Scilab Textbook Companion for Physics by R. Resnick, D. Halliday, K. S. Krane[1]

Created by
Joshi Prajakta Sanjay
3rd year,Engineering
Mechanical Engineering
Pune University
College Teacher
Gautam Chandekar
Cross-Checked by
K. V. P. Pradeep

July 11, 2017

[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: Physics

Author: R. Resnick, D. Halliday, K. S. Krane

Publisher: John Wiley And Sons , New Delhi

Edition: 5

Year: 2009

ISBN: 978-81-265-1088-7

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 4

1 MEASUREMENT 5

2 MOTION IN ONE DIMENSION 9

3 FORCE AND NEWTONS LAWS 25

4 MOTION IN TWO AND THREE DIMENSIONS 34

5 APPLICATIONS OF NEWTONS LAWS 46

6 MOMENTUM 58

7 SYSTEMS OF PARTICLES 72

8 ROTATIONAL KINEMATICS 81

9 ROTATIONAL DYNAMICS 88

10 ANGULAR MOMENTUM 104

11 ENERGY 1 WORK AND KINETIC ENERGY 108

12 ENERGY 2 POTENTIAL ENERGY 122

13 ENERGY 3 CONSERVATION OF ENERGY 126

14 GRAVITATION 135

15 FLUID STATICS **146**

16 FLUID DYNAMICS **151**

17 OSILLATIONS **155**

18 WAVE MOTION **165**

19 SOUND WAVES **172**

20 THE SPECIAL THEORY OF RELATIVITY **179**

21 TEMPERATURE **191**

22 MOLECULAR PROPERTIES OF GASES **194**

23 THE FIRST LAW OF THERMODYNAMICS **204**

24 ENTROPY AND THE SECOND LAW OF THERMODYNAMICS **216**

List of Scilab Codes

Exa 1.1 C1P1 5
Exa 1.2 C1P2 6
Exa 1.3 C1P3 7
Exa 1.5 C1P5 8
Exa 2.1 C2P1 9
Exa 2.2 C2P2 10
Exa 2.3 C2P3 11
Exa 2.4 C2P4 12
Exa 2.5 C2P5 14
Exa 2.6 C2P6 15
Exa 2.7 C2P7 17
Exa 2.8 C2P8 18
Exa 2.9 C2P9 20
Exa 2.10 C2P10 21
Exa 2.11 C2P11 23
Exa 3.1 C3P1 25
Exa 3.2 C3P2 26
Exa 3.3 C3P3 27
Exa 3.4 C3P4 28
Exa 3.5 C3P5 29
Exa 3.6 C3P6 30
Exa 3.7 C3P7 31
Exa 4.1 C4P1 34
Exa 4.2 C4P2 37
Exa 4.3 C4P3 38
Exa 4.4 C4P4 40
Exa 4.5 C4P5 42
Exa 4.6 C4P6 43

Exa 4.7 C4P7 44
Exa 5.1 C5P1 46
Exa 5.2 C5P2 47
Exa 5.3 C5P3 49
Exa 5.4 C5P4 50
Exa 5.7 C5P7 52
Exa 5.9 C5P9 53
Exa 5.10 C5P10 54
Exa 5.11 C5P11 56
Exa 6.1 C6P1 58
Exa 6.2 C6P2 60
Exa 6.3 C6P3 62
Exa 6.4 C6P4 63
Exa 6.5 C6P5 65
Exa 6.6 C6P6 66
Exa 6.7 C6P7 67
Exa 6.8 C6P8 68
Exa 6.9 C6P9 70
Exa 7.2 C7P2 72
Exa 7.3 C7P3 75
Exa 7.7 C7P7 76
Exa 7.8 C7P8 77
Exa 7.9 C7P9 78
Exa 7.10 C7P10 80
Exa 8.1 C8P1 81
Exa 8.2 C8P2 82
Exa 8.3 C8P3 83
Exa 8.4 C8P4 84
Exa 8.5 C8P5 85
Exa 8.6 C8P6 87
Exa 9.1 C9P1 88
Exa 9.2 C9P2 89
Exa 9.3 C9P3 91
Exa 9.6 C9P6 92
Exa 9.7 C9P7 93
Exa 9.8 C9P8 95
Exa 9.9 C9P9 97
Exa 9.10 C9P10 99

Exa 9.12 C9P12 100
Exa 9.13 C9P13 101
Exa 10.2 C10P2 104
Exa 10.4 C10P4 106
Exa 10.5 C10P5 107
Exa 11.1 C11P1 108
Exa 11.2 C11P2 109
Exa 11.3 C11P3 110
Exa 11.4 C11P4 112
Exa 11.6 C11P6 113
Exa 11.7 C11P7 114
Exa 11.8 C11P8 115
Exa 11.9 C11P9 115
Exa 11.10 C11P10 117
Exa 11.11 C11P11 118
Exa 11.12 C11P12 120
Exa 12.1 C12P1 122
Exa 12.2 C12P2 123
Exa 12.3 C12P3 123
Exa 12.4 C12P4 124
Exa 12.7 C12P7 125
Exa 13.1 C13P1 126
Exa 13.2 C13P2 127
Exa 13.3 C13P3 129
Exa 13.4 C13P4 130
Exa 13.5 C13P5 131
Exa 13.6 C13P6 132
Exa 13.7 C13P7 133
Exa 14.1 C14P1 135
Exa 14.2 C14P2 136
Exa 14.3 C14P3 138
Exa 14.4 C14P4 139
Exa 14.5 C14P5 139
Exa 14.7 C14P7 140
Exa 14.8 C14P8 141
Exa 14.9 C14P9 142
Exa 14.10 C14P10 143
Exa 15.1 C15P1 146

Exa 15.2 C15P2 147
Exa 15.3 C15P3 148
Exa 15.4 C15P4 149
Exa 15.5 C15P5 149
Exa 16.1 C16P1 151
Exa 16.2 C16P2 152
Exa 16.3 C16P3 154
Exa 17.1 C17P1 155
Exa 17.2 C17P2 156
Exa 17.3 C17P3 158
Exa 17.4 C17P4 160
Exa 17.6 C17P6 160
Exa 17.7 C17P7 161
Exa 17.8 C17P8 162
Exa 17.9 C17P9 163
Exa 18.1 C18P1 165
Exa 18.2 C18P2 166
Exa 18.3 C18P3 168
Exa 18.4 C18P4 169
Exa 18.5 C18P5 169
Exa 19.1 C19P1 172
Exa 19.2 C19P2 173
Exa 19.3 C19P3 174
Exa 19.4 C19P4 175
Exa 19.5 C19P5 176
Exa 19.6 C19P6 177
Exa 20.1 C20P1 179
Exa 20.2 C20P2 180
Exa 20.3 C20P3 181
Exa 20.4 C20P4 182
Exa 20.5 C20P5 183
Exa 20.6 C20P6 184
Exa 20.7 C20P7 185
Exa 20.8 C20P8 186
Exa 20.9 C20P9 187
Exa 20.10 C20P10 188
Exa 20.11 C20P11 189
Exa 21.1 C21P1 191

Exa 21.2 C21P2 191
Exa 21.3 C21P3 192
Exa 22.1 C22P1 194
Exa 22.2 C22P2 195
Exa 22.3 C22P3 196
Exa 22.4 C22P4 198
Exa 22.5 C22P5 199
Exa 22.6 C22P6 200
Exa 22.7 C22P7 201
Exa 22.9 C22P9 202
Exa 23.2 C23P2 204
Exa 23.3 C23P3 205
Exa 23.4 C23P4 206
Exa 23.5 C23P5 208
Exa 23.6 C23P6 208
Exa 23.7 C23P7 210
Exa 23.8 C23P8 211
Exa 23.9 C23P9 212
Exa 24.1 C24P1 216
Exa 24.2 C24P2 217
Exa 24.3 C24P3 218
Exa 24.4 C24P4 219
Exa 24.5 C24P5 220
Exa 24.6 C24P6 221
Exa 24.7 C24P7 222
Exa 24.8 C24P8 223
Exa 24.9 C24P9 224

Chapter 1

MEASUREMENT

Scilab code Exa 1.1 C1P1

```
clear
 clc
//To find speed in meters per second
//to find volume in cubic centimeters

// GIVEN::

//speed
speed1 = 55 //miles per hour
//volume of gasoline
volume1 = 16 //gallons

// SOLUTION:

//speed in meters per second
// since 1 mile = 1609 meters and 1 hour = 3600
   seconds
speed = (55)*(1609/1)*(1/3600) //miles per hour

//volume of gasoline in cubic centimeters
// since 1 gallon = 231 cubic inches and 1 inch =
```

```
2.54 centimeter
volume = 16*(231/1)*(2.54/1)^3 // cubic centimeters
speed = round(speed)
printf ("\n\n Speed in meters per second speed =\n\n %2i m/s" ,speed);
printf ("\n\n Volume of gasoline in cubic centimeters volume =\n\n %.1e cm^3" ,volume);
```

Scilab code Exa 1.2 C1P2

```

clear
 clc
//To find conversion factor between light year ang meters
//to find distance to the star proxima centuri

// GIVEN::

//distance
d = 4*10^16 //in light years
// velocity of light
v = 3.00*10^8 // m/s

// SOLUTION:

//conversion factor
// first finding conversion between 1 year and seconds
y = (1)*(365.25/1)*(24/1)*(60/1)*(60/1) // seconds
//now finding conversion between light year ang meters
light_year = (y*v) // meters

//to find distance to the star proxima centuri
```

```
distance = (d)*(1/light_year)// light years
light_year = nearfloat("pred",9.48e15)
printf ("\n\n Conversion between 1 year and seconds y = \n\n %.2e seconds" ,y);
printf ("\n\n Conversion between light year ang meters 1 light year =\n\n  %.2e m" ,light_year);
printf ("\n\n Distance to the star proxima centuri distance =\n\n  %.1f light years" ,distance);
```

Scilab code Exa 1.3 C1P3

```

clear
 clc
//to find fractional and percentage uncertainty in your weight
//to find fractional and percentage uncertainty in cat's weight

// GIVEN::

//your weight
w1 = 119 //in lbs
// your and cat's combined weight
w2 = 128 // in lbs

// SOLUTION:

//fractional uncertainty in your weight
u1 = (1/119)
// percentage uncertainty in your weight
u2 = u1*100 //percentage
//fractional uncertainty in cat's weight
u3 = (1/9)
//percentage uncertainty in cat's weight
```

```
u4 = u3*100 //percentage
printf ("\n\n Fractional uncertainty in your weight
    u1 =\n\n %.3f" ,u1);
printf ("\n\n Percentage uncertainty in your weight
    u2 =\n\n %.1f percent" ,u2);
printf ("\n\n Fractional uncertainty in cats weight
    =\n\n %.2f" ,u3);
printf ("\n\n Percentage uncertainty in cats weight
    =\n\n %1i percent" ,u4);
```

Scilab code Exa 1.5 C1P5

```


clear
clc
//to find value of plank time
// GIVEN::

//speed of light
c = 3.00e8  //m/s
//Newton's gravitational constant
G = 6.67e-11 // m^3/s^2.Kg
//plank's constant
h = 6.63e-34// Kg.m^2/s

// SOLUTION:

//plank time
tp = sqrt((G*h)/c^5)// seconds
//answer in the book is slightly different which is
    printing mistake
printf ("\n\n Plank time tp =\n\n %.2e seconds" ,tp
    );
```

Chapter 2

MOTION IN ONE DIMENSION

Scilab code Exa 2.1 C2P1

```

clear
 clc
//to find distance travelled to the north and east
   does the airplane travel

// GIVEN::

//distance travelled by airplane
d = 209// in km
//angle made by airplane east of due north that is
   angle made with y direction
theta = 22.5// in degrees

// SOLUTION:

//angle made by airplane with x direction
fi = 90-theta//in degrees
// distance travelled to the north
```

```
dy = d*sind(fi)
//distance travelled to the east
dx = d*cosd(fi)
printf ("\n\n Angle made by airplane with x
   direction fi =\n\n %.1f degrees",fi);
printf ("\n\n Distance travelled to the north by
   airplane dx =\n\n %.1f km",dx);
printf ("\n\n Distance travelled to the east by
   airplane dy  =\n\n %3i km",dy);
```

Scilab code Exa 2.2 C2P2

```

clear
 clc
//to find magnitude and direction of vector
   indicating location of car

// GIVEN::

//distance travelled due east on a level of road
//s is represented as ax+by.since b has no x
   component and a has no y componebt we can write
Sx = 32// in km
//distance travelled before stopping after taking
   turn due north
Sy = 47// in km


// SOLUTION:

//magnitude of distance travelled
x = sqrt(Sx^2 + Sy^2)//in meters
//direction of travelling
fi = atand(Sy/Sx)//in degrees
```

```
g = Sy/Sx
x = round(x)
fi = round(fi)
printf ("\n\n Distance travelled due east on a level
    of road Sx =\n\n %2i km",Sx);
printf ("\n\n Distance travelled before stopping
    after taking turn due north Sy =\n\n %2i km",Sy);
printf ("\n\n Magnitude ofdistance travelled by
    automobile x =\n\n %2i km",x);
printf ("\n\n Value of tanfi =\n\n %.2f ",g);
printf ("\n\n Direction of automobile travelling fi
    =\n\n %2i degrees north of east",fi);
```

Scilab code Exa 2.3 C2P3

```

clear
 clc
//to find magnitude and direction of resultant of a
    and b and c vector

// GIVEN::

//coefficient in x direction for vector a
ax = 4.3
//coefficient in y direction for vector a
ay = -1.7
//coefficient in x direction for vector b
bx = -2.9
//coefficient in y direction for vector b
by = 2.2
//coefficient in x direction for vector c
cx = 0
//coefficient in y direction for vector c
cy = -3.6
```

```
//we can write a,b and c in vector form
a = [4.3 -1.7]
b = [-2.9 2.2]
c = [0 -3.6]

// SOLUTION:

//coefficient in x direction for resultant vector
sx = ax + bx + cx
//coefficient in y direction for resultant vector
sy = ay + by + cy
//direction of resultant vector
fi = atand(sy/sx)+360
printf ("\n\n Coefficient of resultant vector in x
    direction sx = \n\n %.1f",sx);
printf ("\n\n Coefficient of resultant vector in y
    direction sy =\n\n %.1f",sy);
printf ("\n\n Resultant vector s =\n\n %.1fi + %.1fj
    ',sx,sy);
printf ("\n\n Direction of resultant vector with
    positive x axis measured counterclockwise fi =\n\
    n %3i degrees",fi);
```

Scilab code Exa 2.4 C2P4

```

clear
 clc
//to find magnitude of position,velocity and
    acceleration

// GIVEN::
//time
t = 3//in seconds
//coefficients
```

```
A = 1.00//in m/s^2
B = -32.0//in m/s
C = 5.0//in m/s^2
D = 12.0//in m


// SOLUTION:

//for position vector
//coefficient in x direction for resultant vector
rx = A*t^3 + B*t
//coefficient in y direction for resultant vector
ry = C*t^2 + D

//for velocity vector
//coefficient in x direction for resultant vector
//as v = dx/dt therefore differentiating rx and ry w
    .r.t t
vx = 3*A*t^2 + B
//coefficient in y direction for resultant vector
vy =2* C*t

//for acceleration vector
//as a = dv/dt therefore differentiating rx and ry
    again w.r.t t
//coefficient in x direction for resultant vector
ax = 6*A*t
//coefficient in y direction for resultant vector
ay = 2*C
printf ("\n\n Position vector r =\n\n %1i m i + %1i
    m j',rx,ry);
printf ("\n\n Velocity vector v =\n\n %1i m/s i +
    %1i m/s j',vx,vy);
printf ("\n\n Acceleration vector a =\n\n %1i m/s^2
    i + %1i m/s^2 j',ax,ay);
```

Scilab code Exa 2.5 C2P5

```
clear
 clc
//to find average velocity of car

// GIVEN::

//distance travelling by car
d1 = 5.2//in mi
//distance travelled while walking
d2 = 1.2//in mi
//time required to reach to gas station while
   walking
t1 = 27//in min
//speed of car
v = 43//in mi/h

// SOLUTION:

//net displacement
delta_x = d1 + d2//in mi
//speed of car in mi/minutes
v1 = v/60//in mi/minutes
//total elapsed time
delta_t1 = (d1/v1) + t1//in min
//total elapsed time in h
delta_t = delta_t1/60//in h
//average velocity
//applying kinematic equations
Vav_x = delta_x/delta_t//in mi/h
printf ("\n\n Net displacement delta_x =\n\n %.1f mi
   ",delta_x);
```

```
printf("\n\n Total elapsed time delta_t =\n\n %.2f h",delta_t);
printf("\n\n Average velocity of car required Vav_x =\n\n %.1f mi/h",Vav_x);
```

Scilab code Exa 2.6 C2P6

```

clear
 clc
//to find average velocity for interval AD and DF
//to find slope of position curve at the points B and F and compare it with the value in velocity curve
//to find average acceleration in the interval AD and AF
//to find slope of velocity curve at the points D and compare it with the value in acceleration curve

// GIVEN::

//distance travelling by the point D has come
xD = 5.0//in m
//distance travelling by the point A has come
xA = 1.0//in m
//distance travelling by the point F has come
xF = 1.4//in m
//time elapsed by the point D has come
tD = 2.5//in seconds
//time elapsed by the point A has come
tA = 0.0//in seconds
//time elapsed by the point F has come
tF = 4.0//in seconds
//velocity at point D
```

```
vD = 0.0//in m/s
//velocity at point A
vA = 4.0//in m/s
//velocity at point F
vF = -6.2//in m/s



// SOLUTION:

//average velocity for the interval AD
//applying kinematic equations
Vav_x = (xD-xA)/(tD-tA)
//average velocity for the interval DF
//applying kinematic equations
Vavx = (xF-xD)/(tF-tD)
//slope of position curve at the point B
slope_B = (4.5-2.8)/(1.5-0.5)//refer to the graph 2.6(b) given in the book on page no. 25
//slope of position curve at the point F
slope_F = (1.4-4.5)/(4.0-3.5)//refer to the graph 2.6(b) given in the book on page no. 25
//average acceleration in the interval AD
//applying kinematic equations
Aav_x = (vD-vA)/(tD-tA)//in m/s^2
//average acceleration in the interval AF
//applying kinematic equations
Aavx = ((vF-vA)/(tF-tA))//in m/s^2
Aavx = nearfloat("pred",-2.6)
//slope of velocity curve at the point D
slope_D = (-0.9-0.9)/(3.0-2.0)//refer to the graph 2.6(c) given in the book on page no. 25

printf ("\n\n Average velocity for the interval AD Vav_x=\n\n %.1f m/s",Vav_x);
printf ("\n\n Average velocity for the interval DF V_avx =\n\n %.1f m/s",Vavx);
printf ("\n\n Slope of position curve at the point B
```

```
    slpoe_B=\n\n %.1f m/s",slope_B);
printf ("\n\n Slope of position curve at the point F
    =\n\n %.1f m/s",slope_F);
//refer velocity time graph 2.6(c) given in the book
    on the page no.25
printf ("\n\n From velocity curve value of velocity
    at point B is \n\n 1.7m/s");
printf ("\n\n From velocity curve value of velocity
    at point Bis \n\n -6.2m/s");
printf ("\n\n Average acceleration for the interval
    AD Aav_x =\n\n %.1f m/s^2",Aav_x);
printf ("\n\n Average acceleration for the interval
    AF Aavx =\n\n %.1f m/s^2",Aavx);
printf ("\n\n Slope of velocity curve at the point D
    slope_D =\n\n %.1f m/s^2",slope_D);
//refer velocity time graph 2.6(d) given in the book
    on the page no.25
printf ("\n\n From acceleration curve value of
    acceleration at point D is \n\n -1.8m/s^2");
```

Scilab code Exa 2.7 C2P7

```

clear
 clc
//to find acceleration of partical
//to find velocity of partical when it leaves the
    tube

// GIVEN::

//length of the tube
x = 2.0//in m
//velocity of partical when it enters in the tube i.
    e.at t=0s
```

```
v0x = 9.5*10^5//in m/s
//time when the partical emerges out of the tube
t = 8.0*10^-7//in m/s

// SOLUTION:

//acceleration of the partical
//applying kinematic equations
ax = (x-(v0x*t))/(0.5*t^2)//in m/s^2
//velocity of the partical when it leaves the tube
//applying kinematic equations
vx = v0x + (ax*t)

printf ("\n\n Acceleration of the partical ax =\n\n
   %.1e m/s^2",ax);
printf ("\n\n Velocity of the partical when it
   leaves the tube vx =\n\n %.1e m/s",vx);
```

Scilab code Exa 2.8 C2P8

```

clear
 clc
//to find timw elapsed
//to find acceleration
//after apply brakes with constant acceleration time
    required to stop vehicle
//additionl distance covered after vehicle has
   stopped

// GIVEN::

//linitial velocity at t=0
v0x = 23.6//in m/s
//final velocity
```

```
vx = 12.5//in m/s
//distance travelled
delta_x = 105//in m
//velocity after vehicle has stopped
vxf = 0//in m/s

// SOLUTION:

//average velocity
//applying kinematic equations
vav_x = (v0x + vx)/2//in m/s
//time elapsed
time = delta_x/vav_x//in seconds
//acceleration
//applying kinematic equations
ax = (vx-v0x)/time//in m/s^2
//time required to stop vehicle
//applying kinematic equations
t = (vxf-v0x)/ax//in s
//total distance covered by vehicle
//applying kinematic equations
x = (v0x*t)+(0.5*ax*t^2)//in m/s^2
//additional distance travelled by vehicle
x_final = x - delta_x//in m
x = round(x)
x_final = round(x_final)
time = nearfloat("pred",5.81)

printf ("\n\n Average velocity vav_x =\n\n %.2f m/s",vav_x)
printf ("\n\n Time elapsed time =\n\n %.2f s",time);
printf ("\n\n Acceleration of vehicle ax =\n\n %.2f m/s^2",ax);
printf ("\n\n After apply brakes with constant acceleration time required to stop vehicle t =\n\n %.1f s",t);
printf ("\n\n Total distance covered by vehicle x =\n\n %3i m",x);
```

```
printf ("\n\n Additionl distance covered after
    vehicle has stopped x_final =\n\n %1i m",x_final)
    ;
```

Scilab code Exa 2.9 C2P9

```

clear
 clc
//to find position and acceleration after t=1,2,3,4s
    have elapsed


// GIVEN::

//linitial velocity due free fall of body
v0y = 0//in m/s
//acceleration due to gravity
g = 9.8//in m/s^2
//time elapsed
t1 = 1.0//in s
t2 = 2.0//in s
t3 = 3.0//in s
t4 = 4.0//in s


// SOLUTION:

//velocity v = -(g*t)
//since initial velocity is zero
v1 = (v0y*t1)-(g*t1)//in m/s
v2 = (v0y*t2)-(g*t2)//in m/s
v3 = (v0y*t3)-(g*t3)//in m/s
v4 = (v0y*t4)-(g*t4)//in m/s
//since body is moving vertically downwards s0,
```

```
velocity has -ve sign
//distance travelled y = -(0.5*g*t^2)
y1 = (v0y*t1)-0.5*(g*t1^2)//in m
y2 = (v0y*t2)-0.5*(g*t2^2)//in m
y3 = (v0y*t3)-0.5*(g*t3^2)//in m
y4 = (v0y*t4)-0.5*(g*t4^2)//in m
// -ve sign indicates body is travelling in -ve y direction
printf ("\n\n Distance travelled after elapsed time t1 =\n\n %.1f m",y1);
printf ("\n\n Distance travelled after elapsed time t2 =\n\n %.1f m",y2);
printf ("\n\n Distance travelled after elapsed time t3 =\n\n %.1f m",y3);
printf ("\n\n Distance travelled after elapsed time t4 =\n\n %.1f m",y4);
printf ("\n\n Velocity after elapsed time t1 =\n\n %.1f m/s",v1);
printf ("\n\n Velocity after elapsed time t2 =\n\n %.1f m/s",v2);
printf ("\n\n Velocity after elapsed time t3 =\n\n %.1f m/s",v3);
printf ("\n\n Velocity after elapsed time t4 =\n\n %.1f m/s",v4);
```

