                    //Beta is momentum flux
   correction factor


//Calculation
F_R=Beta*m*V1           //F_R is the force needed to
   prevent the plate from horizontal movement in 'N'


//Display of result
mprintf('\nForce needed to prevent the plate from
   moving horizontally due to water stream is %d N.'
   ,F_R)
```

Scilab code Exa 6.5 Power Generation and Wind Loading of a Wind Turbine

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                                    //To clear the
   console screen
clear;                                  //To clear all
   the existing variables in the memory
```

```


//Let '1' and '2' be the same naming notations as shown in FIGURE 6-23
//Given data
D=30                                    //D is diameter of wind generater blade in 'ft'
V1=7                                    //V1 is cut in wind speed in 'mph'
V1_new=14                               //V1_new is new cut in wind speed in 'mph'
W_act=0.4                               //W_act is power generated by the turbine in 'kW'
rho1=0.076                              //rho1 is air density in 'lbm/ft3'


//Unit conversion
V1=V1*1.4667                            //Conversion from 'mph' to 'ft/s'
V1_new=V1_new*1.4667                    //Conversion from 'mph' to 'ft/s'
W_act=W_act*1000                        //Conversion from 'kW' to 'W'


//Assumption
betaa=1                                 //betaa is momentum flux correction factor


//Part (a)
//Calculation
A1=%pi*D^2/4                            //A1 is the CSA of blade in 'ft2'
m=rho1*V1*A1                            //m is mass flow rate of air in 'lbm/s'
W_max=m*V1^2/2                          //W_max is the
```

```
maximum power in '(lbm ft2)/s3'
W_max=W_max*0.04214016                  //Conversion from '(lbm ft2)/s3' to 'W'
Eta_wind_turbine=W_act/W_max            //Eta_wind_turbine is the efficiency of turbine generator


//Display of result
mprintf('\n(a) Turbine-generator efficiency is %.3f or %.1f Percentage.',Eta_wind_turbine,100*Eta_wind_turbine)


//Part (b)
//Calculation
Rate=V1_new/V1
V2=V1*sqrt(1-Eta_wind_turbine)          //V2 is exit velocity in 'ft/s'
F_R=(m*(V2-V1))                         //F_R is the force excerted on the mast by the wind in '(lbm ft)/s2'
F_R=F_R/32.2                            //Conversion from '(lbm ft)/s2' to 'lbf'
if(F_R<0)
    F_R=abs(F_R)
end
W_act_new=W_act*(Rate^3)                //W_act_new is the new power generated by the turbine in 'W'
F_R_new=F_R*(Rate^2)                    //F_R_new is the  new force excerted on the mast by the wind in 'lbf'
W_act=W_act/1000                        //Conversion from 'W' to 'kW'
W_act_new=W_act_new/1000                //Conversion from 'W' to 'kW'


```

```
//Display of result
mprintf('\n\n(b) Force excerted on the mast by the
   wind is %.1f lbf.',F_R)
mprintf('\n\nOn increasing the wind velocity by the
   Rate of %.2f,\nPower generated becomes %.1f kW
   from %.1f kW.\nForce excerted on the mast by the
   wind becomes %d lbf from %.1f lbf.',Rate,
   W_act_new,W_act,F_R_new,F_R)
```

Scilab code Exa 6.6 Repositioning of a Satellite

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                        //To clear the console
   screen
clear;                      //To clear all the existing
   variables in the memory



//Given data
m_sat=5000                  //m_sat is the mass of
   satellite in 'kg'
m_f=100                     //m_f is the mass of gas
   discharged in 'kg'
v_f=3000                    //v_f is the velocity at
   which gas is discharged in 'm/s
Delta_t=2                   //Delta_t is the time period
    of discharge in 's'


//Part (a)
//Calculaton
a_sat=m_f*v_f/m_sat/Delta_t//a_sat is the
   acceleration of the satellite in 'm/s2'
```

```
//Display of result
mprintf('\n\n(a) Acceleration of the satelite during this %ds period is %d m/s2.',Delta_t,a_sat)


//Part (b)
//CalculaDelta_ton
Dela_V_sat=a_sat*Delta_t    //Dela_V_sat is the change in velocity of the satellite in 'm/s'


//Display of result
mprintf('\n\n(b) Change in velocity of the satelite during this %ds period is %d m/s.',Delta_t,Dela_V_sat)


//Part (c)
//CalculaDelta_ton
F_sat=(m_f/2)*v_f          //F_sat is the trust exerted on the satellite in 'kN'
F_sat=F_sat/1000           //Conversion from 'N' to 'kN'


//Display of result
mprintf('\n\n(c) The thrust exerted on the satellite is %d kN.',F_sat)
```

Scilab code Exa 6.7 Net Force on a Flange

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
```

```
clc;                                //To clear the console screen
clear;                              //To clear all the existing variables in the memory



//Let '1' and '2' be the same naming notations as shown in FIGURE 6-26 page number 246.
//Given data
V_dot=18.5                          //V_dot is water flow rate through the faucet in 'gal/min'
D=0.780                             //D is inner diameter of the pipe at the location of flange in 'in'
P1_gage=13                          //P1_gage is pressure at the location of flange in 'psi'
W_faucet_water=12.8                 //W_faucet_water is the total weight of the faucet assembly plus the water within it in 'lbf'


//Unit conversion
V_dot=V_dot*0.1337/60               //Conversion from 'gal/min' to 'ft3/s'
D=D/12                              //Conversion from 'in' to 'ft'
P1_gage=P1_gage*12^2                //Conversion from 'psi' to 'lbf/ft2'


//Assumption
Beta=1.03                           //Beta is momentum flux correction factor(Assuming the flow to be turblent)
rho=62.3                            //rho is water density in 'lbm/ft3'

```

```

//Calculation
R=D/2                                   //R is radius of pipe at the location of the flange in 'ft'
A1=%pi*R^2                              //A is the CSA of the pipe at the location of flange in 'ft2'
V1=V_dot/A1                             //V1 is the inflow velocity of water in 'ft/s'
V2=V1                                   //V2 is the outflow velocity of water in 'ft/s'
m=V_dot*rho                             //m is the mass flow rate of water in 'lbm/s'
F_Rx=-(m*V1/32.2)-(P1_gage*A1)   //F_Rx is x-component of the force acting on the flange in 'lbf'
F_Rz=-(m*V2/32.2)+W_faucet_water//F_Rz is z-component of the force acting on the flange in 'lbf'
//From Newton's third law, the force the faucet assembly exerts on the flange is negative of above determined forces
F_Rx=-1*F_Rx                            //lbf
F_Rz=-1*F_Rz                            //lbf


//Display of result
mprintf('\nForce the faucet assembly excerts on the flange is F = %.2f i + %.1f k lbf.',F_Rx,F_Rz)
```

Scilab code Exa 6.8 Bending Moment Acting at the Base of a Water Pipe

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                          //To clear the console screen
```

```
clear;                              //To clear all the existing variables in the memory



//Let '1' and '2' be the same naming notations as shown in FIGURE 6-37 page number 255
//Given data
D=10                                //D is the diameter of the pipe in 'cm'
V2=3                                //V2 is velocity of water in 'm/s'
Mass=12                             //Mass is the mass of the horizontal pipe section when filled with water in 'kg/m'
r1=0.5                              //m
r2=2                                //m


//Unit conversion
D=D/100                             //Conversion from 'cm' to 'm'


//Assumption
g=9.81                              //g is acceleration due to gravity in 'm/s2'
rho=1000                            //rho is water density in 'kg/m3'


//Calculation
R=D/2                               //R is the radius of the pipe in 'm'
A_c=%pi*R^2                         //A_c is the CSA of the pipe in 'm2'
m=rho*V2*A_c                        //m is mass flow rate of water in 'kg/s'
```

```
W=Mass*(2*r1)*g                //W is the weight of the horizontal section of the pipe in 'N'
M_A=(r1*W)-(r2*m*V2)           //M_A is the angular momentum about the point A in 'N m'
L=sqrt((2*r2*m*V2)/W)          //L is length of the horizontal pipe that will cause the moment vanish in 'm'


//Display of result
mprintf('\nAngular M_A around the point A is %.1f N m.\nHorizontal section L required to make the moment at point A zero is %.2f m.',M_A,L)
//The answer provided in the textbook is wrong
```

Scilab code Exa 6.9 Power Generation from a Sprinkler System

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                       //To clear the console screen
clear;                     //To clear all the existing variables in the memory



//Given data
V_dot_total=20        //V_dot_total is rate at which water enters the sprinkler in 'L/s'
n=300                      //n is rotational speed of the sprinkler in 'rpm'
D=1                        //D is diameter of the jet in 'cm'
r=0.6                      //r is distance between axis and centre of each nozzle in 'm'
Arms=4                     //'Arms' is number arms present
```

```
    in the sprinkler


//Unit conversion
V_dot_total=V_dot_total/1000     //Conversion from 'L/s' to 'm3/s'
D=D/100                          //Conversion from 'cm' to 'm'


//Assumption
rho=1000                         //rho is water density in 'kg/m3'


//Calculation
R=D/2                            //R is radius of the jet in 'm'
A_jet=%pi*R^2                    //A_jet is the area covered by a jet in 'm2'
m_total=rho*V_dot_total          //m_total is the total mass flow rate in 'kg/s'
V_dot_nozzle=V_dot_total/Arms    //V_dot_nozzle is volume flow rate in each nozzle in 'm3/s'
V_jet=V_dot_nozzle/A_jet         //V_jet is the average jet exit velocity relative to the nozzle in 'm/s'
Omega=2*%pi*n/60                 //Omega is angular velocity of the nozzle in 'rad/s'
V_nozzle=r*Omega                 //V_nozzle is tangential velocity of the nozzle in 'm/s'
V_r=V_jet-V_nozzle               //V_r is the average velocity of the water jet in 'm/s'
T_shaft=r*m_total*V_r            //T_shaft is torque transmitted through the shaft in 'Nm'
W=Omega*T_shaft                  //W is the power generated in 'W'
W=W/1000                         //Conversion from 'W
```

```
    ' to 'kW'


//Diplay of result
mprintf('\nSprinkler type turbine has the potential
   to produce %.1f kW.',W)
```

Chapter 7

DIMENSIONAL ANALYSIS AND MODELING

check Appendix ?? for dependency:

FIGURE_7_13.jpg

Scilab code Exa 7.4 Extrapolation of Nondimensionalized Data

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                              //To clear the console
   screen
clear;                            //To clear all the
   existing variables in the memory


//Given data
g_earth=9.81                      //g_earth is
   acceleration due to gravity on earth in 'm/s2'
v_initial=21                      //v_initial is the
   initial velocity of the baseball in 'm/s'
theta=5                           //theta is angle made
   with the horizon in ' (degree)'
```

```
Z0=2                         //Z0 is initial height
   above moon surface in 'm'


//Unit conversion
theta=theta*%pi/180          //Conversion from '  (
   degree)' to 'radian'


//Part (a)
//Calculation
W0=v_initial*sin(theta)      //W0 is the vertical
   component of initial speed in 'm/s'
g_moon=g_earth/6             //g_moon is acceleration
    due to gravity on moon in 'm/s2'
Fr_Square=W0^2/(g_moon*Z0)   //DimensionLess Froude
   Number
//Value obtained from the FIGURE 7-13 page number
   275
//Choose the value of t_star that the curve obtained
    Fr^2 value makes when the z_star=0
t_star=2.75                  //This value of t_star
   is for computed Fr^2. Change t_star value
   accordingly if the input variables are changed
t=t_star*Z0/W0               //t is the time taken to
    strike the ground in 's'


//Display of result
mprintf('\n(a) Estimated time to strike the ground
   is %.2f s.',t)


//Part (b)
//Calculation
t_exact=(W0+sqrt((W0^2)+(2*Z0*g_moon)))/g_moon//
   t_exact is the exact time taken strike the ground
    in 's'
```

```


//Display of result
mprintf('\n(b) Exact time to strike the ground is %.2f s.',t_exact)
//The answers vary due to round off error
```

Scilab code Exa 7.5 Similarity between Model and Prototype Cars

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                      //To clear the console screen
clear;                    //To clear all the existing variables in the memory


//Given data
V_p=50                    //V_p is prototype wind tunnel speed in 'mi/h'
T_p=25                    //T_p is air temperature in prototype in ' C '
T_m=5                     //T_m is air temperature in model in ' C '


//Assumption
L_p=1                     //L_p is length of the prototype in 'm'
rho_p=1.184               //rho_p is prototype air density at T_p in 'kg/m3'
Mu_p=1.849E-5             //Mu_p is prototype air viscosity at T_p in 'kg/(m s)'
rho_m=1.269               //rho_m is model air density at T_m in 'kg/m3'
```

```
Mu_m=1.754E-5            //Mu_m is model air
    viscosity at T_m in 'kg/(m s)'


//Calculation
L_m=L_p/5                                    //L_m is
     length of the model in 'm'
V_m=V_p*(Mu_m/Mu_p)*(rho_p/rho_m)*(L_p/L_m) //V_m is
     model wind tunnel speed in 'm/s'


//Display of result
mprintf('\nRequired wind tunnel speed is %d mi/h.',
    V_m)
```

Scilab code Exa 7.6 Prediction of Aerodynamic Drag Force on the Prototype Car

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                    //To clear the console screen
clear;                  //To clear all the existing
    variables in the memory


//Given data
F_D_m=21.2              //F_D_m is drag force on the
    model in 'lbf'
V_p=50                  //V_p is prototype wind tunnel
    speed in 'mi/h'
V_m=221                 //V_m is model wind tunnel
    speed in 'mi/h'
T_p=25                  //T_p is air temperature in
    prototype in ' C '
T_m=5                   //T_m is air temperature in
    model in ' C '
```

```


//Assumption
L_p=1                    //L_p is length of the prototype in 'm'
rho_p=1.184              //rho_p is prototype air density at T_p  in 'kg/m3'
Mu_p=1.849E-5            //Mu_p is prototype air viscosity at T_p  in 'kg/(m s)'
rho_m=1.269              //rho_m is model air density at T_m in 'kg/m3'
Mu_m=1.754E-5            //Mu_m is model air viscosity at T_m in 'kg/(m s)'


//Calculation
L_m=L_p/5                //L_m is length of the model in 'm'
F_D_p=F_D_m*(rho_p/rho_m)*(V_p/V_m)^2*(L_p/L_m)^2    //F_D_p is drag force on the prototype in 'lbf'


//Display of result
mprintf('\nAerodynamic drag force on the prototype is %.1f lbf.',F_D_p)
```

check Appendix ?? for dependency:

`TABLE_7_7.jpg`

Scilab code Exa 7.10 `Model Truck Wind Tunnel Measurements`

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
```

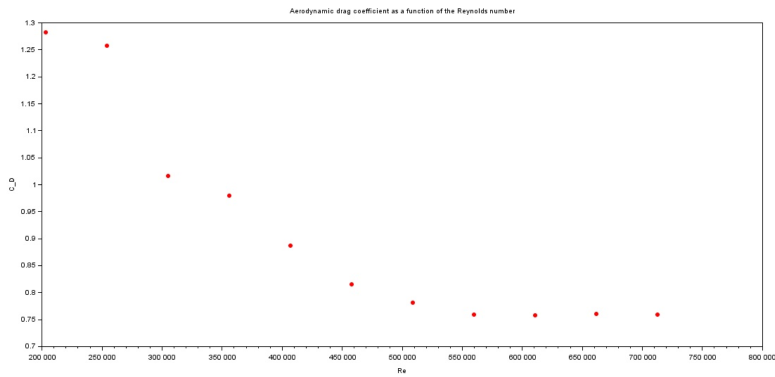


Figure 7.1: Model Truck Wind Tunnel Measurements

```
clc;                          //To clear the console screen
clear;                        //To clear all the existing variables in the memory
clf(0)                        //Clear or reset or reset a figure or a frame uicontrol


//Given data
V_m=[20 25 30 35 40 45 50 55 60 65 70]
                                  //V_m is model wind tunnel speed in 'm/s'
F_D_m=[12.4 19.0 22.1 29.0 34.3 39.9 47.2 55.5 66.0 77.6 89.9] //F_D_m is drag force on the model in 'N'
//Values of V_m and F_D_m are obtained from TABLE 7-7 page number 300.
L_m=0.991                     //L_m is length of the model in 'm'
H_m=0.257                     //H_m is height of the model in 'm'
W_m=0.159                     //W_m is width of the model in 'm'
```

```
T_p=25                  //T_p is air temperature in prototype in ' C '
V_p=26.8                //V_p is prototype wind tunnel speed in 'm/s'


//Assumption
rho_p=1.184             //rho_p is prototype air density at T_p in 'kg/m3'
Mu_p=1.849E-5           //Mu_p is prototype air viscosity at T_p in 'kg/(m s)'


//Calculation
W_p=16*W_m              //W_p is width of the prototype in 'm'
A_p=16^2*W_m*H_m        //A_p is the prototype area in 'm2'
count=length(V_m)       //count is number of data in V_m matrix
A_m=W_m*H_m             //A_m is the model area in 'm2'
for i=1:count
    C_D(i)=F_D_m(i)*2/(rho_p*V_m(i)^2*A_m)    //DimensionLess drag co-efficient
    Re_m(i)=rho_p*V_m(i)*W_m/Mu_p             //Re_m is reynolds number of the model
end
Re_p=rho_p*W_p*V_p/Mu_p                       //Re_p is reynolds number of the prototype
F_D_p=0.5*rho_p*V_p^2*A_p*C_D(count)          //F_D_p is drag force on the prototype in 'N'


//Display of result
if Re_p==Re_m(1) then
    mprintf('\nDynamic similarity has been acheived.!')
```

```
else
    mprintf('\nDynamic similarity has not been acheived.!')
end
plot(Re_m,C_D,'r.')
xlabel('Re')
ylabel('C_D')
title('Aerodynamic drag coefficient as a function of the Reynolds number')
mprintf('\nPredicted aerodynamic drag on the prototype is %d N.',F_D_p)
//The answers vary due to round off error
```

Scilab code Exa 7.11 `Model Lock and RIver`

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                            //To clear the console screen
clear;                          //To clear all the existing variables in the memory


//Given data
L_m_by_L_p=1/100                //L_m_by_L_p is the length scale factor


//Assumption
Nu_p=1.002E-6                   //Nu_p is prototype kinematic viscosity in 'm2/s'


//Calculation
Nu_m=Nu_p*(L_m_by_L_p)^1.5  //Nu_m is model
```

```
kinematic vicosity in 'm2/s'
Nu_m=Nu_m*10^9                //Modification for display of result purpose.


//Display of result
mprintf('\nLiquid with kinematic viscosity of %.2f E-9 m2/s is preffered.\nBut no liquid fall under this kinematic vicosity range. So water can be used.',Nu_m)
//The answers vary due to round off error
```

Chapter 8

FLOW IN PIPES

Scilab code Exa 8.1 Flow Rates in Horizontal and Inclined Pipes

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                                //To clear the
    console screen
clear;                              //To clear all the
    existing variables in the memory


//Given data
rho=888                             //rho is density of
    the oil in 'kg/m3'
Mu=0.8                              //Mu is viscosity of
     the oil in 'kg/(m s)'
D=5                                 //D is diameter of
    the pipe in 'cm'
L=40                                //L is length of the
     pipe in 'm'
P1=745                              //P1 is pressure at
    the pipe inlet in 'kPa'
P2=97                               //P2 is presure at
    the pipe outlet in 'kPa'
```

```
theta1=0                                //theta1 is angle that pipe makes with horizontal in ' (degree)'
theta2=15                               //theta2 is angle that pipe makes with horizontal in ' (degree)'
theta3=-15                              //theta3 is angle that pipe makes with horizontal in ' (degree)'


//Unit conversion
P1=P1*1000                              //Conversion from 'kPa' to 'Pa'
P2=P2*1000                              //Conversion from 'kPa' to 'Pa'
D=D/100                                 //Conversion from 'cm' to 'm'
theta1=theta1*%pi/180                   //Conversion from 'degree' to 'radian'
theta2=theta2*%pi/180                   //Conversion from 'degree' to 'radian'
theta3=theta3*%pi/180                   //Conversion from 'degree' to 'radian'


//Assumption
g=9.81                                  //g is acceleration due to gravity in 'm/s2'


//Part (a)
//Calculation
Delta_P=P1-P2                           //Delta_P is the pressure drop across the pipe in 'Pa'
R=D/2                                   //R is radius of the pipe in 'm'
A_c=%pi*R^2                             //A_c is CSA of the pipe in 'm2'
V_dot_1=(Delta_P-(rho*g*L*sin(theta1)))*%pi*D^4/(128*Mu*L)//V_dot_1 is volume flow rate of
```

```
water in 'm3/s'
V1=V_dot_1/A_c                          //V1 is the average fluid velocity in 'm/s'
Re1=rho*V1*D/Mu                         //Re1 is reynolds number in the pipe


//Display of result
mprintf('\n(a) The flow rate of oil through the pipe is %.5f m3/s.',V_dot_1)
if(Re1<2300)
    mprintf('\n    Flow is laminar.')
else
    if Re1>4000
        mprintf('\n    Flow is turblent.')
    end
end


//Part (b)
//Calculation
V_dot_2=(Delta_P-(rho*g*L*sin(theta2)))*%pi*D^4/(128*Mu*L)   //V_dot_2 is volume flow rate of water in 'm3/s'
V2=V_dot_2/A_c                                                      //V2 is the average fluid velocity in 'm/s'
Re2=rho*V2*D/Mu                                                     //Re2 is reynolds number in the pipe


//Display of result
mprintf('\n\n(b) The flow rate of oil through the pipe is %.5f m3/s.',V_dot_2)
if(Re2<2300)
    mprintf('\n    Flow is laminar.')
else
```

```
    if Re2>4000
        mprintf('\n     Flow is turblent.')
    end
end


//Part (c)
//Calculation
V_dot_3=(Delta_P-(rho*g*L*sin(theta3)))*%pi*D
    ^4/(128*Mu*L)   //V_dot_2 is volume flow rate of
    water in 'm3/s'
V3=V_dot_3/A_c
                                                        //V3
     is the average fluid velocity in 'm/s'
Re3=rho*V3*D/Mu
                                                        //Re3
     is reynolds number in the pipe


//Display of result
mprintf('\n\n(c) The flow rate of oil through the
    pipe is %.5f m3/s.',V_dot_3)
if(Re3<2300)
    mprintf('\n     Flow is laminar.')
else
    if Re3>4000
        mprintf('\n     Flow is turblent.')
    end
end
```

Scilab code Exa 8.2 Pressure Drop and Head Loss in a Pipe

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                        //To clear the console
```

```
screen
clear;                          //To clear all the existing variables in the memorymemory


//Given data
T=40                            //T is water temperature in ' F '
rho=62.24                       //rho is water density in 'lbm/ft3 '
Mu=1.038E-3                     //Mu is water viscosity in 'lbm/(ft s)'
D=0.12                          //D is diameter of the pipe in 'in '
L=30                            //L is length of the pipe in 'ft '
V_avg=3                         //V_avg is average velocity of water in the pipe in 'ft/s'


//Unit conversion
D=D/12                          //Conversion from 'in ' to 'ft '


//Assumption
g=32.2                          //g is acceleration due to gravity in 'm/s2 '


//Part (a)
//Calculation
Re=rho*V_avg*D/Mu               //Re is the reynolds number
if Re<2300 then
    Regime="laminar"
    f=64/Re                     //f is the friction factor
```

```
else
    if Re>4000
        Regime="turblent"
        f=0.316/(Re^0.25)  //f is the friction factor(Assuming the pipe to be smooth)
    end
end
h_L=f*L*V_avg^2/(2*D*g)    //h_L is head loss in 'ft'



//Display of Result
mprintf('\n(a) Flow Regime is %s.\n    Head loss is %.1f ft.',Regime,h_L)


//Part (b)
//Calculation
Delta_P_L=f*L*rho*V_avg^2/(2*D)     //Delta_P_L is pressure drop in the pipe in 'lbm/(ft s2)'


//Display of Result
mprintf('\n\n(b) Pressure drop is %.1f lbm/(ft s2) or %d lbf/ft2 or %.2f psi.',Delta_P_L,Delta_P_L/32.2,Delta_P_L/(32.2*12^2))
//The answers vary due to round off error


//Part (c)
//Unit conversion
Delta_P_L=Delta_P_L/32.2          //Conversion from 'lbm/(ft s2)' to 'lbf/ft2'


//Calculation
R=D/2                             //R is radius of the pipe in 'ft'
```

```
A_c=%pi*R^2                        //A_c is CSA of
    the pipe in 'ft2'
V_dot=V_avg*A_c                    //V_dot is volume
    flow rate of water in 'ft3/s'
W_pump=V_dot*Delta_P_L             //W_pump is power
    required in '(lbf ft)/s'
W_pump=W_pump/0.737                //Conversion from
    '(lbf ft)/s' to 'W'


//Display of Result
mprintf('\n\n(c) The volume flow rate is %.6f ft3/s
    .\n    Pumping power Requirement is %.2f W.',
    V_dot,W_pump)
```

check Appendix ?? for dependency:

TABLE_8_2.jpg

check Appendix AP 43 for dependency:

fsolve2.sci

Scilab code Exa 8.3 Determining the Head Loss in a Water Pipe

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                         //To clear the console
    screen
clear;                       //To clear all the
    existing variables in the memory
exec('.\fsolve2.sci');
//Replace '.' present inside the 'exec('')' with the
     path to the folder location where the dependency
     fsolve2.sci file is saved.


```

```
//Given data
T=60                            //T is the water temperature in ' F '
rho=62.36                       //rho is the water density in 'lbm/ft3 '
Mu=7.536E-4                     //Mu is the water viscosity in 'lbm/(ft s)'
D=2                             //D is diameter of the pipe in 'in '
V_dot=0.2                       //V_dot is volume flow rate of water in 'ft3/s'
L=200                           //L is length of the pipe in 'ft '


//Unit conversion
D=D/12                          //Conversion from 'in ' to 'ft '


//Assumption
Epsilon=0.000007                //Epsilon is equivalent roughness value in 'ft '(Obtained from TABLE 8-2)
g=32.2                          //g is acceleration due to gravity in 'ft/s2 '


//Calculation
R=D/2                           //R is radius of the pipe in 'ft '
A_c=%pi*R^2                     //A_c is CSA of the pipe in 'ft2 '
V=V_dot/A_c                     //V is velocity of water in 'ft/s'
Re=rho*V*D/Mu                   //Re is dimensionless reynolds Mumber
if Re<2300 then
    Regime="laminar"
```

```
    f=64/Re                          //f is the friction factor
else
    if Re>4000
        Regime="turblent"
        Epsilon_by_D=Epsilon/D   //Epsilon_by_D is the roughness factor of the pipe
        f0=0.01                  //f0 is the guess friction factor used to to determine the actual friction factor using fsolve function
        f=fsolve(f0,fsolve2)     //Determination of actual friction factor using fsolve function
    end
end
Delta_P=f*L*rho*V^2/(2*D)       //Delta_P is pressure drop in the pipe in '(lbm ft)/s2'
Delta_P=Delta_P/g               //Conversion from 'lbm/(ft s2)' to 'lbf/ft2'
h_L=Delta_P/rho                 //h_L is head loss in 'ft'
W_pump=V_dot*Delta_P            //W_pump is the required power input in '(lbf ft)/s'
W_pump=W_pump/0.737             //Conversion from '(lbf ft)/s' to 'W'


//Display of result
mprintf('\nFlow regime is %s.\nFriction factor is %.4f.\nPressure drop is %d lbf/ft2 or %.1f psi.\nHead loss is %.1f ft.\nRequired power input is %d W.',Regime,f,Delta_P,Delta_P/12^2,h_L,W_pump)
//The answers vary due to round off error
```

check Appendix AP 41 for dependency:

`fsolve3.sci`

Scilab code Exa 8.4 `Determining the Diameter of an Air Duct`

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                        //To clear the console screen
clear;                      //To clear all the existing variables in the memory
exec('.\fsolve3.sci');
//Replace '.' present inside the 'exec('')' with the path to the folder location where the dependency fsolve3.sci file is saved.