Scilab code Exa 2.10 C2P10

```

clear
 clc
//to find time required to reach highest point
//to find distance travelled by the ball till the highest position is reached
//to find time at which ball will be 27m above the groung
```

```

// GIVEN::

//initial speed of the ball
v0y = 25.2//in m/s
//final speed of the ball
vy = 0//in m/s
//acceleration due to gravity
g = 9.8//in m/s^2
//for calculating time distance of ball above the ground
y1 = 27.0//in meters



// SOLUTION:
//time required to reach highest psition
//applying kinematic equations
t = (v0y-vy)/g//in seconds
//distance travelled by the ball till the highest position is reached
//applying kinematic equations
y = (v0y*t)-(1/2*g*t^2)//in meters
//time at which ball will be 27m above the groung
//solving quadratic equation
y1 = poly([y1 -(v0y) (1/2*g)],'t','coeff')
c = roots(y1)
t1 = c(1)
t2 = c(2)
//velocity of ball at t1
vy1 = v0y-(g*t1)//in m/s
//velocity of ball at t2
vy2 = v0y-(g*t2)//in m/s

printf ("\n\n Time required to reach highest psition t =\n\n %.2f s",t);
printf ("\n\n Distance travelled by the ball till the highest position is reached y =\n\n %.1f m",y
```

```
    );
printf ("\n\n Time at which ball will be 27m above
    the groung t1 =\n\n %.2f s",t1);
printf ("\n\n Time at which ball will be 27m above
    the groung t2 =\n\n %.2f s",t2);
printf ("\n\n Velocity of ball at t1 =\n\n %.1f m/s"
    ,vy1);
printf ("\n\n Velocity of ball at t2 =\n\n %.1f m/s"
    ,vy2);
```

Scilab code Exa 2.11 C2P11

```

clear
 clc
//maximum distance travelled by the rocket above the
    water surface


// GIVEN::

//distance below the water surface
//this can be written as y-y0 = s =125
s = 125//in meters
//initial velocity of rocket
v0y = 0//in m/s
//acceleration due to gravity
g = 9.8//in m/s^2
//time required to reach the water surface
t = 2.15//in seconds
//velocity of rocket at highest position
v2 = 0//in m/s^2

// SOLUTION:
//acceleration of rocket in upward direction
```

```
//applying kinematic equations
ay = (2*s)/t^2//in m/s^2
//final velocity of the rocket at the surface of
    water
//applying kinematic equations
vy = v0y+(ay*t)//in m/s
//now taking v3 as initial velocity of rocketi.e.
    velocity at the water surface level
//so, at highest rocket will have 0 velocity which
    we will take as final velocity of rocket
//time required to reach highest position from water
    surface
//applying kinematic equations
time = (vy-v0y)/g//in seconds
//maximum distance travelled by the rocket above the
    water surface
//applying kinematic equations
y = (vy*time)-(0.5*g*time^2)//in meters
time = nearfloat("pred",11.8)
y = nearfloat("pred",688)
printf ("\n\n Acceleration of rocket in upward
    direction ay = \n\n %.1f m/s^2",ay);
printf ("\n\n Final velocity of the rocket at the
    surface of water vy = \n\n %3i m/s",vy);
printf ("\n\n Time required to reach highest
    position from water surface time = \n\n %.1f
    seconds",time);
printf ("\n\n Maximum distance travelled by the
    rocket above the water surface y = \n\n %2i m",y)
    ;
```

Chapter 3

FORCE AND NEWTONS LAWS

Scilab code Exa 3.1 C3P1

```

 clc

 //To Find the final velocity of sled

 //Given :
 //refer to figure 3-7(a) and3-7(b) from page no. 49
 //mass of sled
 m =240//in kg
 //distance travelled
 d =2.3//in m
 //force applied
 Fsw =130//in N

//solution:

//calculating first acceleration
//applying newton's second law
ax =Fsw/m //m/ s ^2
```

```
//calculating time required to move sled by distance d
//applying kinemtic equation
t =((2*d)/ax)^(1/2) //in seconds
// calculating velocity
//applying kinemtic equation
vx =ax*t //m/ s
printf ("\n\n Acceleration ax = \n\n %.2fm/s^2" ,ax)
printf ("\n\n final velocity vx = \n\n %.1f m/s" ,vx);
```

Scilab code Exa 3.2 C3P2

```

//To Find the force to be apply on sled
//referring to data from problem 3.1 on page no.48
 clc;

 //Given :
 //refer to figure 3-7(a) and3-7(b) from page no. 49
m =240; //kg
d =2.3; //distance travelled in m
Fsw =130; //force in N

//solution:
//calculating acceleration
//applying newton's second law
ax1 = Fsw/m //m/s^2
//calculating time
//applying kinematic equation
t = sqrt((2*d)/ax1) // seconds
//calculating velocity
//applying one dimensinal kinematic equation
vx =ax1*t; //m/ s
v0x = -(ax1*t); //m/s
```

```
t2 = 4.5 // seconds
// calculating first acceleration using equation vx = v0x + ax*t
ax =(v0x-vx)/t2; //m/s^2
ax = nearfloat("succ",0.71)

// calculating force
//applying newton's second law
F_dashsw = m*ax; // N
F_dashsw = nearfloat("pred",-170)
F_dashsw1 = F_dashsw/(0.4535*9.81)

printf ("\n\n Acceleration ax1 = \n\n %.2f m/s^2" ,ax1)
printf ("\n\n Time t = \n\n %.1f s" ,t)
printf ("\n\n Final velocity vx = \n\n %.1f m/s" ,vx)
printf ("\n\n Final velocity v0x = \n\n %.1f m/s" ,v0x)
printf ("\n\n Acceleration ax = \n\n %.2f m/s^2" ,ax)
printf ("\n\n Constant force F_dashsw = \n\n %3i N" ,F_dashsw);
printf ("\n\n Constant force F_dashsw1 = \n\n %2i lb" ,F_dashsw1);
```

Scilab code Exa 3.3 C3P3

```

 clc

//To find force acting on crate

//Given :
//refer to figure 3-8(a) and3-8(b) from page no. 49
```

```
// mass
m =360 //kg
// initial velocity of crate
vx1 =62//km/ph
// final velocity of crate
v0x1 = 105 //km/ph
// time elapsed
t =17 //seconds

//solution:
//calculating initial velocity in m/s
vx =(62*5)/18 //in m/s
// calculating final velocity in m/s
v0x =(105*5)/18 //in m/s
//calculating acceleration
ax =(vx-v0x)/t //in m/s^2
//calculating force
//applying newton's secong law
Fct =m*ax //in seconds
ax = nearfloat("succ",-0.70)
Fct = nearfloat("pred",-250)

printf ("\n\n Acceleration a = \n\n %.2fm/s^2" ,ax)
printf ("\n\n Force acting on crate Fct =\n\n %.3iN"
    ,Fct);
```

Scilab code Exa 3.4 C3P4

```


clear
 clc

//To find force acting on crate

```

```
 //Given :
 //refer to figure 3-17(a) and3-17(b) from page no. 55
 // mass of first crate
 m1 =4.2 //kg
 // mass of second crate
 m2 =1.4 //kg
 // force on first crate
 P1w =3 //in N

//solution://for two crate remain in contact acc(crate 1)=acc(crate 2). we will call this as common acelration as a.
// calculating common acceleration of both crate in m/s^2
//applying newton's secong law
 a =P1w/(m1+m2) //m/s
// calculating force exerted on crate 2 by 1
//applying newton's secong law
 f21 =m2*a //  m/s^2
 f21 = nearfloat("succ",0.76)

printf ("\n\n Calculating common acceleration of both crate a = \n\n %.2fm/s^2" ,a)
printf ("\n\n Force acting on crate f21 = \n\n %.2f N" ,f21);
```

Scilab code Exa 3.5 C3P5

```



clear
 clc

```

```
//To find force acting on crate

 //Given :
  //refer to figure 3-18(a) and3-18(b) from page no.
      55
 // mass of flat-bed cart
 mc =360 //kg
 // mass of box
 mb =150 //kg
 // magnitude of acceleration for cart
 ac =0.167 // m/s^2
  // magnitude of acceleration for box
 ab =1 // m/s^2


//solution:
//force on cart
//applying newton's second law
Fcb =mc*ac //in N
// force on box
//applying newton's second law
Fbw =Fcb+(mb*ab)// in N

printf ("\n\n Force acting on crate Fcb = \n\n %2i N
   " ,Fcb);
printf ("\n\n Force acting on box Fbw = \n\n %2i N"
   ,Fbw);
```

Scilab code Exa 3.6 C3P6

```



clear
 clc
```

```
 //To find frictional force of the box on the cart
// referrinf to same problem as 3−5 on page no.55

 //Given :
 // mass of flat−bed cart
 mc =360 //kg
 // mass of box
 mb =150 //kg
 // magnitude of acceleration for cart
 ac =0.167 // m/s^2
  // magnitude of acceleration for box
 ab =1 // m/s^2


//solution:
// force on cart
//applying newton's second law
 Fcb =mc*ac //in N
// force on box
//applying newton's second law
Fbw =Fcb+(mb*ab)// in N
//frictional force
//applying newton's second law
Fcb =(mc*Fbw)/(mc+mb)// in N
Fcb = nearfloat("succ",150)
//answer of Fcb slightli varies.but answer by scilab
    is same as on calculator
printf ("\n\n Frictional force  of box on the cart
   fcb = \n\n %3iN" ,Fcb);
```

Scilab code Exa 3.7 C3P7

```
clc
//to find net force on passenger ang scale reading
   while descending and ascending

// GIVEN::
 //refer to figure 3-19(a) and3-19(b) from page no.
    56
//mass of passenger
m = 72.2 // in Kg
//acceleration of elevator while descending
a0y = 0// in m/s^2
// acceleration of elevator while ascending
ay = 3.20//in m/s^2
//acceleration due to gravity
g = 9.81//in m/s^2

// SOLUTION:

//passenger while descending
//applying newton's second law
Fps_d = m*(g+a0y)//in m/s^2
Fps_d1 = Fps_d/(g*.4535)//in lb
//passenger while ascending
//applying newton's second law
Fps_a = m*(g+ay)//in m/s^2
Fps_a1 = Fps_a/(g*.4535)//in lb
printf ("\n\n Net force on passenger while
   descending Fps_d = \n\n %3i N" ,Fps_d);
printf ("\n\n Net force on passenger while
   descending Fps_d1 = \n\n %3i lb" ,Fps_d1);
printf ("\n\n Net force on passenger while ascending
    Fps_a = \n\n %3i N" ,Fps_a);
printf ("\n\n Net force on passenger while ascending
    Fps_a1 = \n\n %3i lb" ,Fps_a1);
printf ("\n\n Scale raeding will not change while
   descending due to constant acceleration
   whilescale reading will increase while ascending
   due to increase in acceleration");
```

Chapter 4

MOTION IN TWO AND THREE DIMENSIONS

Scilab code Exa 4.1 C4P1

```
clc
//to find ship's velocity and position relative to
   its locationwhen the tractor beam first appeared

// GIVEN::
//refer to figure 4-1 from page no.66
//problem mainly divides into two parts
//1. t=0 to t=4 seconds    //FIRST PART
//2. t=4 to t=7 seconds    //SECOND PART

//1.for first part i.e.t=0 to t=4 seconds
// time interval for the first part is (4-0)=4
t1 = 4//in seconds
//initial position is (0,0)
x01 = 0
y01 = 0
//initial velocity in x direction for first part
```

```
v0x1 = 15//in km/s
//initial velocity in y direction for first part
v0y1 = 0//in km/s
//acceleration in x direction for the first part
ax1 = 0//in km/s^2
//acceleration in y direction for the first part
ay1 = 4.2//in km/s^2

//1.for second part i.e.t=4 to t=7 seconds
// time interval for the second part is (7-4)=3
t2 = 3//in seconds
//initial velocity in x direction for first part
v0x2 = 15//in km/s
//initial velocity in y direction for first part
v0y2 = 16.8//in km/s
//acceleration in x direction for the first part
ax2 = 18//in km/s^2
//acceleration in y direction for the first part
ay2 = 4.2//in km/s^2

// SOLUTION:

//1.for first part i.e.t=0 to t=4 seconds
//final velocity in x direction
vx1 = v0x1 + ax1*t1//in km/s
//final velocity in y direction
vy1 = v0y1 + ay1*t1//in km/s
//distance travelled in x direction
x1 = x01 + v0x1*t1 + (0.5*ax1*t1^2)//in km
//distance travelled in y direction
y1 = y01 + v0y1*t1 + (0.5*ay1*t1^2)//in km

//1.for second part i.e.t=4 to t=7 seconds
//now the position of ship is (x1,y1)
x02 = x1
y02 = y1
//final velocity in x direction
//applying kinematic equations
```

```
vx2 = v0x2 + ax2*t2//in km/s
//final velocity in y direction
//applying kinematic equatio
vy2 = v0y2 + ay2*t2//in km/s
//distance travelled in x direction
//applying kinematic equatio
x2 = x02 + v0x2*t2 + (0.5*ax2*t2^2)//in km
//distance travelled in y direction
//applying kinematic equatio
y2 = y02 + v0y2*t2 + (0.5*ay2*t2^2)//in km
//distance travelled by of ship
r = sqrt(x2^2 + y2^2)//in km
//velocity of ship
v = sqrt(vx2^2 + vy2^2)//in km/s
//position of ship
theta = atand(vy2/vx2)//in degrees
y2 = round(y2)
r = round(r)
printf ("\n\n Final velocity in x direction for first part vx1 = \n\n %.1f km/s",vx1);
printf ("\n\n Final velocity in y direction for first part vy1 = \n\n %.1f km/s",vy1);
printf ("\n\n Distance travelled in x direction for first part x1 = \n\n %.1f km",x1);
printf ("\n\n Distance travelled in y direction for first part y1 = \n\n %.1f km",y1);

printf ("\n\n Final velocity in x direction for second part vx2 = \n\n %.1f km/s",vx2);
printf ("\n\n Final velocity in y direction for second part vy2 = \n\n %.1f km/s",vy2);
printf ("\n\n Distance travelled in x direction for second part x2 = \n\n %3i km",x2);
printf ("\n\n Distance travelled in y direction for second part y2 = \n\n %3i km",y2);
printf ("\n\n Distance travelled by ship r = \n\n %3i km",r);
printf ("\n\n Velocity of ship v = \n\n %2i km/s",v)
```

```
;
printf ("\n\n Position of ship theta = \n\n %2i
   degrees",theta);
```

Scilab code Exa 4.2 C4P2

```



 clc
//to find direction in which crate moving

// GIVEN::
//refer to figure 4-3(a),(b,(c) from page no.68
//mass of crate
m = 62//in kg
//initial velocity of crate in x direction
v0x = 6.4//in m/s
//initial velocity of crate in y direction
v0y = 0//in m/s
//force applied in opposite direction
Fct = 81//in N
//force applied in perpendicular direction
Fcj = 105//in N
//time interval while application of force
t = 3//in seconds

// SOLUTION:

//in x direction -Fct = m*ax
//in y direction Fcj = m*ay
//acceleration in x direction
//applying newton's second laww of motion
ax = -(Fct/m)//in m/s^2
//acceleration in y direction
```

```
ay = (Fcj/m)//in m/s^2
//component of velocity of crate in x direction
//applying kinematic equatio
vx = v0x + ax*t
//component of velocity of crate in y direction
//applying kinematic equatio
vy = v0y + ay*t
//resultant velocity of crate
v = sqrt(vx^2 + vy^2)//in m/s
//direction of velocity of crate
theta = atand(vy/vx)//in degrees
theta = nearfloat("succ",64)

printf ("\n\n Acceleration in x direction ax = \n\n
    %.2f m/s^2",ax);
printf ("\n\n Acceleration in y direction ay = \n\n
    %.2f m/s^2",ay);
printf ("\n\n Component of velocity of crate in x
    direction vx = \n\n %.1f m/s",vx);
printf ("\n\n Component of velocity of crate in y
    direction vy = \n\n %.1f m/s",vy);
printf ("\n\n Resultant velocity of crate v = \n\n %
    .1f m/s",v);
printf ("\n\n Direction of velocity of crate theta =
     \n\n %2i degrees",theta);
```

Scilab code Exa 4.3 C4P3

```



 clc
//to find direction in which crate moving

// GIVEN::
```

```
//refer to figure 4-8 from page no.70
//velocity of plane
v =155//in km/h
//horizontal velocity of package
v0x = 155//in km/h
//since initial velocity of package is same that of plane but in horizontal direction

//elevation of plane directly above the target
y = -225//in meters
// y is negetive as packages are falling in downward direction
//acceleration due to gravity
g = 9.81//in m/s^2



// SOLUTION:

//time of fall
t = sqrt(-(2*y)/g)//in seconds
//horizontal distance travelled by the package in time t
//applying kinematic equations
x= ((v0x*t)/3600)*1000//in meters
//angle of sight should be
alpha = atand(x/abs(y))//in degrees
x = round(x)
t = nearfloat("succ",6.78)
printf ("\n\n Time of fall t = \n\n %.2f seconds",t);
printf ("\n\n Horizontal distance travelled by the package in time t x = \n\n %3i meters",x);
printf ("\n\n Angle of sight should be alpha = \n\n %2i degrees",alpha);
```

Scilab code Exa 4.4 C4P4

```
clc
//to find time t1 at which the ball reaches highest position of its trajectory
//maximun height at which ball can reach
//total time of flight and range of ball
//velocity of ball when it strikes the ground

// GIVEN::

//initial velocity of ball
v0 =15.5//in m/s
//angle made by the ball with horizontal
fi0 = 36//in degrees


//acceleration due to gravity
g = 9.81//in m/s^2



// SOLUTION:

//vertical component of initial velocity of ball
v0y = v0*sind(fi0)//in m/s
//vertical component of initial velocity of ball
v0x = v0*cosd(fi0)//in m/s
//velocity at the top position of trajectory
vy = 0//in m/s
// time t1 at which the ball reaches highest
```

```
position of its trajectory
//applying kinematic equatio
t1 = (v0y-vy)/g//in seconds
///maximun height at which ball can reach
//as maximum height is reached at time t = t1
//applying kinematic equatio
y_max = v0y*t1-(0.5*g*t1^2)//in meters
//total time of flight and range of ball
//for total time displacement = 0 i.e.y = 0
//applying kinematic equatio
t2 = (2*v0y)/g//in seconds
//range of the ball
//here range is the horizontal distance travelled in time t2
//applying kinematic equatio
x = v0x*t2//in m/s
///velocity of ball when it strikes the ground
 //horizontal componebt of velocty of ball when it strikes the ground
vx = v0*cosd(fi0)//in m/s
//vertical component of velocity of ball when it strikes the ground i.e. at time t2
vy = v0y-(g*t2)//in m/s
//applying kinematic equatiovy = v0y-(g*t2)//in m/s
//magnitude of velocity of ball when it strikes the ground
v = sqrt(vx^2 + vy^2)//in m/s
//direction of ball when it strikes the ground from x axis
fi = atand(vy/vx)//in degrees
fi = round(fi)
printf ("\n\n Time t1 at which the ball reaches highest position of its trajectory t1 = \n\n %.2f seconds",t1);
printf ("\n\n Maximun height at which ball can reach y_max = \n\n %.1f meters",y_max);
printf ("\n\n Total time of flight and range of ball t2 = \n\n %.2f seconda",t2);
```

```
printf ("\n\n Range of the ball x = \n\n %.1f meters
   ",x);
printf ("\n\n Horizontal componebt of velocty of
   ball when it strikes the ground vx = \n\n %.1f m/
   s",vx);
printf ("\n\n Vertical component of velocity of ball
    when it strikes the ground i.e. at time t2 vy =
   \n\n %.1f m/s",vy);
printf ("\n\n Magnitude of velocity of ball when it
   strikes the ground v = \n\n %.1f meters",v);
printf ("\n\n Direction of ball when it strikes the
   ground from x axis fi = \n\n %2i degrees",fi);
```

Scilab code Exa 4.5 C4P5

```



 clc
//to find magnitude of gravitational force exterted
   on the moon by the earth

// GIVEN::

//time required for i revolution
d = 27.3//in days
//radius of orbit
r1 = 238000 //in mi
//radius of orbit in meters
r = (238000*1609.344)//in meters
//mass of the moon
m = 7.36*10^22//in kg

// SOLUTION:

```

```
//time for one complete revolution in seconds
T = (27.3*86400)//in seconds
//speed of the moon
v = (2*3.14*r)/T//in m/s
v = nearfloat("pred",1019)
//centripital force by gravitational force
// equation of centripital force F_ME = mv^2/r
F_ME = (m*v^2)/r//in N
printf ("\n\n Time for one complete revolution in
   seconds T = \n\n %.2e seconds",T);
printf ("\n\n Speed of the moon v  = \n\n %4i m/s",v
   );
printf ("\n\n Magnitude of gravitational force
   extерted on the moon by the earth F_ME = \n\n %.2
   e N",F_ME);
```

Scilab code Exa 4.6 C4P6

```



 clc
//to find weight of the satellite at h = 210km above
    the earth's surface
//to find tangential speed of satellite required

// GIVEN::

//mass of the satellite
m =1250//in kg
//altitude at which satellite is required to be
   placed
h = 210//in km
//radius of the earth
R = 6370//in km
```

```
//acceleration due to gravity
g = 9.2//in m/s^2

// SOLUTION:

//weight of the satellite at the altitude h = 210km
    above earth's surface
w = m*g//in N
//to find tangential speed of satellite required
//force of gravity is weight of the satellite i.e.
    F_SE = w
//radius of orbit of satellite
r = R + h//in km
v =sqrt(w*(r*1000)/m)//in m/s //taking radius in
    meters
v1 = v*(3600/1609.344)//in mi/h
v1 = nearfloat("pred",17401)
printf ("\n\n Weight of the satellite at the
    altitude h = 210km above earths surface w = \n\n
    %.2e N",w);
printf ("\n\n Tangential speed of satellite required
     v = \n\n %4i m/s",v);
printf ("\n\n Tangential speed of satellite required
     v = \n\n %5i mi/h",v1);
```

Scilab code Exa 4.7 C4P7

```



 clc
//to find velocity of plane with respect to ground
//to find compass reading if pilot wishes to fly due
     east

```

```
// GIVEN::
//refer to figure 4-18(a),(b) from page no.77
//speed of air on the indicator
V_PA =215//in km/h
//velocity of wind blowing due north
V_AG = 65//in km/h


// SOLUTION:


//magnitude of velocity of plane with respect to ground
V_PG1 = sqrt(V_PA^2 + V_AG^2)//in km/h
//direction of plane
//angle made by the plane with east direction
alpha = atand(V_AG/V_PA)//in degrees

//magnitude of velocity of plane if pilot wishes to fly due east
//now velocity of plane with respect to groung points east
V_PG2 = sqrt(V_PA^2 - V_AG^2)//in km/h
//direction of plane
//angle made by the plane with east direction
bita = asind(V_AG/V_PA)//in degrees
V_PG1 = round(V_PG1)
V_PG2 = round(V_PG2)
printf ("\n\n Magnitude of velocity of plane with respect to ground V_PG1 = \n\n %3i km/h",V_PG1);
printf ("\n\n Angle made by the plane with east direction alpha = \n\n %.1f degrees",alpha);
printf ("\n\n Magnitude of velocity of plane if pilot wishes to fly due east V_PG2 = \n\n %3i km/h",V_PG2);
printf ("\n\n Angle made by the plane with east direction bita = \n\n %.1f degrees",bita);
```

Chapter 5

APPLICATIONS OF NEWTONS LAWS

Scilab code Exa 5.1 C5P1

```

clear
 clc
//to find tension in three strings , TA,TB and TC in
   strings A,B and C respectively.

// GIVEN::
//refer to figure 5-4(a) on page no. 91
//mass of block
m = 15//in kg
//acceleration due to gravity
g = 9.81//in m/s^2

// SOLUTION:

//considering free body diagram 5-4(b) let TA,TB,TC
   are tensions in string A,B and C respectively.
//applying newton's second law to the knot i.e. SUM(
   forces) = mass*acceleration
```

```
//resolving forces first in y direction refer fig.
   5-4(d)
//resolving forces first in x and y direction refer
   fig. 5-4(c)
//solving equations by generating matrix

A = [-cosd(30) cosd(45) 0 ; sind(30) sind(45) -1 ; 0
    0 1]
B = [0;0;(m*g)]


C = A\B
//tension in sting A
TA = C(1);//in N
//tension in sting B
TB = C(2);//in N
//tension in sting C
TC = C(3);//in N
TA = round(C(1))
TB = round(C(2))
TC = round(C(3))

printf ("\n\n Tension in string A is TA = \n\n %3i N
   ",TA);
printf ("\n\n Tension in string B is TB = \n\n %3i N
   ",TB);
printf ("\n\n Tension in string C is TC = \n\n %3i N
   ",TC);
```

Scilab code Exa 5.2 C5P2

```

clear
 clc
//to find tension in the string (1)when elevator
```

```
descending with constant velocity and (2) ascending with the acceleration of 3.2 m/s^2

// GIVEN::
//refer to figure 5-2(a) on page no. 91
//mass of block
m = 2.4//in kg
//acceleration due to gravity
g = 9.81//in m/s^2
//acceleration of elevator in y direction while descending
ay1 = 0//in m/s^2  since elevator is moving with constant velocity
//acceleration of elevator in y direction while ascending
ay2 = 3.2//in m/s^2


// SOLUTION:

//when elevator is descending
//considering free body diagram 5-4b from page no.91
//resolving forces first in y direction
//applying newton's second law i.e. SUM(forces) = mass*acceleration
T1 = (m*(g+ay1))//in N

//when elevator is descending
//considering free body diagram 5-4b from page no.91
//resolving forces first in y direction
//applying newton's second law i.e. SUM(forces) = mass*acceleration
T2 = m*(g+ay2)//in N
T1 = round(T1)
printf ("\n\n Tension in the string when elevator descending with constant velocity T1 = \n\n %2i N",T1);
printf ("\n\n Tension in the string when elevator
```

```
ascending with acceleration of 3.2m/s^2 T2 = \n\n
 %2i N",T2);
```