//Given data
P=1                         //P is pressure of the air in 'atm'
T=35                        //T is temperature of the air in ' C '
L=150                       //L is length of circular plastic duct in 'm'
V_dot=0.35                  //V_dot is volume flow rate of water in the duct in 'm3/s'
h_L=20                      //h_L is head loss in the pipe in 'm'


//Assumption
Epsilon=0                   //Epsilon is equivalent roughness value in 'm'(Assuming the duct to be smooth)
g=9.81                      //g is acceleration due to gravity in 'm/s2'
rho=1.145                   //rho is density of the air
```

```
in 'kg/m3'
Mu=1.895E-5                    //Mu is dynamic viscosity of the air in 'kg/(m s)'
Nu=1.655E-5                    //Nu is kinematic viscosity of the air in 'm2/s'
//Calculation
x0=[3 1 10000 0.01]            //Guess Value for Velocity (x(1)), Diameter (x(2)), Reynolds Mumber (x(3)) and Friction factor (x(4)) respectively
x=fsolve(x0,fsolve3)           //Calling statement for fsolve function
D_By_SJF=0.66*(((Epsilon^1.25)*(L*V_dot/(g*h_L))^4.75)+(Nu*(V_dot^9.4)*(L/(g*h_L))^5.2))^0.04//D_By_SJF is diameter of the duct determined using Swamee-Jain formula in 'm'
//Display of result
mprintf('\nVelocity is %.2f m/s.\nDiameter is %.3f m.\nReynolds number is %d.\nFriction factor is %.4f.\n\nDiameter by Swamee-Jain formula is %.3f m.',x(1),x(2),x(3),x(4),D_By_SJF)
//The answers vary due to round off error
```

check Appendix AP 40 for dependency:

`fsolve4.sci`

Scilab code Exa 8.5 Determining the Flow Rate of Air in a Duct

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                           //To clear the console screen
```

```
clear;                       //To clear all the existing variables in the memory
exec('.\fsolve4.sci');
//Replace '.' present inside the 'exec('')' with the path to the folder location where the dependency fsolve4.sci file is saved.


//Given data
P=1                          //P is pressure of the air in 'atm'
T=35                         //T is temperature of the air in ' C '
L=300                        //L is length of circular plastic duct in 'm'
h_L=20                       //h_L is head loss in the pipe in 'm'
D=0.267                      //D is diameter of the duct in 'm'
V_dot_old=0.35               //V_dot is old volume flow rate of water in the duct in 'm3/s'


//Assumption
Epsilon=0                    //Epsilon is equivalent roughness value in 'm'(Assuming the duct to be smooth)
g=9.81                       //g is acceleration due to gravity in 'm/s2'
rho=1.145                    //rho is density of the air in 'kg/m3'
Mu=1.895E-5                  //Mu is dynamic viscosity of the air in 'kg/(m s)'
Nu=1.655E-5                  //Nu is kinematic viscosity of the air in 'm2/s'


//Calculation
```

```
Epsilon_by_D=Epsilon/D          //Epsilon_by_D is the roughness factor of the duct
R=D/2                           //R is radius of the duct in 'm'
A=%pi*R^2                       //A is CSA of the duct in 'm2'
x0=[1 0.01 3 50000]             //Guess value for New volume flow rate (x(1)), Friction factor (x(2)), Velocity (x(3)) and Reynolds Number (x(4)) respectively
x=fsolve(x0,fsolve4)            //Calling statement for fsolve function
V_dot_drop=V_dot_old-x(1)       //V_dot_drop is drop in the volume flow rate in 'm3/s'


//Display of result
mprintf('\nOld volume flow rate is %.2f m3/s.\nNew volume flow rate is %.2f m3/s.\nFriction factor is %.4f.\nVelocity is %.2f m/s.\nReynolds Number is %d.\n\nThe drop in the flow rate is %.2f m3/s.',V_dot_old,x(1),x(2),x(3),x(4),V_dot_drop)
//The answers vary due to round off error
```

Scilab code Exa 8.6 Head Loss and Pressure Rise during Gradual Expansion

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                    //To clear the console screen
clear;                  //To clear all the existing variables in the memory


//Let '1' and '2' be the same numbering notation as shown FIGURE 8-40 page number 353
```

```
//Given data
D1=6                    //D1 is diameter of the pipe before expansion in 'cm'
D2=9                    //D2 is diameter of the pipe after expansion in 'cm'
theta=30                //theta is angle made by wall with the horizontal in '  (degree)'
V1=7                    //V1 is the average velocity of water before the expansion in 'm/s'
P1=150                  //P1 is the pressure of water before the expansion in 'kPa'


//Unit conversion
D1=D1/100               //Conversion from 'cm' to 'm'
D2=D2/100               //Conversion from 'cm' to 'm'
P1=P1*1000              //Conversion from 'kPa' to 'Pa'


//Assumption
alpha1=1.06             //alpha1 is the upstream kinetic energy correction factor(Assuming flow to be turblent)
alpha2=1.06             //alpha1 is the downstream kinetic energy correction factor(Assuming flow to be turblent)
rho=1000                //rho is density of the water in 'kg/m3'
K_L=0.07                //K_L is loss co-efficient for gradual expansion
Z1=0                    //Z1 is elevation of pipe before expansion from the reference plane in 'm'
Z2=0                    //Z2 is elevation of pipe after expansion from the reference plane in 'm'
h_pump_u=0              //h_pump_u is the useful pump head delivered to the water in 'm'
h_turbine_e=0           //h_turbine_e is the turbine head extracted from the water in 'm'
```

```
g=9.81                  //g is acceleration due to gravity in 'm/s2'


//Calculation
V2=V1*(D1/D2)^2         //V2 is the average velocity of water after the expansion in 'm/s'
h_L=K_L*V1^2/(2*g)      //h_L is the irreversible head loss in the expansion section in 'm'
P2=rho*g*(((alpha1*V1^2/(2*g))-(alpha2*V2^2/(2*g)))+Z1-Z2+h_pump_u-h_turbine_e-h_L+(P1/(rho*g)))
//P2 is the pressure of water after the expansion in 'Pa'
P2=P2/1000              //Conversion from 'Pa' to 'kPa'


//Display of result
mprintf('\nHead loss in the expansion section is %.3f m.\nPressure in the larger diameter pipe is %d kPa.',h_L,P2)
```

check Appendix AP 39 for dependency:

`fsolve5.sci`

Scilab code Exa 8.7 Pumping Water through Two Parallel Pipes

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                        //To clear the console screen
clear;                      //To clear all the existing variables in the memory
exec('.\fsolve5.sci');
//Replace '.' present inside the 'exec('')' with the path to the folder location where the dependency fsolve5.sci file is saved.
```

```
//Let 'A','B','1' and '2' be same naming notations as that of FIGURE 8-47 page number 358
//Given data
T=20                    //T is water temperature in ' C '
Z_A=5                   //Z_A is elevation of reservoir 1 in 'm'
Z_B=13                  //Z_B is elevation of reservoir 2 in 'm'
NumberOfPipe=2
L1=36                   //L1 is length of the pipe 1 in 'm'
L2=36                   //L2 is length of the pipe 2 in 'm'
D1=4                    //D1 is diameter of the pipe 1 in 'cm'
D2=8                    //D2 is diameter of the pipe 2 in 'cm'
Eta_pump_motor=70       //Eta_pump_motor is efficiency of motor-pump combination in '%'
W_elect=8               //W_elect is power required by motor-pump combination in 'kW'


//Unit conversion
W_elect=W_elect*1000    //Conversion from 'kW' to 'W'
Eta_pump_motor=Eta_pump_motor/100
D1=D1/100               //Conversion from 'cm' to 'm'
D2=D2/100               //Conversion from 'cm' to 'm'


//Assumption
Epsilon=0.000045        //Epsilon is equivalent
```

```
roughness value in 'm'
rho=998                    //rho is the density of water in 'kg/m3'
Mu=1.002E-3                //Mu is the dynamic viscosity of water in 'kg/(m s)'
g=9.81                     //g is acceleration due to gravity in 'm/s2'


//Calculation
Epsilon_by_D1=Epsilon/(3.7*D1)  //Epsilon_by_D1 is the roughness factor of the pipe 1
Epsilon_by_D2=Epsilon/(3.7*D2)  //Epsilon_by_D2 is the roughness factor of the pipe 2
R1=D1/2                         //R1 is radius of the pipe 1 in 'm'
R2=D2/2                         //R2 is radius of the pipe 2 in 'm'
A_c_1=%pi*R1^2                  //A_c_1 is CSA of the pipe 1 in 'm2'
A_c_2=%pi*R2^2                  //A_c_2 is CSA of the pipe 2 in 'm2'
x0=[0.01 0.001 0.01 5 5 10 10 10 10 100000 200000 0.01 0.01] //Guess values for the 13 unknown variables
x=fsolve(x0,fsolve5)            //fsolve function calling statement


//Display of result
mprintf('\nTotal volume flow rate is %.4f m3/s.\nVolume flow rate in pipe 1 is %.5f m3/s.\nVolume flow rate in pipe 2 is %.4f m3/s.\nVelocity in pipe 1 is %.2f m/s.\nVelocity in pipe 2 is %.2f m/s.\nHead loss in pipe is %.1f m.\nHead loss in pipe 1 is %.1f m.\nHead loss in pipe 2 is %.1f m.\nUseful pump head is %.1f m.\nReynolds number in pipe 1 is %d.\nReynolds number in pipe 2 is %d
```

```
.\nFriction factor in pipe 1 is %.4f.\nFriction factor in pipe 2 is %.4f.',x(1),x(2),x(3),x(4),x(5),x(6),x(7),x(8),x(9),x(10),x(11),x(12),x(13))
//The answers vary due to round off error
```

check Appendix AP 38 for dependency:

`fsolve6.sci`

Scilab code Exa 8.8 `Gravity Driven Water Flow in a Pipe`

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                        //To clear the console screen
clear;                      //To clear all the existing variables in the memory
exec('.\fsolve6.sci');
//Replace '.' present inside the 'exec('')' with the path to the folder location where the dependency fsolve6.sci file is saved.


//Let '1' and '2' be the same numbering notation as shown FIGURE 8-48 page number 360
//Given data
T=10                        //T is water temperature in ' C '
D=5                         //D is diameter of cast iron piping system in 'cm'
V_dot=6                     //V_dot is volume flow rate of water in 'L/s'
K_L_entrance=0.5            //K_L_entrance is loss coefficient of entrance
K_L_elbow=0.3               //K_L_elbow is loss coefficient of elbow
```

```
K_L_valve=0.2               //K_L_valve is loss coefficient of valve
K_L_exit=1.06               //K_L_exit is loss coefficient of exit
Z2=4                        //Z2 is elevation of reservoir 2 in 'm'
Distance=9                  //m
Length=80                   //m


//Unit conversion
V_dot=V_dot/1000            //Conversion from 'L/s' to 'm3/s'
D=D/100                     //Conversion from 'cm' to 'm'


//Assumption
Epsilon=0.00026             //Epsilon is equivalent roughness value in 'm'
P1=101325                   //P1 is pressure in reservoir 1 in 'Pa'
P2=101325                   //P2 is pressure in reservoir 2 in 'Pa'
V1=0                        //V1 is water velocity in reservoir 1 in 'm/s'
V2=0                        //V2 is water velocity in reservoir 2 in 'm/s'
g=9.81                      //g is acceleration due to gravity in 'm/s2'
Alpha1=1.03                 //Alpha1 is the kinetic energy correction factor(Assuming flow to be turblent)
Alpha2=1.03                 //Alpha2 is the kinetic energy correction factor(Assuming flow to be turblent)
rho=999.7                   //rho is water density in 'kg/m3'
```

```
Mu=1.307E-3                //Mu is dynamic viscosity of water in 'kg/(m' s)


//Calculation
L=Distance+Length          //L is total length of the pipe in 'm'
R=D/2                      //R is radius of the pipe in 'm'
A=%pi*R^2                  //A is area of the pipe in 'm2'
V=V_dot/A                  //V is velocity of water in the pipe in 'm/s'
Re=rho*V*D/Mu              //Re is the reynolds number in the pipe
if Re<2300 then
    Regime="laminar"
    f=64/Re                //f is the friction factor of the pipe
else
    if Re>4000
        Regime="turblent"
        Epsilon_by_D=Epsilon/D  //Epsilon_by_D is the roughness factor of the pipe
        f0=0.01                 //f0 is the guess friction factor used to to determine actual friction factor using fsolve function
        f=fsolve(f0,fsolve6)    //Determination of actual friction factor using fsolve function
    end
end
Sigma_K_L=K_L_entrance+(2*K_L_elbow)+K_L_valve+K_L_exit                //Sigma_K_L is the total loss co-efficient
h_L=((f*L/D)+(Sigma_K_L))*V^2/(2*g)                                     //h_L is
```

```
   the total head loss in 'm'
Z1=(P2/(rho*g))-(P1/(rho*g))+(Alpha2*V2^2/(2*g))-(
   Alpha1*V1^2/(2*g))+Z2+h_L //Z2 is the elevation
   of the source in 'm'


//Display of result
mprintf('\nFlow Regime is %s.\nFriction factor is %
   .4f.\nTotal head loss is %.1f m.\nElevation Z1 is
    %.1f m.',Regime,f,h_L,Z1)
//The answers vary due to round off error
```

check Appendix AP 36 for dependency:

fsolve7.sci

check Appendix AP 37 for dependency:

fsolve8.sci

Scilab code Exa 8.9 Effect of Flushing on Flow Rate from a Shower

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                //To clear the console screen
clear;              //To clear all the existing
   variables in the memory


//First fsolve7.sci file is executed for Part (a)
   calculation then fsolve8.sci is executed for Part
    (b) calculation.
exec('.\fsolve7.sci');
//Replace '.' present inside the 'exec('')' with the
    path to the folder location where the dependency
    fsolve7.sci file is saved.
exec('.\fsolve8.sci');
```

```
//Replace '.' present inside the 'exec('')' with the path to the folder location where the dependency fsolve8.sci file is saved.


//Let '1','2' and '3' be same naming notations as that of FIGURE 8-49 page number 362
//Given data
D=1.5                       //D is diameter of the copper pipe in 'cm'
P1=200                      //P1 is pressure at the inlet in 'kPa gage'
K_L_shower=12               //K_L_shower is the loss co-efficient of shower
K_L_reservoir=14            //K_L_reservoir is the loss co-efficient of reservoir
n_tee_shower=1              //n_tee_shower is the number of tee in shower-inlet pipe
n_elbow_shower=2            //n_elbow_shower is number of elbow in the shower-inlet pipe
n_valve_shower=1            //n_valve_shower is number of valve in the shower-inlet pipe
n_tee_reservoir=1           //n_tee_reservoir is number of tee in the reservoir-inlet pipe
n_valve_reservoir=1         //n_valve_reservoir is number of valve in the reservoir-inlet pipe
n_elbow_reservoir=1         //n_elbow_reservoir is number of elbow in the reservoir-inlet pipe
Z3=1                        //Z3 is the elevation of the shower in 'm'
Z2=2                        //Z2 is the elevation of the reservoir in 'm'
Z1=0                        //Z1 is the elevation for the inlet location in 'm'
L_shower=11                 //L_shower is the total length of the pipe connecting the inlet and the shower in 'm'
Length1=5                   //m
```

```
Length2=6                //m
Length3=1                //m


//Unit conversion
P1=(P1*1000)+101325      //Conversion from 'kPa gage' to 'Pa absolute'
D=D/100                  //Conversion from 'cm' to 'm'


//Assumption
rho=998                  //rho is density of the water in 'kg/m3'
Mu=1.002E-3              //Mu is dynamic viscosity of the water in 'kg/(m s)'
Nu=1.004E-6              //Nu is kinematic viscosity of the water in 'm2/s'
Epsilon=1.5E-6           //Epsilon is equivalent roughness value in 'm'
K_L_tee=0.9              //K_L_tee is the loss co-efficient of tee
K_L_elbow=0.9            //K_L_elbow is the loss co-efficient of elbow
K_L_valve=10             //K_L_valve is the loss co-efficient of valve
P2=101325                //P2 is pressure at the reservoir in 'Pa'
P3=101325                //P3 is pressure at the shower in 'Pa'
V1=0                     //V1 is velocity of water at the inlet in 'm/s'
V2=0                     //V2 is velocity of water in the reservoir in 'm/s'
V3=0                     //V3 is velocity of water coming from the shower in 'm/s'
Alpha1=1.03              //Alpha1 is the kinetic energy correction factor(Assuming flow to be
```

```
    turblent)
Alpha2=1.03                  //Alpha1 is the kinetic energy correction factor(Assuming flow to be turblent)
Alpha3=1.03                  //Alpha1 is the kinetic energy correction factor(Assuming flow to be turblent)
g=9.81                       //g is acceleration due to gravity in 'm/s2'
h_pump_u=0                   //h_pump_u is the useful pump head delivered to the water in 'm'
h_turbine_e=0                //h_turbine_e is the turbine head extracted from the water in 'm'


//Part (a)
//Calculation
Sigma_K_L_shower=(n_tee_shower*K_L_tee)+(n_elbow_shower*K_L_elbow)+(n_valve_shower*K_L_valve)+(K_L_shower)
//Sigma_K_L_shower is the total loss coefficient in shower-inlet pipeline
h_L=(P1/(rho*g))-(P2/(rho*g))+(Alpha1*V1^2/(2*g))-(Alpha2*V2^2/(2*g))+Z1-Z2+h_pump_u-h_turbine_e //h_l is head loss in 'm'
R=D/2                        //R is radius of the pipe in 'm'
A=%pi*R^2                    //A is area of the pipe in 'm2'
Epsilon_by_D=Epsilon/D  //Epsilon_by_D is the roughness factor of the pipe
x0=[0.0001 0.01 5 20000]//Guess values for calculating actual values using fsolve function
x=fsolve(x0,fsolve7)    //fsolve function calling statement


//Display of result
```

```
mprintf('\n(a) Flow rate of water through the shower head is %.5f m3/s or %.2f L/s.',x(1),x(1)*1000)


//Part (b)
//Calculation
Sigma_K_L_reservoir=(n_tee_reservoir*K_L_tee)+(n_elbow_reservoir*K_L_elbow)+(n_valve_reservoir*K_L_valve)+(K_L_reservoir)
//Sigma_K_L_reservoir is the total loss coefficient in reservoir-inlet pipeline
h_L_3=(P1/(rho*g))-(P3/(rho*g))+(Alpha1*V1^2/(2*g))-(Alpha3*V3^2/(2*g))+Z1-Z3+h_pump_u-h_turbine_e
//h_L_3 is head loss in the reservoir branch in 'm'
y0=[0.001 0.001 0.0001 0.01 0.01 0.01 3 3 3 10000 10000 10000]  //Guess values for calculating actual values using fsolve function
y=fsolve(y0,fsolve8)                                            //fsolve function calling statement
Reduction=(x(1)-y(2))*100/x(1)                                  //Reduction is the reduction in the water flow rate in '%'


//Display of result
mprintf('\n(b) Flushing of toilet reduces flow rate through the shower by %d percent from %.2f to %.2f L/s.',Reduction,x(1)*1000,y(2)*1000)
//The answers vary due to round off error
```

Scilab code Exa 8.10 Measuring Flow Rate with an Orifice Meter

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
```

```
clc;                    //To clear the console screen
clear;                  //To clear all the existing variables in the memory


//Let '1' and '2' be the same numbering notation as shown FIGURE 8-60 page number 369
//Given data
T=20                    //T is temperature of methanol in ' C '
rho_met=788.4           //rho_met is density of methanol in 'kg/m3'
Mu_met=5.857E-4         //Mu_met is viscosity of methanol in 'kg/(m s)'
D_p=4                   //D_P is pipe diameter in 'cm'
D_o=3                   //D_o is orifice diameter in 'cm'
h=11                    //h is differential height of the manometer in 'cm'


//Unit conversion
D_p=D_p/100             //Conversion from 'cm' to 'm'
D_o=D_o/100             //Conversion from 'cm' to 'm'
h=h/100                 //Conversion from 'cm' to 'm'


//Assumption
CD=0.61                 //CD is the dimensionLess discharge coefficient
rho_Hg=13600            //rho_Hg is density of the mercury in 'kg/m3'
g=9.81                  //g is acceleration due to gravity in 'm/s2'


//Calculation
Beta=D_o/D_p            //Beta is the dimensionLess
```

```
diameter ratio
R_o=D_o/2              //R_o is radius of the orifice in 'm'
R_p=D_p/2              //R_p is radius of the pipe in 'm'
A_c=%pi*R_p^2          //A_c is CSA of the pipe in 'm2'
A_o=%pi*R_o^2          //A_o is area of the orifice in 'm2'
V_dot=A_o*CD*sqrt((2*g*h*(rho_Hg-rho_met))/(rho_met*(1-Beta^4)))//V_dot is water flow rate in 'm3/s'
V=V_dot/A_c            //V is average velocity in the pipe in 'm/s'


//Display of result
mprintf('\nVolume flow rate is %f m3/s or %.2f L/s.\nAverage flow velocity in the pipe is %.2f m/s.',V_dot,V_dot*1000,V)
```

Chapter 9

DIFFERENTIAL ANALYSIS OF FLUID FLOW

check Appendix ?? for dependency:

FIGURE_9_25.jpg

Scilab code Exa 9.11 Volume Flow Rate Reduced from Streamline

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                    //To clear the console screen
clear;                  //To clear all the existing
   variables in the memory


//Given data
v=1                    //v is water velocity in the
   channel in 'm/s'
W=2                    //W is wide of the channel in 'm
   '


//Assumption
```

```
Psi_wall=0            //Psi_wall is bottom wall streamline in 'm2/s'
Psi_dividing=1        //Psi_dividing is divding streamline in 'm2/s'


//Values obtained from the FIGURE 9-25 page number 418
Psi_1=1.8             //Psi_1 is streamline 1.8 in 'm2/s'
Psi_2=1.6             //Psi_2 is streamline 1.6 in 'm2/s'
Delta=0.21            //Delta is the distance between the two stream lines in 'm'


//Calculation
V_dot_by_W=Psi_dividing-Psi_wall          //V_dot_by_W is volume flow rate per unit width in 'm2/s'
V_dot=V_dot_by_W*W                        //V_dot is the total volume flow rate through the slot in 'm3/s'
V_A=(Psi_1-Psi_2)/Delta                   //V_A is the speed at point A in 'm/s '


//Display of result
mprintf('\nTotal volume flow rate through the slot is %.1f m3/s.\nSpeed at point A is %.2f m/s.',V_dot,V_A)
```

Chapter 10

APPROXIMATE SOLUTIONS OF THE NAVIER STOKES EQUATION

Scilab code Exa 10.2 Terminal Velocity of a Particle from a Volcano

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                    //To clear the console screen
clear;                  //To clear all the existing
   variables in the memory

//Given data
D=50                    //D is diameter of the particle
   in 'm'
T=-50                   //T is the air temperature in '
    C'
P=55                    //P is the air pressure in 'kPa'
rho_particle=1240       //rho_particle is the particle
   density in 'kg/m3'


//Assumption
```

```
rho_air=0.8588        //rho_air is the air density in 'kg/m3'
Mu_air=1.474E-5       //Mu_air is the air viscosity in 'kg/(m s)'
g=9.81                //g is the acceleration due to gravity in 'm/s2'
Phi_particle=1        //Phi_particle is the sphericity of the particle


//Unit conversion
D=D*10^-6             //Conversion form ' m ' to 'm'


//Calculation
V=g*D^2*(rho_particle-rho_air)/(18*Mu_air)//V is the terminal velocity in 'm/s'
Re=rho_air*V*D/(Mu_air)                    //Re is the dimension less Reynolds number
Regime="stokes"
if Re>1000 then
    Regime="newtons"
    V=1.74077656*sqrt(D*g*(rho_particle-rho_air)/rho_air)                //V is the terminal velocity in 'm/s'
    Re=rho_air*V*D/(Mu_air)                                              //Re is the dimension less reynolds number
else
    if Re>1 & Re<1000 then
        Regime="intermediate"
        Ar=4*D^3*rho_air*g*(rho_paricle-rho_air)/(3*Mu_air^2)              //Ar is the Archimedes number
        dp_star=(3*Ar/4)^(1/3)                                           //dp_star is the dimension less particle diameter
```

```
        ut_star=((18/dp_star^2)+((2.335-(1.744*Phi_particle))/dp_star^0.5))^-1  //ut_star is the dimension less velocity
        V=ut_star/(rho_air^2/(Mu_air*g*(rho_particle-rho_air)))^(1/3)           //V is the terminal velocity in 'm/s'
        Re=rho_air*V*D/(Mu_air)                                                 //Re is the dimension less reynolds number
    end
end


//Display of result
mprintf('\nSettling is in %s regime.\n\nTerminal velocity is %.3f m/s.\nReynolds number is %.3f.',Regime,V,Re)
//The answers vary due to round off error
```

Scilab code Exa 10.6 Velocity in a Flow Composed of Three Components

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                    //To clear the console screen
clear;                  //To clear all the existing variables in the memory

//Given data
V_dot_by_L_1=2          //V_dot_by_L_1 is first line source of strength in 'm/s2'
V_dot_by_L_2=-1         //V_dot_by_L_2 is second line source of strength in 'm/s2'
Gamma=1.5               //Gamma is a line vortex of strength in 'm/s2'
```

```
x_1=0                  //x_1 is the x-coordinate of first line source of strength
y_1=-1                 //y_1 is the y-coordinate of first line source of strength
x_2=1                  //x_2 is the x-coordinate of second line source of strength
y_2=-1                 //y_2 is the y-coordinate of second line source of strength
x_v=1                  //x_v is the x-coordinate of line vortex of strength
y_v=1                  //y_v is the y-coordinate of line vortex of strength
x=1                    //x is the x-coordinate of the point where the fluid velocity is to be found
y=0                    //y is the y-coordinate of the point where the fluid velocity is to be found


//Assumption
Angle=45               //'Angle' is the angle made by the first line source with the x-axis.


//Unit conversion
Angle=Angle*%pi/180 //Conversion from 'radian' to '(degree)'


//Calculation
r_vortex=y_v-y                                          //m
r_source_2=y-y_2                                        //m
r_source_1=1/sin(Angle)                                 //m
V_vortex=Gamma/(2*%pi*r_vortex)                         //V_vortex is the velocity induced by the vortex in 'm/s'
V_source_1=abs(V_dot_by_L_1)/(2*%pi*r_source_1) //V_source_1 is the velocity induced by the first source in 'm/s'
```

```
V_source_2=abs(V_dot_by_L_2)/(2*%pi*r_source_2) //
    V_source_2 is the velocity induced by the second
    source in 'm/s'


//Display of result
mprintf('The superposed velocity at the point (%d,%d
    ) is (%.3f i + %d j) m/s',x,y,(V_vortex + (
    V_source_1)/r_source_1), (V_source_1/r_source_1)
    -(V_source_2))
```

Scilab code Exa 10.9 `Laminar or Turbulent Boundary Layer`

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                //To clear the console screen
clear;              //To clear all the existing
    variables in the memory


//Given data
V=5.0               //V is the speed of canoe in 'mi/h'
T=50                //T is the temperature of the lake
    water in ' F '
X=16                //X is the length of bottom of the
    canoe in 'ft'


//Unit conversion
V=V*5280/3600       //Conversion from 'mi/h' to 'ft/s'


//Assumption
Nu=1.407E-5         //Nu is the kinematic viscosity of
    water in 'ft2/s'
```

```
Re_X_cr=5E5        //Re_cr is the critical Reynolds
    number


//Calculation
Re_X=V*X/Nu        //Re is the Reynolds number
if Re_X>Re_X_cr then
    regime="turblent"
else
    if Re_X<Re_X_cr then
        regime="laminar"
    end
end


//Display of result
mprintf('\nThe boundary layer on the canoe bottom is
    %s.',regime)
```

check Appendix ?? for dependency:

TABLE_10_4.jpg

Scilab code Exa 10.11 Displacement Thickness in the Design of a Wind Tunnel

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                //To clear the console screen
clear;              //To clear all the existing
    variables in the memory


//Given data
T=19                //T is the temperature of the air in
    ' C '
```

```
D=30              //D is the diameter of the wind tunnel in 'cm'
X=30              //X is the length of the wind tunnel in 'cm'
V=4.0             //V is the wind tunnel speed in 'm/s'


//Unit conversion
D=D/100           //Conversion from 'cm' to 'm'
X=X/100           //Conversion from 'cm' to 'm'


//Assumption
Nu=1.507E-5       //Nu is the kinematic vicosity of the air in 'm2/s'
Re_X_cr=5E5       //Re_cr is the critical Reynolds number


//Calculation
//Part (a)
Re_X=V*X/Nu       //Re is the Reynolds number
R=D/2             //R is radius of the wind tunnel in 'm'
//Formula for Delta_star or displacement thickness is obtained from TABLE 10-4 page number 530.
if Re_X<Re_X_cr then
    regime="laminar"
    Delta_star=1.72*X/sqrt(Re_X)        //Delta_star is the displacement thickness at the end of the test section in 'm'
    V_end=V*(R^2/(R-Delta_star)^2)      //V_end is the average air speed at the end of the test section in 'm/s'
else
    regime="turblent"
    Delta_star=0.020*X/Re_X^(1/7)       //Delta_star
```

```
        is the displacement thickness at the end of
        the test section in 'm'
    V_end=V*(R^2/(R-Delta_star)^2)       //V_end is
        the average air speed at the end of the test
        section in 'm/s'
end
Delta_star=Delta_star*1000                   //Conversion
    from 'm' to 'mm'


//Display of result
mprintf('\n(a) Boundary layer on the wall remains %s
    throughout the length of the test section.\n
   Displacement thickness at end of the test section
    is %.2f mm.\n     Average speed at the end of the
    test section is %.2f m/s.',regime,Delta_star,
   V_end)
```

check Appendix ?? for dependency:

`TABLE_10_4.jpg`

Scilab code Exa 10.12 `Comparison of Laminar and Turbulent Boundary Layers`

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                        //To clear the console
   screen
clear;                      //To clear all the existing
   variables in the memory
clf(0)                      //Clear or reset or reset a
   figure 0
clf(1)                      //Clear or reset or reset a
   figure 1


```

```
//Given data
V=10                        //V is air velocity in 'm/s'
L=1.52                      //L is length of the flat plate in 'm'


//Assumption
Nu=1.516E-5                 //Nu is kinematic viscosity of the air in 'm2/s'


//Calculation
//Part (a)
x=L                                     //m
Re_x=V*x/Nu                             //Re_x is Reynolds number at 'x'
delta_laminar=4.91*x/sqrt(Re_x)         //delta_laminar is laminar boundary layer thickness in 'm'
delta_laminar=delta_laminar*1000        //Conversion from 'm' to 'mm'
delta_turblent=0.16*x/(Re_x)^(1/7)      //delta_turblent is turblent boundary layer thickness in 'm'
delta_turblent=delta_turblent*1000      //Conversion from 'm' to 'mm'
y=1:100                                 //y(in 'mm') denotes the vertical distance
n=length(y)                             //n is number of values present in 'y' matrix
for i=1:n
    if(y(i)<delta_laminar)
        u_laminar(i)=V/(delta_laminar^(1/7))*(y(i)^(1/7))    //u_laminar(in m) is velocity when the flow is laminar
    else
        u_laminar(i)=V
    end
    if(y(i)<delta_turblent)
        u_turblent(i)=V/(delta_turblent^(1/7))*(y(i)
```

```
^(1/7)) //u_turblent(in m) is velocity when the flow is turblent
    else
        u_turblent(i)=V
    end
end


//Display of result
plot(u_laminar',y,'k',u_turblent',y,'r')
xlabel("u,m/s")
ylabel("y,mm")
title("FIGURE 10-115: Comparison of laminar and turblent flate plate boundary layer profiles in physical variables at the same x-location.")
legend('laminar','turblent')


//Part (b)
//Calculation
C_f_x_laminar=0.664/sqrt(Re_x)      //C_f_x_laminar is laminar local skin friction coefficient
C_f_x_turblent=0.027/(Re_x)^(1/7)   //C_f_x_turblent is turblent local skin friction coefficient