Scilab code Exa 5.3 C5P3

```

clear
 clc
//to analyse the motion if (1)cord is horizontal and
    (2)the cord is making an angle of 15 degree with
    the  horizontal

// GIVEN::
//refer to figure 5-7(a) on page no. 92
//mass of sled
m = 7.5//in kg
//force by which sled is pulled
P = 21.0//in N
//angle made by sled with horizontal
theta = 15//in degrees
//acceleration due to gravity
g = 9.81//in m/s^2

// SOLUTION:

//when cord is horizontal

//considering free body diagram 5-7b from page no
  .92.
//euating forces in x direction
//applying newton's secong law of motion
//horizontal acceleration
ax = P/m//in m/s^2
////equating forces in y direction
//applying newton's secong law of motion
```

```
//force exerted bu surface
N = round(m*g)//in N

//when cord is  making an angle of 15 degree with
   the horizontal

//considering free body diagram 5-7c from page no
   .93.
//euating forces in x direction and applying newton'
   s secong law of motion
//acceleration
a_x = P*cosd(theta)/m//in m/s^2
////euating forces in y direction
//applying newton's secong law of motion
//normal force exerted bu surface
N_2 = ceil((m*g)-(P*sind(theta)))//in N
N = round(N)
N_2 = ceil(N_2)

printf ("\n\n Normal force exerted bu surface when
   cord is horizontal N1 = \n\n %2i N",N);
printf ("\n\n Acceleration in x direction when cord
   is horizontal ax1 = \n\n %.2f m/s^2",ax);
printf ("\n\n Normal force exerted bu surface when
   cord is making an angle of 15 degree with the
   horizontal N2 = \n\n %2i N",N_2);
printf ("\n\n Acceleration in x direction when cord
   is making an angle of 15 degree with the
   horizontal ax2 = \n\n %.2f m/s^2",a_x);
```

Scilab code Exa 5.4 C5P4

```

clear
 clc
```

```
//to find tension in the string and normal force
    exerted on the block by the plane
//to analyse the motion when the string is cut

// GIVEN::
//refer to figure 5-7(a) on page no. 92
//mass of block
m = 18//in kg
//angle of inclination of plane
theta = 27//in degrees
//acceleration due to gravity
g = 9.81//in m/s^2


// SOLUTION:

//refer to the figure 5-8a from page no. 93
//considering free body diagram 5-8b from page no
    .93.

//whenthe block is stationary on the plane
//equating forces in x direction
//applying newton's second law of motion
//tension in the string
T = m*g*sind(theta)//in N
//equating forces in y direction
//applying newton's second law of motion
//normal reaction by the surface
N = m*g*cosd(theta)//in N

//when the string is cut
//equating forces in x direction
//applying newton's second law of motion
//acceleration of block in x direction ax
ax = -(g*sind(theta))//in m/s^2
//-ve sign indicates acceleration acting in -ve x
    direction i.e. downwards
printf ("\n\n Tension in the string T = \n\n %2i N",
```

```
    T);
printf ("\n\n Normal force exerted on the block by
    the plane N = \n\n %3i N",N);
printf ("\n\n Acceleration of block in x direction
    when the string is cut ax  = \n\n %.2f m/s^2",ax)
    ;
```

Scilab code Exa 5.7 C5P7

```


 clc
//to find tension in the string
//to find acceleration of blocks

// GIVEN::
//refer to figure 5-11(a) on page no. 95
//mass of first block
m1 = 9.5//in kg
//angle of inclination of plane
theta = 34//in degrees
//mass of second block
m2 = 2.6//in kg
//acceleration due to gravity
g = 9.81//in m/s^2


// SOLUTION:

//refer to the free body diagrams 5-11b and 5-11c
    from page no. 95


//for mass m1
//assuming m1 moves in positive x direction
```

```
//equating forces in x direction and applying newton
    's second law of motion
//equating forces in y direction and applying newton
    's second law of motion

//for mass m2
//equating forces in y direction and applying newton
    's second law of motion
//solving above equations simultaneously using
    matrix form
//acceleration of blocks
a = (m2-(m1*sind(theta)))*g/(m1 + m2)//in m/s^2
//if ans. for a is -ve then our assumption is wrong
    i.e. m1 is moving in -ve x direction but
    magnitude of ans is correct
//tension in the string
T = ((m1*m2*g)*(1 + sind(theta)))/(m1 + m2)//in N

printf ("\n\n Acceleration of blocks a = \n\n %.1f m
    /s^2",a);
printf ("\n\n Tension in the string T = \n\n %2i N",
    T);
```

Scilab code Exa 5.9 C5P9

```

clear
 clc
//to find shortest distance in which automobile can
    stop


// GIVEN::
//refer to figure 5-16 on page no. 99
//initial velocity of automobile in mi/h
```

```
v01 = 60//in mi/h
//coefficient of static friction
mew_s = 0.60
//acceleration of gravity
g = 9.81//in m/s^2

// SOLUTION:
//N is normal reaction force by surface.
//refer to the free body diagrams 5-16b from page no
   . 95

//initial velocity of automobile in m/s
v0 = v01*(1609/3600)//in m/s
//applying newton's law in x and y direction
//applying kinematic equation of motion
//shortest distance in which automobile can stop
d = ((v0^2)/(2*mew_s*g))//in meters
d = ceil(d)
printf ("\n\n Shortest distance in which automobile
   can stop d  = \n\n %2i m",d);
```

Scilab code Exa 5.10 C5P10

```

clear
 clc
//to find tension in the string and normal force
   exerted on the block by the plane
//to analyse the motion when the string is cut

// GIVEN::
//refer to problem 5.7 from page no. 95
//mass of first block
m1 = 9.5//in kg
//angle of inclination of plane
```

```
theta = 34//in degrees
//mass of second block
m2 = 2.6//in kg
//acceleration due to gravity
g = 9.81//in m/s^2
//coefficient of static friction
mew_s = 0.24
//coefficient of kinetic friction
mew_k = 0.15



// SOLUTION:
//T is tension in the spring and N is normal reaction force by surface.
//refer to the free body diagrams 5-17a from page no. 99

//for mass m1 and m2
//assuming m1 moves in positive x direction
//equating forces in x direction and applying newton's second law of motion
//equating forces in y direction and applying newton's second law of motion
//acceleration of blocks
a = (m2-(m1*(sind(theta)- (mew_k*cosd(theta)))))*g/(m1 + m2)//in m/s^2
//if ans. for a is -ve then our assumption is wrong i.e. m1 is moving in -ve x direction but magnitude of ans is correct
//tension in the string
T = (((m1*m2*g)*(1 + sind(theta) - (mew_k*cosd(theta))))/(m1 + m2))//in N
T = round(T)

printf ("\n\n Acceleration of blocks a = \n\n %.1f m/s^2",a);
printf ("\n\n Tension in the string T = \n\n %2i N",
```

```
T);
```

Scilab code Exa 5.11 C5P11

```

clear
 clc
//to find time required by the car to come to rest.
//to find the distance travelled by car before
   stopping.


// GIVEN::

//mass of car
m = 1260//in kg
//velocity of car
v0x = 29.2//in m/s
//rate at which breaking force increases with time
c = 3360//in N/s

// SOLUTION:
//assuming car's velocity is in +ve x direction
//applying newton's second law of motion
//applying kinematic equation of motion to derive
   velocity relation and distance travelled relation
//time required by the car to come to rest
t1 = sqrt((2*v0x*m)/c)//in seconds
//distance travelled by car before stopping
//here we are taking time is t1 and x0 is 0
x(t1) = 0 + (v0x*t1)-((c*(t1^3))/(6*m))//in meters
printf ("\n\n Time required by the car to come to
   rest t1 = \n\n %.2f seconds",t1);
printf ("\n\n Distance travelled by car before
   stopping x(t1) = \n\n %.1f m",x(t1));
```

Chapter 6

MOMENTUM

Scilab code Exa 6.1 C6P1

```

clear
 clc
//to find impluse of the force exerted on the ball.
//to find average force assuming collision lasts for
    1.5ms
//to find the change in momentum of the bat

// GIVEN::
//refer to figure 6-8(a) on page no. 123
//mass of baseball
m = 0.14//in kg
//refer to figure 6.1
//horizontal speed of the ball
vi = 42//in m/s
//speed at which ball leaes i.e. final speed of the
   ball
vf = 50//in m/s
//angle at which  ball leaves
fi = 35//in degrees
//time for which collision lasts
```

```
delta_t = 0.0015//in seconds

// SOLUTION:

//refer to figure 6-8(a) on page no. 123
//component of final momentum in x direction
pfx = m*vf*cosd(fi)//in kgm/s
//component of final momentum in y direction
pfy = m*vf*sind(fi)//in kgm/s
//since initial momentum has only x componen
piy = 0//in kgm/s
//component of intial momentum in x direction
//considering our coordinate system as shown 6-8(a)
pix = m*(-vi)//in kgm/s
//using impluse momentum relation
//component of impluse in x direction
Jx = pfx-pix//in kgm/s
//component of impluse in y direction
Jy = pfy-piy//in kgm/s
//final magnitude of impluse
J = sqrt(Jx^2 + Jy^2)//in kgm/s
//direction in which impluse acts
theta = atand(Jy/Jx)//in degrees
//average force
//using impluse force relationship
Fav = J/delta_t//in N
Fav = nearfloat("succ",8200)

//applying newton's third law of motion
//for bat delta_px will be equal and opposite to
    that of ball
//component change in momemtum in x direction
delta_px = -(pfx - pix)//in kgm/s
//component change in momemtum in y direction
delta_py = -(pfy - piy)//in kgm/s

printf ("\n\n Component of final momentum in x
    direction pfx  = \n\n %.1f kgm/s",pfx);
```

```
printf ("\n\n Component of final momentum in y direction pfy  = \n\n %.1f kgm/s",pfy);
printf ("\n\n Component of intial momentum in x direction pix = \n\n %.1f kgm/s",pix);
printf ("\n\n Component of impluse in x direction Jx = \n\n %.1f kgm/s",Jx);
printf ("\n\n Component of impluse in y direction Jy = \n\n %.1f kgm/s",Jy);
printf ("\n\n Final magnitude of impluse J = \n\n %.1f kgm/s",J);
printf ("\n\n Direction in which impluse acts theta = \n\n %2i degrees",theta);
printf ("\n\n Average force Fav = \n\n %4i N",Fav);
printf ("\n\n Component change in momemtum in x direction delta_px = \n\n %.1f kgm/s",delta_px);
printf ("\n\n Component change in momemtum in y direction delta_py = \n\n %.1f kgm/s",delta_py);
```

Scilab code Exa 6.2 C6P2

```

clear
 clc
//to find velocity of carts after collision

// GIVEN::
//we consider +ve x direction as direction of motion of first cart
//refer to figure 6-9 on page no. 123
//mass of first cart
m1 = 0.24//in kg
//initial velocity of first cart
v1i = 0.17//in m/s
//initial velocity of second cart
//as 2nd cart is initially at rest
```

```
v2i = 0//in m/s
//mass of second cart
m2 = 0.68//in kg



// SOLUTION:

//refer to figure 6-9 on page no. 123
//using impluse force relationship
//magnitude of impluse i.e. area under graph 6-9 on
   page 123
J = 0.5*(0.014-0.003)*10//in kgm/s

//assuming direction of motion of first cart is in +
   ve x direction
//change in momentum in x direction for first cart
delta_p1x = -(J)//in kgm/s
//initial momentum of first cart in x direction
p1ix = m1*v1i//in kgm/s
//final momentum for first cart
p1fx = p1ix + delta_p1x//in kgm/s
//final velocity of first cart in x direction
v1fx = p1fx/m1//in m/s
v1fx = nearfloat("pred",-0.058)


//as direction of motion of first cart is in +ve x
   direction for second cart it will be in -ve x
   direction
//using newton's third law of motion
//change in momentum in x direction for second cart
delta_p2x = (J)//in kgm/s
//initial momentum of second cart in x direction
p2ix = m2*v2i//in kgm/s
//final momentum for second cart
p2fx = p2ix + delta_p2x//in kgm/s
//final velocity of second cart in x direction
```

```
v2fx = p2fx/m2//in m/s
printf ("\n\n Magnitude of impluse J = \n\n %.3f Kg.m/s",J);
printf ("\n\n Change in momentum in x direction for first cart delta_p1x  = \n\n %.3f Kg.m/s",delta_p1x);
printf ("\n\n Final momentum for first cart p1fx = \n\n %.3f Kg.m/s",p1fx);
printf ("\n\n Final velocity of first cart in x direction v1fx = \n\n %.3f m/s",v1fx);
printf ("\n\n Final momentum for second cart p2fx = \n\n %.3f Kg.m/s",p2fx);
printf ("\n\n Final velocity of second cart in x direction v2fx = \n\n %.3f m/s",v2fx);
```

Scilab code Exa 6.3 C6P3

```

clear
 clc
//to find direction in which Fred is skating after breaking in contact

// GIVEN::
//we consider +ve x direction for initial motion
//refer to figure 6-11 on page no. 125
//mass of Fred
mF = 75//in kg
//mass of Ginger cart
mG = 55//in kg
//common velocity of Fred ang Ginger
vG = 3.2//in m/s
vF = 3.2//in m/s
//after breaking contact angle of Ginger skating
theta1 = 32//in degrees
```

```
//  SOLUTION:

//refer to figure 6-11(a) on page no. 125
//using consevation of momentum
//x component of Ginger orignal momentum
PGx = mG*vG //in kgm/s
//x component of Fred orignal momentum
PFx = mF*vF //in kgm/s
//after they push off y component of Ginger momentum
PGy = PGx*tand(theta1)//in kgm/s
//after they push off y component of Fred momentum
   will be opposite that of Ginger
//using consevation of momentum
PFy = -(PGy)//in kgm/s
tan_theta = (PFy/PFx)
//direction in which Fred is skating after breaking
   in contact
theta = atand(PFy/PFx)//in degrees
PGy = round(PGy)
theta = round(theta)

printf ("\n\n X component of Ginger orignal momentum
    PGx = \n\n %3i Kg.m/s",PGx);
printf ("\n\n X component of Fred orignal momentum
   PFx = \n\n %3i Kg.m/s",PFx);
printf ("\n\n After they push off y component of
   Ginger momentum PGy = \n\n %3i Kg.m/s",PGy);
printf ("\n\n Value of tan_theta = \n\n %.3f degrees
   ",tan_theta);
printf ("\n\n Direction in which Fred is skating
   after breaking in contact theta = \n\n %2i
   degrees",theta);
```

Scilab code Exa 6.4 C6P4

```
clear
 clc
//to find final velocity of man when seated in
   rowboat

// GIVEN::
//we consider +ve x direction as man's original
   velocity
//refer to figure 6-12 on page no. 126
//mass of man
mm = 65//in kg
//speed of man initially in x direction
vmx = 4.9//in m/s
//mass of rowboat
mb = 88//in kg
//speed of rowboat in x direction
vbx = 1.2//in m/s

// SOLUTION:

//refer to figure 6-12(a) and 6-12(b) on page no.
   126


//before man jumps
//momentum of man in x direction
pmx = mm*vmx//in kgm/s
//momentum of boat in x direction
pbx = mb*vbx//in kgm/s
//total initial momentum in x direction
pix = pmx + pbx//in kgm/s

//after man jumps
//combined final momentum of man and boat in x
   direction
//applying conservation of momentum for boat and man
//final velocity of man when seated in rowboat in x
```

```
direction
vfx = (pix/(mm + mb))//in m/s
printf ("\n\n Total initial momentum in x direction pix = \n\n %.3i Kg.m/s",pix);
printf ("\n\n Final velocity of man when seated in rowboat in x direction vfx = \n\n %.1f m/s",vfx);
```

Scilab code Exa 6.5 C6P5

```

clear
 clc
//to find velocity of second glider after collision

// GIVEN::

//we consider +ve x direction as initial motion of first glider
//mass of first glider
m1 = 1.25//in kg
//initial velocity of first glider in +ve x direction
v1ix = 3.62//in m/s
//mass of second glider
m2 = 2.30//in kg
//final velocity of first glider in +ve x direction
// - sign since after collision first glider is moving in -ve x direction
v1fx = -1.07//in m/s
//initial velocity of second glider in +ve x direction
//since 2nd glider is initially at rest
v2ix = 0//in m/s

// SOLUTION:
```

```
//applying conservation of momentum
//final velocitiy of second glider in +ve x
    direction
v2fx = (m1/m2)*(v1ix-v1fx)//in m/s
//change in momentums for glider having mass m1
delta_p1x = m1*(v1fx-v1ix)//in Kg.m/s
//change in momentums for glider having mass m2
delta_p2x = m2*(v2fx-v2ix)//in Kg.m/s

printf ("\n\n Velocitiy of second glider in +ve x
    direction after collision v2fx = \n\n %.2f m/s",
    v2fx);
printf ("\n\n Change in momentums for glider having
    mass m1 delta_p1x = \n\n %.2f Kg.m/s",delta_p1x);
printf ("\n\n Change in momentums for glider having
    mass m2 delta_p2x = \n\n %.2f Kg.m/s",delta_p2x);
```

Scilab code Exa 6.6 C6P6

```

clear
 clc
//to find final velocity of combination of 1st and 2
    nd glider

// GIVEN::
//refer to problem 6-5 from page no. 127

//we consider +ve x direction as initial motion of
    first glider
//mass of first glider
m1 = 1.25//in kg
//initial velocity of first glider in +ve x
    direction
```

```
v1ix = 3.62//in m/s
//mass of second glider
m2 = 2.30//in kg
//initial velocity of second glider in +ve x direction
//since 2nd glider is initially at rest
v2ix = 0//in m/s


// SOLUTION:

//applying conservation of momentum
//final velocitiy of second glider in +ve x direction
vfx = (m1*v1ix)/(m1 + m2)//in m/s
//change in momentums for glider having mass m1
delta_p1x = m1*(vfx-v1ix)//in Kg.m/s
//change in momentums for glider having mass m2
delta_p2x = m2*(vfx-v2ix)//in Kg.m/s

printf ("\n\n Final velocity of combination of 1st and 2nd glider vfx = \n\n %.2f m/s",vfx);
printf ("\n\n Change in momentums for glider having mass m1 delta_p1x = \n\n %.2f Kg.m/s",delta_p1x);
printf ("\n\n Change in momentums for glider having mass m2 delta_p2x = \n\n %.2f Kg.m/s",delta_p2x);
```

Scilab code Exa 6.7 C6P7

```



 clc
//to find final speed of larger craft

```

```
// GIVEN::
//refer to diagram 6-14 from page no. 127

//we consider +ve x direction as original motion of spaceship(and also that of final velocity of smaller craft)
//total mass of spaceship
//M = m//in kg
//let us consider m = 1
M = 1//in kg
//mass of smaller crafy
//m1 = m/4//in kg
m1 = 1/4//in kg
//mass of larger craft
//m2 =3* m/4//in kg
m2 =3* 1/4//in kg
//initial velocity of spaceship in +ve x direction
vix = 8.45//in km/s
//final speed of smaller craft in +ve x direction
v1fx = 11.63//in km/s


// SOLUTION:

//applying conservation of momentum
//final velocity of larger craft in +ve x direction
v2fx = (((m1 + m2)*vix)-(m1*v1fx))/m2//in m/s

printf ("\n\n Final velocity of larger craft in +ve x direction v2fx = \n\n %.2f km/s",v2fx);
```

Scilab code Exa 6.8 C6P8

```


```

```
clc
//to find final speed and direction of second puck
    after collision

// GIVEN::
//refer to diagram 6-15 from page no. 128

//we consider +ve x direction as initial motion of
    first puck
//mass of first puck
//assume mass of first puck be 1kg
m1 = 1//in kg
//mass of second puck
//mass of second puck is 1.5 times mass of first
    puck
m2 = 1.5//in kg
//initial velocity of first puck in +ve x direction
v1ix = 2.48//in m/s
//initial velocity of second puck in +ve x direction
v2ix = 1.86//in m/s
//initial direction of second puck away from the
    direction of first puck
theta1 = 40//in degrees
//final velocity of first puck after collision
v1fx = 1.59//in m/s
//final direction of first puck after collision
theta2 = 50//in degrees

// SOLUTION:

//applying law of conservation of momentum in x and
    y direction
//solving equation
//final direction of second puck after collision
theta = atand(0.38/2.40)//in degrees
//final speed of second puck after collision
v2f = 2.40/cosd(theta)//in m/s
```

```
printf ("\n\n Final speed  of second puck after collision v2f = \n\n %.2f m/s",v2f);
printf ("\n\n Final direction of second puck after collision theta = \n\n %.1f degrees",theta);
```

Scilab code Exa 6.9 C6P9

```



 clc
//to find velocity of alpha partical after collision
//to find which type of collision is this listed in fig. 6-17

// GIVEN::
//refer to diagram 6-17 from page no. 130

//we consider +ve x direction as initial velocity of  alpha partical
//mass of alpha partical m1 = 4.0u
//assume u = 1
ma = 4.0
//mass of oxygen nucleus m2 = 16.0u
//assume u = 1
mo = 16.0
//initial velocity of alpha partical in +ve x direction
vaix = 1.52*10^7//in m/s
//initial velocity of oxygen nucleus in +ve x direction
//as oxygen nucleus is initially at rest
voix = 0//in m/s
//final velocity of oxygen nucleus after collision
vofx = 6.08*10^6//in
```

```
// SOLUTION:

//applying law of conservation of momentum in x
    direction
//final velocity of alpha partical after collision
    in x direction
vafx = ((ma*vaix)-(mo*vofx))/ma//in m/s
//applying law of conservation of momentum in x
    direction
//we can find out collision is elstic collision as
    alpha partical only reverses the direction of
    momentum after collision
//relative velocity
vx = (ma*vaix+mo*voix)/(ma+mo)//in m/s

printf ("\n\n Final velocity of alpha partical after
     collision in x direction vafx = \n\n %.2e m/s",
    vafx);
printf ("\n\n Relative velocity vx  = \n\n %.3e m/s"
    ,vx);
printf ("\n\n Collision is elstic collision");
```

Chapter 7

SYSTEMS OF PARTICLES

Scilab code Exa 7.2 C7P2

```
clear
clc
//to find center of mass of system
//to find acceleration of center of mass

// GIVEN::

//refer to figure 7-10(a) fron page no. 144
//consider +ve x direction as our reference axis
//mass of first partical
m1 = 4.1//in kg
//mass of second partical
m2 = 8.2//in kg
//mass of third partical
m3 = 4.1//in kg
//from figure 7-20(a)
//x coordinate of first partical
x1 = -2//in cm
//y coordinate of first partical
y1 = 3//in cm
//x coordinate of second partical
```

```
x2 = 4//in cm
//y coordinate of second partical
y2 = 2//in cm
//x coordinate of third partical
x3 = 1//in cm
//y coordinate of third partical
y3 = -2//in cm
//magnitude of first external force
F1 = -6//in N //since acting in -ve x direction
//magnitude of second external force
F2 = 12//in N
//magnitude of third external force
F3 = 14//in N

// SOLUTION:
//refer to figure 7-10(a) and 7-10(b) fron page no.
    144
//assuming all external forces are applied at center
     of mass
//total mass of system
M = m1 + m2 + m3//in kg
//applying center of mass formula
//x coordinate of center of mass
x_cm = (1/M)*(m1*x1 + m2*x2 + m3*x3)//in cm
//y coordinate of center of mass
y_cm = (1/M)*(m1*y1 + m2*y2 + m3*y3)//in cm

//refer to figure 7-10(b)
//component of force F1 in x direction
F1x = F1//in N
//component of force F2 in x direction
F2x = F2*cosd(45)//in N
//component of force F3 in x direction
F3x = F3//in N
//component of force F1 in y direction
F1y = 0//in N
//component of force F2 in y direction
F2y = F2*sind(45)//in N
```

```
//component of force F3 in y direction
F3y = 0//in N
//x component of net external force acting on the center of mass
SUM_fextx = F1x + F2x + F3x//in N
//y component of net external force acting on the center of mass
SUM_fexty = F1y + F2y + F3y//in N
//magnitude of net external force acting on the center of mass
SUM_Fext = sqrt(SUM_fextx^2 + SUM_fexty^2)//in N
//direction in which net force acts
fi = atand(SUM_fexty/SUM_fextx)//in degrees with x axis
//acceleration of center of mass
a_cm = SUM_Fext/(M)//in m/s^2
SUM_Fext = nearfloat("succ",18.6)

printf ("\n\n x coordinate of center of mass x_cm = \n\n %.1f cm",x_cm);
printf ("\n\n y coordinate of center of mass y_cm = \n\n %.1f cm",y_cm);
printf ("\n\n Magnitude of net external force acting on the center of mass in x direction SUM_fextx = \n\n %.1f N",SUM_fextx);
printf ("\n\n Magnitude of net external force acting on the center of mass in y direction SUM_fexty = \n\n %.1f N",SUM_fexty);
printf ("\n\n Magnitude of net external force acting on the center of mass SUM_Fext = \n\n %.1f N",SUM_Fext);
printf ("\n\n Direction in which net force acts with x axis fi = \n\n %2i degrees",fi);
printf ("\n\n Acceleration of center of mass a_cm = \n\n %.1f m/s^2",a_cm);
```