//Display of result
mprintf("\n(b) Laminar local skin friction coefficient is %f.\n    Turblent local skin friction coefficient is %f.",C_f_x_laminar,C_f_x_turblent)


//Part (c)
//Calculation
x=0:5
m=length(x)
for i=1:m
```

```
    delta_laminar(i)=x(i)*4.91/sqrt(Re_x)
    delta_laminar(i)=delta_laminar(i)*1000
       //Conversion from 'm' to 'mm'
    delta_turblent_a(i)=x(i)*0.16/(Re_x)^(1/7)
    delta_turblent_a(i)=delta_turblent_a(i)*1000
       //Conversion from 'm' to 'mm'
    delta_turblent_b(i)=x(i)*0.38/(Re_x)^(1/5)
    delta_turblent_b(i)=delta_turblent_b(i)*1000
       //Conversion from 'm' to 'mm'

end


//Display of result
scf(1)
plot(x',delta_laminar,'k',x',delta_turblent_a,'r',x
   ',delta_turblent_b,'r')
xlabel("x,m")
ylabel("delta,mm")
title("FIGURE 10-116: Comparison of the growth of a
   laminar and turblent flate plate boundary layer
   of Example 10-12.")
legend('laminar','turblent (a)','turblent (b)',2)
```

Scilab code Exa 10.13 Comparison of Turbulent Boundary Layer Profile Equations

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;             //To clear the console screen
clear;           //To clear all the existing
   variables in the memory
```

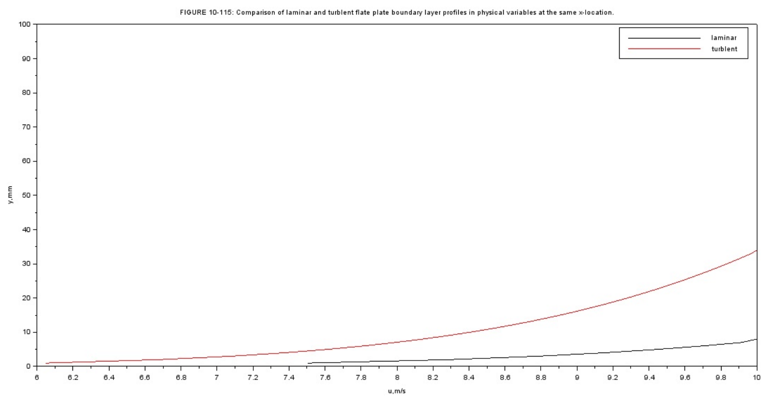


Figure 10.1: Comparison of Laminar and Turbulent Boundary Layers

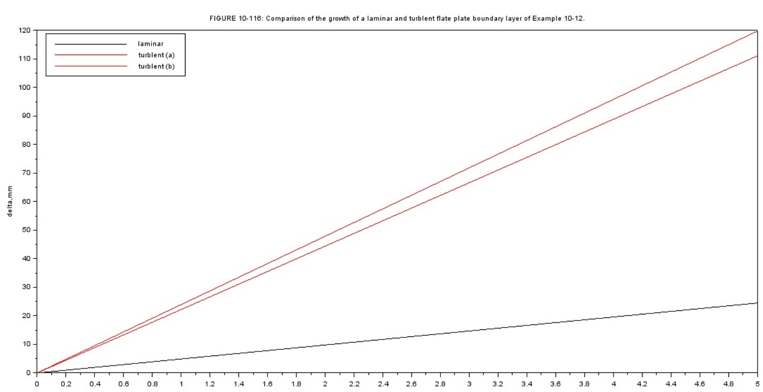


Figure 10.2: Comparison of Laminar and Turbulent Boundary Layers

```
clf(0)            //Clear or reset or reset figure 0
clf(1)            //Clear or reset or reset figure 1


//Given data
T=20              //T is the temperature of air in 'C'
V=10.0            //V is the velocity of air in 'm/s'
L=15.2            //L is length of the smooth flate plate in 'm'


//Assumption
Nu=1.516E-5       //Nu is the kinematic viscosity of air in 'm2/s'
U=10              //U is the velocity component parallel to the wall at a location just above the boundary layer in 'm/s'


//Calculation
X=L               //m
Re_X=V*X/Nu       //Re_X is dimension less Reynolds number
if Re_X<1E5 then
    flow="laminar"
else
    if Re_X>1E5 & Re_X<3E6
    flow="transitional"
    else
        if Re_X>3E6 then
            flow="turblent"
         end
    end
end
Delta=(0.16*X)/(Re_X)^(1/7)      //Delta is the boundary layer thickness in 'm'
C_f_X=0.027/Re_X^(1/7)           //C_f_X is the local
```

```
skin friction coefficient at the end of the plate
u_star=U*sqrt(C_f_X/2)          //u_star is the friction velocity in 'm/s'
u=0:0.1:U                       //m/s
m=length(u)                     //m is the number of data in 'u' matrix
for i=1:m
    y_oneseventh(i)=Delta*(u(i)/U)^7                    //m
    y_loglaw(i)=Nu/u_star*exp(0.4*((u(i)/u_star)-5))        //m
    y_spalding(i)=(Nu/u_star)*((u(i)/u_star)+(exp(-0.4*5)*(exp(0.4*u(i)/u_star)-1-(0.4*u(i)/u_star)-((0.4*u(i)/u_star)^2/2)-((0.4*u(i)/u_star)^3/6))))//m
    y_plus_oneseventh(i)=y_oneseventh(i)*u_star/Nu
    y_plus_loglaw(i)=y_loglaw(i)*u_star/Nu
    y_plus_spalding(i)=y_spalding(i)*u_star/Nu
    u_plus(i)=u(i)/u_star
    y_oneseventh(i)=y_oneseventh(i)*1000                //Conversion from 'm' to 'mm'
    y_loglaw(i)=y_loglaw(i)*1000                        //Conversion from 'm' to 'mm'
    y_spalding(i)=y_spalding(i)*1000                    //Conversion from 'm' to 'mm'
end


//Display of result
mprintf('\nFlow is %s from the beginning of the plate.',flow)


//Graph plotting
scf(0)
```

```
plot(u,y_loglaw','r',u,y_spalding','k',u,y_oneseventh','b')
xlabel('u,m/s')
ylabel('y,mm')
title('FIGURE 10-118: Comparison of turblent flat plate boundary layer profile expressions in physical variables:One-seventh-power approximation,log law, and spalding law of the wall')
legend(['log law';'spalding law';'1/7th power'],2)


scf(1)
plot(y_plus_oneseventh,u_plus,'k',y_plus_loglaw,u_plus,'r',y_plus_spalding,u_plus,'.r')
xlabel('y_plus')
ylabel('u_plus')
title('FIGURE 10-119: Comparison of turblent flat plate biundary layer profile expressions in law of the wall variables: one-seventh-power approximation, log law, and Spalding law of the wall')
legend(['1/7th power';'log law';'spalding law'],4)
```

Scilab code Exa 10.15 Drag on the Wall of a Wind Tunnel Test Section

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                              //To clear the console screen
clear;                            //To clear all the
```

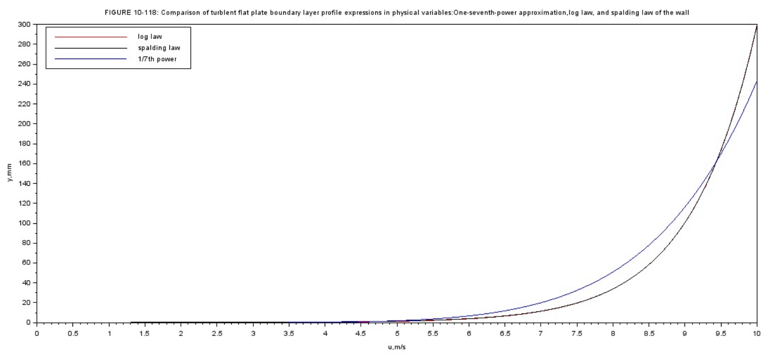


Figure 10.3: Comparison of Turbulent Boundary Layer Profile Equations

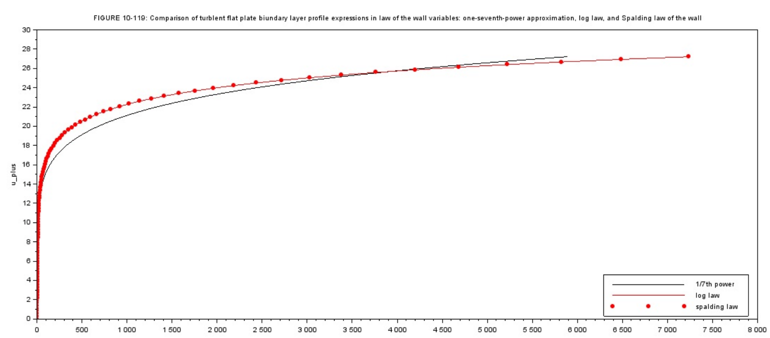


Figure 10.4: Comparison of Turbulent Boundary Layer Profile Equations

```
existing variables in the memory


//Given data
T=20                            //T is the temperature of the air in ' C '
P=101325                        //P is the pressure of the air in 'Pa'
L=1.8                           //L is the length of the test section in 'm'
W=0.50                          //W is the width of the test section in 'm'
Delta1=4.2                      //Delta1 is the boundary layer thickness in 'cm'
Delta2=7.7                      //Delta2 is the boundary layer thickness in 'cm'


//Assumption
Nu=1.516E-5                     //Nu is the kinematic viscosity of air in 'm2/s'
rho=1.204                       //rho is the density of air in 'kg/m3'
U=10.0                          //U is the velocity component parallel to the wall at a location just above the boundary layer in 'm/s'


//Unit conversion
Delta1=Delta1/100               //Conversion from 'cm' to 'm'
Delta2=Delta2/100               //Conversion from 'cm' to 'm'


//Calculation
F_D=W*rho*U^2*(4/45)*(Delta2-Delta1)//F_D is the drag force in 'N'
```

```
//Display of result
mprintf('\nThe drag force is %.2f N.',F_D)
```

Chapter 11

FLOW OVER BODIES DRAG AND LIFT

Scilab code Exa 11.1 Measuring the Drag Coefficient of a Car

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                        //To clear the console
   screen
clear;                      //To clear all the existing
   variables in the memory


//Given data
P=1                         //P is pressure in 'atm'
T=70                        //T is temperature in ' F '
V=60                        //V is velocity in 'mi/h'
A=22.26                     //A is frontal area in 'ft2'
F_D=66                      //F_D is force acting on the
    car in 'lbf'


//Unit conversion
V=V*1.467                   //Conversion from 'mi/h' to
```

```
    'ft/s'
F_D=F_D*32.2                //Conversion from 'lbf' to
    '(lbm ft)/s2'


//Assumption
rho=0.07489                 //rho is density of the air
    in 'lbm/ft3'


//Calculation
CD=2*F_D/(rho*A*V^2)        //CD is the discharge co-
    efficient


//Display of result
mprintf('\nDrag co-efficient of the car is %.2f.',CD
    )
//The answers vary due to round off error
```

Scilab code Exa 11.2 `Effect of Mirror Design on the Fuel Consumption`

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                            //To clear the console
    screen
clear;                          //To clear all the
    existing variables in the memory


//Given data
V=95                            //V is average speed of
    the car in 'km/h'
L=24000                         //L is total distance
    per year in 'km/year'
```

```
rho_fuel=0.8                    //rho_fuel is density of the gasoline in 'kg/L'
GasolinePrice=0.6               //GasolinePrice is price of gasoline in '$/L'
HV=44000                        //HV is the heating value of gasoline in 'kJ/kg'
Eta_car=30                      //Eta_car is overall efficiency of the engine in '%'
D=13                            //D is diameter of the miror in 'cm'


//Unit conversion
Eta_car=Eta_car/100
V=V*1000/3600                   //Conversion from 'km/h' to 'm/s'
D=D/100                         //Conversion from 'cm' to 'm'
rho_fuel=rho_fuel*1000          //Conversion from 'kg/L' to 'kg/m3'
GasolinePrice=GasolinePrice*1000//Conversion from '$/L' to '$/m3'


//Assumption
CD_flat=1.1                     //CD_flat is the dicharge coefficient for circular disk
CD_hemisp=0.4                   //CD_hemisp is the dicharge coefficient for hemishperical body
rho=1.2                         //rho is air density in 'kg/m3'


//Calculation
R=D/2                           //R is radius of the mirror in 'm'
A=%pi*R^2                       //A is area of the mirror in 'm2'
```

```
F_D=CD_flat*A*rho*V^2/2      //F_D is drag force
    acting on the flat mirror in 'N'
W_drag=F_D*L                 //W_drag is the amount
    of work done in 'kJ'/year
E_in=W_drag/Eta_car          //E_in is the required
    energy input in 'kJ/year'
Amount_of_fuel=E_in/(HV*rho_fuel)
    //Amount_of_fuel is required amount of fuel in 'L
    /year'
Cost=Amount_of_fuel*GasolinePrice
    //'Cost' is cost of required fuel in '$/year'
Reduction_Ratio=(CD_flat-CD_hemisp)/CD_flat
Fuel_reduction=Reduction_Ratio*Amount_of_fuel
    //Fuel_reduction is the reduction in fuel
    consumption in 'm3/year'
Fuel_reduction=Fuel_reduction*1000
    //Conversion from 'm3/year' to 'L/year'
Cost_Reduction=Reduction_Ratio*Cost
    //Cost_Reduction is reduction of cost in '$/year'


//Lplay of result
mprintf('\nAmount_of_fuel of money saved is %.2f $/
    year.\nAmount_of_fuel of fuel saved is %.2f L/
    year.',Cost_Reduction,Fuel_reduction)
//The answers vary due to round off error
```

Scilab code Exa 11.3 Flow of Hot Oil over a Flat Pipe

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                    //To clear the console screen
clear;                  //To clear all the existing
    variables in the memory

```

```
//Given data
T=40                    //T is temperature of the engine oil in ' C '
L=5                     //L is length of the flat plate in 'm'
V=2                     //V is velocity of the engine oil in 'm/s'


//Assumption
Re_c=5E5                //Re_c is the critical Reynolds number
rho=876                 //rho is density of the engine oil in 'kg/m3'
Nu=2.485E-4             //Nu is kinematic viscosity of engine oil in 'm2/s'
W=1                     //W is width of the flat plate in 'm'


//Calculation
A=L*W                   //A is area of the flat plate in 'm2'
Re=V*L/Nu               //Re is the Reynolds number
if Re<Re_c then
    C_f=1.33/Re^0.5                         //C_f is the average friction co-efficient
else if (Re>Re_c & Re<1E7)
        C_f=(0.074/Re^(1/5))-(1742/Re)  //C_f is the average friction co-efficient
     end
end
F_D=C_f*A*rho*V^2/2//F_D is drag force acting on the plate in 'N'


//Display of result
mprintf('\nThe drag force acting on the entire plate
```

```
    is %.1f N.',F_D)
//The answers vary due to round off error
```

check Appendix ?? for dependency:

`FIGURE_11_34.jpg`

Scilab code Exa 11.4 `Drag Force Acting on a Pipe in a River`

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                        //To clear the console screen
clear;                      //To clear all the existing variables in the memory


//Given data
D=2.2                       //D is outer diameter of the pipe in 'cm'
L=30                        //L is width of the river in 'm'
V=4                         //V is average flow velocity of water in 'm/s'
T=15                        //T is water temperature in ' C '


//Assumption
rho=999.1                   //rho is density of the water in 'kg/m3'
Mu=1.138E-3                 //Mu is dynamic visocity of the water in 'kg/(m s)'


//Unit conversion
```

```
D=D/100                    //Conversion from 'cm' to 'm'


//Calculation
Re=rho*V*D/Mu              //Re is the dimensionLes reynolds number
CD=1                       //This value is obtained from the FIGURE 11-34 page number 585. When input parameters are changed, 'Re' changes. So change 'CD' accordingly to the new 'Re' using FIGURE 11-34.
A=L*D                      //A is the frontal area for flow past the cylinder in 'm2'
F_D=CD*A*rho*V^2/2         //F_D is the drag force acting on the pipe in 'N'


//Display of result
mprintf('\nThe drag force acting on the pipe is %d N.',F_D)
//The answers vary due to round off error
```

check Appendix ?? for dependency:

FIGURE_11_45.jpg

Scilab code Exa 11.5 Lift and Drag of a Commercial Airplane

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                           //To clear the console screen
clear;                         //To clear all the existing variables in the memory

```

```

//Given data
m=70000                           //m is mass of the
    airplane in 'kg'
A=150                             //A is wing planform
    area in 'm2'
V=558                             //V is speed of plane in
     'km/h'
Altitude=12000                    //m
rho=0.312                         //rho is air density at
    cruising altitude in 'kg/m3'


//Unit conversion
V=V*1000/3600                     //Conversion from 'km/h'
     to 'm/s'


//Assumption
rho_ground=1.20                   //rho_ground is air
    density on the ground in 'kg/m3'
C_L1=1.52                         //C_L1 is maximum lift
    co-efficient of the wing without flaps
C_L2=3.48                         //C_L2 is maximum lift
    co-efficient of the wing with flaps
g=9.81                            //g is acceleration due
    to gravity in 'm/s2'


//Part (a)
//Calculation
W=m*g                                               //W is
    weight of the airplane in 'N'
V_min_1=sqrt(2*W/(rho_ground*C_L1*A))    //V_min_1 is
     minimum velocity without flaps in 'm/s'
V_min_2=sqrt(2*W/(rho_ground*C_L2*A))    //V_min_2 is
     minimum velocity with flaps in 'm/s'
V_min_1_safe=1.2*V_min_1                 //
```

```
V_min_1_safe is safe minimum velocity without flaps in 'm/s'
V_min_2_safe=1.2*V_min_2                    //V_min_2_safe is safe minimum velocity with flaps in 'm/s'


//Display of result
mprintf('\n(a) Safe minimum velocities are %d km/h and %d km/h.\n',V_min_1_safe*3600/1000, V_min_2_safe*3600/1000)


//Part (b)
//Calculation
F_L=W                           //F_L is lift force in 'N'
C_L=F_L/(0.5*rho*V^2*A)         //C_L is the lift co-efficient
mprintf('\n(b) Lift co-efficient is %.2f.',C_L)
Alpha=10                        //Alpha is the angle of attack(in (degree)) corresponding to the 'C_L' value is determined from FIGURE 11-45. When input variables are changed, 'C_L' changes. So, Change the attack angle accordingly using FIGURE 11-45 page number 591.


//Display of result
mprintf('\n    The angle of attack is %d .\n',Alpha)


//Part (c)
C_D=0.03                    //The drag coefficient corresponding to the 'C_L' is determined from FIGURE 11-45 page number 591. When input variables are changed, 'C_L' changes. So, Change
```

```
the 'C_D' accordingly using FIGURE 11-45.
F_D=C_D*A*rho*V^2/2        //F_D is drag force acting on the wings in 'N'
F_D=F_D/1000               //Conversion from 'N' to 'kN'
Power=F_D*V                //'Power' is the power required to overcome the drag in 'kW'


//Display of result
mprintf('\n(c) Drag force acting on the wings is %.1f kN.\n     Power required to overcome this drag is %d kW.',F_D,Power)
//The answers vary due to round off error
```

check Appendix ?? for dependency:

`FIGURE_11_53.jpg`

Scilab code Exa 11.6 `Effect of Spin on a Tennis Ball`

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                           //To clear the console screen
clear;                         //To clear all the existing variables in the memory


//Given data
m=0.125                        //m is mass of tennis ball in 'lbm'
D=2.52                         //D is diameter of tennis ball in 'in'
V=45                           //V is velocity of tennis ball in 'mi/h'
```

```
Omega=4800                    //Omega is backspin
   angular velocity of tennis ball in 'rpm'
P=1                           //P is pressure of air
   in 'atm'
T=80                          //T is temperature of
   air in ' F '


//Assumption
rho=0.07350                   //rho is density of air
   in 'lbm/ft3 '
Nu=1.697E-4                   //Nu is kinematic
   viscosity of air in 'ft2/s'
g=32.2                        //g is acceleration due
   to gravity in 'ft/s2 '

//Unit conversion
V=V*5280/3600                 //COnversion from 'mi/h'
    to 'ft/s'
Omega=Omega*2*%pi/60          //Conversion from 'rpm'
   to 'rad/s'
D=D/12                        //Conversion from 'in '
   to 'ft '


//Calculation
RotationRate=Omega*D/(2*V)  //'RotationRate' is rate
    of rotation in 'rad '
R=D/2                         //R is radius of the
   ball in 'ft '
A=%pi*R^2                     //A is frontal area of
   the ball in 'ft2 '
C_L=0.21                      //Lift coefficient
   corresponding to the 'RotationRate' is determined
    from FIGURE 11-53 page number 595. If input
   parameters are changed then 'RotationRate' also
   changes. So, change 'C_L' accordingly using the
   FIGURE 11-53.
```

```
F_L=C_L*A*rho*V^2/2              //F_L is lift force
    acting on the ball in '(lbm ft)/s2'
W=m*g                            //W is weight of the
    ball in '(lbm ft)/s2'
F_L=F_L/g                        //Conversion from '(lbm
    ft)/s2' to 'lbf'
W=W/g                            //Conversion from '(lbm
    ft)/s2' to 'lbf'
if  W>F_L then
    State="drop"
else
    State="rise"
end


//Display of result
mprintf('\nLift force is %.3f lbf.\nWeight of the
    ball is %.3f lbf.\nThe ball will %s under the
    combined effect of gravity and lift.',F_L,W,State
    )
```

Chapter 12

COMPRESSIBLE FLOW

Scilab code Exa 12.1 Compression of High Speed Air in an Aircraft

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                        //To clear the console
    screen
clear;                      //To clear all the existing
    variables in the memory


//Given data
V_1=250                     //V_1 is cruising speed of
    the aircraft in 'm/s'
h=5000                      //h is altitude in 'm'
P_1=54.05                   //P_1 is atmospheric
    pressure in 'kPa'
T_1=255.7                   //T_1 is the ambient air
    temperature in 'K'
P_02_by_P_01=8              //P_02_by_P_01 is the
    stagnation pressure ratio


//Assumption
```

```
C_p=1.005                  //C_P is constant pressure specific heat in 'kJ/(kg K)'
k=1.4                      //k is specific heat ratio of air


//Part (a)
//Calculation
T_01=T_1+(V_1^2/(2*C_p))/1000    //T_01 is stagnation pressure at the compressor inlet in 'K'
//Division by '1000' on the second term of the R.H.S of the above equation is to convert 'm2/s2' present in the second term to 'kJ/kg'
P_01=P_1*(T_01/T_1)^(k/(k-1))    //P_01 is stagnation pressure at the compressor inlet in 'kPa'


//Display of result
mprintf('\n(a) Stagnation pressure at the inlet of the compressor is %.2f kPa.',P_01)


//Part (b)
//Calculation
T_02=T_01*(P_02_by_P_01)^((k-1)/k)   //T_02 is stagnation temperature at the compressor exit in 'K'
W_in=C_p*(T_02-T_01)                 //W_in is work supplied to the compressor in  'kJ/kg'


//Display of result
mprintf('\n(b) The work supplied to the compressor is %.1f kJ/kg.',W_in)
```

Scilab code Exa 12.2 Mach Number of Air Entering a Diffuser

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                    //To clear the console screen
clear;                  //To clear all the existing variables in the memory


//Given data
V=200                   //V id  velocity of the air in 'm/s'
T=30                    //T is air temperature in ' C '


//Unit conversion
T=T+273                 //Conversion from ' C ' to 'K'


//Assumption
R=0.287                 //R is gas constant in 'kJ/(kg K)'
k=1.4                   //k is specific heat ratio



//Part (a)
//Calculation
C=sqrt(k*R*T*1000)   //C is speed of sound in air in 'm/s'
//Multipilcation by 1000 inside the square root is to convert the kJ/kg present inside the square root to m2/s2.


//Display of result
mprintf('\n(a) Speed of sound in air is %d m/s.',C)

```

```
//Part (b)
//Calculation
Ma=V/C                  //Ma is Mach number


//Display of result
mprintf('\n(b) The Mach number is %.3f.',Ma)
//The answers vary due to round off error
```

check Appendix ?? for dependency:

FIGURE_12_12.jpg

Scilab code Exa 12.3 Gas Flow through a Converging and Diverging Duct

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                          //To clear the console
   screen
clear;                        //To clear all the
   existing variables in the memory


//Given data
m=3                           //m is mass flow rate of
    carbon dioxide in 'kg/s'
P_0=1400                      //P_0 is pressure at the
    duct enterance in 'kPa'
T_0=200                       //T_0 is temperature at
   the duct enterance in ' C '
P_1=200                       //P_1 is pressure at the
    duct exit in 'kPa'
delta_P=200                   //delta_P is the
   pressure drop in 'kPa'
```

```
//Assumption
C_p=0.846                         //C_P is constant pressure specific heat in 'kJ/(kg K)'
k=1.289                           //k is specific heat ratio
R=0.1889                          //R is gas constant in 'kJ/(kg K)'


//Unit conversion
T_0=T_0+273                       //Conversion from 'C' to 'K'


//Calculation
P=P_0-delta_P                     //P is the pressure after the pressure drop in 'kPa'
T=T_0*(P/P_0)^((k-1)/k)           //T is temperature at the location corresponds to given pressure drop 'delta_P' in 'K'
V=sqrt(2*C_p*(T_0-T)*1000)        //V is velocity of the carbon dioxide in 'm/s'
//Multipilcation by '1000' inside the square root is to convert the 'kJ/kg' present inside the square root to 'm2/s2'.
rho=P/(R*T)                       //rho is density of the carbon dioxide in 'kg/m3'
A=m/(rho*V)                       //A is the area of the duct in 'm2'
A=A*1E4                           //Conversion from 'cm2' to 'm2'
C=sqrt(k*R*T*1000)                //C is speed of sound in air in 'm/s'
//Multipilcation by '1000' inside the square root is to convert the 'kJ/kg' present inside the square root to 'm2/s2'.
Ma=V/C                            //Ma is Mach number
```

```
//Display of result
mprintf('\nVelocity is %.1f m/s.\nDensity is %.1f kg
    /m3.\nArea is %.1f cm2.\nMach number is %.3f.',V,
    rho,A,Ma)
//The answers vary due to round off error
```

check Appendix ?? for dependency:

`FIGURE_12_12.jpg`

Scilab code Exa 12.4 `Critical Temperature and Pressure in Gas Flow`

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                          //To clear the console
    screen
clear;                        //To clear all the
    existing variables in the memory


//Given data
m=3                           //m is mass flow rate of
     carbon dioxide in 'kg/s'
P_0=1400                      //P_0 is pressure at the
     duct enterance in 'kPa'
T_0=200                       //T_0 is temperature at
    the duct enterance in ' C '
P_1=200                       //P_1 is pressure at the
     duct exit in 'kPa'
delta_P=200                   //delta_P is the
    pressure drop in 'kPa'



```

```
//Assumption
C_p=0.846                    //C_P is constant pressure specific heat in 'kJ/(kg K)'
k=1.289                      //k is specific heat ratio
R=0.1889                     //R is gas constant in 'kJ/(kg K)'


//Unit conversion
T_0=T_0+273                  //Conversion from 'C' to 'K'


//Calculation
T_star=(2/(k+1))*T_0              //T_star is the critical temperature in 'K'
P_star=(2/(k+1))^(k/(k-1))*P_0   //P_star is critical pressure in 'kPa'


//Display of result
mprintf('\nCritical temperature is %d K.\nCritical pressure is %d kPa.',T_star,P_star)
//The answers vary due to round off error
```

check Appendix ?? for dependency:

`TABLE_A_13.jpg`

Scilab code Exa 12.5 `Effect of Back Pressure on Mass Flow Rate`

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                       //To clear the console screen
```

```
clear;                        //To clear all the existing variables in the memory


//Given data
P_0=1                         //P_0 is air pressure at the nozzle inlet in 'MPa'
T_0=600                       //T_0 is air temperature at the nozzle inlet in ' C '
V_0=150                       //V_0 is air velocity at the nozzle inlet in 'm/s'
A=50                          //A is nozzle throat area in 'cm2'
P_b_1=0.7                     //P_b_1 is the back pressure in 'MPa'
P_b_2=0.4                     //p_b_2 is the back pressure in 'MPa'


//Unit conversion
T_0=T_0+273                   //Conversion from ' C ' to 'K'
A=A*1E-4                      //Conversion from 'cm2' to 'm2'



//Assumption
C_p=1.005                     //C_P is constant pressure specific heat in 'kJ/(kg K)'
k=1.4                         //k is specific heat ratio
R=0.287                       //R is gas constant in '(kPa m3)/(kg K)'


//Part (a)
//Calculation
T_0i=T_0+(V_0^2/(2*C_p))/1000
```

```
                                  //T_0i is stagnation temperature at the nozzle inlet is 'K'
//Division by 1000 on the second term of above equation R.H.S is to convert 'm2/s2' present in the second term to 'kJ/kg'.
P_0i=P_0*(T_0i/T_0)^(k/(k-1))                                  //P_0i is stagnation pressure at the nozzle inlet in 'MPa'
T_0=T_0i;                                                        //K
P_0=P_0i;                                                        //MPa
P_c_Ratio=(2/(k+1))^(k/(k-1))                                  //P_c_Ratio is the critical pressure ratio
P_b_Ratio=P_b_1/P_0;                                        //P_b_Ratio is the back pressure ratio
if P_b_Ratio>P_c_Ratio then
    P_t=P_b_1;                                                   //P_t is throat pressure in 'MPa'
    flow1="not choked";
    Ma1=((2*((P_b_Ratio)^((1-k)/k)-1))/(k-1))^0.5        //Ma1 is Mach number
    T_t_Ratio=(1+(Ma1^2*(k-1)*0.5))^(-1)                  //T_t_Ratio is throat temperature ratio
    //Ma1 and T_t_Ratio is calculated using the formulas given in Page 899 APPENDIX 1.
    T_t=T_t_Ratio*T_0                                             //T_t is temperature at throat in 'K'
    rho_t=P_t*1000/(R*T_t)                                     //rho_t is density at throat in 'kg/m3'
    V_t=Ma1*sqrt(k*R*T_t*1000)
```

```
                                        //V_t is velocity at throat in 'm/s'
    m1=rho_t*A*V_t                                                  //m1 is mass flow rate through the nozzle in kg/s
else
     flow1="choked";
     Ma1=1;                                                         //Ma1 is Mach number
     m1=A*(P_0*1000)*sqrt(k/(R*T_0))*(2/(k+1))^((k+1)/(2*(k-1))) //m1 is mass flow rate through the nozzle in 'kg/s'
end


//Display pf result
mprintf('\n(a) Mass flow rate is %.2f kg/s.',m1)


//Part (b)
//Calculation
P_b_Ratio=P_b_2/P_0                                                 //P_b_Ratio is the back pressure ratio
if P_b_Ratio>P_c_Ratio then
    P_t=P_b_2;                                                      //P_t is throat pressure in 'MPa'
    flow2="not choked";
    Ma1=((2*((P_b_Ratio)^((1-k)/k)-1))/(k-1))^0.5          //Ma1 is Mach number
    T_t_Ratio=(1+(Ma1^2*(k-1)*0.5))^(-1)                   //T_t_Ratio is throat temperature ratio
    //Ma1 and T_t_Ratio is calculated using the formulas given in Page 899 APPENDIX 1.
    T_t=T_t_Ratio*T_0;
```

```
                                                    //T_t is temperature at throat in 'K'
    rho_t=P_t*1000/(R*T_t);                         //rho_t is density at throat in 'kg/m3'
    V_t=Ma1*sqrt(k*R*T_t*1000)                      //V_t is velocity at throat in 'm/s'
    m2=rho_t*A*V_t                                  //m2 is mass flow rate through the nozzle in kg/s
else
     flow2="choked";
     Ma2=1;                                         //Ma2 is Mach number
     m2=A*(P_0*1000)*sqrt(k/(R*T_0))*(2/(k+1))^((k+1)/(2*(k-1))) //m2 is mass flow rate through the nozzle in kg/s
end
m2=m2*sqrt(1000)                                    //Conversion from '(kPa m2)/sqrt(kJ/kg)' to 'kg/s'


//Display of result
mprintf('\n(b) Mass flow rate is %.2f kg/s.',m2)
//The answers vary due to round off error
```

check Appendix ?? for dependency:

`TABLE_A_13.jpg`

check Appendix AP 22 for dependency:

`fsolve9.sci`

Scilab code Exa 12.6 `Gas Flow through a Converging Nozzle`

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                                //To clear the console screen
clear;                              //To clear all the existing variables in the memory
exec('.\fsolve9.sci');
//Replace '.' present inside the 'exec('')' with the path to the folder location where the dependency fsolve9.sci file is saved.