Scilab code Exa 7.3 C7P3

```
clc
//to find location of second fragment

// GIVEN::

//refer to figure 7-11 fron page no. 145
//consider +ve x direction as our reference axis
//mass of projectile
M = 9.6//in kg
//initial velocity of projectile
v0 = 12.4//in m/s
//angle of projectile above horizontal
fi0 = 54//in degrees
//mass of first piece after explosion
m1 = 6.5//in kg
//time after which first piece id observed
t = 1.42//in seconds
//vertical distance at which first piece is observed
y1 = 5.9//in meters
//horizontal distance at which first piece is observed
x1 = 13.6//in meters
//acceleration due to gravity
g = 9.80//in m/s^2

// SOLUTION:

//refer to figure 7-11 from page no. 145
//mass of second piece
```

```
//by mass conservation
m2 = M-m1//in kg
//velocity of projectile in +ve x direction
v0x = v0*cosd(fi0)//in m/s
//velocity of projectile in +ve y direction
v0y = v0*sind(fi0)//in m/s
//using kinematic equation of motion
//x coordinate of position of original projectile
x = v0x*t//in m
//y coordinate of position of original projectile
y = (v0y*t)-(0.5*g*t^2)//in m
//applying center of mass formula
//x coordinate of posion of second piece
x2 = (M*x - m1*x1)/m2//in meters
//y coordinate of posion of second piece
y2 = (M*y - m1*y1)/m2//in meters
x = nearfloat("succ",10.4)
y = nearfloat("pred",4.3)
x2 = nearfloat("succ",3.7)
y2 = nearfloat("pred",0.9)

printf ("\n\n x coordinate of position of original projectile x = \n\n %.1f m",x);
printf ("\n\n y coordinate of position of original projectile y = \n\n %.1f m",y);
printf ("\n\n x coordinate of posion of second piece x2 = \n\n %.1f m",x2);
printf ("\n\n y coordinate of posion of second piece y2 = \n\n %.1f m",y2);
```

Scilab code Exa 7.7 C7P7

```

clear
clc
```

```
//to find speed of block after it has absorbed eight bullets

// GIVEN::
//refer to figure 7-16 fron page no. 148
//mass of bullet
m1 = 3.8//n gram
m = 3.8*10^-3//in kg
//speed of bullet
v = 1100//in m/s
//mass of wooden block
M = 12//in kg
//number of bulletes
N = 8

// SOLUTION:
//refer to figure 7-16 from page no. 148
//consider +ve x direction to the right as seen in fig. 7-16
//applying momentum conservation before bullets are stillin flight and after bullets are in the block
//speed of block after it has absorbed eight bullets
V = ((N*m)/(M + N*m))*v//in m/s
printf ("\n\n Speed of block after it has absorbed eight bullets V = \n\n %.1f m/s",V);
```

Scilab code Exa 7.8 C7P8

```
clear
clc
//to find velocity of the recoiling cannon with respect to the earth
//to find initial velocity vE of the ball with respect to the earth

```

```
// GIVEN::

//refer to figure 7-17 fron page no. 149
//mass of cannon
M = 1300//in kg
//mass of ball fired
m = 72//in kg
//speed of ball in horizontal x direction
vx = 55//in m/s

// SOLUTION:

//refer to figure 7-17 from page no. 149
//considering cannon and ball is our system and consider +ve x as right direction
//finding momentum of our system with respect to the earth
//applying conservation of momentum
Vx = -(m*vx)/(M + m)//in m/s //-ve signs as cannon recoils in left direction
//initial velocity vE of the ball with respect to the earth
vEx = vx + Vx//in m/s

printf ("\n\n Velocity of the recoiling cannon with respect to the earth Vx = \n\n %.1f m/s",Vx);
printf ("\n\n Initial velocity vE of the ball with respect to the earth vEx = \n\n %2i m/s",vEx);
```

Scilab code Exa 7.9 C7P9

```
clear
clc
//to find thrust produced by the rocket
//to find velocity of the spaceship after the
```

```
rockets have fired

// GIVEN::

//total mass of spaceship
M = 13600//in kg
//initial speed of spaceship
vix = 960//in m/s
//rate at which rocket ejects gas
dM_by_dt = 146//in kg/s
//speed at which rocket ejects gas
vrel = 1520//in m/s
//mass of gas burned and ejected from spaceship
m = 9100//in kg

// SOLUTION:

//consider +ve x direction in the direction of spaceship's initial velocity
//thrust produced by the rocket
F = vrel*dM_by_dt//in N
//initial mass of gas
Mi = 13600//inkg
//final mass of gass
Mf = Mi-m//in kg
//rewriting equation of velocity and integrating velocity equation from initial to final conditions
//velocity of the spaceship after the rockets have fired
vfx = vix + (-vrel*(log(Mf/Mi)))//in m/s
vfx = nearfloat("pred",2641)
printf ("\n\n Thrust produced by the rocket F = \n\n %.2e N",F);
printf ("\n\n Velocity of the spaceship after the rockets have fired vfx = \n\n %4i m/s",vfx);
```

Scilab code Exa 7.10 C7P10

```
clear
clc
//to find force to be applied on conveyor belt to
   keep it moving with constant speed

// GIVEN::

//refer to figure 7-20 fron page no. 151
//rate at which sand is being dropped
dM_by_dt = 0.134//in kg/s
//speed at which sand is being dropped
vx = 0.96//in m/s

// SOLUTION:

//refer to figure 7-20 from page no. 151
//consider +ve x as direction of motion of belt and
   applying equation for systems of variable mass
//force to be applied on conveyor belt to keep it
   moving with constant speed
sum_F_extx = (vx*dM_by_dt)//in N
printf ("\n\n Force to be applied on conveyor belt
   to keep it moving with constant speed sum_F_extx
   = \n\n %.3f N",sum_F_extx);
```

Chapter 8

ROTATIONAL KINEMATICS

Scilab code Exa 8.1 C8P1

```
clear
clc
//to find average angular velocity of fan blade
//to find average angular aceleration of fan blade

// GIVEN::

//initial angular velocity of fan blade
wi1 = 48.6//in revolution per minute
wi = wi1/60//in rev/s
//final angular velocity of fan blade
//as finally fan blade comes to rest
wf = 0//in revolution per minute
//time required for fan blade to come to rest
delta_t = 32//in seconds
//no. of revolution completed by fan blade before
   come to rest
delta_fi = 8.8


// SOLUTION:
```

```
//using kinematic equation of motion for rotational
    motion
//average angular velocity of fan blade
w_av = delta_fi/delta_t//in rev/s
//average angular aceleration of fan blade
a_av = (wf-wi)/delta_t//in rev/s^2

printf ("\n\n Average angular velocity of fan blade
    w_av = \n\n %.2f rev/s",w_av);
printf ("\n\n Average angular acceleration of fan
    blade a_av = \n\n %.3f rev/s^2",a_av);
```

Scilab code Exa 8.2 C8P2

```
clear
clc
//to find angular posion at t = 2 seconds
//to find instantaneous angular acceleration of
    reference line at t = 0.5 seconds
// GIVEN:

//refer to the figure 8-1 from page no. 159
//in angular velocity function w = A*t + B*t^2
    values of conatanta
A = 6.2//in rad/s^2
B = 8.7//in red/s^2
//for calculatiing angular position time interval
t1 = 2//in seconds
//for calculatiing angular acceleration time
    interval
t2 = 0.50//in seconds
//initial condition
//reference line initially is at fi = 0 when t = 0

// SOLUTION:
```

```
//using kinematic equation of motion for rotational motion
//angular posion at t = 2 seconds
fi = ((1/2)*A*t1^2) + ((1/3)*B*t1^3)//in rad
//angular instantaneous acceleration at t = 0.5 seconds
a = A + (2*B*t2)//in rad/s^2

printf ("\n\n Angular posion at t = 2 seconds fi = \n\n %.1f rad",fi);
printf ("\n\n Angular instantaneous acceleration at t = 0.5 seconds a = \n\n %.1f rav/s^2",a);
```

Scilab code Exa 8.3 C8P3

```
clear
clc
//to find angular displacement of grindstone 2.7 seconds later
//to find angular speed of grindstone 2.7 seconds later
// GIVEN:

//refer to the figure 8-7 from page no. 165
//constant angular acceleration of grindstone in +ve z direction
az = 3.2//in rad/s^2
//time initervalfor calculating angular acceleration and angular displacement
t = 2.7//in seconds
//initially angular displacement
fi_0 = 0//in rad
//initially angular velocity in +ve z direction
w0z = 0//in rad/s

```

```
// SOLUTION:
//consider angular velocity in +ve z direction
//using kinematic equation of motion for rotational motion
//angular displacement of grindstone 2.7 seconds later
fi = fi_0 + (w0z*t) + (0.5*az*t^2)//in rad
//angular speed of grindstone 2.7 seconds later
wz = w0z + (az*t)//in rad/s

printf ("\n\n Angular displacement of grindstone 2.7 seconds later fi = \n\n %.1f rad",fi);
printf ("\n\n Angular speed of grindstone 2.7 seconds later wz = \n\n %.1f rad/s",wz);
```

Scilab code Exa 8.4 C8P4

```
clear
clc
//to find angular acceleration of grindstone
//to find total angle turned through during slowing down of grindstone

// GIVEN:
//refer to problem 8-3 and from page no. 165
//refer to the figure 8-7 from page no. 165
//initial angular speed of grindstone
w0z = 8.6//in rad/s
//final angular speed of grindstone
//as grindstone comes to rest
wz = 0//in rad/s
//time interval in which grindstone comes to rest
t = 192//in seconds
//initial angular displacement of grindstone
fi_0 = 0//in rad
```

```
// SOLUTION:
//consider angular velocity in +ve z direction
//using kinematic equation of motion for rotational
    motion
//angular acceleration of grindstone
az = (wz-w0z)/t//in rad/s^2
//total angle turned through during slowing down of
    grindstone
fi = fi_0 + (w0z*t) + ((1/2)*az*(t^2))//in rad
fi = nearfloat("pred",823)

printf ("\n\n Angular acceleration of grindstone az
    = \n\n %.3f rad/s^2",az);
printf ("\n\n Total angle turned through during
    slowing down of grindstone fi = \n\n %.3i rad",fi
    );
```

Scilab code Exa 8.5 C8P5

```
clear
clc
//to find linesr or tangential speed of a point on a
    rim
//to find tangential acceleration of a point on a
    rim
//to find radial acceleration of a point on a rim

// GIVEN:
//refer to problem 8-3 from page no. 165
//refer to the figure 8-7 from page no. 165
//radius of grindstone for first case
r1 = 0.24//in meters
//radius of grindstone for second case
r2 = 0.12//in meters
```

```
//initial angular speed of grindstone
w = 8.6//in rad/s
//constant angular acceleration of grindstone
a = 3.2//in rad/s^2
//time interval
t = 2.7//in seconds

// SOLUTION:
//using kinematic equation of motion for rotational motion

//for r1 = 0.24m
//linesr or tangential speed of a point on a rim
vT = w*r1//in m/s
//tangential acceleration of a point on a rim
aT = a*r1//in m/s^2
//radial acceleration of a point on a rim
aR = w^2*r1//in m/s^2

//for r1 = 0.12m
//linesr or tangential speed of a point on a rim
v_T = w*r2//in m/s
//tangential acceleration of a point on a rim
a_T = a*r2//in m/s^2
//radial acceleration of a point on a rim
a_R = w^2*r2//in m/s^2
aR = round(aR)

printf ("\n\n Linesr or tangential speed of a point on a rim for r1 = 0.24m vT = \n\n %.1f m/s",vT);
printf ("\n\n Tangential acceleration of a point on a rim for r1 = 0.24m aT = \n\n %.2f m/s^2",aT);
printf ("\n\n Radial acceleration of a point on a rim for r1 = 0.24m aR = \n\n %2i m/s^2",aR);
printf ("\n\n Linesr or tangential speed of a point on a rim for r1 v_T = 0.12m v_T = \n\n %.1f m/s",v_T);
printf ("\n\n Tangential acceleration of a point on
```

```
    a rim for r1 = 0.12m a_T = \n\n %.2f m/s^2",a_T);
printf ("\n\n Radial acceleration of a point on a
    rim for r1 = 0.12m a_R = \n\n %.1f m/s^2",a_R);
```

Scilab code Exa 8.6 C8P6

```
clear
clc
//to find tangential speed of point on the equator
    of pulsar

// GIVEN:
//rotational period of pulsar
T = 0.033//in seconds
//radius of pulsar
r = 15//in km

// SOLUTION:
//using kinematic equation of motion for rotational
    motion
//angular speed
w = (2*3.14)/T//in rad/s
//tangential speed of point on the equator of pulsar
vT = w*r//in km/s

printf ("\n\n Angular speed w = \n\n %3i rad/s",w);
//answer of vT is slightly varying.But answer of
    scilab program and calculator is same
printf ("\n\n Tangential speed of point on the
    equator of pulsar vT = \n\n %4i km/s",vT);
```

Chapter 9

ROTATIONAL DYNAMICS

Scilab code Exa 9.1 C9P1

```
clear
clc
//To find magnitude of torque due to gravity about
   the pivot point o

// GIVEN::

//refer to figure 9-5 from page no. 178
//mass of body
m = 0.17//in kg
//length of rod
L = 1.25//in meters
//angle of pendulum with vertical
theta = 10//in degrees
//acceleration due to gravity
g = 9.8//in m/s^2

// SOLUTION:

//magnitude of torque
tow = L*m*g*sind(theta)//in N.m
```

```
printf ("\n\n Magnitude of torque tow = \n\n %.2f N.
    m" ,tow);
```

Scilab code Exa 9.2 C9P2

```
clear
clc
//To find rotational inertia
//to find angular acceleration

// GIVEN::

//refer to figure 9-9 from page no. 181
//mass of first partical
m1 = 2.3//in kg
//mass of second partical
m2 = 3.2//in kg
//mass of third partical
m3 = 1.5//in kg
//force applied to m2
F = 4.5//in N
//angle made by force with horizontal
theta = 30//in degrees

// SOLUTION:

//consider firstly the axis passes through m1
r1f = 0.0//in m
r2f = 3.0//in m
r3f = 4.0//in m
//rotational inertia about the axis
I1 = (m1*r1f^2)+(m2*r2f^2)+(m3*r3f^2)//in Kg.m^2

//consider secondly the axis passes through m2
```

```
r1s = 3.0//in m
r2s = 0.0//in m
r3s = 5.0//in m
//rotational inertia about the axis
I2 = (m1*r1s^2)+(m2*r2s^2)+(m3*r3s^2)//in Kg.m^2

//consider thirdly the axis passes through m3
r1t = 4.0//in m
r2t = 5.0//in m
r3t = 0.0//in m
//rotational inertia about the axis
I3 = (m1*r1t^2)+(m2*r2t^2)+(m3*r3t^2)//in Kg.m^2
I1 = round(I1)
I2 = round(I2)
I3 = round(I3)

//from figure fi
fi = asind(3/5)//in degrees
//angle between F and line connecting m3 and m2
fi1 = theta + fi//in degrees
//value of moment arm
r_perpendicular = r3s*sind(fi1)//in m
//magnitude of torque about m3
tow_z = r_perpendicular*F//in N.m
//using rotational inertia about axis through m3
//angular acceleration
az = -(tow_z)/I3//in rad/s^2

printf ("\n\n Rotational inertia about the axis when
    the axis passes through m1 is I1 = \n\n %2i Kg.m
   ^2",I1);
printf ("\n\n Rotational inertia about the axis when
    the axis passes through m2 is I2 = \n\n %2i Kg.m
   ^2",I2);
printf ("\n\n Rotational inertia about the axis when
    the axis passes through m3 is I3 = \n\n %3i Kg.m
   ^2",I3);
printf ("\n\n Magnitude of torque about m3 tow_z = \
```

```
   n\n %.1f N.m",tow_z);
printf ("\n\n Angular acceleration az = \n\n %.2f
   rad/s^2",az);
```

Scilab code Exa 9.3 C9P3

```

clear
 clc
//To find rotational inertia

// GIVEN::

//refer to figure 9-9 from page no. 181
//mass of first partical
m1 = 2.3//in kg
//mass of second partical
m2 = 3.2//in kg
//mass of third partical
m3 = 1.5//in kg

// SOLUTION:
//locating center of mass

x1 = 0//in m
x2 = 0//in m
x3 = 4.0//in m
//x coordinate of center of mass
x_cm = (m1*x1+m2*x2+m3*x3)/(m1+m2+m3)//in m

y1 = 0//in m
y2 = 3.0//in m
y3 = 0//in m
//y coordinate of center of mass
y_cm = (m1*y1+m2*y2+m3*y3)/(m1+m2+m3)//in m
```

```
//squqred distance from center of mass to each of particals
//for first partical
r1_square = x_cm^2 + y_cm^2//in m^2
//for second partical
r2_square = x_cm^2 + (y2-y_cm)^2//in m^2
//for third partical
r3_square = (x3-x_cm)^2 + y_cm^2//in m^2
//rotational inertia
I_cm = (m1*r1_square+m2*r2_square+m3*r3_square)//in Kg.m^2

r2_square = nearfloat("succ",3.40)
r3_square = nearfloat("pred",11.74)
I_cm = ceil(I_cm)

printf ("\n\n x coordinate of center of mass x_cm = \n\n %.2f m",x_cm);
printf ("\n\n y coordinate of center of mass y_cm = \n\n %.2f m",y_cm);
printf ("\n\n Squqred distance from center of mass for first partical r1_square = \n\n %.2f m^2",r1_square);
printf ("\n\n Squqred distance from center of mass for second partical r2_square = \n\n %.2f m^2",r2_square);
printf ("\n\n Squqred distance from center of mass for third partical r3_square = \n\n %2i m^2",r3_square);
printf ("\n\n Rotational inertia I_cm = \n\n %.1f Kg.m^2",I_cm);
```

Scilab code Exa 9.6 C9P6

```
clear
 clc
//to find forces that is scale reading


// GIVEN::

//refer to figure 9-22(a) from page no. 189
//mass od beam
m = 1.8//in kg
//massof block
M = 2.7//in kg
//acceleration due to gravity
g = 9.8//in m/s^2

// SOLUTION:

//refer to figure 9-22(b) from page no. 189
//consider our system as beam and block together
//equating net torque to zero
//force Fr
Fr = (g/4)*(M+2*m)//in N
//equating forces iny direction as 0 for
    equillibrium condition
//force F1
F1 = (M+m)*g - Fr//in N
F1 = round(F1)

printf ("\n\n Force Fr = \n\n %2i N",Fr);
printf ("\n\n Force F1 = \n\n %2i N",F1);
```

Scilab code Exa 9.7 C9P7

```

clear
```

```
clc
//to find forces exerted on the ladder by the ground
    and by the wall


// GIVEN::

//refer to figure 9-23(a) from page no. 189
//length of ladder
L = 12//in meters
//mass of ladder
m = 45//in kg
//distance of upper end of ladder above the ground
h = 9.3//in meters
//mass of firefighter
M = 72//in kg
//acceleration due to gravity
g = 9.8//in m/s^2

// SOLUTION:

//refer to figure 9-23(b) from page no. 189
//distance from the wall to the foot of ladder
a = sqrt(L^2 - h^2)//in meters
//considering equillibrium conditions
//finding normal reaction by ground
N = (M+m)*g//in N
//force exerted on ladder by the wall
Fw = (g*a*(M/2 + m/3))/h//in N
N = round(N)
Fw = round(Fw)
printf ("\n\n Distance from the wall to the foot of
    ladder a = \n\n %.1f m",a);
//answer is slightly different than book.But answer
    of scilab program is same as that of calculator
printf ("\n\n Forces exerted on the ladder by the
    ground N = \n\n %3i N",N);
//answer is slightly different than book.But answer
```

```
of scilab program is same as that of calculator
printf ("\n\n Forces exerted on the ladder by the
    wall Fw = \n\n %3i N",Fw);
```

Scilab code Exa 9.8 C9P8

```

clear
 clc
//to find tension in the wire
//to find force exerted by the hinge on the beam


// GIVEN::

//refer to figure 9-24(a) from page no. 190
//length of the beam
L = 3.3//in meters
//mass of beam
m = 8.5//in kg
//distance at which wire is connected
d = 2.1//in meters
//angle made by beam with horizontal
theta = 30//in degrees
//mass of body
M = 56//in kg
//acceleration due to gravity
g = 9.8//in m/s^2

// SOLUTION:

//refer to figure 9-24(b) from page no. 190
//angle alpha from geometry
alpha = atand((d-(L*sind(theta)))/(L*cosd(theta)))//
    in degrees
```

```
k = M*g+m*g;
j = m*g/2;
//applying equllibrium conditions to get 4 equations
A = [0 1 0 -1 ; 1 0 1 0 ; 1 -tand(theta) 0 0 ; 0 0 1 -tand(alpha)];
b = [0 ; k ; j ; 0];
c = A\b
Fv = c(1)
Fh = c(2)
Tv = c(3)
Th = c(4)

Fv = round(Fv)
Fh = round(Fh)
Th = round(Th)
//resultant tension in the wire
T = sqrt(Th^2 + Tv^2)//in N
//resultant force exerted by the hinge on the beam
F = sqrt(Fh^2+ Fv^2)//in N
T = round(T)
F = round(F)
//angle made by vector F with horizontal
fi = atand(Fv/Fh)//in degrees

printf ("\n\n Vertical force Fv = \n\n %3i N",Fv);
printf ("\n\n Horizontal force Fh = \n\n %3i N",Fh);
printf ("\n\n vertical tension in in wire Tv = \n\n %3i N",Tv);
printf ("\n\n Horizontal tension in in wire Th = \n\n %3i N",Th);
printf ("\n\n Resultant tension in the wire T = \n\n %3i N",T);
printf ("\n\n Resultant force exerted by the hinge on the beam F = \n\n %3i N",F);
printf ("\n\n angle made by vector F with horizontal fi = \n\n %.1f degrees",fi);
```

Scilab code Exa 9.9 C9P9

```

clear
 clc
//to find magnitude of torque
//to find resultant angular acceleration of the
    system


// GIVEN::

//refer to figure 9-25 from page no. 191
//force exerted
F = 115//in N
//distance from axis of rotation at which force is
    exerted
r = 1.50//in meters
//angle of apllication of force
theta1 = 32//in degrees
//direction of horizontal component
theta2 = 15//in degrees
//acceleration due to gravity
g = 9.8//in m/s^2
//radius od disk
R = 1.5//in meters
//thicknes of disk
d = 0.40//in cm
//mass of child
m = 25//in kg
//radius of position of child
r1 = 1.0//in meters


```

```
// SOLUTION:

//refer to figure 9-25 from page no. 191
//horizontal component of force
Fh = F*cosd(theta1)//in N
//component of force perpendicular to r
F_perpendicular = Fh*cosd(theta2)//in N
//vertical torque along the axis of rotation
tow = r*F_perpendicular//in N.m

//volume of disk
volume = %pi*(R*100)^2*d//in m^3
//consider density of steel
density = 7.9//in g/cm^3
//mass of merry-go-round
M = (volume*density)*10^-3//in kg
//rotational inertia of disk
Im = ((1/2)*M*R^2)///in kg.m^2
//rotational inertia of child
Ic = m*r1^2///in kg.m^2
//total rotational inertia
It = Im + Ic//in kg.m^2
//angular acceleration of the system
alpha_z = tow/It//in rad/s^2

printf ("\n\n Horizontal component of force Fh = \n\
    n %.1f N",Fh);
printf ("\n\n Component of force perpendicular to r
    F_perpendicular  = \n\n %.1f N",F_perpendicular);
printf ("\n\n Vertical torque along the axis of
    rotation tow = \n\n %3i N.m",tow);
printf ("\n\n Rotational inertia of disk Im = \n\n
    %3i kg.m^2",Im);
printf ("\n\n Rotational inertia of child Ic = \n\n
    %3i kg.m^2",Ic);
printf ("\n\n Total rotational inertia It = \n\n %3i
    kg.m^2",It);
printf ("\n\n Angular acceleration of the system
```

```
alpha_z = \n\n %.2f rad/s^2",alpha_z);
```