//Let '1' and '2' be the same naming notations as shown in 12-25 page number 629
//Given data
T_1=400                             //T_1 is temperature of the nitrogen at duct inlet in 'K'
P_1=100                             //P_1 is pressure of the nitrogen at duct inlet in 'kPa'
Ma1=0.3                             //Ma1 is Mach number at duct inlet
Area_drop=20                        //Area_drop is percentage drop in area


//Assumption
k=1.4                               //k is specific heat ratio


//Calculation
//T_2/T_0, T_1/T_0, P_2/P_0, P_1/P_0, A_1/A_star, A_2/A_star are represented as T_2_ratio, T_1_ratio, P_2_ratio, P_1_ratio, A_1_ratio, A_2_ratio respectively in the following code.
A_1_ratio=(1/Ma1)*((2/(k+1))*(1+(0.5*Ma1^2*(k-1))))
```

```
^((0.5*(k+1))/(k-1))//A_1_ratio is inlet area ratio
T_1_ratio=(1+(0.5*(k-1)*Ma1^2))^(-1)              //T_1_ratio is inlet temperature ratio
P_1_ratio=(1+(0.5*(k-1)*Ma1^2))^(-k/(k-1))        //P_1_ratio is inlet pressure ratio
//A_1_ratio,T_1_ratio and P_1_ratio are calculated using the formulas given in Page number 899 APPENDIX 1.
A_2_ratio=(1-(Area_drop/100))*A_1_ratio           //A_2_ratio is area ratio where area drop is 'Area_drop'
Ma2_guess=0.5                                     //Ma2_guess is guess Mach number and it is used to find actual Mach number using fsolve function
Ma2=fsolve(Ma2_guess,fsolve9)                     //Calling of fsolve function
T_2_ratio=(1+(0.5*(k-1)*Ma2^2))^(-1)              //T_2_ratio is temperature ratio where area drop is 'Area_drop'
P_2_ratio=(1+(0.5*(k-1)*Ma2^2))^(-k/(k-1))        //P_2_ratio is pressure ratio where area drop is Area_drop
//T_2_ratio,P_2_ratio are calculated using the formulas given in Page number 899 APPENDIX 1.
T_2=T_1*(T_2_ratio/T_1_ratio);                    //T_2 is temperature at the desired location in 'K'
P_2=P_1*(P_2_ratio/P_1_ratio);                    //P_2 is pressure at the desired location in 'kPa'


//Display of result
mprintf('\nTemperature at the desired location is %d K.\nPressure at the desired location is %.1f kPa.',T_2,P_2)
//The answers vary due to round off error
```

check Appendix ?? for dependency:

TABLE_A_13.jpg

Scilab code Exa 12.7 Airflow through a Converging DIverging Nozzle

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                                //To clear the
   console screen
clear;                              //To clear all the
   existing variables in the memory


//Given data
P_in=1                              //P_in is pressure
   of air at nozzle inlet in 'MPa'
T_in=800                            //T_in is
   temperature of air at nozzle inlet in 'K'
k=1.4                               //k is specific heat
    ratio
Ma_e=2                              //Ma_e is exit Mach
   number
A_star=20                           //A_star is the
   throat area in 'cm2'
V_in=0                              //V_in is air
   velocity at the nozzle inlet in 'm/s'


//Assumption
R=0.287                             //R is gas constant
   in '(kPa m3)/(kg K)'
Ma=1                                //Ma is Mach number
   at the throat


//Calculation
```

```
P_0=P_in;                          //P_0 is stagnation pressure in 'MPa'
T_0=T_in;                          //T_0 is stagnation temperature in 'K'
rho_0=(P_0*1000)/(R*T_0)           //rho_0 is stagnation density in 'kg/m3'
//Multiplication by 1000 on the numerator of the above equation R.H.S is to convert P_0 from 'MPa' to 'kPa'


//Part (a)
P_star_Ratio=(1+(0.5*(k-1)*Ma^2))^(-k/(k-1))        //P_star_Ratio is throat pressure ratio
T_star_Ratio=(1+(0.5*(k-1)*Ma^2))^(-1)        //T_star_Ratio is throat temperature ratio
rho_star_Ratio=(1+(0.5*(k-1)*Ma^2))^(-1/(k-1))        //rho_star_Ratio is throat density ratio
//P_star_Ratio, T_star_Ratio and rho_star_Ratio are calculated using the formulas given in Page number 899 APPENDIX 1.
P_star=P_star_Ratio*P_0        //P_star is pressure at throat in 'MPa'
T_star=T_star_Ratio*T_0        //T_star is temperature at throat in 'K'
rho_star=rho_star_Ratio*rho_0        //rho_star is density at throat in 'kg/m3'
V_star=sqrt(k*R*T_star*1000)        //V_star is velocity at throat in 'm/s'
//Multiplication by 1000 on the R.H.S of the above equation is to convert 'kJ/kg' present in the above to 'm2/s2'


//Display of result
mprintf('\n(a) Pressure at throat is %.4f MPa.\n    Temperature at throat is %.1f K.\n    Density at throat is %.3f kg/m3.\n    Velocity at throat is
```

```
%.1f m/s.',P_star,T_star,rho_star,V_star)
//The answers vary due to round off error


//Part (b)
P_e_Ratio=(1+(0.5*(k-1)*Ma_e^2))^(-k/(k-1))    //P_e_Ratio is exit pressure ratio
T_e_Ratio=(1+(0.5*(k-1)*Ma_e^2))^(-1)    //T_e_Ratio is exit temperature ratio
rho_e_Ratio=(1+(0.5*(k-1)*Ma_e^2))^(-1/(k-1))    //rho_e_Ratio is exit density ratio
Ma_e_star=Ma_e*sqrt((k+1)/(2+((k-1)*Ma_e^2)))    //A_e_Ratio is exit Mach number ratio
A_e_Ratio=(1/Ma_e)*((2/(k+1))*(1+(0.5*Ma_e^2*(k-1))))^((0.5*(k+1))/(k-1)) //A_e_Ratio is exit area ratio
//P_e_Ratio, T_e_Ratio, rho_e_Ratio, Ma_e_star and A_e_Ratio are calculated using the formulas given in Page number 899 APPENDIX 1.
P_e=P_e_Ratio*P_0    //P_e is exit pressure in 'MPa'
T_e=T_e_Ratio*T_0    //T_e is exit temperature in 'K'
rho_e=rho_e_Ratio*rho_0    //rho_e is exit density in 'kg/m3'
A_e=A_e_Ratio*A_star    //A_e is exit area in 'cm2'
V_e=Ma_e_star*V_star    //V_e is exit velocity in 'm/s'


//Display of result
mprintf('\n\n(b) Pressure at exit is %.4f MPa.\n Temperature at exit is %.1f K.\n    Density at exit is %.3f kg/m3.\n    Velocity at exit is %.1f m/s.\n    Exit area is %.2f cm2.',P_e,T_e,rho_e,V_e,A_e)
//The answers vary due to round off error
```

```
//Part (c)
m=rho_star*A_star*V_star/1E4
    //m is mass flow rate in 'kg/m3'
//Division by 1E4 on the R.H.S of the above equation
     is to convert area from 'cm2' to 'm2'


//Display of result
mprintf('\n\n(c) Mass flow rate is %.2f kg/s.',m)
```

check Appendix ?? for dependency:

`FIGURE_12_35.jpg`

check Appendix ?? for dependency:

`TABLE_A_13.jpg`

Scilab code Exa 12.9 `Shock Wave In a Converging Diverging Nozzle`

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;
    //To clear the console screen
clear;
    //To clear all the existing variables in the
    memory


//Let '1' and '2' be the same naming notations as
    shown in 12-35 page number 638.
//Given data
```

```
P_in=1
    //P_in is pressure of air at nozzle inlet in 'MPa'
T_in=800
    //T_in is temperature of air at nozzle inlet in 'K'
k=1.4
    //k is specific heat ratio
Ma_1=2
    //Ma_e is exit Mach number
A_star=20
    //A_star is the throat area in 'cm2'
V_in=0
    //V_in is air velocity at the nozzle inlet in 'm/s'


//Assumption
R=0.287
    //R is gas constant in '(kPa m3)/(kg K)'
Ma_star=1
    //Ma is Mach number at the throat
C_p=1.005
    //C_p is constant pressure specific heat in 'kJ/(kg K)'


//Calculation
P_01=P_in;
```

```
    //P_0 is stagnation pressure in 'MPa'
T_01=T_in;
    //T_0 is stagnation temperature in 'K'
P_1_Ratio=(1+(0.5*(k-1)*Ma_1^2))^(-k/(k-1))                    //P_1_Ratio is static pressure ratio before the shock
T_1_Ratio=(1+(0.5*(k-1)*Ma_1^2))^(-1)                          //T_1_Ratio is static temperature ratio before the shock
rho_1_Ratio=(1+(0.5*(k-1)*Ma_1^2))^(-1/(k-1))                  //rho_1_Ratio is static density ratio before the shock
//P_1_Ratio, T_1_Ratio and rho_1_Ratio are calculated using the formulas given in Page 899 APPENDIX 1.
rho_0=(P_01*1000)/(R*T_01)                                     //rho_0 is stagnation density in 'kg/m3'
//Multiplication by 1000 on the numerator of the above equation R.H.S is to convert P_01 from 'MPa' to 'kPa'
P_1=P_1_Ratio*P_01
    //P_1 is static pressure before the shock in 'MPa'
T_1=T_1_Ratio*T_01
    //T_1 is static temperature before the shock in 'K'
rho_1=rho_1_Ratio*rho_0                                        //rho_1 is satic density before the shock in 'kg/m3'
Ma_2=sqrt((((k-1)*Ma_1^2)+2)/((2*k*Ma_1^2)-k+1))               //Ma_2 is Mach number after the shock
```

```
P_2_by_P_1=(1+(k*Ma_1^2))/(1+(k*Ma_2^2))                    //P_2_by_P_1 is static pressure ratio after the shock
P_02_by_P_01=(Ma_1/Ma_2)*((1+(Ma_2^2*0.5*(k-1)))/(1+(Ma_1^2*0.5*(k-1))))^((k+1)/(2*(k-1)))//P_02_by_P_01 is stagnation pressure ratio after the shock
T_2_by_T_1=(2+(Ma_1^2*(k-1)))/(2+(Ma_2^2*(k-1)))                  //T_2_by_T_1 is static temperature ratio after the shock
rho_2_Ratio=((k+1)*Ma_1^2)/(2+(Ma_1^2*(k-1)))                   //rho_2_ratio is static density ratio after the shock
//Ma_2, P_2_by_P_1, P_02_by_P_01, T_2_by_T_1 and rho_2_Ratio are calculated using the formulas given in Page 899 APPENDIX 1.
P_02=P_02_by_P_01*P_01                                                      //P_02 is stagnation pressure after the shock in 'MPa'
P_2=P_2_by_P_1*P_1                                                          //P_2 is static pressure after the shock in 'MPa'
T_2=T_2_by_T_1*T_1                                                          //T_2 is static temperature after the shock in 'K'
rho_2=rho_2_Ratio*rho_1                                                     //rho_2 is satic densoty after the shock in 'kg/m3'


//Display of result
mprintf('\n(a) Stagnation pressure after shock is %.3f MPa.\n    Static pressure after shock is %.3f MPa.\n    Static temperature after shock is %d K.\n    Static density after shock is %.2f kg/m3.',P_02,P_2,T_2,rho_2)
```

```
//The answers vary due to round off error


//Part (b)
//Calculation
delta_s=(C_p*log(T_2_by_T_1))-(R*log(P_2_by_P_1))                    //delta_s is entropy change across the shock in 'kJ/(kg K)'


//Display of result
mprintf('\n\n(b) The entropy change across the shock is %.4f kJ/(kg K).',delta_s)


//Part (c)
//Calculation
V_2=Ma_2*sqrt(k*R*T_2*1000)                                           //V_2 is air velocity after the shock in 'm/s'
//Multiplication by 1000 on the R.H.S of the above equation is to convert 'kJ/kg' present in the above to 'm2/s2'


//Display of result
mprintf('\n\n(c) Air velocity after shock is %d m/s.',V_2)
//The answers vary due to round off error


//Part (d)
//Calculation
rho_star_ratio=(1+(0.5*(k-1)*Ma_star^2))^(-1/(k-1))                 //rho_star_Ratio is throat density ratio
T_star_ratio=(1+(0.5*(k-1)*Ma_star^2))^(-1)                         //T_star_Ratio is throat
```

```
temperature ratio
//rho_star_ratio and T_star_ratio are calculated
    using the formulas given in Page 899 APPENDIX 1.
T_star=T_star_ratio*T_01
                                                    //
    T_star is temperature at throat in 'K'
rho_star=rho_star_ratio*rho_0
                                               //rho_star
    is density at throat in 'kg/m3'
v_star=sqrt(k*R*T_star*1000)
                                               //V_star
    is velocity at throat in 'm/s'
m=rho_star*A_star*v_star/1E4
                                               //m is
    mass flow rate in 'kg/m3'


//Display of result
mprintf('\n\n(d) Mass flow rate after shock is %.2f
    kg/s.',m)
```

check Appendix ?? for dependency:

`FIGURE_12_36.jpg`

Scilab code Exa 12.10 Estimation of the Mach Number from Mach Lines

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                    //To clear the console
    screen
clear;                  //To clear all the existing
    variables in the memory


//Given data
```

```
Ma_experimental=3        //Ma_experimental is the experimental Mach Number


//Assumption
Mu=19                    //Mu is angle of Mach lines in the free stream flow in ' (degree)'


//Unit conversion
Mu=Mu*%pi/180            //Conversion from ' (degree)' to 'radian'


//Calculation
Ma_estimated=1/sin(Mu)   //Ma_estimated is the estimated Mach number


//Display of result
mprintf('\nEstimated Mach number is %.2f.\nExperimental Mach number is %d.',Ma_estimated,Ma_experimental)
if abs(Ma_estimated-Ma_experimental)<0.1 then
    mprintf('\nOur estimated Mach number agrees with the experimental Mach number.')
else
    mprintf('\nOur estimated Mach number do not agrees with the experimental Mach number.')
end
```

check Appendix AP 29 for dependency:

`fsolve10.sci`

Scilab code Exa 12.11 `Oblique Shock Calculations`

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                          //To clear the console screen
clear;                        //To clear all the existing variables in the memory
exec('.\fsolve10.sci');
//Replace '.' present inside the 'exec('')' with the path to the folder location where the dependency fsolve10.sci file is saved.


//Let '1' and '2' be the same naming notations as shown in 12-48 page number 646.
//Given data
Ma_1=2                        //Ma_1 is Mach number of super sonic air
P_1=75                        //P_1 is pressure of the super sonic air in 'kPa'
delta=10                      //delta is the half angle of a 2D wedge in ' (degree)'


//Assumption
k=1.4                         //k is specific heat ratio
Beta_weak_Guess=10            //Beta_weak_Guess is guess weak shock angle(in ' (degree)') used to calculate actual weak shock angle
Beta_strong_Guess=90          //Beta_strong_Guess is guess strong shock angle(in ' (degree)') used to calculate actual strong shock angle


//Unit conversion
Beta_weak_Guess=Beta_weak_Guess*%pi/180                    //Conversion from ' (degree)' to 'radian'
```

```
Beta_strong_Guess=Beta_strong_Guess*%pi/180                                //Conversion from '(degree)' to 'radian'


//Calculation
theta=delta;                                              //theta is oblique shock deflection angle in ' (degree)'
theta=theta*%pi/180                                       //Conversion from ' (degree)' to 'radian'
Beta_weak=fsolve(Beta_weak_Guess,fsolve10)                //fsolve function to calcualte weak shock angle
Beta_strong=fsolve(Beta_strong_Guess,fsolve10)            //fsolve function to calcualte strong shock angle
Ma_1_n_weak=Ma_1*sin(Beta_weak)                           //Ma_1_n_weak is weak upstream normal Mach number
P_2_weak=P_1*(((2*k*Ma_1_n_weak^2)-k+1)/(k+1))            //P_2_weak is weak downstream pressure in 'kPa'
Ma_2_n_weak=sqrt((((k-1)*Ma_1_n_weak^2)+2)/(((2*k*Ma_1_n_weak^2)-k)+1)) //Ma_2_n_weak is weak downstream normal Mach number
Ma2_weak_downstream=Ma_2_n_weak/sin(Beta_weak-theta)      //Ma2_weak_downstream is weak downstream Mach number
Ma_1_n_strong=Ma_1*sin(Beta_strong)                       //Ma_1_n_strong is strong upstream normal Mach number
P_2_strong=P_1*(((2*k*Ma_1_n_strong^2)-k+1)/(k+1))        //P_2_strong is strong downstream pressure in 'kPa'
```

```
Ma_2_n_strong=sqrt((((k-1)*Ma_1_n_strong^2)+2)/(((2*k*Ma_1_n_strong^2)-k)+1))    //Ma_2_strong is strong downstream normal Mach number
Ma2_strong_downstream=Ma_2_n_strong/sin(Beta_strong-theta)                          //Ma2_strong_downstream is strong downstream Mach number


//Display of result
mprintf('\nWeak shock angle is %.1f  .\nWeak shock downstream pressure is %d kPa.\nWeak shock downstream Mach number is %.2f.\n\nStrong shock angle is %.1f  .\nStrong shock downstream pressure is %d kPa.\nStrong shock downstream Mach number is %.3f.',Beta_weak*180/%pi,P_2_weak,Ma2_weak_downstream,Beta_strong*180/%pi,P_2_strong,Ma2_strong_downstream)
//The answers vary due to round off error
```

check Appendix AP 28 for dependency:

`fsolve11.sci`

Scilab code Exa 12.12 Prandtl Meyer Expansion Wave Calculations

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                                         //To clear the console screen
clear;                                       //To clear all the existing variables in the memory
exec('.\fsolve11.sci');
//Replace '.' present inside the 'exec('')' with the path to the folder location where the dependency fsolve11.sci file is saved.
```

```


//Let '1' and '2' be the same naming notations as shown in 12-49 page number 647.
//Given data
Ma_1=2                               //Ma_1 is upstream Mach number of super sonic air
P_1=230                              //P_1 is pressure of super sonic air in 'kPa'
delta=10                             //delta is wall expansion angle in '  (degree)'


//Assumption
k=1.4                                //k is specific heat ratio


//Calculation
theta=delta;                         //theta is total defletion angle in '  (degree)'
Nu_Ma1=(sqrt((k+1)/(k-1))*atan(sqrt((k-1)/(k+1)*(Ma_1^2-1))))-(atan(sqrt(Ma_1^2-1))) //Nu_Ma1 is upstream Prandtl-Meyer function in '  (degree)'
Nu_Ma1=Nu_Ma1*180/%pi                //Conversion from 'radian' to '  (degree)'
Nu_Ma2=theta+Nu_Ma1                  //Nu_Ma2 is downstream Prandtl-Meyer function in '  (degree)'
Ma_2_guess=2                         //Ma_2_guess is guess downstream Mach number which is used calculate actual downstream Mach number using fsolve function
Ma_2=fsolve(Ma_2_guess,fsolve11)     //Using fsolve function to find the downstream Mach number.
P_2=((1+(0.5*(k-1)*Ma_2^2))/(1+(0.5*(k-1)*Ma_1^2)))^(-k/(k-1))*P_1 //P_2 is downstream pressure in 'kPa'

```

```
//Display of result
mprintf('\nDownstream Mach number is %.3f.\
   nDownstream pressure is %d kPa.',Ma_2,P_2)
```

check Appendix ?? for dependency:

`TABLE_A_15.jpg`

check Appendix AP 27 for dependency:

`fsolve12.sci`

Scilab code Exa 12.15 `Reyleigh Flow in a Tubular Combuster`

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                                                  //To
    clear the console screen
clear;                                                //To
    clear all the existing variables in the memory
exec('.\fsolve12.sci');
//Replace '.' present inside the 'exec('')' with the
    path to the folder location where the dependency
    fsolve12.sci file is saved.


//Let '1' and '2' be the same naming notations as
   shown in 12-58 page number 656.
//Given data
D=15                                                  //D
   is diameter of combustor in 'cm'
T_1=550                                               //
   T_1 is intlet temperature of compressed air in 'K
   '
P_1=480                                               //
   P_1 is inlet pressure of compressed air in 'kPa'
```

```
V_1=80 //V_1 is inlet velocity of compressed air in 'm/s'
HV=42000 //HV is heating value of fuel in 'kJ/kg'
AF=40 //AF is air-fuel mass ratio


//Unit conversion
D=D/100 //Conversion from 'cm' to 'm'


//Assumption
k=1.4 //k is specific heat ratio
C_p=1.005 //C_p is specific heat of air in 'kJ/(kg K)'
R=0.287 //R is gas constant in 'kJ/(kg K)'


//Calculation
//T_02/T_0_star, T_1/T_star, P_1/P_star, V_1/V_star, T_2/T_star, P_2/P_star, V_2/V_star are represented as T_02_ratio, T_1_ratio, P_1_ratio, V_1_ratio, T_2_ratio, P_2_ratio, V_2_ratio in the following code.
rho_1=P_1/(R*T_1) //rho_1 is inlet air density in 'kg/m3'
A_1=%pi*D^2/4 //A_1 is cross-sectional area of the combustor in 'm2'
m_air=rho_1*A_1*V_1 //m_air is inlet mass flow rate of air in 'kg/s'
m_fuel=m_air/AF //m_fuel is mass flow rate of the fuel in 'kg/s'
Q=m_fuel*HV //Q
```

```
is the rate of heat transfer in 'kW'
q=Q/m_air                                   //q is heat transfer per 'kg' of air in 'kJ/kg'
T_01=T_1+(V_1^2/(2*C_p*1000))               //T_01 is inlet stagnation temperature in 'K'
//Division by '1000' on the second term of above equation RHS is to convert 'm2/s2' present in the second term to 'kJ/kg'.
C_1=sqrt(k*R*T_1*1000)                      //C_1 is speed of sound in air in 'm/s'
//Multipilcation by '1000' inside the square root is to convert the 'kJ/kg' present inside the square root to 'm2/s2'.
Ma_1=V_1/C_1                                //Ma_1 is inlet Mach number
T_02=T_01+(q/C_p)                           //T_02 is exit stagnation temperature in 'K'
T_0_star=T_01*(1+(k*Ma_1^2))^2/((k+1)*Ma_1^2*(2+((k-1)*Ma_1^2)))//T_0_star is maximum stagnation temperature in 'K'
T_02_ratio=T_02/T_0_star                    //T_02_ratio is exit stagnation temperature ratio
Ma_2_guess=0.3                              //Ma_2_guess is guess exit Mach number and it is used to determine actual exit Mach number using fsolve function
Ma_2=fsolve(Ma_2_guess,fsolve12)            //fsolve function is determine the exit Mach number
T_1_ratio=((Ma_1*(k+1))/(1+(k*Ma_1^2)))^2   //T_1_ratio is inlet temperature ratio
P_1_ratio=(1+k)/(1+(k*Ma_1^2))              //P_1_ratio is inlet pressure ratio
V_1_ratio=((1+k)*Ma_1^2)/(1+(k*Ma_1^2))     //V_1_ratio is inlet velocity ratio
T_2_ratio=((Ma_2*(k+1))/(1+(k*Ma_2^2)))^2   //T_2_ratio is outlet temperature ratio
P_2_ratio=(1+k)/(1+(k*Ma_2^2))              //P_2_ratio is outlet pressure ratio
```

```
V_2_ratio=((1+k)*Ma_2^2)/(1+(k*Ma_2^2))          //V_2_ratio is outlet velocity ratio
//T_1_ratio, P_1_ratio, V_1_ratio, T_2_ratio, P_2_ratio and V_2_ratio are calculated using the formulas given in Page 901 APPENDIX 1.
T_2=T_1*(T_2_ratio/T_1_ratio)          //T_2 is exit temperature in 'K'
P_2=P_1*(P_2_ratio/P_1_ratio)          //P_2 is exit pressure in 'kPa'
V_2=V_1*(V_2_ratio/V_1_ratio)          //V_2 is exit velocty in 'm/s'


//Display of result
mprintf('\nExit Mach number is %.4f.\nExit temperature is %d K.\nExit pressure is %d kPa.\nExit velocity is %d m/s.',Ma_2,T_2,P_2,V_2)
//The answers vary due to round off error
```

check Appendix AP 25 for dependency:

`fsolve13.sci`

Scilab code Exa 12.16 Choked Fanno Flow in a Duct

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;          //To clear the console screen
clear;          //To clear all the existing variables in the memory
exec('.\fsolve13.sci');
//Replace '.' present inside the 'exec('')' with the path to the folder location where the dependency fsolve13.sci file is saved.
```

```
//Let '1' and '2' be the same naming notations as shown in 12-66 page number 664.
//Given data
Ma_1=0.4    //Ma_1 is Mach number at the inlet
Ma_2=1    //Ma_2 is Mach number at the exit
T_1=300    //T_1 is inlet air temperature in 'K'
D=3    //D is diameter of the duct in 'cm'
P_1=150    //P_1 is inlet air pressure in 'kPa'
//Assumption
k=1.4    //k is specific heat ratio
C_p=1.005    //C_p is constant pressure specific heat in 'kJ/(kg K)'
R=0.287    //R is gas constant in 'kJ/(kg K)'
Nu=1.58E-5    //Nu is kinematic viscosity of the air in 'm2/s'
epsilon=0    //epsilon is roughness of the duct in 'm'
//Unit conversion
D=D/100    //Conversion from 'cm' to 'm'
//Calculation
//P_01/P_0_star, T_1/T_star, P_1/P_star, V_1/V_star
```

```
are represented as P_01_ratio, T_1_ratio, P_1_ratio, V_1_ratio respectively in the following codes.
epsilon_by_D=epsilon/D;        //epsilon_by_D is roughness factor
C_1=sqrt(k*R*T_1*1000)        //C_1 is speed of light in the air in 'm/s'
//Multipilcation by '1000' inside the square root is to convert the 'kJ/kg' present inside the square root to 'm2/s2'.
V_1=Ma_1*C_1        //V_1 is the inlet air velocity in 'm/s'
Re_1=V_1*D/Nu        //Re_1 is the inlet Reynolds number
f0=0.01        //f0 is guess friction factor and it is used to determine the actual friction factor using fsolve function
f=fsolve(f0,fsolve13)        //fsolve function calling statement to determine the friction factor
P_01_ratio=(1/Ma_1)*((2+((k-1)*Ma_1^2))/(k+1))^((k+1)/(2*(k-1)))//P_01_ratio is inlet stagnation pressure ratio
T_1_ratio=(k+1)/(2+((k-1)*Ma_1^2))        //T_1_ratio is inlet temperature ratio
P_1_ratio=(1/Ma_1)*((k+1)/(2+((k-1)*Ma_1^2)))^0.5        //P_1_ratio is inlet pressure ratio
V_1_ratio=Ma_1*((k+1)/(2+((k-1)*Ma_1^2)))^0.5        //V_1_ratio is inlet velocity raito
fl_by_D=((1-Ma_1^2)/(k*Ma_1^2))+(((k+1)/(2*k))*log(((k+1)*Ma_1^2)/(2+((k-1)*Ma_1^2))))//fl_by_D is inlet pipe number
//P_01_ratio, T_1_ratio, P_1_ratio, V_1_ratio and fl_by_D are calculated using the formulas given in Page 902 APPENDIX 1.
L_1_star=fl_by_D*D/f        //L_1_star is the duct length in 'm'
```

```
T_star=T_1/T_1_ratio    //T_star is exit pressure in 'K'
P_star=P_1/P_1_ratio    //P_star is exit pressure in 'kPa'
V_star=V_1/V_1_ratio    //V_star is exit velocity in 'm/s'
FractionLost=1-1/P_01_ratio


//Display of Re_1sult
mprintf('\nDuct length is %.2f m.\nExit temperature is %d K.\nExit pressure is %.1f kPa.\nExit Velocity is %d m/s.\nFraction of stagnation pressure lost is %.3f or %.1f Percentage.',L_1_star,T_star,P_star,V_star,FractionLost,FractionLost*100)
//The answers vary due to round off error
```

check Appendix AP 24 for dependency:

`fsolve14.sci`

Scilab code Exa 12.17 Exit Conditions of Fanno Flow in a Duct

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                                    //To clear the console screen
clear;                                  //To clear all the existing variables in the memory
exec('.\fsolve14.sci');
//Replace '.' present inside the 'exec('')' with the path to the folder location where the dependency fsolve14.sci file is saved.


```

```
//Let '1' and '2' be the same naming notations as shown in 12-67 page number 665.
//Given data
L=27                                         //L is length of the duct in 'm'
D=5                                          //D is diameter of the duct in 'cm'
V_1=85                                       //V_1 is inlet air velocity in 'm/s'
T_1=450                                      //T_1 is inlet air temperature in 'K'
P_1=220                                      //P_1 is inlet air pressure in 'kPa'
f=0.023                                      //f is average friction factor for the duct


//Unit conversion
D=D/100                                      //Conversion from 'cm' to 'm'


//Assumption
k=1.4                                        //k is specific heat ratio
C_p=1.005                                    //C_p is constant pressure specific heat in 'kJ/(kg K)'
R=0.287                                      //R is gas constant in 'kJ/(kg K)'


//Calculation
C_1=sqrt(k*R*T_1*1000)                       //C_1 is speed of light in the air in 'm/s'
//Multipilcation by '1000' inside the square root is to convert the 'kJ/kg' present inside the square root to 'm2/s2'.
```

```
Ma_1=V_1/C_1                                    //Ma_1 is Mach number at the inlet
fl_star_by_D_in=((1-Ma_1^2)/(k*Ma_1^2))+(((k+1)/(2*k))*log(((k+1)*Ma_1^2)/(2+((k-1)*Ma_1^2))))
//fl_star_by_D_in is inlet pipe number calculated using sonic length instead of actual length
fl_by_D=f*L/D                                   //fl_by_D is actual pipe number
if fl_by_D<fl_star_by_D_in then
    flow="not choked"
    fl_star_by_D_out=(fl_star_by_D_in)-(fl_by_D)
    Ma_2_guess=1                                //'Ma_2_guess' is guess exit Mach number and it is used to determine the actual Mach number using fsolve function
    Ma_2=fsolve(Ma_2_guess,fsolve14)            //Function calling statement to determine the 'Ma_2'
else
    flow="chocked";
    Ma_2=1                                      //Ma_2 is Mach number at the exit
end
A_1=%pi*D^2/4                                   //A_1 is area of the duct in 'm2'
m_air=P_1*A_1*V_1/(R*T_1)                       //m_air is mass flow rate of air in 'kg/s'


//Display of result
mprintf('\nFlow is %s.\nThe Mach number at the duct exit is %.2f.\nThe mass flow rate of air is %.3f kg/s.',flow,Ma_2,m_air)
//The answers vary due to round off error
```

Chapter 13

OPEN CHANNEL FLOW

check Appendix AP 19 for dependency:

fsolve15.sci

Scilab code Exa 13.1 Character of Flow and Alternate Depth

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                                //To clear the
   console screen
clear;                              //To clear all the
   existing variables in the memory
exec('.\fsolve15.sci');
//Replace '.' present inside the 'exec('')' with the
    path to the folder location where the dependency
    fsolve15.sci file is saved.


//Given data
b=0.4                               //b is width of
   rectangular channel in 'm'
V_dot=0.2                           //V_dot is volume
   flow rate of water in 'm3/s'
```

```
y_1=0.15                         //y_1 is the flow depth in 'm'


//Assumption
g=9.81                           //g is acceleration due to gravity in 'm/s2'


//Calculation
A_c=b*y_1                        //A_c is area of the channel in 'm2'
V=V_dot/(A_c)                    //V is the average flow velocity in 'm/s'
y_1_c=(V_dot/(g*b^2))^(1/3)      //y_1_c is the critical depth for the flow in 'm'
Fr=V/sqrt(g*y_1)                 //Fr is the Froude number
if Fr<1 then
    Flow="subcritical or tranquil"
else
    if Fr==1
        Flow="critical"
    else
        if Fr>1
            Flow="supercritical"
        end
    end
end
E_s1=y_1+((V_dot^2)/(2*g*b^2*y_1^2))     //E_s1 is specific energy in 'm'
E_s2=E_s1                                //m
y_2_Guess=1                              //y_2_Guess is guess alternate depth used is to find actual alternative 'y_1' using the fsolve function
y_2=fsolve(y_2_Guess,fsolve15)           //Calling statement for fsolve function

```

```
//Display of result
mprintf('\nAverage flow velocity is %.2f m/s.\nFlow
    is %s.\nAlternate flow depth is %.2f m.',V,Flow,
    y_2)
```

check Appendix AP 18 for dependency:

`ManningEquation1.sci`

Scilab code Exa 13.2 `Flow Rate in an Open Channel in Uniform Flow`

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                              //To clear the
    console screen
clear;                            //To clear all the
    existing variables in the memory
exec('.\ManningEquation1.sci');
//Replace '.' present inside the 'exec('')' with the
     path to the folder location where the dependency
     ManningEquation1.sci file is saved.


//Given data
b=0.8                             //b is bottom width
    of the channel in 'm'
theta=60                          //theta is trapezoid
     angle in '  (degree)'
alpha_1=0.3                       //alpha_1 is bottom
    slope 1 of the channel in '  (degree)'
alpha_2=1                         //alpha_2 is bottom
    slope 2 of the channel in '  (degree)'
y=0.52                            //y is flow depth in
     'm'

```

```
//Unit conversion
theta=theta*%pi/180              //Convesion from '(degree)' to 'radian'
alpha_1=alpha_1*%pi/180          //Convesion from '(degree)' to 'radian'
alpha_2=alpha_2*%pi/180          //Convesion from '(degree)' to 'radian'


//Assumption
n=0.030                          //n is dimension less Manning coefficient
a=1                              //a is a factor with uint 'm^(1/3)/s'


//Calculation
A_c=y*(b+(y/tan(theta)))         //A_c is cross sectional area of the channel in 'm2
p=b+((2*y)/(sin(theta)))         //p is perimeter of the channel in 'm'
R_h=A_c/p                        //R_h is radius of the channel in 'm'
S0_1=tan(alpha_1)                //S0_1 is the bottom slope 1
S0_2=tan(alpha_2)                //S0_2 is the bottom slope 2
V_dot_1=ManningEquation1(alpha_1)//V_dot_1 is volume flow rate in 'm3/s' measured at bottom slope S0_1
V_dot_2=ManningEquation1(alpha_2)//V_dot_2 is volume flow rate in 'm3/s' measured at bottom slope S0_2


//Display of result
mprintf('\nVolume flow rate at bottom slope of %.1
```

```
f   is %.2f m3/s.\nVolume flow rate at bottom
   slope of %d  is %.1f m3/s.',alpha_1*180/%pi,
   V_dot_1,alpha_2*180/%pi,V_dot_2)
```

check Appendix AP 16 for dependency:

`fsolve16.sci`

check Appendix AP 17 for dependency:

`fsolve17.sci`

Scilab code Exa 13.3 `The Height of an Rectangular Channel`

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                          //To clear the console
   screen
clear;                        //To clear all the existing
   variables in the memory


//First fsolve16.sci file is executed for finding '
   y1' then fsolve17.sci is executed for finding 'y2
   '.
exec('.\fsolve16.sci');
//Replace '.' present inside the 'exec('')' with the
    path to the folder location where the dependency
    fsolve16.sci file is saved.
exec('.\fsolve17.sci');
//Replace '.' present inside the 'exec('')' with the
    path to the folder location where the dependency
    fsolve17.sci file is saved.


//Given data
```

```
b=4                         //b is bottom width of the channel in 'ft'
V_dot=51                    //V_dot is volume flow rate of water in 'ft3/s'
Bottom_Drop_1=2             //ft/(1000 ft length)
Bottom_Drop_2=1             //ft/(1000 ft length)


//Assumption
n=0.014                     //n is dimension less Manning coefficient
a=1.486                     //a is a factor with uint 'm^(1/3)/s'


//Calculation
S0_1=Bottom_Drop_1/1000             //S0_1 is the bottom slope 1
S0_2=Bottom_Drop_2/1000             //S0_2 is the bottom slope 2
y1_Guess=1                          //y1_Guess(in 'ft') is the guess value of height 1 used in fsolve function to determine the actual height 1 'y1'.
y2_Guess=1                          //y2_Guess(in 'ft') is the guess value of height 2 used in fsolve function to determine the actual height 2 'y2'.
y1=fsolve(y1_Guess,fsolve16)     //Calling statement for fsolve function to find height 1 'y1'
y2=fsolve(y2_Guess,fsolve17)     //Calling statement for fsolve function to find height 2 'y2'


//Display of result
mprintf('\nHeight of the channel when bottom drop is %d ft/(1000 ft length) is %.1f ft.\nHeight of the channel when bottom drop is %d ft/(1000 ft length) is %.1f ft.',Bottom_Drop_1,y1,Bottom_Drop_2,y2)
```

check Appendix ?? for dependency:

FIGURE_13_20.jpg

Scilab code Exa 13.4 Channels with Nonuniform Roughness

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                          //To clear the console
    screen
clear;                        //To clear all the existing
    variables in the memory


//Let '1' and '2' be the same numbering notations as
     that of FIGURE 13-20 in page number 696
//Given data
S0=0.003                      //S0 is bottom slope of the
    channel
n1=0.030                      //n1 is manning coefficient
    in section 1
n2=0.050                      //n2 is manning coefficient
    in section 2
Height_1=2                    //Height_1 is height of
    section 1 in 'm'
Width_1=6                     //Width_1 is width of the
    section 1 in 'm'
Slope_Height_1=3              //Slope_Height_1 is slope
    length of section 1 in 'm'
Height_2=2                    //Height_2 is height of the
    section 2 in 'm'
Width_2=8                     //Width_2 is width of the
    section 2 in 'm'


```

```
//Assumption
a=1                    //a is a factor with uint 'm^(1/3)/s'


//Calculation
Slope_Length_1=sqrt(Slope_Height_1^2+(Width_1/2)^2)       //Slope_Height_2 is slope length of section 2 in 'm'
A_c_1=(Height_1*Width_1)+(0.5*Slope_Height_1*Width_1)    //A_c_1 is section 1 flow area in 'm2'
p1=Height_1+(2*Slope_Length_1)                            //p1 is perimeter of section 1 in 'm'
R_h_1=A_c_1/p1                                            //R_h_1 is hydraulic radius of section 1 in 'm'
A_c_2=Height_2*Width_2                                    //A_c_2 is section 2 flow area in 'm2'
p2=Width_2+Height_2                                       //p2 is perimeter of section 2 in 'm'
R_h_2=A_c_2/p2                                            //R_h_2 is hydraulic radius of section 2 in 'm'
A_c=A_c_1+A_c_2                                           //A_c is flow area of entire channel in 'm2'
p=p1+p2                                                   //p is perimeter of the entire channel in 'm'
R_h=(A_c_1+A_c_2)/(p1+p2)                                 //R_h is hydraulic radius of the entire section in 'm'
Q=((A_c_1*R_h_1^(2/3)/n1)+(A_c_2*R_h_2^(2/3)/n2))*S0^(1/2)*a//Q is volume flow rate through the channel in 'm3/s'
```

```
n_eff=a*A_c*(R_h)^(2/3)*(S0)^(1/2)/Q                                  //n_eff is manning
    coefficient for the entire channel


//Display of result
mprintf('\nVolume flow rate through the channel is
    %d m3/s.\nThe effective manning co-efficient is %
    .3f.',Q,n_eff)
//The answers vary due to round off error
```

check Appendix ?? for dependency:

FIGURE_13_26.jpg

Scilab code Exa 13.5 Best Cross Section of an Open Channel

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                    //To clear the console screen
clear;                  //To clear all the existing
    variables in the memory


//Given data
V_dot=2                 //V_dot is volume flow rate in '
    m3/s'
S0=0.001                //S0 is bottom slope


//Assumption
n=0.016                 //n is dimension less Manning
    coefficient
a=1                     //a is a factor with uint 'm
    ^(1/3)/s'
```

```
theta=60              //theta is trapezoid angle in '(degree)'


//Unit conversion
theta=theta*%pi/180 //Conversion from ' (degree)' to 'radian'


//Part (a)
//Calculation
b1=(2*n*V_dot*4^(2/3)/a/sqrt(S0))^(3/8) //b1 is width of the channel in 'm'
y1=b1/2                                  //y1 is height of the channel in 'm'


//Display of result
mprintf('\n(a) Best height of rectangular channel is %.2f m.\n     Best width of the rectangular channel is %.2f m.',y1,b1)


//Part (b)
//Calculation
b2=(n*V_dot/((1+cos(theta))*sqrt(3)*0.5*(sqrt(3)/4)^(2/3)*a*sqrt(S0)))^(3/8)//b2 is width of the channel in 'm'
A_c=0.5*sqrt(3)*b2^2*(1+cos(theta))                                    //A_c is area of the channel in 'm2'
p=3*b2                                                                 //p is perimeter of the channel in 'm'
y2=0.5*sqrt(3)*b2                                                      //y2 is height of the channel in 'm'

```

```
//Display of result
mprintf('\n\n(b) Best dimensions of trapezoidal
    channel are,\n     Width = %.2f m.\n     Height = %
    .3f m.\n     Trapezoid angle = %d .',b2,y2,theta
    *180/%pi)
//The answers vary due to round off error
```

check Appendix AP 13 for dependency:

`ManningEquation2.sci`

Scilab code Exa 13.6 Classification of Channel Slope

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                             //To clear the console
    screen
clear;                           //To clear all the
    existing variables in the memory
exec('.\ManningEquation2.sci');
//Replace '.' present inside the 'exec('')' with the
     path to the folder location where the dependency
     ManningEquation2.sci file is saved.


//Given data
b=6                              //b is width of the
    channel in 'm'
y=2                              //y is flow depth in 'm'
S0=0.004                         //S0 is bottom slope of
    the channel


//Assumption
```

```
n=0.014                        //n is dimension less
   Manning coefficient
a=1                            //a is a factor with
   uint 'm^(1/3)/s'
g=9.81                         //g is acceleration due
   to gravity in 'm/s2'


//Calculation
A_c=y*b                        //A_c is cross sectional
    area of the channel in 'm2'
p=b+(2*y)                      //p is perimeter of the
   channel in 'm'
R_h=A_c/p                      //R_h is hydraulic
   radius of the channel in 'm'
y_n=y                          //y_n is normal depth in
    'm' when flow is uniform
V_dot=ManningEquation2()        //'ManningEquation'
   function calling statement
y_c=V_dot^2/(g*A_c^2)          //y_c is critical depth
   of the flow in 'm'
if y_c<y_n then
    channelslope="mild"
else
    if y_c>y_n then
        channelslope="steep"
    else
        if y_c == y_n then
            channelslope="critical"
        end
    end
end


//Display of result
mprintf('\nVolume flow rate is %.1f m3/s.',V_dot)
mprintf('\nCritical depth is %.1f m.',y_c) //The
   answer provided in the textbook is wrong
```

```
mprintf('\nSo the channel is %s.',channelslope)
```

Scilab code Exa 13.7 Hydraulic Jump

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                    //To clear the console screen
clear;                  //To clear all the existing variables in the memory


//Let '1' and '2' be the same numbering notations as that of FIGURE 12-35 page number 712.
//Given data
b=10                    //b is width of the rectangular channel in 'm'
y1=0.8                  //y1 is flow depth before the jump in 'm'
v1=7                    //v1 is velocity before the jump in 'm/s'


//Assumption
rho=1000                //rho is density of water in 'kg/m3'
g=9.81                  //g is acceleration due to gravity in 'm/s2'


//Calculation
Fr1=v1/sqrt(g*y1)     //Fr1 is Froude number before the hydraulic jump
if Fr1>1 then
    //Part (a)
    Flow="supercritical"
```

```
y2=0.5*y1*(-1+sqrt(1+(8*Fr1^2)))     //y2 is flow depth after the jump in 'm'
v2=y1*v1/y2                          //v2 is velocity after the jump in 'm/s'
Fr2=v2/sqrt(g*y2)                    //Fr2 is Froude number after the jump


//Display of result
mprintf('\n(a) Flow is %s.\n    Depth after jump is %.2f m.\n    Velocity after jump is %.2f m/s.\n    Froude number after jump is %.3f.',Flow,y2,v2,Fr2)
//The answers vary due to round off error


//Part (b)
h_L=y1-y2+((v1^2-v2^2)/(2*g))        //h_L is head loss in 'm'
E_s1=y1+(v1^2/(2*g))                 //E_s1 is specific energy of water before the jump in 'm'
Dissipation_Ratio=h_L/E_s1


//Display of result
mprintf('\n\n(b) Head loss is %.3f m.\n    Specific energy of water before jump is %.2f m.\n    Dissipation ratio is %.3f.',h_L,E_s1,Dissipation_Ratio)
//The answers vary due to round off error


//Part (c)
m=rho*b*y1*v1                        //m is mass flow rate of water in 'kg/s'
E_dissipated=m*g*h_L                 //E_dissipated is power dissipation in 'W'
```

```
    E_dissipated=E_dissipated/1000      //Conversion
        from 'W' to 'kW'


    //Display of result
    mprintf('\n\n(c) Mass flow rate of water is %d
       kg/s.\n    Power dissipated is %d kW.',m,
       E_dissipated)
    //The answers vary due to round off error
else
    mprintf('\n(a) Flow is subcritical so, hydraulic
        jump is not possible...!')
end
```

check Appendix ?? for dependency:

FIGURE_13_38.jpg

Scilab code Exa 13.8 Sluice Gate with Drowned Outflow

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                          //To clear the console
   screen
clear;                        //To clear all the
   existing variables in the memory


//Given data
y1=3                          //y1 is depth of the
   reservoir in 'm'
b=6                           //b is the width of the
   channel to which water is released in 'm'
a=0.25                        //a is height of the
   sluice gate in 'm'
```

```
y2=1.5                        //y2 is flow depth after
    all turbulence subside in 'm'


//Assumption
g=9.81                        //g is acceleration due
   to gravity in 'm/s2'


//Calculation
y1_by_a=y1/a                  //y1_by_a is the depth
   ratio
y2_by_a=y2/a                  //y2_by_a is the
   contraction coefficient
C_d=0.47                      //Determined from FIGURE
    13-38 (page number 715) using 'y1_by_a' and '
   y2_by_a'.
V_dot=C_d*b*a*sqrt(2*g*y1)   //V_dot is rate of
   discharge of the water in 'm3/s'


//Display of result
mprintf('\nThe rate of discharge is %.2f m3/s.',
   V_dot)
```

Scilab code Exa 13.9 Subcritical Flow over a Bump

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                          //To clear the console
   screen
clear;                        //To clear all the
   existing variables in the memory


```

```
//Given data
delta_z_b=15                     //delta_z_b is height of bumb in 'cm'
y1=0.80                          //y1 is flow depth in 'm'
v1=1.2                           //v1 is water velocity before the bump in 'm/s'


//Unit conversion
delta_z_b=delta_z_b/100          //Conversion from 'cm' to 'm'


//Assumption
g=9.81                           //g is acceleration due to gravity in 'm/s2'


//Calculation
Fr1=v1/sqrt(g*y1)                //Fr1 is upstream Froude number
y_c=(y1^2*v1^2/g)^(1/3)          //y_c is upstream critical depth in 'm'
E_s1=y1+(v1^2/(2*g))             //E_s1 is upstream specific energy in 'm'
if Fr1<1 then
    flow="Subcritical"
    Possible_y2=roots([1,-(E_s1-delta_z_b),0,(v1^2*y1^2/(2*g))])
    for i=1:3//This loop is to eliminate negative depth and depth which is less than the critical depth
            if real(Possible_y2(i))>0 then
                if real(Possible_y2(i))>y_c then
                    y2=real(Possible_y2(i))      //y2 is flow depth over the bump in 'm'
```

```
                end
            end
    end
    Height_Over_Bump=delta_z_b+y2 //Height_Over_Bump is flow depth after the bump in 'm'
    if Height_Over_Bump<y1 then
        condition="depressed"
        Depression=y1-Height_Over_Bump //Depression is difference of flow depth before the bump and over the bump in 'm'
        mprintf('\nFlow is %s.\nWater surface if %s over the bump in the amount of %.2f m.',flow,condition,Depression)
    else
        condition="elevated"
        Elevation=Height_Over_Bump-y1 //Elevation is difference of flow depth over the bump and before the bump in 'm'
        mprintf('\nFlow is %s.\nWater surface if %s over the bump in the amount of %.2f m.',flow,condition,Elevation)
    end
else
    flow="supercritical"
    Possible_y2=roots([1,-(E_s1-delta_z_b),0,(v1^2*y1^2/(2*g))])
    for i=1:3//This loop is to eliminate negative depth and depth which is less than the critical depth
            if real(Possible_y2(i))>0 then
                if real(Possible_y2(i))<y_c then
                    y2=real(Possible_y2(i)) //y2 is flow depth over the bump in 'm'
                end
            end
    end
```

```
    Height_Over_Bump=delta_z_b+y2 //Height_Over_Bump is flow depth after the bump in 'm'
    if Height_Over_Bump<y1 then
        condition="depressed"
        Depression=y1-Height_Over_Bump //Depression is difference of flow depth before the bump and over the bump in 'm'
        mprintf('\nFlow is %s.\nWater surface if %s over the bump in the amount of %.2f m.',flow,condition,Depression)
    else
        condition="elevated"
        Elevation=Height_Over_Bump-y1 //Elevation is difference of flow depth over the bump and before the bump in 'm'
        mprintf('\nFlow is %s.\nWater surface if %s over the bump in the amount of %.2f m.',flow,condition,Elevation)
    end
end
```

Scilab code Exa 13.10 Measuring Flow Rate by a Weir

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                        //To clear the console screen
clear;                      //To clear all the existing variables in the memory


//Given data
b=5                         //b is width of the open channel in 'm'
```

```
P_w=0.6                    //P_w height of the weir in
    'm'
W_w=b                      //W_w is width of the weir
    in 'm'
y1=1.5                     //y1 is depth of water in
    upstream in 'm'


//Assumption
g=9.81                     //g is acceleration due to
    gravity in 'm/s2'


//Calculation
H=y1-P_w                   //H is weir height in 'm'
if (H/P_w)<2 then
    C_wd_rec=0.598+((0.0897)*(H/P_w))
                            //C_wd_rec is discharge
        coefficient of the weir
    V_dot_rec=C_wd_rec*(2/3)*b*(H)^(3/2)*sqrt(2*g)
               //V_dot_rec is flow rate through the
        channel in 'm3/s'
    mprintf('\nWater flow rate through the channel
        is %.2f m3/s.',V_dot_rec)
    //The answers vary due to round off error
else
   mprintf('\nConditions required to calculate weir
       discharge co-efficient is not satisfied!!!\
       nThat is (H/P_w)>2')
end
```

Chapter 14

TURBOMACHINERY

check Appendix ?? for dependency:

TABLE_14_1.jpg

check Appendix AP 11 for dependency:

fsolve18.sci

Scilab code Exa 14.1 Operating Point of a Fan in a Ventilation System

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                                                    //To
   clear the console screen
clear;                                                  //To
   clear all the existing variables in the memory
clf(0)                                                  //Clear
   or reset or reset a figure 0
exec('.\fsolve18.sci');
//Replace '.' present inside the 'exec('')' with the
    path to the folder location where the dependency
    fsolve18.sci file is saved.


```

```
//Given data
V_dot_from_table=[0 250 500 750 1000 1200]   //
    V_dot_from_table is volume flow rate in 'cfm'
    from TABLE 14-1
H_available=[0.90 0.95 0.90 0.75 0.40 0]     //
    H_available is head in 'in of water' from TABLE
    14-1
D=9.06                                       //D is
    the inner diameter of the duct in 'in'
L=44                                         //L is
    the total length of duct in 'ft'
epsilon=0.15                                 //
    epsilon is the equivalent roughness height in 'mm
    '
V_dot_given=600                              //
    V_dot_given is the minimum volume flow rate
    through the duct in 'ft3/min'
T=25                                         //T is
    the temperature of duct in ' C '
Elbow=5                                      //Elbow
    is the number of elbows along the duct
Damper=1                                     //Damper
     is the number of dampers present along the duct
Entry_K_L=1.3                                //
    Entry_K_L is the entry loss coefficient
Damper_K_L=1.8                               //
    Damper_K_L is the damper loss coefficient
Elbow_K_L=0.21                               //
    Elbow_K_L is the elbow loss coefficient
Fan_ID=9                                     //Fan_ID
     is fan inlet diameter in 'in'
Fan_OD=0                                     //Fan_OD
     is the fan outlet diameter in 'in'


//Assumption
v=1.562E-5                                   //v is
    the air vicosity in 'm2/s'
```

```
rho_air=1.184                                    //rho_air is the air density in 'kg/m3'
rho_water=998                                    //rho_water is the water density in 'kg/m3'
P1=101325                                        //P1 is the pressure in the reservoir in 'Pa'
P2=101325                                        //P2 is the prssure in the pipe outlet in 'Pa'
V1=0                                             //V1 is the velocity in the reservoir in 'm/s'
Alpha1=1.05                                      //Assuming the flow to be turblent
Alpha2=1.05                                      //Assuming the flow to be turblent
g=9.81                                           //g is the acceleration due to gravity in 'm/s2'
V_dot_min=1                                      //V_dot_min is the minimum volume flow rate for calculation in 'ft3/min'
V_dot_max=1200                                   //V_dot_max is the maximum volume flow rate for calculation in 'ft3/min'


//Unit conversion
D=D*0.0254                                       //Conversion from 'in' to 'm'
epsilon=epsilon/1000                             //Conversion from 'mm' to 'm'
L=L*12*0.0254                                    //Conversion from 'ft' to 'm'


//Calculation
f0=0.01                                          //f0 is guess friction factor which is used to determine the actual friction factor using fsolve
```

```
function
Sigma_K_L=Entry_K_L+(Damper*Damper_K_L)+(Elbow*Elbow_K_L)                //Sigma_K_L is total loss coefficient
R=D/2                                                                     //R is radius of the duct in 'm'
A=%pi*R^2                                                                 //A is the CSA of the duct in 'm2'
epsilon_by_D=epsilon/D                                                    //epsilon_by_D is the roughness factor
V_dot=V_dot_min:1:V_dot_max
m=length(V_dot)                                                           //m is the number of volume flow rate generated
for i=1:m
    V_dot(i)=V_dot(i)*(12*0.0254)^3/60                                    //Conversion from 'ft3/min' to 'm3/s'
    Re(i)=4*V_dot(i)/(v*%pi*D)                                            //Re is dimension less Reynolds number
    Rey=Re(i)                                                             //Rey is Reynolds number used in the 'fsolve18' function
    f(i)=fsolve(f0,fsolve18)                                              //calling statement for fsolve function
    V2(i)=V_dot(i)/A                                                      //V2 is the velocity at the pipe outlet in 'm/s'
    H_required(i)=(Alpha2+(f(i)*L/D)+Sigma_K_L)*(V2(i)^2/(2*g))           //H_required is head in 'm of air'
```

```
    H_required_inchesofwater(i)=H_required(i)*(
       rho_air/rho_water)        //
       H_required_inchesofwater is head in 'm of
       water'
    H_required_inchesofwater(i)=
       H_required_inchesofwater(i)/0.0254       //
       Conversion from 'm of water' to 'in of water'
    V_dot(i)=V_dot(i)/(12*0.0254)^3*60
                                          //Conversion
        from 'm3/s' to 'ft3/min'
end
plot(V_dot',H_required_inchesofwater,'k',
   V_dot_from_table',H_available','r.-')
xlabel("V_dot,cfm")
ylabel("H,inches H2O")
legend(['H_required';'H_available'],2)
title("FIGURE 41-13: Net head as a function of
   volume flow rate for the ventilation system of
   Example 14-1.The point where the available and
   required values of H interest is the operating
   point")
//Following values are obtained from the point of
   intersection of the H_required and H_available
   curve present in FIGURE 14-13
V_dot_operating=650

   //x co-ordinate of the point of intersection in '
   cfm'
H_required_Operating=0.83
                                                  //y
    co-ordinate of the point of intersection in '
   inches of water'
H_available_operating=0.83
                                                 //y
   co-ordinate of the point of intersection in '
   inches of water'


```

```
//Display of result
mprintf('\nOperating point volume flow rate is %d cfm.\nRequired and available at this operating point are %.2f and %.2f inches of water respectively.',V_dot_operating,H_required_Operating,H_available_operating)
```

check Appendix ?? for dependency:

`FIGURE_14_15.jpg`

Scilab code Exa 14.2 `Selection of Pump Impeller Size`

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                            //To clear the console screen
clear;                          //To clear all the existing variables in the memory


//Given data
V_dot=370                       //V_dot is the volume flow rate in 'gpm'
head=24                         //head is the required net head in 'ft'
Impeller1=8.25                  //in
Impeller2=12.75                 //in
speed=1160                      //'speed' is the pump speed in 'rpm'
H_Impeller1=24                  //H_Impeller1 is impeller1's head in 'ft'
Impeller1_Eta_pump=0.7          //Impeller1_Eta_pump is the pump efficiency when Impeller1 is used
Impeller2_Eta_pump=0.765        //Impeller2_Eta_pump is the pump efficiency when Impeller2 is used
```

```
//Unit conversion
V_dot=V_dot*0.1337/60        //Conversion from 'gpm' to 'ft3/s'

//Assumption
rho=62.30                    //rho is water density in 'lbm/ft3'
g=32.2                       //g is acceleration due to gravity in 'ft/s2'

//Calculation
bhp_Impeller1=rho*g*V_dot*H_Impeller1/(Impeller1_Eta_pump)  //bhp_Impeller1 is the bhp required for impeller1 in '(lbm ft2)/s3'
bhp_Impeller1=bhp_Impeller1/32.2                              //Conversion from '(lbm ft2)/s3' to '(lbf ft)/s'
bhp_Impeller1=bhp_Impeller1/550                               //Conversion from '(lbf ft)/s' to 'hp'
H_Impeller2=72                                                //H_Impeller2 is impeller2's head in 'ft' (Obtained from FIGURE 14-15 using the point of intersection of 12.5 inche curve and V_dot=370 gpm line)
bhp_Impeller2=rho*g*V_dot*H_Impeller2/(Impeller2_Eta_pump)  //bhp_Impeller2 is the bhp required for impeller 2 in '(lbm ft2)/s3'
bhp_Impeller2=bhp_Impeller2/32.2                              //Conversion from '(lbm ft2)/s3' to '(lbf ft)/s'
bhp_Impeller2=bhp_Impeller2/550                               //Conversion from '(
```

```
lbf ft)/s' to 'hp'


//Display of result
mprintf('\nbhp for %.2f in impeller option is %.2f hp.\nbhp for %.2f in impeller option is %.2f hp.',Impeller1,bhp_Impeller1,Impeller2,bhp_Impeller2)
//The answers vary due to round off error
if bhp_Impeller1<bhp_Impeller2 then
    mprintf('\nClearly, the smaller diameter impeller option is the better choice because it uses less power.')
else
    mprintf('\nClearly, the larger diameter impeller option is the better choice because it uses less power.')
end
```

---

check Appendix AP 7 for dependency:

`Colebrook.sci`

check Appendix **??** for dependency:

`FIGURE_14_15.jpg`

check Appendix AP 5 for dependency:

`fsolve19.sci`

check Appendix AP 6 for dependency:

`fsolve20.sci`

Scilab code Exa 14.3 `Maximum Flow Rate to Avoid Pump Cavitation`

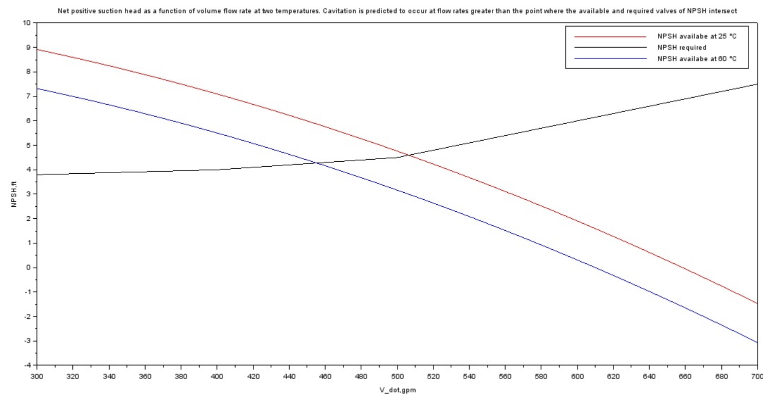


Figure 14.1: Maximum Flow Rate to Avoid Pump Cavitation

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                              //To clear the console screen
clear;                            //To clear all the existing variables in the memory
clf(0)                            //Clear or reset or reset a figure 0


//Order of dependency file execution: 1.Colebrook.sci, 2.fsolve19.sci, 3.fsolve20.sci
exec('.\Colebrook.sci');
//Replace '.' present inside the 'exec('')' with the path to the folder location where the dependency Colebrook.sci file is saved.
exec('.\fsolve19.sci');
//Replace '.' present inside the 'exec('')' with the path to the folder location where the dependency fsolve19.sci file is saved.
exec('.\fsolve20.sci');
//Replace '.' present inside the 'exec('')' with the
```

```
path to the folder location where the dependency fsolve20.sci file is saved.


//Given data
T1=25                   //T1 is temperature1 in ' C '
Impeller=11.25          //Impeller is the impeller option in 'in'
Z1=4                    //Z1 is the height of the reservoir in 'ft'
Z2=0                    //Z2 is the height of the pump in 'ft'
L=10.5                  //L is the length of the pipe used in 'ft'
D=4                     //D is the inner diameter of the pipe in 'in'
Epsilon=0.02            //Epsilon is the average inner roughness height in 'in'
Inlet_K_L=0.5           //Inlet_K_L is the inlet loss coefficient
Inlet=1                 //Inlet is the number of inlets in the system
Elbow_K_L=0.3           //Elbow_K_L is the elbow loss coefficient
Elbow=3                 //Elbow is the number of elbow in the system
Valve_K_L=6             //Valve_K_L is the valve loss coefficient
Valve=1                 //Valve is the number of valve in the system


//Assumption
T2=60                   //T2 is the temperature2 in  C
P_atm=101325            //P_atm is the atmospheric pressure in 'Pa'
P_v_at_T1=3169          //P_v_at_T1 is the vapor pressure at T1 in 'Pa'
```

```
P_v_at_T2=19940       //P_v_at_T2 is the vapor
   pressure at T2 in 'Pa'
rho_at_T1=997         //rho_at_T1 is the water density
    at T1 in 'kg/m3'
rho_at_T2=983.3       //rho_at_T2 is the water density
    at T2 in 'kg/m3'
Mu_at_T1=8.91E-4      //Mu_at_T1 is the water
   viscosity at T1 in 'kg/(m s)'
Mu_at_T2=4.67E-4      //Mu_at_T2 is the water
   viscosity at T2 in 'kg/(m s)'
V1=0                  //V1 is the velocity in 'm/s'
P1=P_atm              //P1 is the pressure at
   reservoir in 'Pa'
h_pump=0              //h_pump is the head loss due to
    pump in 'm'
h_turbine=0           //h_turbine is the head loss due
    to turbine in 'm'
Alpha1=1.05           //Alpha1 is the velocity
   correction factor(Assuming the flow to be
   turblent)
Alpha2=1.05           //Alpha2 is the velocity
   correction factor(Assuming the flow to be
   turblent)


//Assumption
V_dot_min=300

   //V_dot_min is the minimum volume flow rate in '
   gpm' for calculation and graph ploting
V_dot_max=700

   //V_dot_max is the maximum volume flow rate in '
   gpm' for calculation and graph ploting
g=9.81

   //g is the acceleration due to gravity in 'm/s2'

```

```
//Unit conversion
D=D*0.0254                                        //Conversion from 'in' to 'm'
Epsilon=Epsilon*0.0254                                        //Conversion from 'in' to 'm'
L=L*12*0.0254                                        //Conversion from 'ft' to 'm'
Z1=Z1*12*0.0254                                        //Conversion from 'ft' to 'm'
Z2=Z2*12*0.0254                                        //Conversion from 'ft' to 'm'


//Calculation
R=D/2                                        //R is the radius in 'm'
Area=%pi*R^2                                        //Area is the pipe CSA in 'm2'
Epsilon_by_D=Epsilon/D                                        //Epsilon_by_D is the roughness factor
Sigma_K_L=(Inlet_K_L*Inlet)+(Elbow_K_L*Elbow)+(
    Valve_K_L*Valve) //Sigma_K_L is the total loss
    coefficient
V_dot=V_dot_min:1:V_dot_max
m=length(V_dot)                                        //'length' function is used to determine the
    number volume flow rate generated
for i=1:m
```

```
    V_dot(i)=V_dot(i)*0.00378541/60 //Conversion from 'gpm' to 'm3/s'
    V2(i)=V_dot(i)/Area //V2 is velocity in 'm/s'
    Re_at_T1(i)=rho_at_T1*V2(i)*D/Mu_at_T1 //Re_at_T1 is the Reynolds number at T1
    f_at_T1(i)=Colebrook(Epsilon_by_D,Re_at_T1(i)) //f_at_T1 is the friction factor at T1 computed using colebrook function
    h_L_at_T1(i)=((f_at_T1(i)*L/D)+Sigma_K_L)*(V2(i)^2/(2*g)) //h_L_at_T1 is the head loss at T1 in 'm'
    NPSH_at_T1(i)=((P_atm-P_v_at_T1)/(rho_at_T1*g))+Z1-Z2-h_L_at_T1(i)-((Alpha2-1)*V2(i)^2/(2*g))
end
for i=1:m
    V2(i)=V_dot(i)/Area //V2 is velocity in 'm/s'
    Re_at_T2(i)=rho_at_T2*V2(i)*D/Mu_at_T2 //Re_at_T2 is the reynolds number at T2
    f_at_T2(i)=Colebrook(Epsilon_by_D,Re_at_T2(i)) //f_at_T2 is the friction factor at T2 computed using colebrook function
    h_L_at_T2(i)=((f_at_T2(i)*L/D)+Sigma_K_L)*(V2(i)^2/(2*g)) //h_L_at_T2 is the head loss at T2 in 'm'
    NPSH_at_T2(i)=((P_atm-P_v_at_T2)/(rho_at_T2*g))+Z1-Z2-h_L_at_T2(i)-((Alpha2-1)*V2(i)^2/(2*g))
    V_dot(i)=V_dot(i)/0.00378541*60 //Conversion from 'gpm' to 'm3/s'
end
NPSH_required=[3.8 4 4.5 6 7.5]
```

```
                                    //NPSH_required is required NPSH in 'ft'
V_dot_required=[300 400 500 600 700]                              //V_dot_required is required volume flow rate in 'gpm'
//NPSH_required and V_dot_required is obtained from FIGURE 14-15 page number 744.
plot(V_dot',NPSH_at_T1,'r',V_dot_required',NPSH_required','k',V_dot',NPSH_at_T2,'b')
xlabel("V_dot,gpm")
ylabel("NPSH,ft")
title("Net positive suction head as a function of volume flow rate at two temperatures. Cavitation is predicted to occur at flow rates greater than the point where the available and required valves of NPSH intersect")
legend('NPSH availabe at 25  C ','NPSH required','NPSH availabe at 60  C ')

k=length(V_dot_required)                                    //'length' function is used to determine the number of flow rate in 'V_dot_required' matrix
//Following loop is to determine the slope and intercept of the 'V_dot_required' vs 'NPSH_required' line. So that point of intersection is determined by sustitution method.
//Assuming the plot between 'V_dot_required' and 'NPSH_required' to be a straight line is the cause for the variation of the final answer from original answer
for i=1:k
    Modified_V_dot_required(i,1)=V_dot_required(i)
    Modified_V_dot_required(i,2)=1
end
Slope_intercept=Modified_V_dot_required\NPSH_required'
VFRGuess_at_T1=550                                          //
```

```
VFRGuess_at_T1 is the guess volume flow rate to find the volume flow rate at the point of intersection of 'NPSH_required' curve and 'NPSH_at_T1' curve by using fsolve function

VFRGuess_at_T1=VFRGuess_at_T1*0.00378541/60      // Conversion from 'gpm' to 'm3/s'
VFR_at_T1=fsolve(VFRGuess_at_T1,fsolve19)        // Calling of fsolve function to determine the actual VFR_at_T1
VFR_at_T1=VFR_at_T1*60/0.00378541                // Conversion from 'm3/s' to 'gpm'
VFRGuess_at_T2=500                               // VFRGuess_at_T2 is the guess volume flow rate to find the volume flow rate at the point of intersection of 'NPSH_required' curve and 'NPSH_at_T2' curve by using fsolve function
VFRGuess_at_T2=VFRGuess_at_T2*0.00378541/60      // Conversion from 'gpm' to 'm3/s'
VFR_at_T2=fsolve(VFRGuess_at_T2,fsolve20)        // Calling of fsolve function to determine the actual VFR_at_T2
VFR_at_T2=VFR_at_T2*60/0.00378541                // Conversion from 'm3/s' to 'gpm' for display purpose
if VFR_at_T2<VFR_at_T1 then
    condition="decreases"
else
    condition="increases"
end


//Display of result
mprintf('\nAt %d  C , cavitation occurs at flow rates above approximately %d gpm.\nMaxiumum volume flow rate without cavitation %s with temperature.\nFor example at %d  C , flow rate above which cavitation occurs is approximately %d
```

```
gpm.',T1,VFR_at_T1,condition,T2,VFR_at_T2)
//Variation of answer from the actual answer is
   beacuse assuming plot between V_dot_required and
   NPSH_required to be a straight line.
```

---

check Appendix ?? for dependency:

FIGURE_14_27.jpg

Scilab code Exa 14.4 Volume Flow Rate Through a Positive Displacement Pump

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                        //To clear the console
   screen
clear;                      //To clear all the existing
   variables in the memory


//Given data
V_lobe=0.45                 //V_lobe is the volume of
   motor oil in 'cm3'
n_pump=900                  //n_pump is the rotation
   speed of the pump in 'rpm'


//Assumption
n=0.5                       //n is rotations for 180
   rotation. Obtained from FIGURE 14-27 page number
   752.


//Calculation
V_closed=2*V_lobe           //V_closed is total volume
   of oil pumped in 'cm3'
```

```
V_dot=n_pump*V_closed/n //V_dot is the volume flow rate in 'cm3/min'


//Display of result
mprintf('\nThe volume flow rate is %d cm3/min.',V_dot)
```

Scilab code Exa 14.5 Idealized Blower Performance

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                        //To clear the console screen
clear;                      //To clear all the existing variables in the memory


//Given data
n=1750                      //n is the rotational speed of blower in 'rpm'
Alpha1=0                    //Alpha1 is the air inlet angle in ' (degree)'
Alpha2=40                   //Alpha2 is the air exit angle in ' (degree)'
r1=4                        //r1 is the inlet radius in 'cm'
b1=5.2                      //b1 is the inlet blade width in 'cm'
r2=8                        //r2 is the outlet radius in 'cm'
b2=2.3                      //b2 is the outlet blade width in 'cm'
V_dot=0.13                  //V_dot is the volume flow rate in 'm3/s'
```

```
Eta=1                    //Eta is the efficiency


//Unit conversion
r1=r1/100                //Conversion from 'cm' to 'm'
b1=b1/100                //Conversion from 'cm' to 'm'
r2=r2/100                //Conversion from 'cm' to 'm'
b2=b2/100                //Conversion from 'cm' to 'm'
Alpha1=Alpha1*%pi/180    //Conversion from '  (degree)' to 'radian'
Alpha2=Alpha2*%pi/180    //Conversion from '  (degree)' to 'radian'


//Assumption
g=9.81                   //g is the acceleration due to gravity in 'm/s2'
rho_air=1.20             //rho_air is the density of air in 'kg/m3'
rho_water=998            //rho_water is the density of water in 'kg/m3'


//Calculation
V1_n=V_dot/(2*%pi*r1*b1)                //V1_n is normal velocity component at the inlet in 'm/s'
V1_t=V1_n*tan(Alpha1)                   //V1_t is the tangential velocity component at the inlet in 'm/s'
V2_n=V_dot/(2*%pi*r2*b2)                //V2_n is normal velocity component at the outlet in 'm/s'
V2_t=V2_n*tan(Alpha2)                   //V2_t is the tangential velocity component at the outlet in 'm/s'
```

```
Omega=n*(2*%pi)/60                            //Omega is the angular velocity in 'rad/s'
H=Omega*((r2*V2_t)-(r1*V1_t))/g               //H is the net head in 'm of air'
H_water_column=H*rho_air/rho_water            //H_water_column is the net head in 'm of water'
H_water_column=H_water_column*1000            //Conversion from 'm of water' to 'mm of water'
bhp=rho_air*g*V_dot*H                         //bhp is the brake horse power in 'W'


//Display of result
mprintf('\nNet head is %.1f mm of water.\nRequired brake horse power is %.1f W.',H_water_column,bhp)
```

Scilab code Exa 14.6 Preliminary Design of a Centrifugal Pump

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                      //To clear the console screen
clear;                    //To clear all the existing variables in the memory


//Given data
r1=100                    //r1 is the inlet radius in 'mm'
r2=180                    //r2 is the outlet radius in 'mm'
b1=50                     //b1 is the inlet blade width in 'mm'
b2=30                     //b2 is the outlet blade width in 'mm'
```

```
V_dot=0.25                  //V_dot is the volume flow
    rate in 'm3/s'
H=14.5                      //H is the net head in 'm'
n=1720                      //n is the rotational speed
    of impeller in 'rpm'
V1_t=0                      //V1_t is the tangential
    velocity component at the inlet in 'm/s'


//Unit conversion
r1=r1/1000                   //Conversion from 'mm' to '
   m'
b1=b1/1000                   //Conversion from 'mm' to '
   m'
r2=r2/1000                   //Conversion from 'mm' to '
   m'
b2=b2/1000                   //Conversion from 'mm' to '
   m'


//Assumption
g=9.81                      //g is the acceleration due
    to gravity in 'm/s2'
Eta=1                       //Eta is the efficiency
V_f=0.0008157               //V_f is specific volume of
    the refrigerent in 'm3/kg'
rho=1226                    //rho is the density in 'kg/
   m3'


//Calculation
W_water_horsepower=rho*g*V_dot*H          //
    W_water_horsepower is the required water horse
    power in 'W'
bhp=W_water_horsepower/Eta                 //bhp is the
     required brake horse power in 'W'
bhp=bhp/745.7                              //Conversion
     from 'W' to 'hp'
```

```
Omega=n*(2*%pi)/60                       //Omega is
   the angular velocity in 'rad/s'
Beta1=atan(V_dot/(2*%pi*b1*Omega*r1^2)) //Beta1 is
   the balde angle at the inlet in 'radian'
Beta1=Beta1*180/%pi                      //Conversion
    from 'radian' to '  (degree)'
V2_n=V_dot/(2*%pi*r2*b2)                 //V2_n is
   normal velocity component at the outlet in 'm/s'
V2_t=((H*g)+(Omega*r1*V1_t))/(Omega*r2) //V2_t is
   the tangential velocity component at the outlet
   in 'm/s'
Beta2=atan(V2_n/((Omega*r2)-V2_t))      //Beta2 is
   the balde angle at the outlet in 'radian'
Beta2=Beta2*180/%pi                      //Conversion
    from 'radian' to '  (degree)'


//Display of result
mprintf('\nThe brake horsepower required by the pump
    is %d hp.\nBlade angle at the inlet is %.1 f  .\
   nBlade angle at the outlet is %.1 f  .',bhp,Beta1,
   Beta2)
//The answers vary due to round off error
```

Scilab code Exa 14.7 Calculation of Twist in an Airplane Propeller

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                          //To clear the console
   screen
clear;                        //To clear all the
   existing variables in the memory


//Given data
```

```
D_propeller=34                //D_propeller is the propeller diameter in 'cm'
D_hub=5.5                     //D_hub is the hub assembly diameter in 'cm'
n=1700                        //n is the rotational speed of propeller in 'rpm'
alpha=14                      //alpha is the attack angle in ' (degree)'
v=30                          //v is the velocity of the airplane in 'mi/h'


//Unit conversion
D_propeller=D_propeller/100 //Conversion from 'cm' to 'm'
D_hub=D_hub/100               //Conversion from 'cm' to 'm'
v=v*1609.34/3600              //Conversion from 'mi/h' to 'm/s'


//Calculation
Omega=n*(%pi*2)/60                                                     //Omega is the angular velocity in 'rad/s'
r_propeller=D_propeller/2                                              //r_propeller is the radius of the propeller in 'm'
r_hub=D_hub/2                                                          //r_hub is the radius of the hub assembly in 'm'
theta_propeller=alpha+(atan(v/(Omega*r_propeller))*180/%pi) //theta_propeller is the blade pitch angle at the tip in ' (degree)'
theta_hub=alpha+(atan(v/(Omega*r_hub))*180/%pi)             //theta_hub is the balde pitch angle at the root in ' (degree)'
//Multiplication by '180/%pi' on the second term of
```

```
'theta_propeller' and 'theta_hub' equations R.H.S is to convert the second term from 'radian' to '(degree)'


//Display of result
mprintf('\nThe pitch angle at the root is %.1f  .\nThe pitch angle at the tip is %.1f  .',theta_hub,theta_propeller)
```

Scilab code Exa 14.8 Design of a Vane Axial Flow Fan for a Wind Tunnel

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                          //To clear the console screen
clear;                        //To clear all the existing variables in the memory


//Given data
Beta_sl=0                     //Beta_sl is the stator blade leading edge angle in '  (degree)'
Beta_st=60                    //Beta_st is the stator blade trailing edge angle in '  (degree)'
stator=16                     //stator is the number of stator blade
V_in=47.1                     //V_in is the axial flow speed through the blades in 'm/s'
n=1750                        //n is the rotational speed of propeller in 'rpm'
r=0.4                         //r is the radius of the fan in 'm'


```

```
//Unit conversion
Beta_sl=Beta_sl*%pi/180      //Conversion from ' (degree)' to 'radian'
Beta_st=Beta_st*%pi/180      //Conversion from ' (degree)' to 'radian'


//Calculation
V_st=V_in/cos(Beta_st)       //V_st is the velocity leaving the trailing edge of the stator blade in 'm/s'
Omega=n*(2*%pi)/60           //Omega is the angular velocity in 'rad/s'
u_theta=Omega*r              //u_theta is the tangential velocity of the rotor blade in 'm/s'
Beta_rl=atan((u_theta+(V_in*tan(Beta_st)))/V_in) //Beta_rl is the rotor blade leading edge angle in 'radian'
Beta_rl=Beta_rl*180/%pi      //Conversion from ' radian' to ' (degree)'
Beta_rt=atan(u_theta/V_in)   //Beta_rt is the rotor blade trailing edge angle in 'radian'
Beta_rt=Beta_rt*180/%pi      //Conversion from ' radian' to ' (degree)'
//Following code is to determine the number of rotor blades by finding the numbers which don't have a common denominator with the number of stator blade
i=stator
k=1
for j=i-4:i+4
    if gcd([i j])==1 then   //if gcd([i j]) returns 1, it means there is no common divisor for i and j other than '1'.
        Rotar_blade(k)=j
        k=k+1
    end
end
```

```
//Display of result
mprintf('\nThe leading edge angle of rotor blade is
    %.2f  .\nThe trailing edge angle of the rotor
    blade is %.2f  .',Beta_rl,Beta_rt)
mprintf('\nNumber of rotor blades can be a number
    like')
disp(Rotar_blade')
```

check Appendix ?? for dependency:

FIGURE_14_72.png

check Appendix ?? for dependency:

FIGURE_14_73.png

Scilab code Exa 14.9 Using Pump Specific Speed for Preliminary Pump Design

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                          //To clear the console
    screen
clear;                        //To clear all the
    existing variables in the memory


//Given data
V_dot=320                     //V_dot is the volume
    flow rate in 'gpm'
n=1170                        //n is the rotational
    speed of the pump shaft in 'rpm'
H=23.5                        //H is the required net
    head in 'ft of gasoline'


```

```
//Calculation
N_sp_US=n*V_dot^0.5/H^(3/4) //N_sp_US is the pump specific speed in customary U.S. units
N_sp=3.658E-4*N_sp_US        //N_sp is the normalized pump specific speed    (Formula is obtained from FIGURE 14-72 page number 776)
//Condition for N_sp is obtained from FIGURE 14-73 page number 777.
if N_sp>0 & N_sp<=1.8 then
    pump="Centrifugal"
else
    if N_sp>1.8 & N_sp<=3.5
        pump="Mixed"
    else
        pump="Axial"
    end
end


//Display of result
mprintf('\nPump specific speed in customary U.S. units is %d.\nNormalized pump specific speed is %.3f\n%s pump is the suitable choice.',N_sp_US,N_sp,pump)
```

check Appendix ?? for dependency:

TABLE_14_2.jpg

Scilab code Exa 14.11 Design of a New Geometrically Similar Pump

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                                            //To clear the console screen
```

```
clear;                                          //To clear all the existing variables in the memory
clf(0)                                          //Clear or reset or reset a figure 0
clf(1)                                          //Clear or reset or reset a figure 1


//Given data
D_A=6                                           //D_A is impeller diameter of pump A in 'cm'
n_A=1725                                        //n_A is the rotational speed of pump A in 'rpm'
Omega_A=180.6                                   //Omega_A is angular velocity of pump A in 'rad/s'
V_dot_A=[100 200 300 400 500 600 700]    //V_dot_A is the volume flow rate obtained from TABLE 14-2 in 'cm3/s'
H_A=[180 185 175 170 150 95 54]          //H_A is the head obtained from TABLE 14-2 in 'cm'
Eta_pump_A=[32 54 70 79 81 66 38]        //Eta_pump_A is the pump efficiency obtained from TABLE 12-2 in '%'
V_dot_B=2400                                    //V_dot_B is the volume flow rate in pump B in 'cm3/s'
H_B=450                                         //H_B is the net head of pump B in 'cm'


//Unit conversion
V_dot_B=V_dot_B*10^-6                           //Conversion from 'cm3/s' to 'm3/s'
H_B=H_B/100                                     //Conversion from 'cm' to 'm'


//Assumption
g=9.81                    //g is acceleration due to
```

```
gravity in 'm/s2'
rho_water=998        //rho_water is the water density in 'kg/m3'
rho_R134a=1226       //rho_R134a is the density of the refrigerant in 'kg/m3'


//Calculation
m=length(V_dot_A)    //m is number of data in 'V_dot_A' matrix found using 'length' function
for i=1:m
    bhp_A(i)=rho_water*g*V_dot_A(i)*H_A(i)/((Eta_pump_A(i)/100)*100^4)//W
    //Division by '10^4' on the R.H.S of the above equation is convert the 'V_dot_A' and 'H_A' into 'meter3/s' and 'meter' respectively.
end
scf(0)
plot(V_dot_A,H_A,'r',V_dot_A,Eta_pump_A,'k.-',V_dot_A',bhp_A.*20','-.r')
xlabel("V_dot,cm3/s")
legend(['H,cm';'Eta,%';'bhp*20,W'],-1)
title("FIGURE 14-76: Dimensional upmp performance curves for the water pump of Example 14-11")
for i=1:m
    C_H_A(i)=g*(H_A(i)/100)/(Omega_A^2*(D_A/100)^2)
    C_Q_A(i)=V_dot_A(i)/(Omega_A*D_A^3)
    C_P_A(i)=bhp_A(i)/(rho_water*Omega_A^3*(D_A/100)^5)
end
scf(1)
//Following 'for' loop is to find the maximum value present in the 'Eta_pump_A' matrix to determine the 'BEP'
Eta_max=Eta_pump_A(1)
m=length(Eta_pump_A)
for i=2:m
    if Eta_pump_A(i)>Eta_max then
```

```
        Eta_max=Eta_pump_A(i)//Eta_max is maximum value in the Eta_pump_A matrix
    end
end
Eta_max=Eta_max/100      //Conversion from '%(percentage)' to 'fraction'
plot(C_Q_A'.*100,Eta_pump_A./100,'k',C_Q_A.*100,C_P_A.*100,'-.r',C_Q_A.*100,C_H_A.*10,'r')
xlabel("C_Q_A*100")
legend(['Eta_pump_A*100';'C_P_A*100';'C_H_A*10'],1)
title("FIGURE 14-77: Non dimensional pump performance curves for the pumps of Example 14-11; BEP is estimated as the operating point where Eta_pump_A is a maximum")
Eta_pump_star=Eta_max    //Maxima of the efficiency curve
//Following dimensionless pump parameters are obtained from FIGURE 14_77 corresponding to the maximum efficiency.
C_Q_star=0.0112          //point of intersection of the vertical line from the maxima of efficiency curve and C_V_dot curve
C_H_star=0.133           //point of intersection of the vertical line from the maxima of efficiency curve and C_H curve
C_P_star=0.00184         //point of intersection of the vertical line from the maxima of efficiency curve and C_P curve



//Display of result
mprintf('\n(a) Dimension less pump parameters are as follows:\n     C_Q_star=%.4f\n     C_H_star=%.3f\n     C_P_star=%.5f\n     Eta_pump_star=%.3f',C_Q_star,C_H_star,C_P_star,Eta_pump_star)
//The answers vary due to round off error

```

```
//Part (b)
//Calculation
D_B=((V_dot_B^2*C_H_star)/(C_Q_star^2*g*H_B))^(1/4)
    //D_B is the diameter of the pump B in 'm'
Omega_B=V_dot_B/(C_Q_star*D_B^3)
    //Omega_B is the angular velocity in pump B in '
    rad/s'
n_B=Omega_B*60/(2*%pi)
    //n_B is the rotational speed in pump B in 'rpm'
bhp_B=C_P_star*rho_R134a*Omega_B^3*D_B^5
    //bhp_B is the required brake horse power of pump
     B in 'W'


//Display of result
mprintf('\n\n(b) Diameter of the pump B is %.3f m\n
       Rotational speed in pump B is %d rpm\n
    Required brake horse power of pump B is %d W',D_B
    ,n_B,bhp_B)
//The answers vary due to round off error
```

Scilab code Exa 14.12 Hydroturbine Design

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                        //To clear the console
    screen
clear;                      //To clear all the
    existing variables in the memory
clf(0)                      //Clear or reset or
```