Scilab code Exa 9.10 C9P10

```

clear
 clc
//to find acceleration of the falling block
//to find tension in the chord
//to find angular acceleration of the disk


// GIVEN::

//refer to figure 9-26(a) from page no. 192
//mass of disk
M = 2.5//in kg
//radius of disk
R = 20//in cm
//mass of block
m = 1.2//in kg
//acceleration due to gravity
g = 9.8//in m/s^2

// SOLUTION:

//refer to figure 9-26(b) from page no. 192
//applying newton's second law in y direction for block
//and applying rotational form of newton's second law for disk
//we get 2 equations and 2 unknowns
A = [m 1;(1/2*M) -1]
B = [(m*g);0]
c = A\B
```

```
//acceleration of block
a = c(1)//in m/s^2
//tension in the string
T = c(2)//in N
//angular acceleration of disk
az = a/(R*10^-2)//in rad/s^2
a_z = az/(2*%pi)//in rev/s^2

printf ("\n\n Acceleration of block a = \n\n %.1f m/s^2",a);
printf ("\n\n Tension in the string T = \n\n %.1f N",T);
printf ("\n\n Angular acceleration of disk az in rad/s^2 = \n\n %.1f rad/s^2",az);
printf ("\n\n Angular acceleration of disk a_z in rev/s^2 = \n\n %.1f rev/s^2",a_z);
```

Scilab code Exa 9.12 C9P12

```

clear
 clc
//to find velocity of center of mass at time t
//to find value of t

// GIVEN::

//refer to figure 9-33(a) from page no. 192
//radius of solid cylinder
R = 12//in cm
//mass of solid cylinder
M = 3.2//in kg
//initial angular velocity of solid cylinder
w0 = 15//in rev/s
//coefficient of kinetic friction between surface
```

```
    and cylinder
mew_k = 0.21
//acceleration due to gravity
g = 9.8//in m/s^2

// SOLUTION:

//refer to figure 9-33(b) from page no. 192
w_0 = w0*2*%pi//in rad/rev
//applying newton's second law in x direction
//and applying rotational form of newton's second
    law
//velocity of center of mass
vcm = (1/3*w_0*(R*10^-2))//in m/s
//value of t
t = vcm/(mew_k*g)//in seconds

printf ("\n\n Velocity of center of mass vcm = \n\n
    %.1f m/s",vcm);
printf ("\n\n Value of t = \n\n %.1f seconds",t);
```

Scilab code Exa 9.13 C9P13

```

clear
 clc
//to find rotational velocity when it reaches end of
     the string

// GIVEN::

//refer to figure 9-34(a) from page no. 196
//total mass of yo-yo
M = 0.24//in kg
//radius of disk
```

```
R = 2.8//in cm
//radius of shaft
R0 = 0.25//in cm
//length of the string
L = 1.2//in meters
//initial velocity of yo-yo
v0 = 1.4//in m/s
//acceleration due to gravity
g = 9.8//in m/s^2

// SOLUTION:

//refer to figure 9-34(b) from page no. 196
//momemt of inertia
I = (1/2*(M*R^2))
//applying newton's second law
//and applying rotational form of newton's second law
//angular acceleration
az = (g*100/R0)*(1/(1+R^2/(2*R0^2)))//in rad/s^2
//angle through which yo-yo rotates
fi = L/(R0*10^-2)//in rad
//initial angular velocity
w0z = v0/(R0*10^-2)//in rad/s
//solving using equation to find out time
y = poly([-fi w0z (1/2*az)],'t','coeff')
c = roots(y)
//taking only positive value as it is time
t2 = c(2)//in seconds
//rotational velocity when it reaches end of the string
wz = w0z+(az*t2)//in rad/s^2

printf ("\n\n Angular acceleration az = \n\n %.1f rad/s^2",az);
printf ("\n\n Time for calculating rotational velocity  t2 = \n\n %.2f seconds",t2);
printf ("\n\n initial angular velocity w0z = \n\n
```

```
    %3i rad/s",w0z);
printf("\n\n Rotational velocity when it reaches
    end of the string wz = \n\n %3i rad/s^2",wz);
```

Chapter 10

ANGULAR MOMENTUM

Scilab code Exa 10.2 C10P2

```
clear
clc
//To find which magnitude is greater
//angular momentum of earth associated with its
   rotation on its axis
                              //OR
//angular momentum of earth associated with its
   orbital motion around the sun

//Given:
//refer to figure 10-8 from page no. 213
//rotation period of the earth about its axis in
   hour
t1 = 24//in hour
//rotation period of earth about its axis in seconds
T1 = (t1*60*60)//in seconds
//T2 is time required by earth to complete one
   revolution around the sun
T2 = 3.16*10^7//in seconds
//mass of the earth
M = 5.98*10^24//in kg
```

```
//radius of the earth
RE = 6.37*10^6//in meters

//Solution:
//considering earth as a uniform sphere mmoment of
    inertia
I = (2/5)*M*RE^2
//angular speed
w1 = (2*3.14)/T1//in per seconds
//angular momentum of earth associated with its
    rotation
L_rot = I*w1//in kg m^2/s
//radius of orbit
R_orb = 1.50*10^11//in meters

//angular speed
w2 = (2*3.14)/T2//in per second
//velocity of rotation of earth around the sun
v = w2*R_orb//in m/s
//linear momentum
p = M*v
//angular momentum of earth associated with its
    orbital motion around the sun
L_orb = R_orb*p//in  kg m^2/s

printf ("\n\n Angular momentum of earth associated
    with its rotation on its axis is L_rot = \n\n %.2
    e kg m^2/s" ,L_rot);
printf ("\n\n Angular momentum of earth associated
    with its orbital motion around the sun L_orb = \n
    \n %.2e kg m^2/s",L_orb);
if (L_rot>L_orb) then
    printf('\n\n Angular momentum of earth
        associated with its rotation on its axis is
        greater than angular momentum of earth
        associated with its orbital motion around the
         suns');
else
```

```
    printf('\n\n Angular momentum of earth
        associated with its orbital motion around
        the sun is greater than angular momentum of
        earth associated with its rotation on its
        axis');
end
```

Scilab code Exa 10.4 C10P4

```
clear
clc
//to find centripital force austronautshould apply
    at distance 50 m from spacecraft
//to find centripital force austronautshould apply
    at distance 5 m from spacecraft

//Given:
//mass of austronaut
M = 120//in kg
//length of cord
ri = 180//in meters
// initial tangential velocity acquired by astronaut
vi = 2.5//in m/s

//Solution:
//appiying conservation of angular momentum
//initially required centripital force
F = (M*vi^2)/ri//in N
//when astonaut is at a distance of 50 m from
    spacecraft
r1 = 50//in meters
//velocity at this stage
v = (vi*ri)/r1//in m/s
//centripital force
f = (M*v^2)/r1//in N
```

```
printf ("\n\n Initially required centripital force F
    = \n\n %.1f N" ,F);
printf ("\n\n Tangential speed v = \n\n %.1f m/s" ,v
    );
printf ("\n\n Centripital force austronautshould
    apply at distance 50 m from spacecraft f = \n\n
    %3i N",f);
```

Scilab code Exa 10.5 C10P5

```
clear
clc
//to find angular speed of combination of disk

//Given:
//refer to figure 10-17(a)and(b) from page no. 219
//mass of disk
M = 125//in g
//radius of disk
r = 7.2//in centimeters
// initial angular speed of disc about vertical axis
omega_i = 0.84//in rev/s

//Solution:
//completely inelastic collision.
//appllying conservation of angular momentum
//ratio of rotational inertia of disks
R = (1/3)
//angular speed of combination of disk
omega_f = omega_i*(R)//in rev/s

printf ("\n\n Angular speed of combination of disk
    omega_f = \n\n %.2f rev/s" ,omega_f);
```

Chapter 11

ENERGY 1 WORK AND KINETIC ENERGY

Scilab code Exa 11.1 C11P1

```

clear
 clc
//to find work done

// GIVEN::

//refer to figure 11-8(a) from page no. 232
//mass of block
m = 11.7//in kg
//distance by which block is pushed on inclined
   plane
s = 4.65//in meters
//height by which block is raised
h = 2.86//in meters
//acceleration due to gravity
g = 9.8//in m/s^2

// SOLUTION:
```

```
//refer to figure 11-8(b) from page no. 232
//from diagram sin(theta) can be calculated as
sin_theta = (h/s)
//angle between applied force and displacement of
    block
fi = 0//in degrees
//using newton's second law of motion
//force pushing the block
F = m*g*sin_theta//in N
//work done by force F
W = F*s*cosd(fi)//in J
//work done by raising block vertically
Work = m*g*h//in J
W = round(W)
Work = round(Work)
printf ("\n\n Force pushing the block F = \n\n %.1f
    N",F);
printf ("\n\n Work done by force F W = \n\n %3i J",W
    );
printf ("\n\n Work done by raising block vertically
    \n\n Work = \n\n %3i J",Work);
```

Scilab code Exa 11.2 C11P2

```

clear
 clc
//to find work done by the chid

// GIVEN::

//refer to figure 11-9(a) from page no. 233
//mass of sled
m = 5.6//in kg
```

```
//distance by which sled is pushed horizontally
s = 12//in meters
//coefficient of kinetic friction
mew_k = 0.20
//angle made by the rope with horizontal
fi = 45//in degrees
//acceleration due to gravity
g = 9.8//in m/s^2

// SOLUTION:

//refer to figure 11-9(b) from page no. 233
//using newton's second law of motion
//we get three equations and three unknowns
A = [cosd(fi) -1 0;sind(fi) 0 1;0 1 -mew_k]
B = [0; m*g; 0]
c = A\B
//force applied by the child
F = c(1)//in N
//frictional force
f = c(2)//in N
//normal reaction
N = c(3)//in N
//work done by the child
W = F*s*cosd(fi)//in J


F = round(F)
W = round(W)
printf ("\n\n Force applied by the child F = \n\n
    %2i N",F);
printf ("\n\n Work done by the child W = \n\n %3i J"
    ,W);
```

Scilab code Exa 11.3 C11P3

```
clear
 clc
//to find average power must be applied by the
   elevator motor

// GIVEN::

//weight of elevator
w = 5160//in N
//average weight of passenger
wp = 710//in N
//number of passengers
n = 20
//distance between floors
sf = 3.5//in meters
//time elasped
t = 18//in seconds
//acceleration due to gravity
g = 9.8//in m/s^2

// SOLUTION:

//total weight of elevator and passenger
//upward force exerted by motor
F = w+n*wp//in N
//total height by which elevator moves
s = sf*25//in meters
//work done must be applied by the elevator motor
W = F*s//in J
//average power
Pav = (W/t)*10^-3//in kW

//value of force F is slightly different than scilab
    answer
//but silab answer is same as calculator answer
printf ("\n\n Upward force exerted by motor F = \n\n
    %5i N",F);
```

```
printf ("\n\n Work done must be applied by the elevator motor W  = \n\n %.1e J",W);
printf ("\n\n Average power Pav = \n\n %2i kW",Pav);
```

Scilab code Exa 11.4 C11P4

```
clear
 clc
//to find work done by gravity
//to find work done by the spring
//to find work done by the hand
// GIVEN::
//refer to figure 11-15(a) from page no. 237
//mass of block
m = 6.40//in kg
//distance streched by spring
d = 0.124//in meters
//acceleration due to gravity
g = 9.8//in m/s^2
// SOLUTION:
//refer to figure 11-8(b)and 11-5(c) from page no. 237
//applying equillibrium condition in y direction
//force constant of spring
k = m*g/d//in N/m
//work done by gravity
Wg = m*g*d//in J
//work done by the spring
Ws = (-1/2)*k*d^2//in J
```

```
//-ve sign as force and displacement are in opposite
    directions
//work done by the hand
//intergrating force in y direction
Wh = m*g*(-d)+(1/2)*k*(-d)^2//in J
k = round(k)
printf ("\n\n Force constant of spring k = \n\n %3i
   N/m",k);
printf ("\n\n Work done by gravity Wg = \n\n %.2f J"
   ,Wg);
printf ("\n\n Work done by the spring Ws = \n\n %.2f
    J",Ws);
printf ("\n\n Work done by the hand Wh = \n\n %.2f J
   ",Wh);
```

Scilab code Exa 11.6 C11P6

```

clear
 clc
//to find kinetic energy

// GIVEN::

//distance travelled by neutron
d = 6.2//in meters
//time for neutron travel
t = 160//in micrometers
//mass of neutron
m = 1.67e-27//in kg

// SOLUTION:

//speed of neutron
v = d/(t*10^-6)//in m/s
```

```
//applying formula for kinetic energy
//kinetic energy of neutron
K = (1/2)*m*v^2//in J
K1 = K*(6.242e18)//in eV
K = nearfloat("succ",1.26e-18)
K1 = nearfloat("succ",7.9)

printf ("\n\n Speed of neutron v = \n\n %.2e m/s",v);
printf ("\n\n Kinetic energy of neutron in J K  = \n\n %.2e J",K);
printf ("\n\n Kinetic energy of neutron in eV K  = \n\n %.1f eV",K1);
```

Scilab code Exa 11.7 C11P7

```

clear
 clc
//to find speed of body when it strikes the ground

// GIVEN::
//mass of body
m = 4.5//in kg
//height from which body is dropped
h = 10.5//in meters
//acceleration due to gravity
g = 9.80//in m/s^2

// SOLUTION:
//using work-energy principle
//speed of body when it strikes the ground
v = sqrt(2*g*h)//in m/s
printf ("\n\n Speed of body when it strikes the ground v = \n\n %.1f m/s",v);
```

Scilab code Exa 11.8 C11P8

```
clear
 clc
//to find spring compression

// GIVEN::
//mass of body
m = 3.63//in kg
//speed of block
v = 1.22//in m/s
//force constant for spring
k = 135//in

// SOLUTION:
//using work-energy principle
//spring compression
d = v*sqrt(m/k)//in meters
d1 = d*10^2//in
printf ("\n\n Spring compression d = \n\n %.3f m",d)
    ;
printf ("\n\n Spring compression d = \n\n %.1f cm",
    d1);
```

Scilab code Exa 11.9 C11P9

```
clear
 clc
//to find speed of crate according to observer o
```

```
////to find work and change in kinetic energy

// GIVEN:
//refer to figure 11-18(a),(b)from page no. 242
//force applied
Fx = 5.63//in N
//mass of crate
m = 12.0//in kg
//speed of train
vx = 15.0//in m/s
//distance travelled by crate
s = 2.4//in meters

// SOLUTION:
//using work-energy principle
//work done
W = Fx*s//in J
//initial kinetic energy according to observer in
    car
Ki = 0
////final kinetic energy according to observer in
    car
Kf = W -Ki
//speed of crate according to observer o
vf = sqrt(2*Kf/m)//in m/s
//applying impulse-momentum theorem
//time interval
delta_t = (m*vf/Fx)//in seconds
//forward distance travelled
d = vx*delta_t//in meters
//total distance moved by crate
s_dash = d+s//in meters
//work done
W_dash = Fx*s_dash//in J
//final speed of crate
vf_dash = vx+vf//in m/s
//change in kinetic energy
deltaK_dash = (1/2*m*(vf_dash^2))-(1/2*m*(vx^2))
```

```
W_dash = round(W_dash)
deltaK_dash = round(deltaK_dash)
printf ("\n\n Final kinetic energy according to
    observer in car Kf = \n\n %.1f J",Kf);
printf ("\n\n Speed of crate according to observer o
    vf = \n\n %.2f m/s",vf);
printf ("\n\n Time interval delta_t = \n\n %.2f
    seconds",delta_t);
printf ("\n\n Work done W_dash = \n\n %3i J",W_dash)
    ;
printf ("\n\n Change in kinetic energy deltaK_dash =
    \n\n %3i J",deltaK_dash);
printf ("\n\n As W_dash = deltaK_dash work-energy
    principle is valid")
```

Scilab code Exa 11.10 C11P10

```

clear
 clc
//to find conatance force to be applied

// GIVEN:
//refer to figure 11-21 from page no. 244
//initial angular velocity of spacecraft
wi = 2.4//in rev/s
//radius of spacecraft
R = 1.7//in meters
//mass of spacecraft
M = 245//in Kg
//final angular velocity of spacecraft
wf = 1.7//in rev/s
//rotation of spacecraft
theta = 3//in revolutions

```

```
// SOLUTION:

//moment of inertia of spacecraft
I = (2/3*M*R^2)//in Kg.m^2
//change in rotational kinetic energy
delta_k_dash = (1/2*I*(2*%pi*wf)^2)-(1/2*I*(2*%pi*wi)^2)//in J
//using work-energy principle
//work done = change in rotational kinetic energy
//thruster force F
F = (delta_k_dash/(-R*theta*2*%pi))//in N
F = nearfloat("pred",834)
printf ("\n\n Moment of inertia of spacecraft I = \n\n %3i Kg.m^2",I);
printf ("\n\n Change in rotational kinetic energy delta_k-dash = \n\n %.2e J",delta_k_dash);
printf ("\n\n Thruster force F = \n\n %3i N",F);
```

Scilab code Exa 11.11 C11P11

```

clear
 clc
//to find kinetic energy lost by neutron

// GIVEN:

//initial kinetic energy of neutron
K1i = 5.0//in MeV
//mass of neuron mn
mn = 1//considering it as unity as other masses are given with reference to mn
//mass of neucleus of lead
mPb = 206*mn
```

```
//mass of neucleus of carbon
mC = 12*mn
//mass of neucleus of hydrogen
mH = mn

// SOLUTION:

//As collision is elastic collision
//using conservation of energy principle

//collision with neucleus of lead
//final kinetic energy of neutron
K1f = K1i*((mn-mPb)/(mn+mPb))^2//in MeV
//kinetic energy lost by neutron
K_lostl = K1i-K1f//in MeV


//collision with neucleus of carbon
//final kinetic energy of neutron
K1f_C = K1i*((mn-mC)/(mn+mC))^2//in MeV
//kinetic energy lost by neutron
K_lostC = K1i-K1f_C//in MeV


//collision with neucleus of lead
//final kinetic energy of neutron
K1f_H = K1i*((mn-mH)/(mn+mH))^2//in MeV
//kinetic energy lost by neutron
K_lostH = K1i-K1f_H//in MeV

printf ("\n\n Collision with neucleus of lead")
printf ("\n\n Final kinetic energy of neutron K1f =
    \n\n %.1f MeV",K1f);
printf ("\n\n Kinetic energy lost by neutron K_lostl
     = \n\n %.1f MeV",K_lostl);
printf ("\n\n Collision with neucleus of carbon")
printf ("\n\n Final kinetic energy of neutron K1f_C
    = \n\n %.1f MeV",K1f_C);
```

```
printf ("\n\n Kinetic energy lost by neutron K_lostC
    = \n\n %.1f MeV",K_lostC);
printf ("\n\n Collision with neucleus of hydrogen")
printf ("\n\n Final kinetic energy of neutron K1f_H
   = \n\n %.1f MeV",K1f_H);
printf ("\n\n Kinetic energy lost by neutron K_lostH
    = \n\n %.1f MeV",K_lostH);
```

Scilab code Exa 11.12 C11P12

```

clear
 clc
//to find initial speed of bullet
//to find lost in kinetic energy

// GIVEN:
//refer to figure 11-23 from page no. 246
//mass of block
M = 5.4//in Kg
//mass of bullet
m = 9.5e-3//in Kg
//height to which block rises
h = 6.3e-2//in meters
//acceleration due to gravity
g = 9.8//in m/s^2

// SOLUTION:

//applying work-energy principle
//initial speed of bullet
vi = ((M+m)/m)*(sqrt(2*g*h))//in m/s
//ratio of final to initial kinetic enerdy
Kf_by_Ki = (m/(M+m))
//initialkinetic energy remains after collision
```

```
Kr = (Kf_by_Ki)*100//in percentage
//kinetic energy stored inside pendullum
Ks = 100-Kr//in percentage
//answer of vi is slightly different than textbook.
    but answer by calculator is same as that of
    scilab
printf ("\n\n Initial speed of bullet vi = \n\n %3i
   m/s",vi);
printf ("\n\n Ratio of final to initial kinetic
    enerdy Kf/Ki = \n\n %.4f ",Kf_by_Ki);
printf ("\n\n Initial kinetic energy remains after
    collision Kr = \n\n %.2f percent",Kr);
printf ("\n\n Kinetic energy stored inside pendullum
     Ks = \n\n %.2f percent",Ks);
```

Chapter 12

ENERGY 2 POTENTIAL ENERGY

Scilab code Exa 12.1 C12P1

```

clear
 clc
//to find change in gravitational potential energy

// GIVEN:
//mass of elevator
m = 920//in Kg
//height above the groung
h = 412//in meters
//acceleration due to gravity
g = 9.8//in m/s^2

// SOLUTION:
//applying potential energy formula
//change in gravitational potential energy
delta_U = m*g*h//in J
delta_U1 = delta_U*10^-6//in MJ

```

```
printf ("\n\n Change in gravitational potential
   energy delta_U = \n\n %.1e J",delta_U);
printf ("\n\n Change in gravitational potential
   energy delta_U1 = \n\n %.1f MJ",delta_U1);
```

Scilab code Exa 12.2 C12P2

```

clear
 clc
//to find potential energy stored in the spring

// GIVEN:
//foce constant of spring
k = 1.25e8//in N/m
//compression in spring
x = 5.6e-2//in meters

// SOLUTION:
//applying spring force formula
//potential energy stored in the spring
U = (1/2*k*x^2)//in J
printf ("\n\n Potential energy stored in the spring
   U = \n\n %.2e J",U)
```

Scilab code Exa 12.3 C12P3

```

clear
 clc
//to find speed of ball

// GIVEN:
```

```
//refer to figure 12-1
//compression in spring
d = 3.2e-2//in meters
//mass of ball
m = 12e-3//in Kg
//force constant of spring
k = 7.5//in N/cm

// SOLUTION:
//applying conservation of energy principle
//speed of ball
vm = d*sqrt((k*10^2)/m)//in m/s

printf ("\n\n Speed of ball vm = \n\n %.1f m/s",vm)
```

Scilab code Exa 12.4 C12P4

```


clear
 clc
//to find speed of ball

// GIVEN:
//refer to figure 12-6 on page no. 263
//lift of car
y = 25//in meters
//acceleration due to gravity
g = 9.8//in m/s^2

// SOLUTION:
//applying conservation of energy principle
//speed of car
v = sqrt(2*g*y)//in m/s
printf ("\n\n Speed of car v = \n\n %2i m/s",v)
```

Scilab code Exa 12.7 C12P7

```

clear
 clc
//to find speed of ball

// GIVEN:
//refer to problem 9-10
//mass of disk
M = 2.5//in kg
//distance of fall
y = 0.56//in meters
//mass of block
m = 1.2//in kg
//acceleration due to gravity
g = 9.8//in m/s^2

// SOLUTION:
//applying conservation of mechanocal energy
   principle
//speed of block
v = sqrt((4*m*g*y)/(M+2*m))//in m/s
printf ("\n\n Speed of ball v = \n\n %.1f m/s",v)
```

Chapter 13

ENERGY 3 CONSERVATION OF ENERGY

Scilab code Exa 13.1 C13P1

```

clear
 clc
//to find change in internal energy

// GIVEN:
//mass of baseball
m = 0.143//in kg
//height of tower
h = 443//in m
//terminal velocity
v = 42//in m/s
//acceleration due to gravity
g = 9.8//in m/s^2

// SOLUTION:

//initial potential energy
Ui = m*g*h//in J
```

```
//final potential energy
Uf = 0//in J
//change in potential energy
delta_U = (Uf-Ui)//in J
//final kinetic energy
Kf = (1/2)*(m*v^2)//in J
//initial kinetic energy
Ki = 0//in J
//change in kinetic energy
delta_K = (Kf-Ki)//in J
//applying conservation of energy principle
//change in internal energy
delta_Eint = (-delta_U-delta_K)//in J
delta_U = round (Uf-Ui)
delta_K = round(Kf-Ki)
delta_Eint = round(-delta_U-delta_K)

printf ("\n\n Change in potential energy delta_U =
   \n\n %3i J",delta_U)
printf ("\n\n Change in kinetic energy delta_K = \n
   \n %3i J",delta_K)
printf ("\n\n Change in internal energy delta_Eint
   = \n\n %3i J",delta_Eint)
```

Scilab code Exa 13.2 C13P2

```

clear
 clc
 //to find gain in internal energy
//to find speed of block


// GIVEN:
//mass of block
```

```
m = 4.5//in Kg
//angle of inclination
theta = 30//in degrees
//initial speed
v = 5.0//in m/s
//distance travelled
d = 1.5//in meters
//acceleration due to gravity
g = 9.8//in m/s^2

// SOLUTION:
//applying conservation of energy principle
//consider block+plane+earth as our system
//final potential energy
Uf = m*g*(d*sind(theta))//in J
//initial potential energy
Ui = 0//in J
//change in potential energy
delta_U = Uf-Ui//in J
//final kinetic energy
Kf = 0//in J
//initial kinetic energy
Ki =(1/2)*m*v^2//in J
//change in kinetic energy
delta_K = Kf-Ki//in J
//change in mechanical energy in system
delta_U_plus_delta_K = delta_U+delta_K//in J
//applying conservation of energy principle
//gain in internal energu
delta_E_int = -(delta_U_plus_delta_K)//in J
//final kinetic energy for downhill journy
//here delta_K = 2*delta_E_int as round tripi.e. uphill and downhill motion
KF = (-(2*delta_E_int))+(-delta_K)//in J
//speed of block
vf = sqrt(2*KF/m)//in m/s
KF = round(KF)

```

```
printf ("\n\n Change in potential energy delta_U  =
   \n\n %2i J",delta_U)
printf ("\n\n Change in kinetic energy delta_K  = \n
   \n %2i J",delta_K)
printf ("\n\n Change in mechanical energy in system
   delta_U_plus_delta_K   = \n\n %2i J",
   delta_U_plus_delta_K)
printf ("\n\n Gain in internal energy delta_E_int  =
    \n\n %2i J",delta_E_int)
printf ("\n\n Final kinetic energy for downhill
   journy KF  = \n\n %2i J",KF)
printf ("\n\n Speed of block vf = \n\n %.1f m/s",vf)
```

Scilab code Exa 13.3 C13P3

```

clear
 clc
//to find speed of center of mass
//to find change in stored internal energy


// GIVEN:
//refer to figure 13-5 on page no. 285
//mass of ice skater
M = 50//in Kg
//force exerted
F = 55//in N
//distance moved by center of mass
scm = 32e-2//in m
// SOLUTION:
//consider newton's third law and center of mass
   equation
//speed of center of mass
vcm = sqrt(2*F*scm/M)//in m/s
```

```
//applying conservation of energy principle
//change in stored internal energy
delta_Eint = -(1/2)*(M*vcm^2)//in J

printf ("\n\n Speed of center of mass vcm = \n\n %
    .2f m/s",vcm)
printf ("\n\n Change in stored internal energy
    delta_Eint = \n\n %.1f J",delta_Eint)
```

Scilab code Exa 13.4 C13P4

```

clear
 clc
//to find speed of John after contact is broken
//to find change in stored internal energy of skater


// GIVEN:
//refer to figure 13-9(a),(b) on page no. 288
//mass of John skater
M = 50//in Kg
//mass of Jim skater
M1 = 72//in Kg
//force exerted by Jim
Fext = 55//in N
//distance through which force is applied
s = 32e-2//in m
//distabce moved by center of mass
scm = 58e-2//in m

// SOLUTION:
//consider John as our system
//applyinq consevation of energy principle
//applying center of mass equation
```

```
//change in kinetic energy
delta_Kcm = Fext*scm//in J
//speed of John after contact is broken
vcm = sqrt(2*delta_Kcm/M)//in m/s
//change in John's internal energy
delta_E_int_John = Fext*s-Fext*scm//in J
//change in Jim's internal energy
delta_E_int_Jim = -(Fext*s)//in J

printf ("\n\n Change in kinetic energy delta_Kcm =
    \n\n %.1f J",delta_Kcm)
printf ("\n\n Speed of John after contact is broken
    vcm = \n\n %.2f m/s",vcm)
printf ("\n\n Change in Johns internal energy
    delta_E_int_John = \n\n %.1f J",delta_E_int_John)
printf ("\n\n Change in Jim internal energy
    delta_E_int_Jim = \n\n %.1f J",delta_E_int_Jim)
```

Scilab code Exa 13.5 C13P5

```

clear
 clc
//to find change in stored internal energy of system
    of block+surface
//distance travelled by block befire coming to rest

// GIVEN:
//mass of block
M = 5.2//in Kg
//initial horizontal velocity of block
vcm = 0.65//in m/s
//coefficient of kinetic friction
mew = 0.12
//acceleration due to gravity
```

```
g = 9.8//in m/s^2

// SOLUTION:
//applyinq consevation of energy principle
//change in stored internal energy of system of
   block+surface
//final kinetic energy is zero as block comes to
   rest
delta_Eint = -(0-(1/2*M*vcm^2))//in J //-ve sign as
   kinetic energy is lost
//distance travelled by block befire coming to rest
scm = (vcm^2/(2*mew*g))//in m

printf ("\n\n Final kinetic energy is zero as block
   comes to rest delta_Eint = \n\n %.1f J",
   delta_Eint)
printf ("\n\n Distance travelled by block befire
   coming to rest scm = \n\n %.2f m",scm)
```

Scilab code Exa 13.6 C13P6

```

clear
 clc
//to find energy and direction of outgoing particl 3
   H

// GIVEN:
//refer to figure 13-11 from page no. 290
//difference in internal energy of initial and final
    partical
delta_Eint = 4.03//in MeV
//initial kinetic energy of deuteron
Ki = 1.50//in MeV
//initial kinetic energy of proton
```

```
K1 = 3.39//in MeV
//mass of hydrogen
m1 = 1.01//u
//mass of deuteron
m2 = 2.01//u
//mass of proton
m3 = 3.02//u

// SOLUTION:
//applyinq consevation of energy principle
//final kinetic energy
Kf = delta_Eint+Ki//in MeV
//final kinetic energy of outgoing partical 3H
K3 = Kf-K1//in MeV
//applying conservation of momentum principle
//value of cosfi
f = sqrt((m2*Ki)/(m3*K3))
//direction of outgoing particl 3H
fi = acosd(sqrt((m2*Ki)/(m3*K3)))//in degrees

printf ("\n\n Final kinetic energy Kf = \n\n %.2f
    MeV",Kf)
printf ("\n\n Final kinetic energy of outgoing
    partical 3H K3 = \n\n %.2f MeV",K3)
printf ("\n\n Value of cosfi = \n\n %.3f ",f)
printf ("\n\n Direction of outgoing particl 3H fi =
    \n\n %.1f degree",fi)
```

Scilab code Exa 13.7 C13P7

```

clear
 clc
//to find kinetic energy of radon and alpha partical

```

```
// GIVEN:
//decrease in internal energy
delta_E = 4.87//in MeV
//mass of alpha partical
mHe = 4.00//in u
//mass of radon partical
mRn = 222.0//in u

// SOLUTION:
//applying conservation of energy principle
//we get two equations
//one for ratio of kinetic energies and second for total kinetic energy
//solving two equations using matrix
A = [1 (-mHe/mRn);1 1]
b = [0;4.87]
c = A\b
//ratio of kinetic energies
KRn_by_KHe = mHe/mRn
//total kinetic energy of products
Kf = delta_E//in MeV
//kinetic energy of radon partical
K_Rn = c(1)//in MeV
//kinetic energy of alpha partical
K_He = c(2)//in MeV

printf ("\n\n Ratio of kinetic energies KRn_by_KHe = \n\n %.4f",KRn_by_KHe)
printf ("\n\n Total kinetic energy of products Kf = \n\n %.2f MeV",Kf)
printf ("\n\n Kinetic energy of radon partical K_Rn = \n\n %.3f MeV",K_Rn)
printf ("\n\n Kinetic energy of alpha partical K_He = \n\n %.2f MeV",K_He)
```

Chapter 14

GRAVITATION

Scilab code Exa 14.1 C14P1

```
clear
 clc
//to find magnitude of gravitational force exerted
   on cantaloupe on the surface of earth
//due to (a)the Earth (b)the Moon (c)the Sun

// GIVEN:
//mass of cantaloupe
mc = 1.00//in Kg
//acceleration due to gravity
g = 9.8//in m/s^2
//Gravitational constant
G = 6.67e-11//in N.m^2/Kg^2
//mass of moon
m_M = 7.36e22//in Kg
//mass of sun
m_S = 1.99e30//in Kg
//radius of moon
r_M = 3.82e8//in m
//radius of sun
r_S = 1.50e11//in m
```

```

// SOLUTION:
//applying newton's law of universal gravitation
//gravitational force exerted on cantaloupe on the surface of earth
//due to (a)the Earth
FcE = mc*g//in N
//gravitational force exerted on cantaloupe on the surface of earth
//due to (a)the Moon
FcM = G*((mc*m_M)/(r_M)^2)//in N
//gravitational force exerted on cantaloupe on the surface of earth
//due to (a)the Sun
FcS = G*((mc*m_S)/(r_S)^2)//in N

printf ("\n\n Gravitational force exerted on cantaloupe on the surface of earth\n due to (a) the Earth FcE = \n\n %.1f N",FcE)
printf ("\n\n Gravitational force exerted on cantaloupe on the surface of earth\n due to (b) the Moon FcM = \n\n %.2e N",FcM)
printf ("\n\n Gravitational force exerted on cantaloupe on the surface of earth\n due to (c) the Sun FcS = \n\n %.2e N",FcS)
```

Scilab code Exa 14.2 C14P2

```
clear
 clc
//to find magnitude and direction of gravitational force

// GIVEN:
//refer to figure 14-4 on page no. 302
```

```
//mass of astronaut
ma = 105//in Kg
//mass of first asteroid
m1 = 346//in Kg
//radius of first asteroid
r1 = 215//in m
//mass of second asteroid
m2 = 184//in Kg
//radius of second asteroid
r2 = 142//in m
//angle between forces
theta = 120//in degrees
//Gravitational constant
G = 6.67e-11//in N.m^2/Kg^2

// SOLUTION:
//applying newton's law of universal gravitation
//magnitude of gravitational force due to first asteroid
Fa1 = G*((ma*m1)/(r1^2))//in N
//magnitude of gravitational force due to second asteroid
Fa2 = G*((ma*m2)/(r2^2))//in N
//magnitude of total gravitational force
//using parallelogram method
Fa = sqrt((Fa1^2)+(Fa2^2)+(2*Fa1*Fa2*cosd(theta)))
//direction of gravitational force
fi = atand((Fa2*sind(theta))/(Fa1+(Fa2*cosd(theta))))//in degrees
Fa = nearfloat("pred",5.80e-11)

printf ("\n\n Magnitude of gravitational force due to first asteroid Fa1 = \n\n %.2e N",Fa1)
printf ("\n\n Magnitude of gravitational force due to second asteroid Fa2 = \n\n %.2e N",Fa2)
printf ("\n\n Magnitude of total gravitational force Fa = \n\n %.2e N",Fa)
printf ("\n\n Direction of gravitational force fi =
```

```
\n\n %.1f degree",fi)
```

Scilab code Exa 14.3 C14P3

```
clear
 clc
//to find free fall acceleration of neutron ster and
    asteroid ceres

// GIVEN:
//mass of neutron star
Mn = 1.99e30//in Kg
//radius of neutron star
Rn = 12e3//in m
//mass of asteroid ceres
Mc = 1.2e21//in Kg
//radius of asteroid ceres
Rc = 4.7e5//in m
//Gravitational constant
G = 6.67e-11//in N.m^2/Kg^2

// SOLUTION:
//applying newton's law of universal gravitation and
    newton's second law of motion
//free fall acceleration of neutron sterid
g0 = G*(Mn/(Rn^2))//in m/s^2
//free fall acceleration of austeroid ceres
go = G*(Mc/(Rc^2))//in m/s^2

printf ("\n\n Free fall acceleration of neutron
   sterid g0 = \n\n %.1e m/s^2",g0)
printf ("\n\n Free fall acceleration of austeroid
   ceres go = \n\n %.2f m/s^2",go)
```

Scilab code Exa 14.4 C14P4

```
clear
 clc
//to find speed of partical at r = 0

// GIVEN:

//mass of Earth
ME = 5.98e24//in Kg
//radius of Earth
RE = 6.37e6//in m
//Gravitational constant
G = 6.67e-11//in N.m^2/Kg^2

// SOLUTION:
//applying newton's law of universal gravitation and
    law of conservation of energy
//speed of partical at r = 0
v = sqrt((G*ME)/(RE))//in m/s
printf ("\n\n Speed of partical at r = 0 is v = \n\n
    %.2e m/s",v)
```

Scilab code Exa 14.5 C14P5

```
clear
 clc
//to find speed of canister when it enters the Earth
   's atmosphere

// GIVEN:

```

```
//mass of Earth
ME = 5.98e24//in Kg
//radius of Earth
RE = 6.37e6//in m
//initial speed of canister
vi = 525//in m/s
//distance above earth's surface
h = 100e3//in m
//Gravitational constant
G = 6.67e-11//in N.m^2/Kg^2

// SOLUTION:
//applying newton's law of universal gravitation and law of conservation of energy
//speed of canister when it enters the Earth's atmosphere
vf_square = vi - ((2*G*ME)*((1/(3*RE))-(1/(RE+h)))) //in m^2/s^2
vf = sqrt(vi - ((2*G*ME)*((1/(3*RE))-(1/(RE+h)))))//in m/s
vf = nearfloat("succ",9.05e3)
vf_square = nearfloat("succ",8.18e7)

printf ("\n\n Square of speed of canister when it enters the Earths atmosphere vf_square = \n\n %.2e m^2/s^2",vf_square)
printf ("\n\n Speed of canister when it enters the Earths atmosphere vf = \n\n %.2e m/s",vf)
```

Scilab code Exa 14.7 C14P7

```
clear
clc
//to find mass of Sun and mass of Jupiter

```

```
// GIVEN:

//orbital radius of earth
re = 1.50e11//in m
//period of revolution for earth
Te = 3.15e7//in seconds
//orbital radius of Moon
rm = 4.22e8//in m
//period of revolution for Moon
Tm = 1.53e5//in seconds
//Gravitational constant
G = 6.67e-11//in N.m^2/Kg^2

// SOLUTION:
//applying Kepler's law of peroids
//mass of Sun using Earth's orbital motion
M = (4*(%pi^2)*(re^3))/(G*(Te^2))//in Kg
//mass of Jupiter using Moon's orbital motion
M_ = (4*(%pi^2)*(rm^3))/(G*(Tm^2))//in Kg

printf ("\n\n Mass of Sun using Earth orbital motion
    M = \n\n %.2e Kg",M)
printf ("\n\n Mass of Jupiter using Moon orbital
    motion M = \n\n %.2e Kg",M_)
```

Scilab code Exa 14.8 C14P8

```
clear
clc
//to find height above the Earth

// GIVEN:

//period of the satellite
T = 86400//in seconds
```

```
//mass of Earth
ME = 5.98e24//in Kg
//radius of Earth
RE = 6.37e6//in meters
//Gravitational constant
G = 6.67e-11//in N.m^2/Kg^2

// SOLUTION:
//applying Kepler's law of peroids
//radius of orbit of satellite
r = ((G*T^2*ME)/(4*%pi^2))^(1/3)//in meters
//height above the Earth
h = r-RE//in meters
r = nearfloat("pred",4.22e7)
h = nearfloat("pred",3.58e7)

printf ("\n\n Radius of orbit of satellite r = \n\n
   %.2e m",r)
printf ("\n\n Height above the Earth h = \n\n %.2e m
   ",h)
```

Scilab code Exa 14.9 C14P9

```
clear
clc
//to find aphelion or farthest distance of Halley's
   comet from the Sun and eccentricity of it's orbit

// GIVEN:
//refer to figure 14-16 from page no. 313
//period of the Halley's comet
T = 76 //in years
//mass of Sun
M = 2.0e30//in Kg
//closest approach to the Sun
```

```
//minimum distance of Halley's comet from Sun
Rp = 8.8e10//in meters
//Gravitational constant
G = 6.67e-11//in N.m^2/Kg^2

// SOLUTION:
//applying Kepler's law of peroids
//semimajor axis
a = ((G*(T*365*24*60*60)^2*M)/(4*%pi^2))^(1/3)//in
    meters //taking T in seconds
//refer to figure 14-14
//maximum distance of Halley's comet from Sun
Ra = (2*a)-Rp//in meters
//eccentricity of Halley's orbit
e = 1-(Rp/a)

printf ("\n\n Semimajor axis a = \n\n %.1e m",a)
printf ("\n\n Maximum distance of Halley comet from
    Sun Ra = \n\n %.1e m",Ra)
printf ("\n\n Eccentricity of Halley orbit e = \n\n
    %.2f ",e)
```

Scilab code Exa 14.10 C14P10

```
clear
clc
//to find energy,period,semimajor axis of B before
    and after burn

// GIVEN:
//refer to figure 14-19 from page no. 315
//mass of spacecraft
m = 3250//in Kg
//height above Earth
h = 270//in Km
```

```
//radius of earth
RE = 6370//in Km
//mass of earth
ME = 5.98e24//in Kg
//decrease in velocity after burn
d = 0.95//in percent
//Gravitational constant
G = 6.67e-11//in N.m^2/Kg^2

// SOLUTION:
//before burn
//semimajor axis before burn
a = RE+h//in Km
//energy before burn
E = -(G*m*ME)/(2*a*(1000))//in J
//period before burn
//applying Krpler's law of peroids
T = ((4*(%pi^2)*((a*1000)^3))/(G*ME))^(1/2)//in
    seconds
//kinetic energy before burn
K = -(E)//in J
//velocity before burn
v = sqrt((2*K)/m)//in m/s

//after burn
//velocity after burn
v_dash = (1-(d*0.01))*v//in m/s
//kinetic energy after burn
K_dash = 1/2*(m)*(v_dash)^2//in J
//potential energy after burn
U_dash = -(K)//in J
//total energy after burn
E_dash = K_dash+(2*U_dash)//in J
//semimajor axis after burn
a_dash = -((G*m*ME)/(2*E_dash))//in meters
//period after burn
T_dash = ((4*(%pi^2)*((a_dash)^3))/(G*ME))^(1/2)//in
     seconds
```

```
T = nearfloat("pred",5381)
E_dash = nearfloat("succ",-9.94e10)
T_dash = nearfloat("succ",5240)

printf ("\n\n Semimajor axis before burn a = \n\n %4i Km",a)
printf ("\n\n Energy before burn E = \n\n %.2e J",E)
printf ("\n\n Period before burn T = \n\n %4i s",T)
printf ("\n\n Kinetic energy before burn K = \n\n %.2e J",K)
printf ("\n\n Velocity before burn v = \n\n %.2e m/s",v)
printf ("\n\n Velocity after burn v_dash = \n\n %.2e m/s",v_dash)
printf ("\n\n Kinetic energy after burn K_dash = \n\n %.2e J",K_dash)
printf ("\n\n Total energy after burn E_dash = \n\n %.2e J",E_dash)
printf ("\n\n Semimajor axis after burn a_dash = \n\n %.2e m",a_dash)
printf ("\n\n Period after burn T_dash = \n\n %4i s",T_dash)
```

Chapter 15

FLUID STATICS

Scilab code Exa 15.1 C15P1

```
clear
clc
//to find density of oil

// GIVEN:
//refer to figure 15-6 from page no. 336
//height of water level above oil on one side
d = 12.3//in mm
//height of water level above oil on second side
a = 67.5//in mm
//density of water
rho_w = 1.000e3//in Kg/m^3

// SOLUTION:
//equating pressure on both sides
//density of oil
rho = rho_w*((2*a)/((2*(a)+d)))//in Kg/m^3

printf ("\n\n Density of oil rho = \n\n %3i Kg/m^3",
   rho)
```

Scilab code Exa 15.2 C15P2

```
clear
clc
//to find applied force
//to find distance by which car is raised

// GIVEN:
//refer to figure 15-9 from page no. 338
//diameter of smaller piston
Di = 2.2//in cm
//combined mass
M = 1980//in Kg
//diameter of larger piston
DO = 16.4//in cm
//length of pump handle
L = 36//in cm
//distance of pivot to the piston
x = 9.4//in cm
//acceleration due to gravity
g = 9.8//in m/s^2
//vertical distance by which hand moves
h = 28//in cm

// SOLUTION:
//area of larger piston
AO = %pi*(DO/2)^2//in cm^2
//area of smaller piston
Ai = %pi*(Di/2)^2//in cm^2
//applied force to the smaller piston
Fi = M*g*(Ai/AO)//in N
//using Newton's third law of motion
//applied force at the end of pump handle
Fh = Fi*(x/L)//in N
```

```
//distance moved by smaller piston
di = h*(x/L)//in cm
//equating pressure on each side
//distance moved by larger piston and car is raised
   by
d0 = di*(Ai/A0)//in cm

printf ("\n\n Applied force to the smaller piston Fi
    = \n\n %3i N",Fi)
printf ("\n\n Applied force at the end of pump
   handle Fh = \n\n %2i N",Fh)
printf ("\n\n Distance moved by smaller piston di =
   \n\n %.1f cm",di)
printf ("\n\n Distance moved by larger piston and
   car is raised by d0 = \n\n %.2f cm",d0)
```

Scilab code Exa 15.3 C15P3

```
clear
clc
//to find fraction of total volume of iceberg is
   exposed

// GIVEN:
//density of water
rho_w = 1024//in Kg/m^3
//density of ice
rho_i = 917//in Kg/m^3

// SOLUTION:
//applying Archimedes' principle
//ratio of volume of water displaced to volume of
   submerged portion of ice
Vw_by_Vi = (rho_i/rho_w)*100//in percent
//percent of iceberg exposed
```

```
V = 100-(Vw_by_Vi)//in percent

printf ("\n\n Ratio of volume of water displaced to
    volume of submerged portion of ice Vw_by_Vi = \n\
    n %.1f percent",Vw_by_Vi)
printf ("\n\n Percent of iceberg exposed  = \n\n %.1
    f percent",V)
```

Scilab code Exa 15.4 C15P4

```
clear
clc
//to find atmospheric pressure

// GIVEN:
//height of mercury column in barometer
h = 740.35//in mm
//temperature
T = -5.0//in degree
//density of mercure
rho = 1.3608e4//in Kg/m^3
//acceleration due to gravity
g = 9.7835//in m/s^2

// SOLUTION:
//atmospheric pressure
po = rho*g*h*10^-3//in Pa
//taking h in meters

printf ("\n\n Atmospheric pressure po  = \n\n %.4e
    Pa",po)
```

Scilab code Exa 15.5 C15P5

```
clear
clc
//to find surface tension of liquid

// GIVEN:
//refer to figure 15-15(a) on page no. 343
//upward force
p = 3.45e-3//in N
//length of wire
d = 4.85//in cm
//linear mass density
mew = 1.75e-3//in Kg/m
//acceleration due to gravity
g = 9.7835//in m/s^2

// SOLUTION:
//refer to figure 15-15(a) on page no. 343
//using equilibrium condition
//surface tension of liquid
Gamma = (p-(mew*(d*10^-2)*g))/(2*d*(10^-2))//in N/m
//taking d in meters

printf ("\n\n Surface tension of liquid Gamma = \n\n %.3f N/m",Gamma)
```

Chapter 16

FLUID DYNAMICS

Scilab code Exa 16.1 C16P1

```
clear
clc
//to find volume flow rate of water

// GIVEN:
//refer to figure 16-5 on page no. 354
//cross sectional area
A1 = 1.2//in cm^2
//cross sectional area
A2 = 0.35//in cm^2
//vertical distance between two levels
h = 45//in mm
//acceleration due to gravity
g = 9.8//in m/s^2

// SOLUTION:
//applying equation of continuity and conservation
   of energy between two levels
//speed of water at level 1
v1 = sqrt((2*g*(h*10^-3)*(A2^2))/(A1^2-A2^2))//in m/
   s //taking h in meters
```

```
//volume flow rate
V1 = v1*100//in speed v1 in cm/s
R = A1*V1//in cm^2/s

printf("\n\n Speed of water at level 1 v1 = \n\n %.3f m/s",v1)
printf("\n\n Volume flow rate R = \n\n %2i cm^3/s",R)
```

Scilab code Exa 16.2 C16P2

```
clear
clc
//to find pressure in horizontal pipe and flow speed in pressure in smaller pipe

// GIVEN:
//refer to figure 16-7 on page no. 356
//height of storage tower
h = 32//in m
//diameter of storage tower
D = 3.0//in m
//diameter of horizontal pipe
d = 2.54//in m
//delivery rate of water
R = 0.0025//in m^3/s
//diameter of smaller pipe
d_dash = 1.27//in cm
//distance above the ground for water supply
yC = 7.2//in m
//initial pressure
p0 = 1.01e5//in Pa
//density of water
rho = 1.0e3//in Kg/m^3
//acceleration due to gravity
```

```
g = 9.8//in m/s^2

// SOLUTION:
//area at leval A
A_A = %pi*(1.5)^2//in m^2
//area at leval B
A_B = %pi*(0.0127)^2//in m^2
//area at leval C
A_C = %pi*((d_dash*10^-2)/2)^2//in m^2
//applying equation of continuity
//speed of water at point A
vA = R/A_A//in m/s
//speed of water at point B
vB = R/A_B//in m/s
//applying Bernoulli's equation
//pressure in pipe at B
pB = p0+(rho*g*h)-((1/2)*rho*(vB^2))//in Pa
//applying equation of continuity
//speed of water at point C
vC = R/A_C//in m/s
//take h = yA
//applying Bernoulli's equation
//pressure in pipe at C
pC = p0-((1/2)*rho*(vC^2))+(rho*g*(h-yC))//in Pa
pB = nearfloat("succ",4.03e5)


printf ("\n\n Speed of water at point A vA = \n\n %.1e m/s",vA)
printf ("\n\n Speed of water at point B vB = \n\n %.1f m/s",vB)
printf ("\n\n Pressure in pipe at B pB = \n\n %.2e Pa",pB)
printf ("\n\n Speed of water at point C vC = \n\n %.1f m/s",vC)
printf ("\n\n Pressure in pipe at C pC = \n\n %.2e Pa",pC)
```