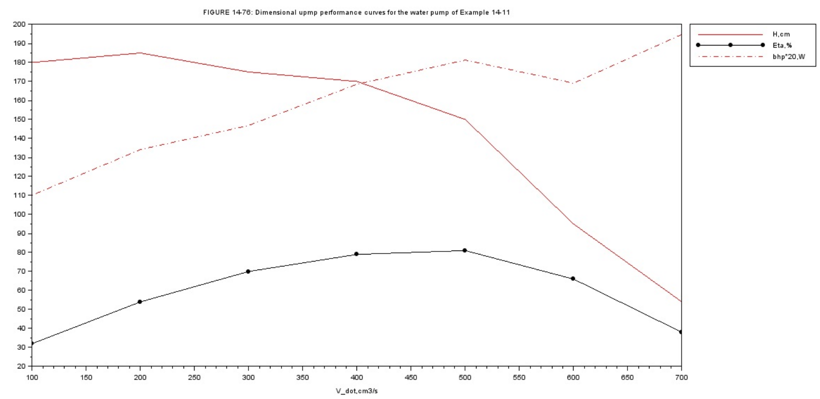


Figure 14.2: Design of a New Geometrically Similar Pump

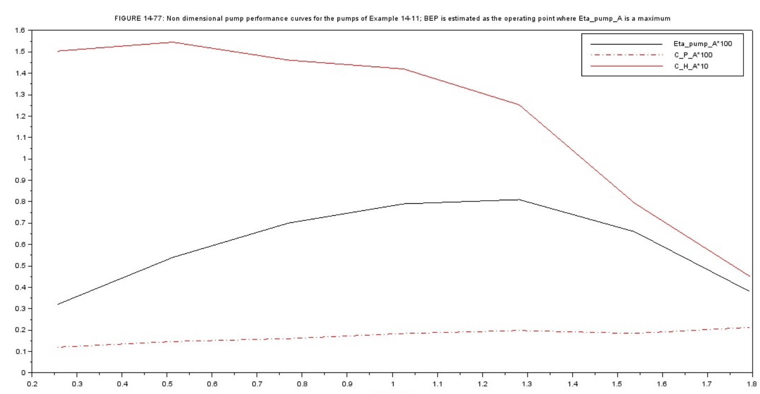


Figure 14.3: Design of a New Geometrically Similar Pump

```
reset a figure 0
clf(1)                          //Clear or reset or reset a figure 1


//Given data
r2=8.20                         //r2 is the runner inlet radius in 'ft'
r1=5.80                         //r1 is the runner outlet radius in 'ft'
b1=8.60                         //b1 is the runner blade outlet width in 'ft'
b2=3.00                         //b2 is the runner blade inlet width in 'ft'
n=120                           //n is the rotational speed of the runner in 'rpm'
f=60                            //f is the electric generator frequency in 'Hz'
Alpha2=33                       //Alpha2 is the flow angle at runner inlet in ' (degree)'
V_dot=9.50E6                    //V_dot is the volume flow rate in 'gpm'
H_gross=303                     //H_gross is the gross head provided by the dam in 'ft'
Alpha1=[10 0 -10]               //Alpha1 is the flow angle at runner outlet in ' (degree)'


//Assumption
rho=998.0                       //rho is the water density in 'kg/m3'
Eta=1                           //Eta is efficiency of the turbine
g=9.81                          //g is the acceleration due to gravity in 'm/s2'


//Unit conversion
```

```
Alpha2=Alpha2*%pi/180          //Conversion from ' (degree)' to 'radian'
Alpha1=Alpha1.*%pi/180         //Conversion from ' (degree)' to 'radian'
r1=r1*12*0.0254                //Conversion from 'ft' to 'm'
r2=r2*12*0.0254                //Conversion from 'ft' to 'm'
b1=b1*12*0.0254                //Conversion from 'ft' to 'm'
b2=b2*12*0.0254                //Conversion from 'ft' to 'm'
H_gross=H_gross*12*0.0254      //Conversion from 'ft' to 'm'
V_dot=V_dot*0.00378541/60      //Conversion from 'gpm' to 'm3/s'


//Calculation
Omega=n*(2*%pi)/60             //Omega is the angular velocity in 'rad/s'
V_2_n=V_dot/(2*%pi*r2*b2)      //V_2_n is the inlet normal velocity component in 'm/s'
V_2_t=V_2_n*tan(Alpha2)        //V_2_t is the inlet tangential velocity component in 'm/s'
Beta2=180/%pi*atan(V_2_n/((Omega*r2)-(V_2_t))) //Beta2 is the runner leading edge angle in ' (degree)'
V_1_n=V_dot/(2*%pi*r1*b1)      //V_1_n is the outlet normal velocity component in 'm/s'
m=length(Alpha1)               //m is number of elements in Alpha1 matrix
for i=1:m
    V_1_t(i)=V_1_n*tan(Alpha1(i))                              //V_1_t is the outlet tangential velocity component in 'm/s'
    Beta1(i)=180/%pi*atan(V_1_n/((Omega*r1)-(V_1_t(i))))     //Beta1 is the runner trailing edge
```

```
angle in ' (degree)'
    W_shaft(i)=rho*Omega*V_dot*((r2*V_2_t)-(r1*V_1_t(i)))    //W_shaft is the shaft output power in 'W'
    bhp(i)=W_shaft(i)/Eta                                                          //bhp is the brake horse power in 'W'
    W_shaft(i)=W_shaft(i)/745.7                                                    //Conversion from 'W' to 'hp'
    H(i)=bhp(i)/(rho*g*V_dot)                                                      //H is the required net head in 'm'
end
Alpha1=Alpha1./%pi*180           //Conversion from 'radian' to ' (degree)'
bhp=bhp./10^6                    //Conversion from 'W' to 'MW'


//Display of result
//Part (a)
mprintf('\n(a) When runner outlet angle is %d ,\n    Runner leading edge angle is %.1f  .\n    Runner trailing edge angle is %.1f  .\n    The shaft output power is %d hp.\n    Required net head is %.1f m.',Alpha1(1),Beta2,Beta1(1),W_shaft(1),H(1))
//Part (b)
mprintf('\n\n(b) When runner outlet angle is %d ,\n    Runner leading edge angle is %.1f  .\n    Runner trailing edge angle is %.1f  .\n    The shaft output power is %d hp.\n    Required net head is %.1f m.',Alpha1(2),Beta2,Beta1(2),W_shaft(2),H(2))
//Part (c)
mprintf('\n\n(c) When runner outlet angle is %d ,\n    Runner leading edge angle is %.1f  .\n
```

```
    Runner trailing edge angle is %.1f .\n    The
    shaft output power is %d hp.\n    Required net
    head is %.1f m.',Alpha1(3),Beta2,Beta1(3),W_shaft
    (3),H(3))


//Graph plotting
scf(0)
plot(Alpha1',bhp,'k')
xlabel('Alpha1,degree')
ylabel('bhp,MW')
title('FIGURE 14-101: Brake horse power output as a
    function of runner outlet flow angle for the
    turbine of Example 14-12')
scf(1)
slope_H_gross=0                                    //
    Slope of the H_gross line is zero
Intercept_H_gross=H_gross                          //
    Intercept is H_gross in 'm'
H1=slope_H_gross.*Alpha1.^0+Intercept_H_gross     //
    Defining the equation of the H_gross line in y=mx
    +c form
plot(Alpha1',H,'r',Alpha1',H1','-.k')
xlabel('Alpha1,degree')
ylabel('H,m')
legend(['H';'H_gross'])
title('FIGURE 14-101: Ideal required net head as a
    function of runner outlet flow angle for the
    turbine of Example 14-12')
```

Scilab code Exa 14.13 Application of Turbine Affinity Laws

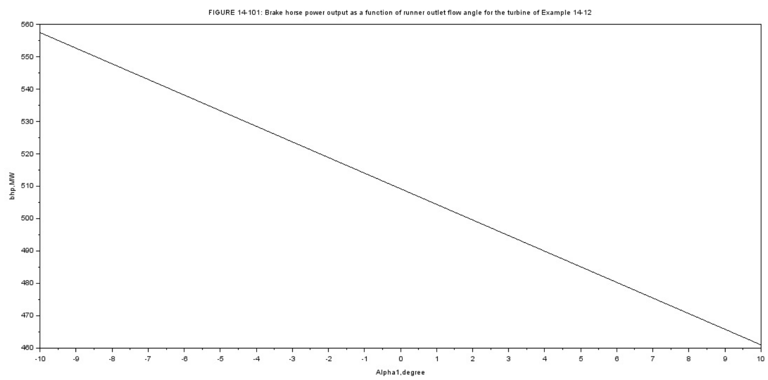


Figure 14.4: Hydroturbine Design

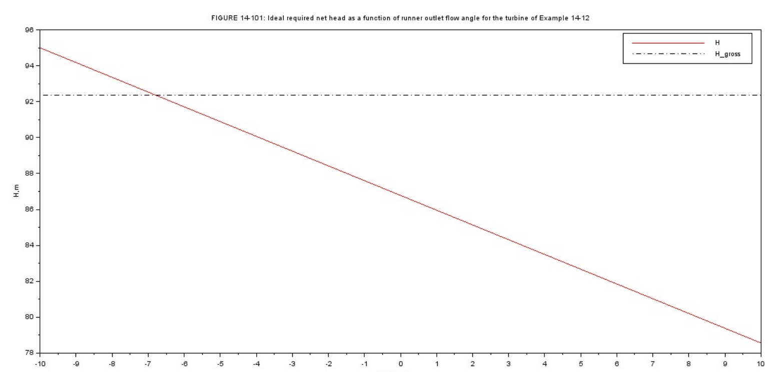


Figure 14.5: Hydroturbine Design

```
//SCILAB version: 5.5.2
//Operating system: Windows 7 Ultimate
clc;                            //To clear the console screen
clear;                          //To clear all the existing variables in the memory


//Given data
D_A=2.05                        //D_A is the diameter of the turbine A in 'm'
n_A=10                          //n_A is the rotational speed of turbine A in 'rpm'
Omega_A=12.57                   //Omega_A is the angular velocity of turbine A in 'rad/s'
V_dot_A=350                     //V_dot_A is the volume flow rate in the tubine A in 'm3/s'
H_A=75                          //H_A is the net head in the turbine A 'm of water'
bhp_A=242                       //bhp_A is the brake horse power of turbine A in 'MW'
n_B=n_A                         //n_B is the rotational speed of turbine B in 'rpm'
H_B=104                         //H_B is the net head in the turbine B 'm of water'


//Assumption
rho_A=998                       //rho_A is the water density in turbine A in 'kg/m3'
rho_B=998                       //rho_B is the water density in turbine B in 'kg/m3'
g=9.81                          //g is the acceleration due to gravity in 'm/s2'


//Unit conversion
bhp_A=bhp_A*10^6                //Conversion from 'MW'
```

```
    to 'W'


//Calculation
Omega_B=Omega_A    //Omega_B is the angular velocity of the turbine B in 'rad/s'
D_B=D_A*(n_A/n_B)*sqrt(H_B/H_A)    //D_B is the diameter of turbine B in 'm'
V_dot_B=V_dot_A*(n_B/n_A)*(D_B/D_A)^3    //V_dot_B is the volume flow rate in turbine B in 'm3/s'
bhp_B=bhp_A*(rho_B/rho_A)*(n_B/n_A)^3*(D_B/D_A)^5    //bhp_B is the brake horse power of turbine B in 'W'
Eta_turbine_A=bhp_A/(rho_A*g*H_A*V_dot_A)    //Eta_turbine_A is the efficiency of the turbine A
Eta_turbine_B=1-((1-Eta_turbine_A)*(D_A/D_B)^(1/5))    //Eta_turbine_B is the efficiency of the turbine B
C_H_A=g*H_A/(Omega_A^2*D_A^2)    //C_H_A is the heat coefficient of turbine A
C_H_B=g*H_B/(Omega_B^2*D_B^2)    //C_H_B is the heat coefficient of turbine B
C_Q_A=V_dot_A/(Omega_A*D_A^3)    //C_Q_A is the capacity coefficient of turbine A
C_Q_B=V_dot_B/(Omega_B*D_B^3)    //C_Q_B is the capacity coefficient of turbine B
C_P_A=bhp_A/(rho_A*Omega_A^3*D_A^5)    //C_P_A is the power coefficient of turbine A
C_P_B=bhp_B/(rho_B*Omega_B^3*D_B^5)    //C_P_B is the power coefficient of turbine B
bhp_B=bhp_B/10^6    //Conversion from 'W' to 'MW'


//Display of result
```

```
mprintf('\nDiameter of turbine B is %.2f m.\nVolume flow rate in turbine B is %d m3/s.\nBrake horse power of turbine B is %d MW.\nEfficiency of turbine B is %.3f.',D_B,V_dot_B,bhp_B,Eta_turbine_B)
//The answers vary due to round off error
mprintf('\n\nDimension less turbine parameters for both turbines are as follows:')
mprintf('\n\n\tTurbine A\tTrbine B\nC_H\t%f\t%f\nC_Q\t%f\t%f\nC_P\t%f\t%f',C_H_A,C_H_B,C_Q_A,C_Q_B,C_P_A,C_P_B)
```

check Appendix ?? for dependency:

`FIGURE_14_107.jpg`

Scilab code Exa 14.14 `Turbine Specific Speed`

```
clc                            //To clear the console screen
clear                          //To clear all the existing variables in the memory


//Given data
D_A=2.05                       //D_A is the diameter of the turbine A in 'm'
n_A=10                         //n_A is the rotational speed of turbine A in 'rpm'
Omega_A=12.57                  //Omega_A is the angular velocity of turbine A in 'rad/s'
V_dot_A=350                    //V_dot_A is the volume flow rate in the tubine A in 'm3/s'
H_A=75                         //H_A is the net head in the turbine A 'm of water'
```

```
bhp_A=242                          //bhp_A is the brake horse power of turbine A in 'MW'
n_B=n_A                            //n_B is the rotational speed of turbine B in 'rpm'
H_B=104                            //H_B is the net head in the turbine B 'm of water'


//Assumption
rho_A=998                          //rho_A is the water density in turbine A in 'kg/m3'
rho_B=998                          //rho_B is the water density in turbine B in 'kg/m3'
g=9.81                             //g is the acceleration due to gravity in 'm/s2'


//Unit conversion
bhp_A=bhp_A*10^6                   //Conversion from 'MW' to 'W'


//Calculation
Omega_B=Omega_A
    //Omega_B is the angular velocity of the turbine B in 'rad/s'
D_B=D_A*(n_A/n_B)*sqrt(H_B/H_A)
    //D_B is the diameter of turbine B in 'm'
bhp_B=bhp_A*(rho_B/rho_A)*(n_B/n_A)^3*(D_B/D_A)^5
    //bhp_B is the brake horse power in 'W'
N_st_A=Omega_A*(bhp_A)^0.5/(rho_A^0.5*(g*H_A)^(5/4))
    //N_st_A is the dimension less turbine specific speed for turbine A
N_st_B=Omega_B*(bhp_B)^0.5/(rho_B^0.5*(g*H_B)^(5/4))
    //N_st_B is the dimension less turbine specific speed for turbine B
//Formula for N_st_US is obtained from FIGURE 14-107 page number 799.
```

```
N_st_US_A=43.46*N_st_A //N_st_US_A is the turbine specific speed in customary U.S. units of turbine A
N_st_US_B=43.46*N_st_B //N_st_US_B is the turbine specific speed in customary U.S. units of turbine B


//Display of result
if (N_st_A-N_st_B)<0.001 then
    mprintf('\nTurbine specific speed of the two turbines are same..!')
else
    mprintf('\nTurbine specific speed of the two turbines are not same..!')
end
mprintf('\nDimensionless turbine specific speed for turbine A and B are %.2f and %.2f respectively.\nTurbine specific speed in customary U.S. units of turbine A and B are %.2f and %.2f respectively.',N_st_A,N_st_B,N_st_US_A,N_st_US_B)
```

Appendix

Scilab code AP 5 `fsolve19.sci`

```
//Chapter 14 example 14-3
//Following fsolve19 function is to find the point
    of intersection of 'V_dot' vs 'NPSH_at_T1' curve
    and 'V_dot' vs 'NPSH_required' curve using fsolve
     function
function f1_at_T1=fsolve19(V_dot_op)
    Velocity2=V_dot_op/Area
    Re=rho_at_T1*Velocity2*D/Mu_at_T1
    fd=Colebrook(Epsilon_by_D,Re)
    f1_at_T1=(Slope_intercept(2)+(V_dot_op*
       Slope_intercept(1)))-(((P_atm-P_v_at_T1)/(
       rho_at_T1*g))+Z1-Z2-(((fd*L/D)+(Sigma_K_L))*(
       V_dot_op/Area)^2/2/g)-((Alpha2-1)*(V_dot_op/
       Area)^2/(2*g)))
endfunction
```

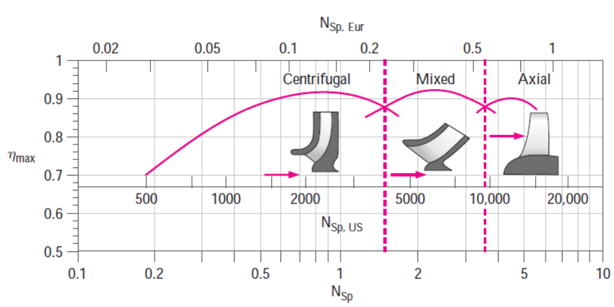


FIGURE 14–73
Maximum efficiency as a function of pump specific speed for the three main types of dynamic pump. The horizontal scales show nondimensional pump specific speed (N_{Sp}), pump specific speed in customary U.S. units ($N_{Sp,\,US}$), and pump specific speed in customary European units ($N_{Sp,\,Eur}$).

FIGURE$_1$4$_7$3

Conversion ratios

$$\frac{N_{Sp}}{N_{Sp,\,US}} = 3.568 \times 10^{-4}$$

$$\frac{N_{Sp}}{N_{Sp,\,Eur}} = 2\pi$$

$$\frac{N_{Sp,\,Eur}}{N_{Sp,\,US}} = 5.822 \times 10^{-5}$$

$$\frac{N_{Sp,\,US}}{N_{Sp}} = 2734$$

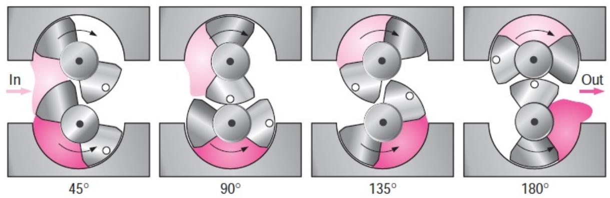
$$\frac{N_{Sp,\,Eur}}{N_{Sp}} = \frac{1}{2\pi}$$

$$\frac{N_{Sp,\,US}}{N_{Sp,\,Eur}} = 17{,}180$$

FIGURE 14–72

Conversions between the dimensionless, conventional U.S., and conventional European definitions of pump specific speed. Numerical

FIGURE 14–27
Four phases (one-eighth of a turn apart) in the operation of a two-lobe rotary pump, a type of positive-displacement pump. The light blue region represents a chunk of fluid pushed through the top rotor, while the dark blue region represents a chunk of fluid pushed through the bottom rotor, which rotates in the opposite direction. Flow is from left to right.



FIGURE$_1$4$_2$7

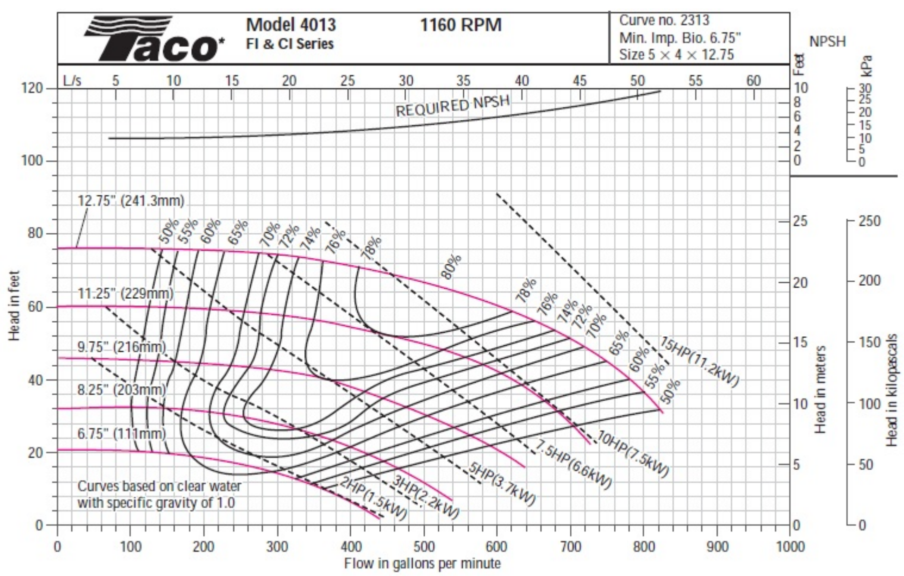


FIGURE 14–15
Example of a manufacturer's performance plot for a family of centrifugal pumps. Each pump has the same casing, but a different impeller diameter.
Courtesy of Taco, Inc., Cranston, RI. Used by permission.

FIGURE$_1$4$_1$5

Scilab code AP 6 fsolve20.sci

```
//Chapter 14 example 14−3
//Following fsolve20 function is to find the point
   of intersection of 'V_dot' vs 'NPSH_at_T2' curve
   and 'V_dot' vs 'NPSH_required' curve using fsolve
    function
function f1_at_T2=fsolve20(V_dot_op)
    Velocity2=V_dot_op/Area
    Re=rho_at_T2*Velocity2*D/Mu_at_T2
    fd=Colebrook(Epsilon_by_D,Re)
    f1_at_T2=(Slope_intercept(2)+(V_dot_op*
       Slope_intercept(1)))-(((P_atm-P_v_at_T2)/(
       rho_at_T2*g))+Z1-Z2-(((fd*L/D)+(Sigma_K_L))*(
       V_dot_op/Area)^2/2/g)-((Alpha2-1)*(V_dot_op/
       Area)^2/(2*g)))
endfunction
```

Scilab code AP 7 Colebrook.sci

```
//Chapter 14 example 14−3
//Following is the function to determine the
   friction factor using colebrook equation.
//Due to the implicit nature of the colebrook
   equation, it is solved using Newton Raphson
   Method.
function f=Colebrook(Epsilon_by_D,Re)
    A=Epsilon_by_D/3.7
    B=2.51/Re
    f_guess=64/Re
    Function=1/(sqrt(f_guess))+(2*log10(A+(B/sqrt(
       f_guess))))
    FunctionDerivative=(-0.5*f_guess^-(3/2))+((-1*B)
       /((A*f_guess^(3/2))+(B*f_guess)))
    f_new=f_guess-(Function/FunctionDerivative)
    while (abs(f_new-f_guess)>1e-10)
        f_guess=f_new
```

```
        Function=1/(sqrt(f_guess))+(2*log10(A+(B/
          sqrt(f_guess))))
        FunctionDerivative=((-1*f_guess^-1.5)/(2))
          +((-1*B)/((A*f_guess^1.5)+(B*f_guess)))
        f_new=f_guess-(Function/FunctionDerivative)
    end
    f=f_new;
endfunction
```

Scilab code AP 11 `fsolve18.sci`

```
//Chapter 14 example 14-1
//Following is the function to determine the
   friction factor using colebrook equation
function f=fsolve18(f0)
    f=(1/f0)-(4*(log10((epsilon_by_D/3.7)+(2.51/(Rey
       *sqrt(f0)))))^2)
endfunction
```

Scilab code AP 13 `ManningEquation2.sci`

```
//Chapter 13 example 13-6
//Following is the function to determine the volume
   flow rate 'V_dot' using Manning equation.
function V_dot=ManningEquation2()
    V_dot=a*A_c*R_h^(2/3)*S0^(1/2)/n//V_dot is
       volume flow rate of water in 'm3/s'
endfunction
```

Scilab code AP 16 `fsolve16.sci`

```
//Chapter 13 example 13-3
//Following is the function to determine the height
   1 'y1' for 'Bottom_Drop_1' using Manning equation
function f1=fsolve16(y)
```

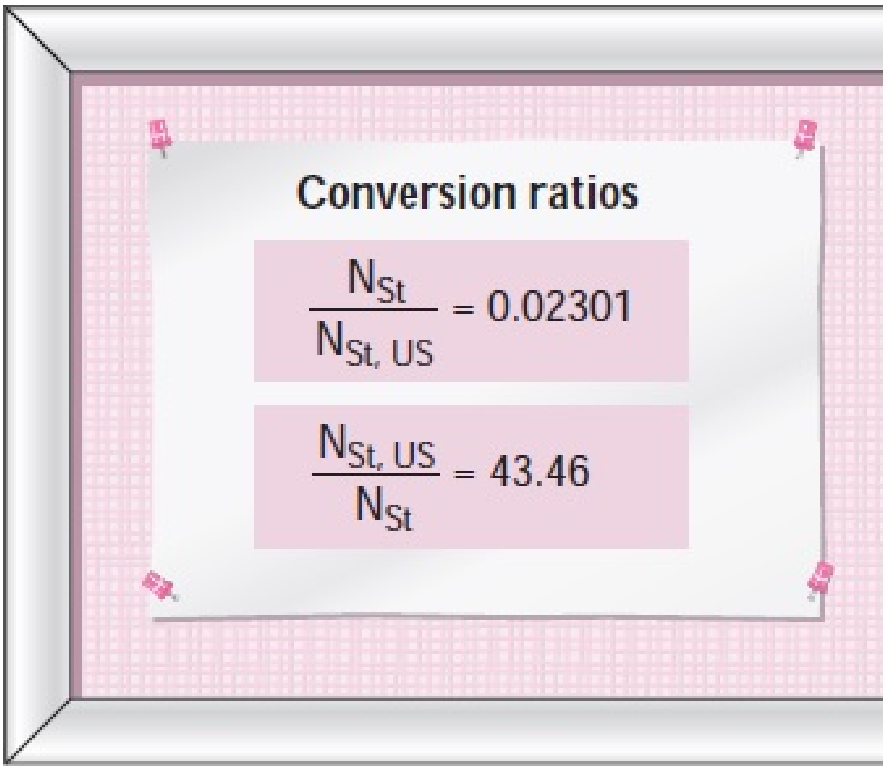


FIGURE 14–107
Conversions between the dimensionless and the conventional U.S. definitions of turbine specific speed. Numerical values are given to four significant digits. The conversions assume earth gravity and water as the working fluid.

FIGURE$_1$4$_1$07

TABLE 14–2

Manufacturer's performance data for a water pump operating at 1725 rpm and room temperature (Example 14–11)*

$\dot{V}$, cm^3/s	H, cm	η_{pump}, %
100	180	32
200	185	54
300	175	70
400	170	79
500	150	81
600	95	66
700	54	38

* Net head is in centimeters of water.

TABLE_14_2

TABLE 14–1

Manufacturer's performance data for the fan of Example 14–1*

$\dot{V}$, cfm	$(\delta P)_{fan}$, inches H_2O
0	0.90
250	0.95
500	0.90
750	0.75
1000	0.40
1200	0.0

* Note that the pressure rise data are listed as inches of *water,* even though *air* is the fluid. This is common practice in the ventilation industry.

$TABLE_14_1$

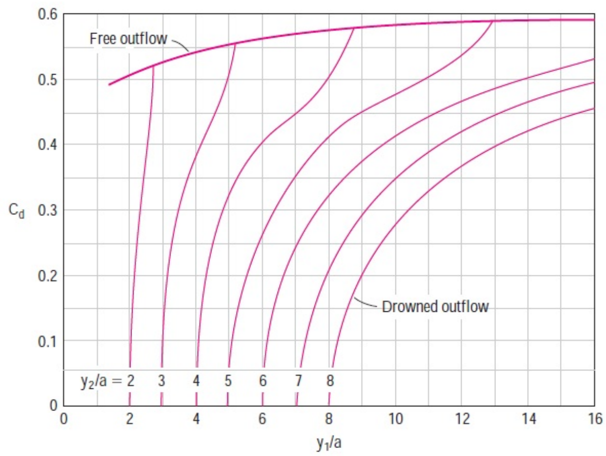


FIGURE 13–38
Discharge coefficients for drowned and free discharge from underflow gates.

From Henderson, Open Channel Flow, 1st Edition, © 1966. Reprinted by permission of Pearson Education, Inc., Upper Saddle River, NJ.

FIGURE$_{1}$3$_{3}$8

```
    f1=(V_dot*n)-(a*b*y*((b*y)/((b)+(2*y)))^(2/3)*(
        S0_1)^(1/2))
endfunction
```

Scilab code AP 17 `fsolve17.sci`

```
//Chapter 13 example 13−3
//Following is the function to determine the 'y2'
    for 'Bottom_Drop_2' using Manning equation
function f2=fsolve17(y)
    f2=(V_dot*n)-(a*b*y*((b*y)/((b)+(2*y)))^(2/3)*(
        S0_2)^(1/2))
endfunction
```

Scilab code AP 18 `ManningEquation1.sci`

```
//Chapter 13 example 13−2
//Following is the function to calculate Volume flow
    rate using Manning equation
function V_dot=ManningEquation1(S0)
    V_dot=a*A_c*R_h^(2/3)*S0^(1/2)/n//V_dot is
        volume flow rate through the channel in 'm3/s
        '
```

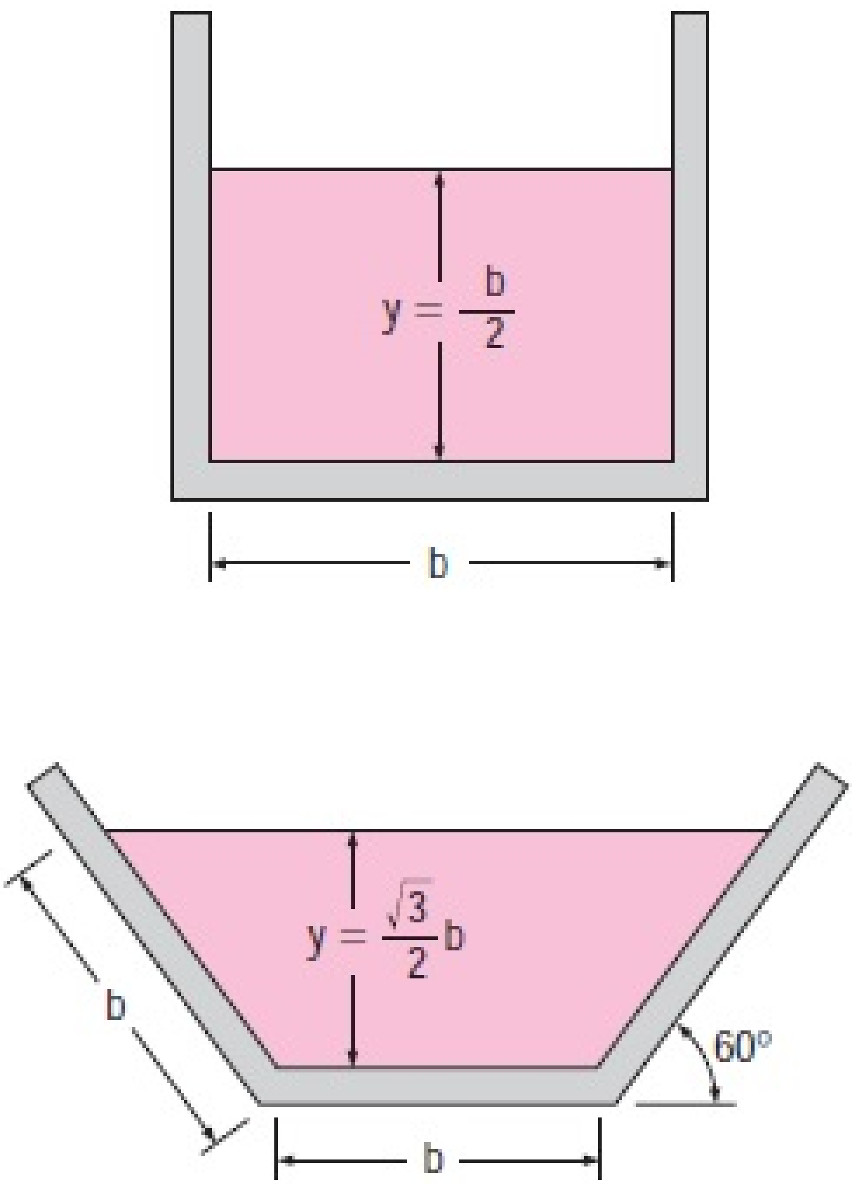


FIGURE 13–26
Schematic for Example 13–5.