Scilab code Exa 16.3 C16P3

```
clear
clc
//to find coefficient of viscocity of castor oil

// GIVEN:
//density of castor oil
rho = 0.96e3//in Kg/m^3
//gauge pressure of pump
delta_p = 950//in Pa
//diameter of pipe
D =  2.6//in cm
//length of pipe
L = 65//in cm
//time interval in which oil is collected
dt = 90//in seconds
//mass of oil collected in dt time interval
dm = 1.23//in Kg
// SOLUTION:
//radius of pipe
R = (D*10^-2)/2//in meters
//mass flux
dm_by_dt = (dm/dt)//in Kg/s
//coefficient of viscocity of castor oil
eta = (rho*%pi*(R^4)*delta_p)/(8*(dm/dt)*(L*10^-2))
    //in N.s/m^2 //taking Lin meters

printf ("\n\n Mass flux dm_by_dt = \n\n %.4f Kg/s",
    dm_by_dt)
printf ("\n\n Coefficient of viscocity of castor oil
     eta = \n\n %.2f N.s/m^2",eta)
```

Chapter 17

OSILLATIONS

Scilab code Exa 17.1 C17P1

```
clear
clc
//to find force constant k of spring
//to find magnitude of horizontal force and period
   of oscillation

// GIVEN:
//refer to figure 17-5 from page no. 375
//mass of boby
M = 1.65//in Kg
//increase in length
y = 7.33//in cm
//mass of block
m = 2.43//in Kg
//distance by which spring is streched
x = 11.6//in cm
//acceleration due to gravity
g = 9.81//in m/s^2

// SOLUTION:
//applying simple harmonic motion equation
```

```
//equating forces in y direction
//force constant k of spring
k = (-M*g)/(-y*10^-2)//in N/m //taking y in meters
//magnitude of horizontal force
F = k*(x*10^-2)//in N //taking x in meters
//period of oscillation
T = (2*%pi*(sqrt(m/k)))*10^3//in miliseconds
k = round(k)

printf ("\n\n Force constant k of spring k = \n\n %3i N/m",k)
printf ("\n\n Magnitude of horizontal force F = \n\n %.1f N",F)
printf ("\n\n Period of oscillation T = \n\n %3i ms",T)
```

Scilab code Exa 17.2 C17P2

```
clear
clc
//to find total energy stored in the system
//to find maximum speed and magnitude of maximum acceleration of block
//to find position,velocity and acceleration of block at t = 0.215s

// GIVEN:
//refer to problem 17-1
//mass of boby
M = 1.65//in Kg
//increase in length
y = 7.33//in cm
//mass of block
m = 2.43//in Kg
//distance by which spring is streched
```

```
x_m = 11.6//in cm
//time
t = 0.215//seconds
//acceleration due to gravity
g = 9.81//in m/s^2

// SOLUTION:
//applying simple harmonic motion equation
//equating forces in y direction
//force constant k of spring
k = (-M*g)/(-y*10^-2)//in N/m //taking y in meters
//total energy stored in the system
E = (1/2)*k*((x_m*10^-2)^2)//in J
//magnitude of kinetic energy
K_max = E//in J
//maximum speed of block
v_max = sqrt((2*K_max)/m)//in m/s
//maximum acceleration of block
a_max = (k*(x_m*10^-2))/m//in m/s^2
//period of oscillation
T = (2*%pi*(sqrt(m/k)))*10^3//in miliseconds
//angular frequency
omega = (2*%pi)/(T*10^-3)//in rad/s
z = omega*t
//position of block at t = 0.215s
x = (x_m*10^-2)*(cos(z))//in m
//velocity of block at t = 0.215s
vx = -(omega*(x_m*10^-2))*(sin(z))//in m/s
//acceleration of block at t = 0.215s
ax = -(omega^2)*x//in m/s^2
omega = nearfloat("succ",9.536)
a_max = nearfloat("succ",10.6)
x = nearfloat("succ",-0.0535)
ax = nearfloat("succ",4.87)

printf ("\n\n Total energy stored in the system E =
    \n\n %.2f J",E)
printf ("\n\n Maximum speed of block v_max = \n\n %
```

```
    .2f m/s",v_max)
printf("\n\n Maximum acceleration of block a_max =
    \n\n %.1f m/s^2",a_max)
printf("\n\n Angular frequency omega = \n\n %.3f
    rad/s",omega)
printf("\n\n Position of block at t = 0.215s x = \n
    \n %.4f m",x)
printf("\n\n Velocity of block at t = 0.215s vx = \
    n\n %.3f m/s",vx)
printf("\n\n acceleration of block at t = 0.215s ax
     = \n\n %.2f m/s^2",ax)
```

Scilab code Exa 17.3 C17P3

```
clear
clc
//to find equation for x(t)

// GIVEN:
//refer to problem 17-1
//mass of boby
M = 1.65//in Kg
//increase in length
y = 7.33//in cm
//mass of block
m = 2.43//in Kg
//distance by which spring is streched
x_m = 11.6//in cm
//displacement of block
x = 0.0624//in meters
//velocity of block
vx = 0.847//in m/s
//acceleration due to gravity
g = 9.81//in m/s^2

```

```
// SOLUTION:
//applying simple harmonic motion equation
//equating forces in y direction
//force constant k of spring
k = (-M*g)/(-y*10^-2)//in N/m //taking y in meters
//total energy of system
E = ((1/2)*m*(vx^2))+((1/2)*k*(x^2))//in J
//maximum amplitude of motion
xm = sqrt((2*E)/k)//in meters
//using cosin equation of x
//value of cos(fi)
cos_fi = x/xm
//phase constant
fi1 = acosd(cos_fi)
fi2 = 360-(fi1)
fi = fi2*(%pi/180)//in rad
//period of oscillation
T = (2*%pi*(sqrt(m/k)))*10^3//in miliseconds
//angular frequency
omega = (2*%pi)/(T*10^-3)//in rad/s
//initial velocity
v_x1 = -(omega*xm)*sind(fi1)//in m/s
v_x2 = -(omega*xm)*sind(fi2)//in m/s
xm = nearfloat("pred",0.1085)
cos_fi = nearfloat("succ",0.5751)
omega = nearfloat("succ",9.54)
fi = nearfloat("succ",5.33)

printf ("\n\n Total energy of system E = \n\n %.3f J
   ",E)
printf ("\n\n Maximum amplitude of motion xm = \n\n
   %.4f m",xm)
printf ("\n\n Value of cos(fi) = \n\n %.4f",cos_fi)
printf ("\n\n Initial velocity = \n\n %.3f for fi =
   %.1f degree \n Or \n %.3f for fi = %.1f degree",
   v_x1,fi1,v_x2,fi2)
printf ("\n\n Equation for x(t) = \n\n (%.3f m)*(cos
   (%.2f rad/s)t + %.2f rad)",xm,omega,fi)
```

Scilab code Exa 17.4 C17P4

```
clear
clc
//to find rotaional inertia of traingle

// GIVEN:
//mass of rod
M = 0.112//in Kg
//length of rod
L = 0.096//in m
//period of oscillations of rod
T_rod = 2.14//in seconds
//period of oscillations of traingular shape body
T_triangle = 5.83//in seconds


// SOLUTION:
//using equation of physical pendulum
//rotational inertia of body
I_rod = (M*L^2)/12//in Kg.m^2
//rotaional inertia of traingle
I_triangle = I_rod*(T_triangle/T_rod)^2//in Kg.m^2

printf ("\n\n Rotational inertia of body I_rod = \n\
    n %.2e Kg.m^2",I_rod)
printf ("\n\n Rotaional inertia of traingle
    I_triangle = \n\n %.2e Kg.m^2",I_triangle)
```

Scilab code Exa 17.6 C17P6

```
clear
clc
//to find value of acceleration due to gravity

// GIVEN:
//radius of disk
R = 10.2//in cm
//period
T = 0.784//in seconds

// SOLUTION
//refer to problem 17-5
//acceleration due to gravity
g = (6*(%pi^2)*(R*10^-2))/(T^2)//in m/s^2

printf ("\n\n Value of acceleration due to gravity g
    = \n\n %.2f m/s^2",g)
```

Scilab code Exa 17.7 C17P7

```
clear
clc
//to find time required by body to come halfway

// GIVEN:
//refer to figure 17-15 from page no. 385
//from the equation given
//radius of reference circle
r = 0.35//in m
//angular speed
omega = 8.3//in rad/s

// SOLUTION
//refer to problem 17-5
//angle turned to come halfway
```

```
wt = 60//in degree
//time required by body to come halfway
t = ((wt*%pi)/180)/omega//in seconds //taking angle
    in radians

printf ("\n\n Angle turned to come halfway wt = \n\n
     %2i degree",wt)
printf ("\n\n Time required by body to come halfway
    t = \n\n %.2f seconds",t)
```

Scilab code Exa 17.8 C17P8

```
clear
clc
//to find periods of oscillations

// GIVEN:
//refer to figure 17-11 from page no. 386
//mass of block
m = 250//in gram
//force constant
k = 85//in N/m
//damping constant
b = 0.070//in Kg/s

// SOLUTION
//using equation of damped oscillatory motion
//for small damping period
T = 2*%pi*(sqrt((m*10^-3)/k))//in seconds //taking m
     in Kg
//periods of oscillations
t = ((m*10^-3)*(log(2)))/b//in seconds //taking m in
     Kg

printf ("\n\n For small damping period T = \n\n %.2f
```

```
   seconds",T)
printf ("\n\n Periods of oscillations t = \n\n %.1f
   seconds",t)
```

Scilab code Exa 17.9 C17P9

```
clear
clc
//to find reduced mass of molecule
//to find effective force constant

// GIVEN:
//refer to figure 17-22 from page no. 390
//mass of hydrogen atom
m1 = 1.007825//in u
//mass of isotop cl-35
m2 = 34.968853//in u
//mass of isotop cl-37
m3 = 36.965903//in u
//vibrational frequency
f = 8.5e13//in Hz
//mass
M = 1.66e-27//in Kg/u

// SOLUTION
//reduced mass of H35cl
m = (m1*m2)/(m1+m2)//in u
//reduced mass of H37cl
m_1 = (m1*m3)/(m1+m3)//in u
//effective force constant
k = 4*(%pi^2)*(f^2)*m_1*M//in N/m

printf ("\n\n Reduced mass of H35cl m = \n\n %.6f u"
   ,m)
printf ("\n\n Reduced mass of H37cl m_1 = \n\n %.6f
```

```
u",m_1)
//answer is slightly different than ans. in book.But
    ans. by scilab program is same as that of
   calculator.
printf ("\n\n Effective force constant k = \n\n %3i
   N/m",k)
```

Chapter 18

WAVE MOTION

Scilab code Exa 18.1 C18P1

```
clear
clc
//to find amplitude,frequency,speed and wave length
   of the wave motion
//to find equation of wave

// GIVEN:
//distance moved up and down
x = 1.30//in cm
//frequency
f = 125//in per second
//wavelength
lambda = 15.6//in cm

// SOLUTION
//using equations of sinusoidal wave motion
//amplitude of wave motion
ym = x/2//in cm
//wave speed
v = (lambda*10^-2)*f//in m/s //taking lambda in
   meters
```

```
//wave number
k = (2*%pi)/(lambda*10^-2)//in rad/m //taking lambda
    in meters
//angular frequency
omega = v*k//in rad/s
omega = nearfloat("succ",786)

printf ("\n\n Amplitude of wave motion ym = \n\n %.2
   f cm",ym)
printf ("\n\n Wave speed v = \n\n %.1f m/s",v)
printf ("\n\n Wave number k = \n\n %.1f rad/m",k)
printf ("\n\n Angular frequency omega = \n\n %3i rad
   /s",omega)
printf ("\n\n Equation of wave is \n\n y(x,t) = (%.2
   f cm)*sin[(%.1f rad/m)x - (%3i rad/s)t] ",ym,k,
   omega)
```

Scilab code Exa 18.2 C18P2

```
clear
clc
//to find expression of velocity and acceleration of
    partical p
//to find displacement,velocity and accleration of
   partical

// GIVEN:
//refer to problem 18-1
//distance moved up and down
x = 1.30//in cm
//frequency
f = 125//in per second
//wavelength
lambda = 15.6//in cm
//location of partical p
```

```
xp = 0.245//in meters
//time
t = 15.0//in ms

// SOLUTION
//using equations of sinusoidal wave motion
//amplitude of wave motion
ym = x/2//in cm
//wave speed
v = (lambda*10^-2)*f//in m/s //taking lambda in meters
//wave number
k = (2*%pi)/(lambda*10^-2)//in rad/m //taking lambda in meters
//angular frequency
omega = v*k//in rad/s
omega = nearfloat("succ",786)
//value of constant
ym_into_omega = ym*omega//in cm/s
k_into_x = k*xp//in rad
omega2_into_ym = (omega^2)*ym//in cm/s^2
//displacement of partical at t
y = (ym)*(sin((k_into_x) - (omega*(t*10^-3))))//in cm/s
//velocity of partical at t
uy = -(ym_into_omega)*(cos((k_into_x) - (omega*(t*10^-3))))//in cm/s
//acceleration of partical at t
ay = -(omega2_into_ym)*(sin((k_into_x) - (omega*(t*10^-3))))//in cm/s^2
ym_into_omega = round(ym_into_omega)

printf ("\n\n Expression of velocity of partical p is \n\n uy(xp,t) = -(%3i cm/s)*cos[(%.2f rad) - (%3i rad/s)t] ",ym_into_omega,k_into_x,omega)
printf ("\n\n Expression of accleration of partical p is \n\n ay(xp,t) = -(%.2e cm/s^2)*sin[(%.2f rad) - (%3i rad/s)t] ",omega2_into_ym,k_into_x,omega
```

```
    )
printf ("\n\n Value of omega^2*ym = \n\n %.2e cm/s^2
    ",omega2_into_ym)
printf ("\n\n Displacement of partical at t y = \n\n
     %.2f cm",y)
//answer of uy is slightly different than book.
    answer of scilab program is same as that of
    calculator.
printf ("\n\n Velocity of partical at t uy = \n\n %
    .2f cm/s",uy)
printf ("\n\n Acceleration of partical at t ay = \n\
    n %.1e cm/s^2",ay)
```

Scilab code Exa 18.3 C18P3

```
clear
clc
//to find amplitude of combined wave
//to find value by which phase difference be changed

// GIVEN:
//amplitude of each wave
ym = 9.7//in mm
//phase difference
fi = 110//in degree

// SOLUTION
//using equations of interference of waves
//amplitude of combined wave
y = 2*ym*(cosd(fi/2))//in mm
//value by which phase difference be changed
delta_fi = 2*(acosd(1/2))//in degree
delta_fi1 = -(delta_fi)//in degree

printf ("\n\n Amplitude of combined wave y = \n\n %
```

```
.1f mm",y)
printf("\n\n Value by which phase difference be changed delta_fi = \n\n %3i degree or %3i degree ",delta_fi,delta_fi1)
```

Scilab code Exa 18.4 C18P4

```
clear
clc
//to find tension in string to get 4 loops

// GIVEN:
//refer to figure 18-23 from page no. 418
//freqyuency
fn = 120//in Hz
//length of string
L = 1.2//in meters
//linear mass density of string
mew = 1.6//in g/m
//no. of loops
n = 4

// SOLUTION
//using equation of wave motion
//tension in string to get 4 loops
F = (4*(L^2)*(fn^2)*(mew*10^-3))/(n^2)//in N //taking mew in Kg/m

printf("\n\n Tension in string to get 4 loops F = \n\n %.1f N",F)
```

Scilab code Exa 18.5 C18P5

```
clear
clc
//to find longest wavelengths of resonance of the string
//to fing corresponding wavelengths that reach the ear of the listener

// GIVEN:
//freqyuency
f = 440//in Hz
//length of string
L = 0.34//in meters
//wave speed in air
v_air = 343//in m/s

// SOLUTION
//using equation of wave for resonance condition
//longest wavelengths of resonance of the string
lambda1 = (2*L)/1//in meters
lambda2 = (2*L)/2//in meters
lambda3 = (2*L)/3//in meters
//wave speed
v_string = f*lambda1//in m/s
//multiplication factor
v_air_by_v_string = (v_air/v_string)
//corresponding wavelengths that reach the ear of the listener
lambda_1 = (lambda1)*(v_air/v_string)//in meters
lambda_2 = (lambda2)*(v_air/v_string)//in meters
lambda_3 = (lambda3)*(v_air/v_string)//in meters

printf ("\n\n Longest wavelengths of resonance of the string \n lamda1 = %.2f m \n lamda2 = %.2f m \n lamda3 = %.2f m ",lambda1,lambda2,lambda3)
printf ("\n\n Wave speed v_string = \n\n %3i m/s ", v_string)
printf ("\n\n Relation between lambda_air and lambda_string is \n\n lambda_air = %.2f(
```

```
    lambda_string) ",v_air_by_v_string)
printf ("\n\n Corresponding wavelengths that reach
    the ear of the listener \n lamda_1 = %.2f m \n
    lamda_2 = %.2f m \n lamda_3 = %.2f m ",lambda_1,
    lambda_2,lambda_3)
```

Chapter 19

SOUND WAVES

Scilab code Exa 19.1 C19P1

```
clear
clc
//to find density and displacement amplitude

// GIVEN:
//maximum pressure variation
delta_pm = 28//in Pa
//frequency
f = 1000//in Hz
//pressure amplitude
delta_p1 = 2.8e-5//in Pa
//bulk modulus of air
B = 1.4e5//in Pa
//speed of sound in air
v = 343//in m/s
//density of air
rho_0 = 1.21//in Kg/m^3

// SOLUTION
//using equation of sound wave
//wave number
```

```
k = (2*%pi*f)/v//in rad/m
//density amplitude
delta_rho_m = delta_pm*(rho_0/B)//in Kg/m^3
//displacement amplitude
s_m = delta_pm/(k*B)//in meters
//for faintest sounds
//density amplitude
delta_rhom = delta_p1*(rho_0/B)//in Kg/m^3
//displacement amplitude
sm = delta_p1/(k*B)//in meters

printf ("\n\n Wave number k = \n\n %.1f rad/m ",k)
printf ("\n\n Density amplitude delta_rho_m = \n\n %
    .1e Kg/m^3 ",delta_rho_m)
printf ("\n\n Displacement amplitude s_m = \n\n %.1e
     m ",s_m)
printf ("\n\n Density amplitude for faintest sounds
    delta_rhom = \n\n %.1e Kg/m^3 ",delta_rhom)
printf ("\n\n Displacement amplitude for faintest
    sounds sm = \n\n %.1e m ",sm)
```

Scilab code Exa 19.2 C19P2

```
clear
clc
//to find intensity  and sound level of sound wave

// GIVEN:
//radiated power
p = 25//in W
//distance from source
r = 2.5//in meters
//intensity of sound having sound level 0 dB
I0 = 1*10^-12//in W/m^2

```

```
// SOLUTION
//using equation of sound wave
//intensity of sound wave
I = p/(4*%pi*r^2)//in W/m^2
//sound level of sound wave
SL = 10*(log10(I/I0))//in dB

printf ("\n\n Intensity of sound wave I = \n\n %.2f W/m^2 ",I)
printf ("\n\n Sound level of sound wave SL = \n\n %3i dB ",SL)
```

Scilab code Exa 19.3 C19P3

```
clear
clc
//to find wavelength for minimum sound intensity

// GIVEN:
//refer figure 19-6 from page no. 433
//distance of listener
r2 = 1.2//in meter
//distance between two speaker
D = 2.3//in meters

// SOLUTION
//using equation of interference of sound wave
//using pythagorean formula
//distance from speaker 1
r1 = sqrt((r2^2)+(D^2))//in meters
//difference between distance from two sources
r1_minus_r2 = r1-r2//in meters
//wavelengths for minimum sound intensity
lambda1 = r1_minus_r2*2//in meters
lambda2 = (r1_minus_r2*2)/3//in meters
```

```
lambda3 = (r1_minus_r2*2)/5//in meters

printf ("\n\n Distance from speaker 1 r1 = \n\n %.1f m ",r1)
printf ("\n\n Difference between distance from two sources r1_minus_r2 = \n\n %.1f m ",r1_minus_r2)
printf ("\n\n Wavelengths for minimum sound intensity \n\n lambda = %.1f m,%.2f m,%.2f m ",lambda1,lambda2,lambda3)
```

Scilab code Exa 19.4 C19P4

```
clear
clc
//to find speed of sound

// GIVEN:
//refer figure 19-8 from page no. 436
//frequeny
f = 1080//in Hz
//distances of water level at resonance
x1 = 6.5//in cm
x2 = 22.2//in cm
x3 = 37.7//in cm

// SOLUTION
//using equation of sound wave for resonance
//from first two resonances
half_lambda = x2-x1//in cm
//from second and third resonance
halflambda = x3-x2//in cm
//average of both lambda values
half_lambda1 = (half_lambda+halflambda)/2//in cm
//wavelength of sound wave
lambda = 2*(half_lambda1)//in cm
```

```
//speed of sound
v = (lambda*10^-2)*f//in m/s //taking lambda in
    meters
v = round(v)

printf ("\n\n From first two resonances half_lambda
    = \n\n %.1f cm ",half_lambda)
printf ("\n\n From second and third resonance
    halflambda = \n\n %.1f cm ",halflambda)
printf ("\n\n Wavelength of sound wave lambda = \n\n
    %.1f cm ",lambda)
printf ("\n\n Speed of sound v = \n\n %3i m/s ",v)
```

Scilab code Exa 19.5 C19P5

```
clear
clc
//to find fundamental frequncy of string
//to find fundamental frequncy of string for first
    overtone
//to find original frequency

// GIVEN:
//refer figure 19-8 from page no. 436
//frequeny
f = 440//in Hz
//frequency of tuning fork
f2 = 3//in Hz
//frequency of tuning fork for first overtone
f3 = 880//in Hz

// SOLUTION
//using equation of sound wave
//fundamental frequncy of string
f1 = f+f2//in Hz
```

```
f_1 = f-f2//in Hz
//frequncy of string for first overtone frequency
f4 = f3+(2*f2)//in Hz
f_4 = f3-(2*f2)//in Hz
//original frequency
f5 = f1//in Hz

printf ("\n\n Fundamental frequncy of string \n\n f1
    = %3i Hz or %3i Hz ",f1,f_1)
printf ("\n\n Frequncy of string for first overtone
    frequency \n\n %3i Hz or %3i Hz ",f4,f_4)
printf ("\n\n Original frequency = \n\n %3i Hz",f5)
```

Scilab code Exa 19.6 C19P6

```
clear
clc
//to find frequency we would perceive

// GIVEN:
//frequency of siren
f = 1125//in Hz
//speed of car
vs = 29//in m/s
//speed of car and your speed
v_0 = 14.5//in m/s
//speed of sound
v = 343//in m/s


// SOLUTION
//using equation of sound wave
//frequency we would perceiv when police car is
    moving
f_dash = f*(v/(v-vs))//in Hz
```

```
f_dash = round(f_dash)
//frequency we would perceiv when your car is moving
v0 = vs//in m/s
fdash = f*((v+v0)/v)//in Hz
//frequency we would perceiv when both police car and your car is moving
v0 = v_0
F_dash = f*((v+v0)/(v-v0))//in Hz
//frequency we would perceiv when your car moving at 9m/s and police car is behind you with 38m/s
v0 = 9//in m/s
vs = 38//in m/s
Fdash = f*((v-v0)/(v-vs))//in Hz
Fdash = round(Fdash)
printf ("\n\n Frequency we would perceiv when police car is moving f_dash = \n\n %4i Hz",f_dash)
printf ("\n\n Frequency we would perceiv when your car is moving fdash = \n\n %4i Hz",fdash)
printf ("\n\n Frequency we would perceiv when both police car and your car is moving F_dash = \n\n %4i Hz",F_dash)
printf ("\n\n Frequency we would perceiv when your car moving at 9m/s and police car is behind you with 38m/s Fdash = \n\n %4i Hz",Fdash)
```