FIGURE$_1$3$_2$6

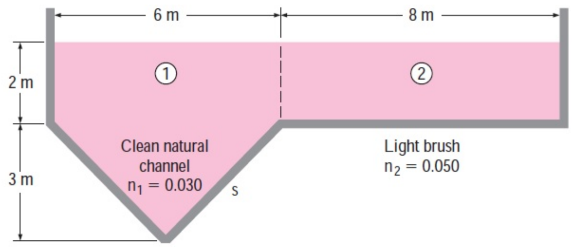


FIGURE 13–20
Schematic for Example 13–4.

FIGURE$_1$3$_2$0

```
endfunction
```

Scilab code AP 19 `fsolve15.sci`

```
//Chapter 13 example 13-1
//Following is the function to determine the
   alternate depth by using fsolve function
function f=fsolve15(y_2)
    f=(E_s2*y_2^2)-(y_2^3+(V_dot^2/(2*g*b^2)))
endfunction
```

Scilab code AP 22 `fsolve9`

```
//Chapter 12 example 12-6
//Following is the function to determine Mach number
    'Ma2' using the formula for 'A_2_ratio' given in
    page number 899 APPENDIX 1.
function f=fsolve9(Ma2)
    f=(A_2_ratio*Ma2)-((2/(k+1))*(1+(0.5*Ma2^2*(k-1)
       )))^((0.5*(k+1))/(k-1))
endfunction
```

Scilab code AP 24 `fsolve14.sci`

```
//Chapter 12 example 12-17
//Following is the function to 'Ma_2' from '
   fl_star_by_D_out' by using the formula present in
    page number 902 APPENDIX 1.
```

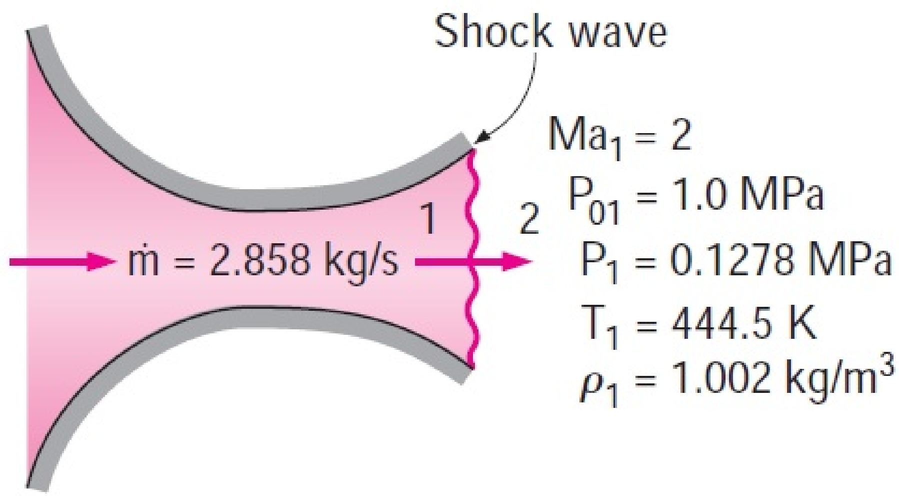


FIGURE 12–35
Schematic for Example 12–9.

FIGURE$_1 2_3 5$

$$Ma^* = Ma\sqrt{\frac{k+1}{2+(k-1)Ma^2}}$$

$$\frac{A}{A^*} = \frac{1}{Ma}\left[\left(\frac{2}{k+1}\right)\left(1+\frac{k-1}{2}Ma^2\right)\right]^{0.5(k+1)/(k-1)}$$

$$\frac{P}{P_0} = \left(1+\frac{k-1}{2}Ma^2\right)^{-k/(k-1)}$$

$$\frac{\rho}{\rho_0} = \left(1+\frac{k-1}{2}Ma^2\right)^{-1/(k-1)}$$

$$\frac{T}{T_0} = \left(1+\frac{k-1}{2}Ma^2\right)^{-1}$$

TABLE$_{A1}$3

```
function f=fsolve14(Ma_2)
   f=(fl_star_by_D_out*Ma_2^2)-(((1-Ma_2^2)/k)+((((k
      +1)*Ma_2^2)/(2*k))*log(((k+1)*Ma_2^2)/(2+((k
      -1)*Ma_2^2)))))
endfunction
```

Scilab code AP 25 `fsolve13`

```
//Chapter 12 example 12-16
//Following is the function to determine the
   friction factor using colebrook equation.
function f=fsolve13(f0)
    f=(1/f0)-(4*(log10((epsilon_by_D/3.7)+(2.51/(
       Re_1*sqrt(f0)))))^2)
endfunction
```

Scilab code AP 27 `fsolve12`

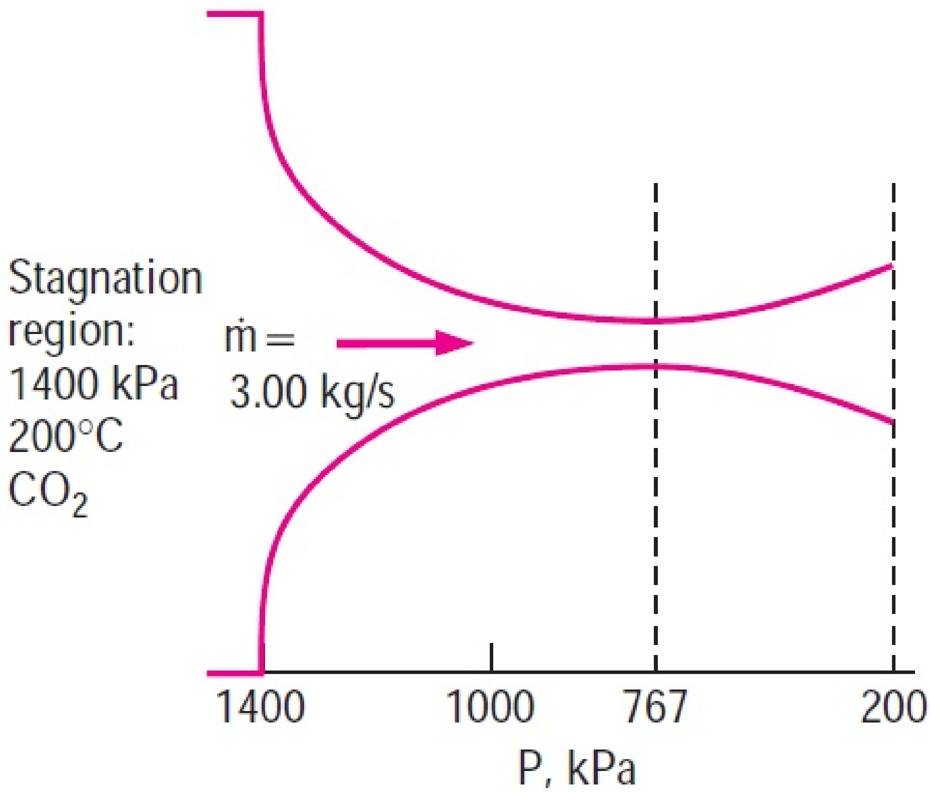


FIGURE 12–12
Schematic for Example 12–3.

FIGURE$_1$2$_1$2

$$\frac{T_0}{T_0^*} = \frac{(k+1)Ma^2[2+(k-1)Ma^2]}{(1+kMa^2)^2}$$

$$\frac{P_0}{P_0^*} = \frac{k+1}{1+kMa^2}\left(\frac{2+(k-1)Ma^2}{k+1}\right)^{k/(k-1)}$$

$$\frac{T}{T^*} = \left(\frac{Ma(1+k)}{1+kMa^2}\right)^2$$

$$\frac{P}{P^*} = \frac{1+k}{1+kMa^2}$$

$$\frac{V}{V^*} = \frac{\rho^*}{\rho} = \frac{(1+k)Ma^2}{1+kMa^2}$$

$TABLE_{A1}5$

```
//Chapter 12 example 12-15
//Following is the function to determine 'Ma_2' using the exit stag-temperature ratio by using the formula present in APPENDIX 1 page number 901.
function f=fsolve12(Ma_2)
    f=(T_02/T_0_star*(1+(k*Ma_2^2))^2)-((k+1)*(2+((k-1)*Ma_2^2))*Ma_2^2)
endfunction
```

Scilab code AP 28 `fsolve11.sci`

```
//Chapter 12 example 12-12
//Following is the function to calculate the downstream Mach number using downstream Prandtl Meyer function(PMF).
function f=fsolve11(Ma_2)
    f=Nu_Ma2-((sqrt((k+1)/(k-1))*atan(sqrt((k-1)/(k+1)*(Ma_2^2-1))))-(atan(sqrt(Ma_2^2-1))))*180/%pi
    //Multiplication by the factor '180/%pi' on the second R.H.S term of above equation is to convert the second term from 'radian' to '  (degree)'.
endfunction
```

Scilab code AP 29 `fsolve10.sci`

```
//Chapter 12 example 12-11
//Following is the function to find shock angle  .
function f=fsolve10(Beta)
    f=tan(theta)-((2/tan(Beta)*(Ma_1^2*sin(Beta)*sin(Beta)-1))/((Ma_1^2*(k+cos(2*Beta))+2))
endfunction
```

Scilab code AP 36 `fsolve7.sci`

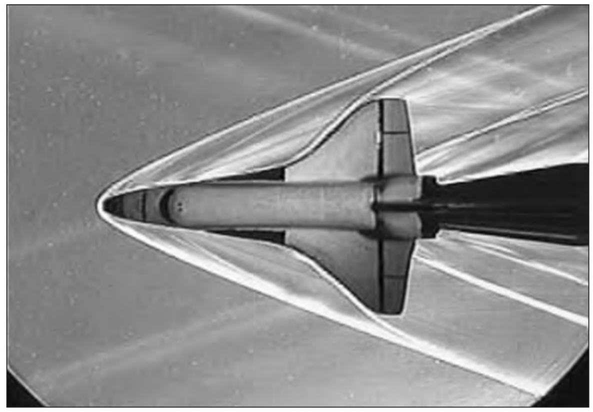


FIGURE 12–36
Schlieren image of a small model of the space shuttle Orbiter being tested at Mach 3 in the supersonic wind tunnel of the Penn State Gas Dynamics Lab. Several oblique shocks are seen in the air surrounding the spacecraft.
Photo by G. S. Settles, Penn State University. Used by permission.

FIGURE$_1$2$_3$6

```
//Chapter 8 example 8-9
//Following is the function to determine Volume flow
    rate x(1), Friction factor x(2), Velocity x(3),
   Reynolds number x(4) by solving 4 non linear
   equations using fsolve function
function f1=fsolve7(x)
    f1(1)=(h_L*2*g)-((Sigma_K_L_shower+((L_shower/D)
        *x(2)))*x(3)^2)
    f1(2)=x(3)-(x(1)/A)
    f1(3)=x(4)-(x(3)*D/Nu)
    f1(4)=(1/x(2))-(4*(log10((Epsilon_by_D/3.7)
        +(2.51/(x(4)*sqrt(x(2))))))^2)
endfunction
```

Scilab code AP 37 `fsolve8.sci`

```
//Chapter 8 example 8-9
//Following is the function to determine Total
   volume flow rate (y(1)), volume flow rate 1 (y(2)
   ), volume flow rate 2 (y(3)), Friction factor 1 (
   y(4)), Friction factor 2 (y(5)), Friction factor
   3 (y(6)), Velocity 1 (y(7)), Velocity 2 (y(8)),
   Velocity 3 (y(9)), Reynolds number 1 (y(10)),
   Reynolds number 2 (y(11)) and  Reynolds number 3
```

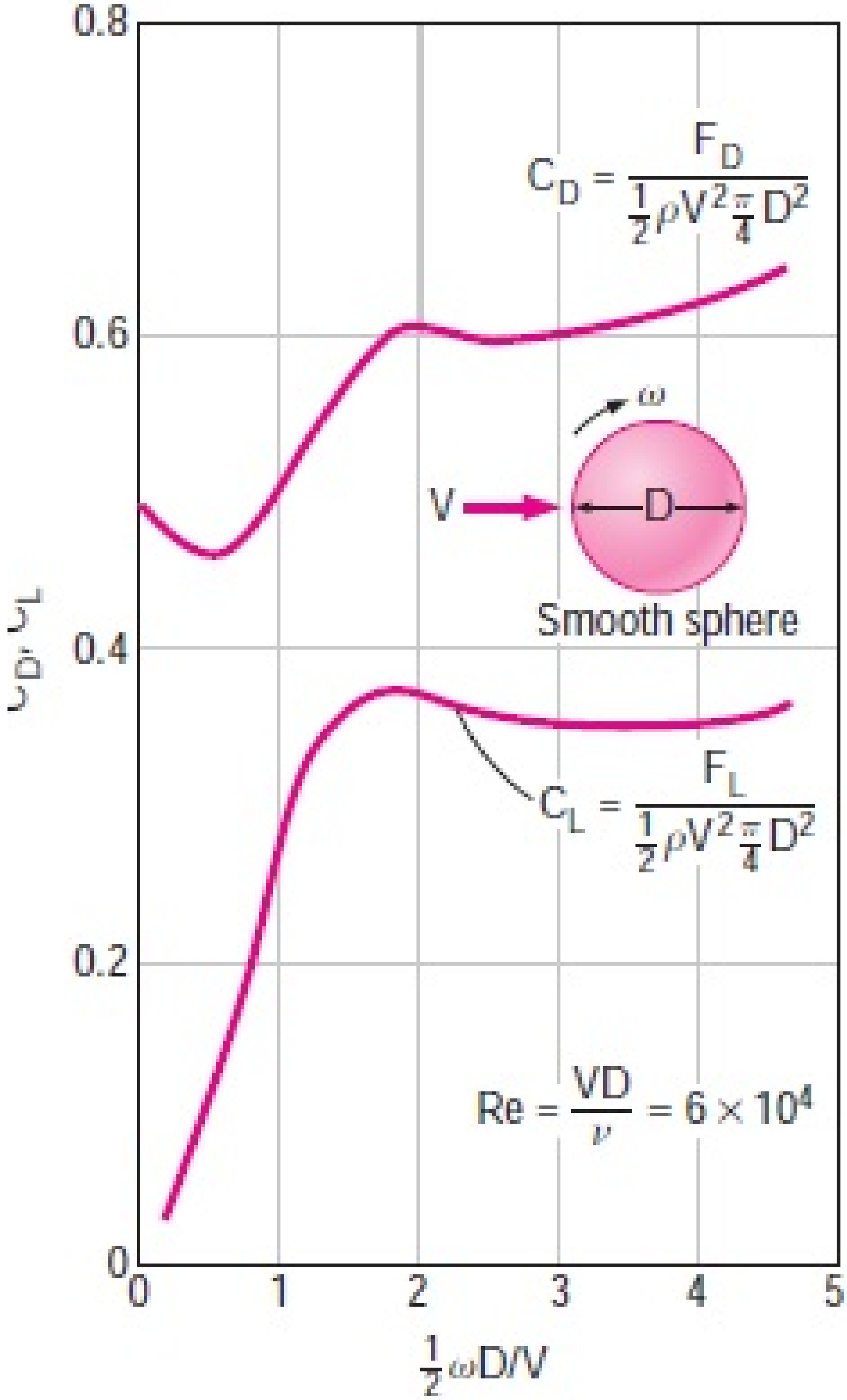


FIGURE 11–53
The variation of lift and drag coefficients of a smooth sphere with

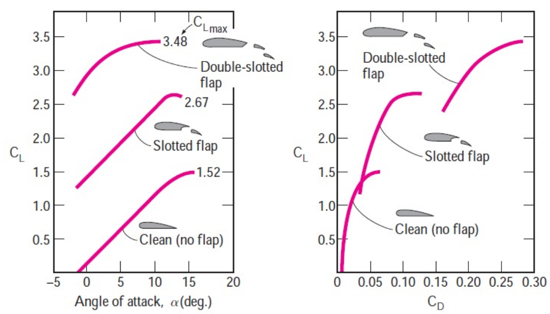


FIGURE 11–45
Effect of flaps on the lift and drag coefficients of an airfoil.
From Abbott and von Doenhoff, for NACA 23012 (1959).

FIGURE$_{1}$1$_{4}$5

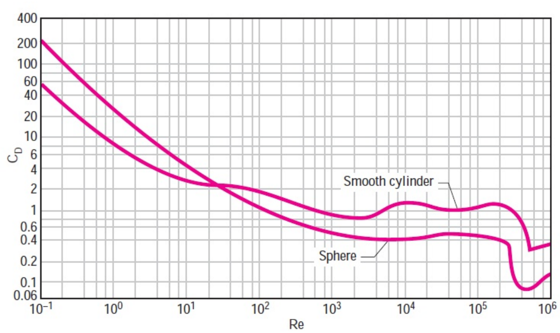


FIGURE 11–34
Average drag coefficient for cross-flow over a smooth circular cylinder and a smooth sphere.
From H. Schlichting, Boundary Layer Theory 7e. Copyright © 1979 The McGraw-Hill Companies, Inc. Used by permission.

FIGURE$_{1}$1$_{3}$4

TABLE 10–4

Summary of expressions for laminar and turbulent boundary layers on a smooth flat plate aligned parallel to a uniform stream*

Property	Laminar	(a) Turbulent(†)	(b) Turbulent(‡)
Boundary layer thickness	$\frac{\delta}{x} = \frac{4.91}{\sqrt{Re_x}}$	$\frac{\delta}{x} \cong \frac{0.16}{(Re_x)^{1/7}}$	$\frac{\delta}{x} \cong \frac{0.38}{(Re_x)^{1/5}}$
Displacement thickness	$\frac{\delta^*}{x} = \frac{1.72}{\sqrt{Re_x}}$	$\frac{\delta^*}{x} \cong \frac{0.020}{(Re_x)^{1/7}}$	$\frac{\delta^*}{x} \cong \frac{0.048}{(Re_x)^{1/5}}$
Momentum thickness	$\frac{\theta}{x} = \frac{0.664}{\sqrt{Re_x}}$	$\frac{\theta}{x} \cong \frac{0.016}{(Re_x)^{1/7}}$	$\frac{\theta}{x} \cong \frac{0.037}{(Re_x)^{1/5}}$
Local skin friction coefficient	$C_{f,x} = \frac{0.664}{\sqrt{Re_x}}$	$C_{f,x} \cong \frac{0.027}{(Re_x)^{1/7}}$	$C_{f,x} \cong \frac{0.059}{(Re_x)^{1/5}}$

* Laminar values are exact and are listed to three significant digits, but turbulent values are listed to only two significant digits due to the large uncertainty affiliated with all turbulent flow fields.

† Obtained from one-seventh-power law.

‡ Obtained from one-seventh-power law combined with empirical data for turbulent flow through smooth pipes.

TABLE$_{1}0_{4}$

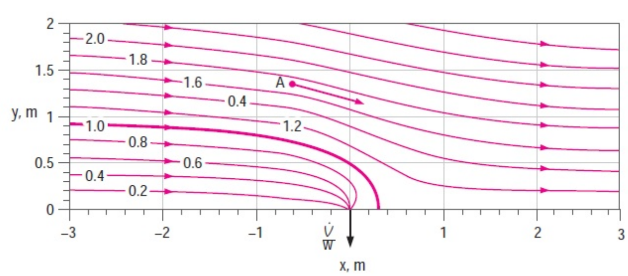


FIGURE 9–25
Streamlines for free-stream flow along a wall with a narrow suction slot; streamline values are shown in units of m^2/s; the thick streamline is the dividing streamline. The direction of the velocity vector at point A is determined by the left-side convention.

FIGURE$_{92}5$

```
  (y(12)) by solving 12 non linear equations using
   fsolve function
function f2=fsolve8(y)
    f2(1)=y(1)-y(2)-y(3)
    f2(2)=(h_L*2*g)-((y(4)*y(7)^2*Length1/D)+(((y(5)
       *Length2/D)+Sigma_K_L_shower)*(y(8)^2)))
    f2(3)=(h_L_3*2*g)-((y(4)*y(7)^2*Length1/D)+(((y
       (6)*Length3/D)+Sigma_K_L_reservoir)*(y(9)^2))
       )
    f2(4)=y(7)-(y(1)/A)
    f2(5)=y(8)-(y(2)/A)
    f2(6)=y(9)-(y(3)/A)
    f2(7)=y(10)-(D*y(7)/Nu)
    f2(8)=y(11)-(D*y(8)/Nu)
    f2(9)=y(12)-(D*y(9)/Nu)
    f2(10)=(1/y(4))-(4*(log10(Epsilon_by_D+(2.51/(y
       (10)*sqrt(y(4))))))^2)
    f2(11)=(1/y(5))-(4*(log10(Epsilon_by_D+(2.51/(y
       (11)*sqrt(y(5))))))^2)
    f2(12)=(1/y(6))-(4*(log10(Epsilon_by_D+(2.51/(y
       (12)*sqrt(y(6))))))^2)
endfunction
```

Scilab code AP 38 fsolve6.sci

```
//Chapter 8 example 8-8
//Following function is used to determination of
   friction factor from colebrook equation using
   fsolve function
function f=fsolve6(f0)
    f=(1/f0)-(4*(log10((Epsilon_by_D/3.7)+(2.51/(Re*
       sqrt(f0)))))^2)
endfunction
```

Scilab code AP 39 fsolve5.sci

```
//Chapter 8 example 8-7
```

```
//Following is the function to determine 13 unknown using 13 equations and fsolve function where x(1) is Total volume flow rate, x(2) is Volume flow rate in pipe 1, x(3) is Volume flow rate in pipe 2, x(4) is Velocity in pipe 1, x(5) is Velocity in pipe 2, x(6) is Head loss in pipe, x(7) is Head loss in pipe 1, x(8) is Head loss in pipe 2, x(9) is useful pump head, x(10) is Reynolds number in pipe 1, x(11) is Reynolds number in pipe 2, x(12) is Friction factor in pipe 1 and x(13) is Friction factor in pipe 2.
function f=fsolve5(x)
    f(1)=(W_elect*Eta_pump_motor)-(g*rho*x(1)*x(9))
    f(2)=x(9)-(Z_B-Z_A)-x(6)
    f(3)=x(6)-x(7)
    f(4)=x(6)-x(8)
    f(5)=x(4)-(x(2)/A_c_1)
    f(6)=x(5)-(x(3)/A_c_2)
    f(7)=x(10)-(rho*D1*x(4)/Mu)
    f(8)=x(11)-(rho*D2*x(5)/Mu)
    f(9)=(1/x(12))-(4*(log10(Epsilon_by_D1+(2.51/(x(10)*sqrt(x(12))))))^2)
    f(10)=(1/x(13))-(4*(log10(Epsilon_by_D2+(2.51/(x(11)*sqrt(x(13))))))^2)
    f(11)=(D1*2*g*x(7))-(L1*x(12)*x(4)^2)
    f(12)=(D2*2*g*x(8))-(L2*x(13)*x(5)^2)
    f(13)=x(1)-x(2)-x(3)
endfunction
```

Scilab code AP 40 `fsolve4.sci`

```
//Chapter 8 example 8-5
//Following is the function to determine New volume flow rate (x(1)), Friction factor (x(2)), Velocity (x(3)) and Reynolds Number (x(4)) by solving 4 nonlinear equation with fsolve function
function f=fsolve4(x)
    f(1)=x(3)-(x(1)/(%pi*D^2/4))
```

```
    f(2)=x(4)-(x(3)*D/Nu)
    f(3)=(h_L*D)-((L/(2*g))*x(2)*x(3)^2)
    f(4)=(1/x(2))-(4*(log10(2.51/(x(4)*sqrt(x(2)))))
       ^2)
endfunction
```

Scilab code AP 41 `fsolve3.sci`

```
//Chapter 8 example 8-4
//Following is the function to determine Velocity (x
   (1)), Diameter (x(2)), Reynolds number (x(3)) and
    Friction factor (x(4)) by solving 4 nonlinear
   equation using fsolve function.
function f=fsolve3(x)
    f(1)=(x(1)*x(2)^2)-((4*V_dot)/%pi)
    f(2)=x(3)-((1/Nu)*x(1)*x(2))
    f(3)=(h_L*x(2))-((L/(2*g))*x(4)*x(1)^2)
    f(4)=(1/x(4))-(4*(log10(2.51/(x(3)*sqrt(x(4)))))
       ^2)
endfunction
```

Scilab code AP 43 `fsolve2.sci`

```
//Chapter 8 exmaple 8-3
//Following function is used to determination of
   friction factor from colebrook equation using
   fsolve function
function f=fsolve2(f0)
    f=(1/f0)-(4*(log10((Epsilon_by_D/3.7)+(2.51/(Re*
       sqrt(f0)))))^2)
endfunction
```

Scilab code AP 46 `fsolve1.sci`

```
//Chapter 1 example 1-5
//Following is the function to determine the two
   required number using 'fsolve' function
```

TABLE 8–2

Equivalent roughness values for new commercial pipes*

	Roughness, ε	
Material	ft	mm
Glass, plastic	0 (smooth)	
Concrete	0.003–0.03	0.9–9
Wood stave	0.0016	0.5
Rubber, smoothed	0.000033	0.01
Copper or brass tubing	0.000005	0.0015
Cast iron	0.00085	0.26
Galvanized iron	0.0005	0.15
Wrought iron	0.00015	0.046
Stainless steel	0.000007	0.002
Commercial steel	0.00015	0.045

* The uncertainty in these values can be as much as ±60 percent.

TABLE$_{82}$

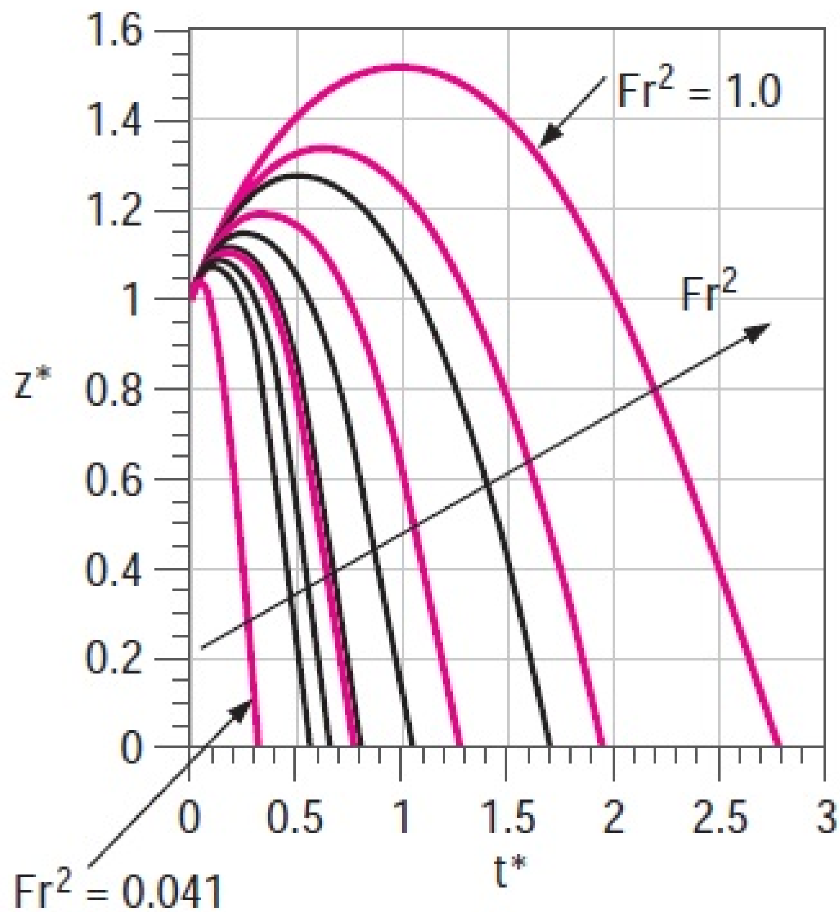


FIGURE 7–13
Trajectories of a steel ball falling in a vacuum. Data of Fig. 7–12a and b are nondimensionalized and combined onto one plot.

$\text{FIGURE}_{71}3$

TABLE 7–7

Wind tunnel data: aerodynamic drag force on a model truck as a function of wind tunnel speed

V, m/s	F_D, N
20	12.4
25	19.0
30	22.1
35	29.0
40	34.3
45	39.9
50	47.2
55	55.5
60	66.0
65	77.6
70	89.9

$TABLE_{77}$

```
function f=fsolve1(Number)
    f(1)=Number(1)-Number(2)-Difference
    f(2)=Number(1)^2-Number(1)+Number(2)^2-Number(2)
        -Sum
endfunction
```