Chapter 20

THE SPECIAL THEORY OF RELATIVITY

Scilab code Exa 20.1 C20P1

```
clear
clc
//to find minimum speed of muon in the Earth's fram
   of reference
//to find minimum speed of muon in the muon's fram
   of reference

//Given:
//refer to figure 20-8(a)and (b) from page no. 457
//lifetime of muon
delta_t0 = 2.2//in microsesonds
//height of atmosphere
L0 = 100//in Km
//speed of light
c = 3.00e8//in m/s

//Solution:
//appiying Einstein's posulates
//in the Earth's fram of reference
```

```
//time of travel
delta_t = (L0*10^3)/c//in microseconds
//minimum speed of muon
u = sqrt((1-((delta_t0/(delta_t)*10^-6)^2)))//in m/s

//in the muon's fram of reference
//height of atmosphere
L = c*(delta_t0*10^-6)//in meters
//minimum speed of muon
u1 = sqrt((1-(((L)/(L0*1000))^2)))//in m/s

printf ("\n\n Time of travel in the Earth fram of
    reference delta_t = \n\n %.2e seconds",delta_t);
printf ("\n\n Minimum speed of muon in the Earth
    fram of reference u = \n\n %.6fc",u);
printf ("\n\n Height of atmosphere in the muon fram
    of reference L = \n\n %3i meters",L);
printf ("\n\n Minimum speed of muon in the muon fram
     of reference u = \n\n %.6fc",u1);
```

Scilab code Exa 20.2 C20P2

```
clear
clc
//to find speed of missile measured by observer on
    the Earth

//Given:
//refer to figure 20-9 from page no. 457
//speed of spaceship
u = 0.80// times c
//speed of missile
v0 = 0.60//times c


```

```
//Solution:
//appiying formule for relativistic addition of
   velocities
//speed of missile measured by observer on Earth
v = (v0+u)/(1+(v0*u))//times c

printf ("\n\n Speed of missile measured by observer
   on the Earth v = \n\n %.2fc" ,v);
```

Scilab code Exa 20.3 C20P3

```
clear
clc
//to find distance between two flashes and time
   between two flashes

//Given:
//seperated distance
delta_x = 2.45//in Km
//time intervel
delta_t = 5.35//in microseconds
//speed of frame S'
u = 0.855//times c
//speed of light
c = 3.00e8//in m/s

//Solution:
//appiying Lorentz transformations
//Lorentz parameters
gama = 1/(sqrt(1-u^2))
//refer to table 20-2
//using interval transformations
//distance between two flashe
delta_x_dash = gama*((delta_x*1000)-(u*c*(delta_t
   *10^-6)))//in meters //taking delta_t in seconds
```

```
and delta_x in meters
//time between two flashes
delta_t_dash = gama*((delta_t*10^-6)-(u*c*(((delta_x*1000))/(c^2))))//in seconds //taking delta_t in seconds and delta_x in meters
delta_t_dash = nearfloat("succ",-3.147e-6)

printf ("\n\n Lorentz parameters gama = \n\n %.3f" , gama);
printf ("\n\n Distance between two flashe delta_x_dash = \n\n %4i meters" ,delta_x_dash);
printf ("\n\n Time between two flashes delta_t_dash = \n\n %.3e seconds" ,delta_t_dash);
```

Scilab code Exa 20.4 C20P4

```
clear
clc
//to find final velocity of particalas measured in the laboratory frame

//Given:
//refer to figure 20-14 from page no. 461
//velocity of partical
vx_dash = 0.60//times c
////velocity of partical w.r.t. frame moving with it
u = 0.60//times c
//speed of light
c = 3.00e8//in m/s

//Solution:
//appiying transformations of velocities
//final velocity of particalas measured in the laboratory frame
vx = (vx_dash+u)/(1+(u*vx_dash))//times c
```

```
printf ("\n\n Final velocity of particalas measured
    in the laboratory frame vx = \n\n %.2fc" ,vx);
```

Scilab code Exa 20.5 C20P5

```
clear
clc
//to find time necessary for the rocket to pass
    particular point
//to find rest length for the rocket
//to find length of D of platform according to
    observer S'
//to find time required for S to pass entire length
    according to observer S'
//to find //time interval between two events
//Given:
//refer to figure 20-19(a),(b),(c) from page no. 465
//llength of platform
L = 65//in meters
//relative speed of rocket
u = 0.80//times c
//speed of light
c = 3.00e8//in m/s

//Solution:
//appiying formule for relativity of length
//time necessary for the rocket to pass particular
    point
delta_t0 = L*10^6/(u*c)//in microseconds
//rest length for the rocket
L0 = L/(sqrt(1-(u^2)))//in meters
//length of D of platform according to observer S'
D0 = L
D = D0*(sqrt(1-(u^2)))//in meters
```

```
D = round(D)
//time required for S to pass entire length
    according to observer S'
delta_t_dash = L0*10^6/(u*c)//in microseconds
//time measured by S and S' usind time dilation
    formula
delta_tdash = delta_t0/(sqrt(1-(u^2)))//in
    microseconds
//refer to table 20-2
//time interval between two events
deltat_dash = -(u*c*(-L))*10^6/((c^2)*(sqrt(1-(u^2))
    ))//in microseconds
//time interval between two events according to S'
deltatdash = (L0-D)*10^6/(u*c)//in microseconds

printf ("\n\n Time necessary for the rocket to pass
    particular point delta_t0 = \n\n %.2f
    microseconds",delta_t0);
printf ("\n\n Rest length for the rocket L0 = \n\n
    %3i meters",L0);
printf ("\n\n Length of D of platform according to
    observer S-dash D = \n\n %2i meters",D);
printf ("\n\n Time required for S to pass entire
    length according to observer S-dash delta_t_dash
    = \n\n %.2f microseconds",delta_t_dash);
printf ("\n\n Time measured by S and S-dash usind
    time dilation formula delta_tdash = \n\n %.2f
    microseconds",delta_tdash);
printf ("\n\n Time interval between two events
    deltat_dash = \n\n %.2f microseconds",
    deltat_dash);
printf ("\n\n Time interval between two events
    according to S-dash deltatdash = \n\n %.2f
    microseconds",deltatdash);
```

Scilab code Exa 20.6 C20P6

```
clear
clc
//to find momentum of proton

//Given:
//speed of proton
v = 0.86//times c
//speed of light
c = 3.00e8//in m/s
//mass of proton
m = 1.67e-27//in Kg

//Solution:
//appiying fomule for relativistic momentum
//momentum of proton
P = (m*v*c)/(sqrt(1-(v^2)))//in Kg.m/s
//value of pc
Pc = P*c*(6.24e12)//in MeV //(6.24e12) is conversion
    factor between J and MeV

printf ("\n\n Momentum of proton P = \n\n %.2e Kg.m/
   s" ,P);
printf ("\n\n Value of pc = \n\n %4i MeV" ,Pc);
printf ("\n\n Momentum of proton p = \n\n %4i MeV/c"
    ,Pc);
```

Scilab code Exa 20.7 C20P7

```
clear
clc
//to find speed of electron as fraction of c and as
   difference from c

```

```
//Given:
//kinetic energy of electron
K = 50//in GeV
//value of mc_square
mc_square = 0.511e-3//in GeV
//speed of light
c = 3.00e8//in m/s

//Solution:
//appiying fomule for relativistic energy
//speed of electron as fraction of c
v = sqrt(1-(1/(1+(K/mc_square)^2)))//times c
//speed of electron as difference from c
c_minus_v = (5.2e-11)*c//in m/s

printf ("\n\n Speed of electron as fraction of c v = \n\n %.12fc" ,v);
printf ("\n\n Speed of electron as difference from c c_minus_v = \n\n %.3f m/s" ,c_minus_v);
```

Scilab code Exa 20.8 C20P8

```
clear
clc
//to find difference between masses of combined ball from sum of masses of original balls

//Given:
//mass of ball
m = 35//in gram
//speed of ball
v = 1.7//in m/s
//speed of light
c = 3.00e8//in m/s
```

```
//Solution:
//appiying fomule for energy and mass in special relativity
//applying conservation of energy
//increase in rest energy
delta_E0 = 2*((1/2)*(m*10^-3)*(v^2))//in J //taking mass in Kg
//increase in mass
delta_m = delta_E0/(c^2)//in Kg

printf ("\n\n Increase in rest energy delta_E0 = \n\n %.3f J",delta_E0);
printf ("\n\n Difference between masses of combined ball from sum of masses of original balls delta_m = \n\n %.1e Kg",delta_m);
```

Scilab code Exa 20.9 C20P9

```
clear
clc
//to find kinetic energy needed to produce Z0

//Given:
//refer to sample problem 20-8
//rest energy
E0 = 91.2//in GeV
//rest energy of electron and positron
E = 0.511//in MeV
//speed of light
c = 3.00e8//in m/s

//Solution:
//appiying fomule for energy and mass in special relativity
```

```
//change in rest energy
delta_E0 = E0-(2*(E*10^-3))//in GeV //coveting E
   into GeV
//applying conservation of energy
//kinetic energy needed to produce Z0
delta_K = -(delta_E0)//in GeV

printf ("\n\n Change in rest energy delta_E0 = \n\n
   %.1f GeV" ,delta_E0);
printf ("\n\n Kinetic energy needed to produce Z0
   delta_K = \n\n %.1f GeV" ,delta_K);
```

Scilab code Exa 20.10 C20P10

```
clear
clc
//to find kinetic energy of each pion

//Given:
//value of mc^2 for Kaon
mk_c_square = 498//in MeV
//kinetic energy of Kaon
K = 325//in MeV
////value of mc^2 for pion
mpi_c_square = 140//in MeV
//speed of light
c = 3.00e8//in m/s

//Solution:
//appiying fomule for coservation of total
   relativistic energy
//applying conservation of energy
//initial total relativistic energy
Ek = K+mk_c_square//in MeV
//total initial momentum
```

```
pk_c = sqrt((Ek^2)-(mk_c_square)^2)//in MeV
//total energy of final system
E = Ek//in MeV
//applying conservation of momentum
//value of p1c
p1c = 668//in MeV
p_1c = -13//in MeV
//kinetic energy of each pion
//kinetic energy of first pion
K1 = (sqrt((p1c^2)+(mpi_c_square^2)))-mpi_c_square//in MeV
//kinetic energy of second pion
K2 = (sqrt((p_1c^2)+(mpi_c_square^2)))-mpi_c_square//in MeV
K1 = round(K1)

printf ("\n\n Initial total relativistic energy Ek = \n\n %3i MeV",Ek);
printf ("\n\n Total initial momentum pk_c = \n\n %3i MeV",pk_c);
printf ("\n\n Total energy of final system E = \n\n %3i MeV",E);
printf ("\n\n Value of p1c = \n\n %3i MeV or %3i MeV",p1c,p_1c);
printf ("\n\n Kinetic energy of first pion K1 = \n\n %3i MeV",K1);
printf ("\n\n Kinetic energy of second pion K2 = \n\n %.1f MeV",K2);
```

Scilab code Exa 20.11 C20P11

```
clear
clc
//to find threshold kinetic energy to produce antiproton
```

```
//Given:
//refer to figure 20-23 from page no. 470
//rest energy of proton
mp_c_square = 938//in MeV
//speed of light
c = 3.00e8//in m/s

//Solution:
//appiying fomule for relativistic momentum
//applying conservation of energy
//value of mpc^2/E1'
mpc_square_by_E1dash = 1/2
//value of v1'/c
v1_dash_by_c = sqrt(1-(mpc_square_by_E1dash)^2)
//refer to table 20-3
//speed of incident proton
v_dash = v1_dash_by_c//times c
u = v1_dash_by_c//times c
v = (v_dash+u)/(1+(v1_dash_by_c)^2)//times c
//total energy of incident proton
E = 1/(sqrt(1-(v^2)))//times mp_c_square
E = round(E)
//threshold kinetic energy to produce antiproton
K = (E*mp_c_square)-mp_c_square//in MeV

printf ("\n\n Value of v1_dash/c = \n\n %.3f" ,
    v1_dash_by_c);
printf ("\n\n Speed of incident proton v = \n\n %.3
    fc" ,v);
printf ("\n\n Total energy of incident proton E = \n
    \n %1imp_c_square ",E);
printf ("\n\n Threshold kinetic energy to produce
    antiproton K = \n\n %4i MeV",K);
```

Chapter 21

TEMPERATURE

Scilab code Exa 21.1 C21P1

```
clear
clc
//to find temperature measured by thermometer

//Given:
//factor by which resistance is increased
R_by_Rtr = 1.392
//temperature of triple point of water
Ttr = 273.16//in K

//Solution:
//using formula for measuring temperatures
//temperature measured by thermometer
T_R = Ttr*R_by_Rtr//in K

printf ("\n\n Temperature measured by thermometer
    T_R = \n\n %.1f K" ,T_R);
```

Scilab code Exa 21.2 C21P2

```
clear
clc
//to find maximum temperature variation allowable
   during ruling

//Given:
//refer to table 21-3
//accuracy for milimeter interval
delta_L = 5e-5//in mm
//coefficient of linear expansion
alpha = 11e-6//in per degree celsius
//consider length of steel
L = 1//in mm

//Solution:
//using formula for temperature expansion
//maximum temperature variation allowable during
   ruling
delta_T = delta_L/(alpha*L)//in degree celsius

printf ("\n\n Maximum temperature variation
   allowable during ruling delta_T = \n\n %.1f
   degree celsius",delta_T);
```

Scilab code Exa 21.3 C21P3

```
clear
clc
//to find final pressure of gas

//Given:
//refer to figure 21-13 from page 488
//initial temperature of oxygen
Ti = 20//in degree celsius
//initial pressure of oxygen
```

```
pi = 15//in atm
//initial volume of oxygen
vi = 22//in liters
//final temperature of oxygen
Tf = 25//in degree celsius
//final volume of oxygen
vf = 16//in liters

//Solution:
//consider oxygen as ideal gas and applying
   equations of ideal gas
//final pressure of gas
pf = pi*((Tf+273)/(Ti+273))*(vi/vf)//in atm //taking
    temp. in kelvin
pf = round(pf)

printf ("\n\n Final pressure of gas pf = \n\n %2i
   atm" ,pf);
```

Chapter 22

MOLECULAR PROPERTIES OF GASES

Scilab code Exa 22.1 C22P1

```
clear
clc
//to find root mean square speed of hydrogen
   molecule

//Given:
//pressure
p = 1//in atm
//density of hydrogen
rho = 8.99e-2//in Kg/m^3

//Solution:
//assume hydron as ideal gas
//applying formula of root mean square speed for
   ideal gas
//root mean square speed of hydrogen molecule
vrms = sqrt((3*p*1.01e5)/(rho))//in m/s //taking
   pressure in Pa

```

```
//answer of vrms is slightly different than book
   answer.But ans. by scilab program is same as that
    of calculator
printf ("\n\n Root mean square speed of hydrogen
   molecule vrms = \n\n %4i m/s" ,vrms);
```

Scilab code Exa 22.2 C22P2

```
clear
clc
//to find number of moles of oxygen
//to find number  of molecules of oxygen
//to find approximate rate at which oxygen molecule
   strike one face of the box

//Given:
//refer to figure 22-2 from page no. 499
//length of edge of cubical box
L = 10//in cm
//pressure of oxygen
p = 1.0//in atm
//temperature of oxygen
T = 300//in K
//molar gas constant
R = 8.31//in J/mol.K
//Avogadro constant
NA = 6.02e23//in molecules/mol

//Solution:
////assumong oxygen as ideal gas
//applying ideal gas equations
//volume of box
V = ((L*10^-2)^3)//in m^3
//number  of moles of oxygen
n = ((p*1.01*10^5)*V)/(R*T)//taking p into Pa
```

```
//number  of molecules of oxygen
N = n*NA
N = nearfloat("succ",2.5e22)
//refer to table 22-1
//root mean square speed of oxygen
vrms = 483//in m/s
//approximate rate at which oxygen molecule strike
    one face of the box
Rate = (N*vrms)/(6*(L*10^-2))//in collisions/s

printf ("\n\n Number  of moles of oxygen n = \n\n %
    .3f mol" ,n);
printf ("\n\n Number  of molecules of oxygen N = \n\
    n %.1e molecules" ,N);
printf ("\n\n Root mean square speed of oxygen vrms
    = \n\n %3i m/s" ,vrms);
printf ("\n\n Approximate rate at which oxygen
    molecule strike one face of the box Rate = \n\n %
    .1e collisions/s" ,Rate);
```

Scilab code Exa 22.3 C22P3

```
clear
clc
//to find ratio of rms speed of gas molecules
    containing 235-U and gas molecules containing
    238-U
//to find relative abundance of gas molecules
    containing 235-U
//to find number of times gas molecule should be
    passed through barrier

//Given:
//abundance of 235-U
a1 = 0.7//in percentage
```

```
//abundance of 238-U
a2 = 99.3//in percentage
//final abundance of 235-U
a3 = 3//in percentage

//Solution:
//applying equations for root mean square speed
//molecular mass of 235-U
m_235 = 235+6*(19)//in u
//molecular mass of 238-U
m_238 = 238+6*(19)//in u
//ratio of rms speed of gas molecules containing 235-U and gas molecules containing 238-U
vrms_235_by_vrms_238 = sqrt(m_238/m_235)
//ratio of abundances
r = a1/a2
//relative abundance of gas molecules containing 235-U
ratio_1_pass = r*vrms_235_by_vrms_238
//isotope ratio
i = (a3)/(100-(a3))
//number of times gas molecule should be passed through barrier
n = (log(i/r))/(log(vrms_235_by_vrms_238))

printf ("\n\n Molecular mass of 235-U m_235 = \n\n %3i u",m_235);
printf ("\n\n Molecular mass of 238-U m_238 = \n\n %3i u",m_238);
printf ("\n\n Ratio of rms speed of gas molecules containing 235-U and gas molecules containing 238-U vrms_235_by_vrms_238 = \n\n %.4f", vrms_235_by_vrms_238);
printf ("\n\n Ratio of abundances = \n\n %.5f",r);
printf ("\n\n Relative abundance of gas molecules containing 235-U ratio_1_pass = \n\n %.5f", ratio_1_pass);
printf ("\n\n Isotope ratio = \n\n %.5f",i);
```

```
//answer of n slightly changes than book.But answer by scilab answer is same as that of answer by calculator
printf ("\n\n Number of times gas molecule should be passed through barrier = \n\n %3i" ,n);
```

Scilab code Exa 22.4 C22P4

```
clear
clc
//to find mean free path and average collision rate of nytrogen at room temperature

//Given:
//room temperature
T = 300//in K
//atmospheric pressure
p = 1.01e5//in Pa
//effective diameter of nytrogen
d = 3.15e-10//in meters
//average speed
vav = 478//in m/s
//Boltzmann constant
k = 1.38e-23//in J/K

//Solution:
//applying formula of mean path
//mean free path of nytrogen at room temperature
lambda = (k*T)/(sqrt(2)*%pi*(d^2)*p)//in meters
//average collision rate of nytrogen at room temperature
rate = vav/lambda//in collisions/second

printf ("\n\n Mean free path of nytrogen at room temperature lambda = \n\n %.1e meters" ,lambda);
```

```
printf ("\n\n Average collision rate of nytrogen at
   room temperature rate = \n\n %.1e collisions/
   second" ,rate);
```

Scilab code Exa 22.5 C22P5

```
clear
clc
//to find average speed,root-mean speed,root-mean
   square speed and most probable speed of particals

//Given:
//number of particals
N = 10
//speed of particals
v1 = 0.0//in m/s
v2 = 1.0//in m/s
v3 = 2.0//in m/s
v4 = 3.0//in m/s
v5 = 3.0//in m/s
v6 = 3.0//in m/s
v7 = 4.0//in m/s
v8 = 4.0//in m/s
v9 = 5.0//in m/s
v10 = 6.0//in m/s

//Solution:
//applying formula for average speed
//average speed of particals
vav = (1/N)*(v1+v2+v3+v4+v5+v6+v7+v8+v9+v10)//in m/s
//applying formula for root-mean speed
//root-mean speed of particals
v_square_av = (1/N)*(v1^2+v2^2+v3^2+v4^2+v5^2+v6^2+
   v7^2+v8^2+v9^2+v10^2)//in m^2/s^2
//applying formula for root-mean square speed
```

```
//root-mean square speed of particals
vrms = sqrt(v_square_av)//in m/s
//most probable speed of particals
//taking into consideration all speeds of particals
vp = v4//in m/s
printf ("\n\n Average speed of particals vav = \n\n
   %.1f m/s" ,vav);
printf ("\n\n Root-mean speed of particals
   v_square_av = \n\n %.1f m^2/s^2" ,v_square_av);
printf ("\n\n Root-mean square speed of particals
   vrms = \n\n %.1f m/s" ,vrms);
printf ("\n\n Most probable speed of particals vp =
   \n\n %.1f m/s" ,vp);
```

Scilab code Exa 22.6 C22P6

```
clear
clc
//to find fraction of molecules having speed in
   range 599-601m/s

//Given:
//temperature
T = 300//in K
//molar mass of oxygen
M = 0.032//in Kg/mol
//molar gas constant
R = 8.31//in J/mol.K
//velocity
v = 600//in m/s

//Solution:
//fraction of molecules having speed in range
   599-601m/s
//difference in speed
```

```
dv = 2//in m/s
f = 4*%pi*((M/(2*%pi*R*T))^(3/2))*(v^2)*%e^((-M*(v
    ^2)/(2*R*T)))*dv
f1 = f*100//in percent

printf ("\n\n Fraction of molecules having speed in
    range 599-601m/s f = \n\n %.1e" ,f);
printf ("\n\n Percentage of molecules having speed
    in range 599-601m/s f = \n\n %.2f percent" ,f1);
```

Scilab code Exa 22.7 C22P7

```
clear
clc
//to find most probable speed,average speed,root-
    mean square speed of oxygen

//Given:
//temperature
T = 300//in K
//molar gas constant
R = 8.31//in J/mol.K
//molar mass
M = 0.032//in Kg/mol

//Solution:
//applying formula for most probable speed
//most probable speed of oxygen
vp = sqrt((2*R*T)/(M))//in m/s
//applying formula for average speed
//average speed of oxygen
vav = sqrt((8*R*T)/(%pi*M))//in m/s
//applying formula for root-mean square speed
//root-mean square speed of oxygen
vrms = sqrt((3*R*T)/(M))//in m/s
```

```
vp = round(vp)
a = vav/vp
a1 = vrms/vp
a1 = nearfloat("succ",1.225)

printf ("\n\n Most probable speed of oxygen vp = \n\n %3i m/s" ,vp);
printf ("\n\n Average speed of oxygen vav = \n\n %3i m/s" ,vav);
printf ("\n\n Root-mean square speed of oxygen vrms = \n\n %3i m/s" ,vrms);
printf ("\n\n For any gas vp:vav:vrms = 1:%.3f:%.3f" ,a,a1);
```

Scilab code Exa 22.9 C22P9

```
clear
clc
//to find pressure according to ideal gas law
//to find pressure according to van der Waals equations

//Given:
//for qxygen van der Waals coefficients
a = 0.138//in J.m^3/mol^2
b = 3.18e-5//in m^3/mol
//number mol of oxygen
n = 1//in mol
//volume of box
V = 0.0224//in m^3
//molar gas constant
R = 8.31//in J/mol.K
//molar mass
M = 0.032//in Kg/mol
//temperature
```

```
T = 50//in K

//Solution:
//applying ideal gas equation
//pressure according to ideal gas law
p = (n*R*T)/V//IN Pa
//applying van der Waals equations
//pressure according to van der Waals equations
P = ((n*R*T)/(V-(n*b)) )-((a*n^2)/V^2)//in Pa

printf ("\n\n Pressure according to ideal gas law p
    = \n\n %.2e Pa" ,p);
printf ("\n\n Pressure according to van der Waals
    equations P = \n\n %.2e Pa" ,P);
```

Chapter 23

THE FIRST LAW OF THERMODYNAMICS

Scilab code Exa 23.2 C23P2

```
clear
clc
//to find rate of heat energy pass through the
   insulation
//to find additional insulation required to reduce
   heat transfer rate by half

//Given:
//refer to figure 23-6 from page no. 520
////temperature of steam
TS = 100//in degree celsius
//diameter of pipe
d = 5.4//in cm
//thickness of insulation
t = 5.2//in cm
//length of pipe
D = 6.2//in meters
//temperature of room
TR = 11//in degree celsius
```

```
//thermal conductivity
k = 0.048//in W/m.K

//Solution:
//radius of cylinder
r1 = d/2//in cm
//radius of cylinder with insulation
r2 = r1+t//in cm
//applying fourier's law of heat conduction
//rate of heat energy pass through the insulation
H = (2*%pi*k*D*(TS-TR))/(log(r2/r1))//in W
//additional insulation required to reduce heat
    transfer rate by half
r2_dash = (r2^2)/r1//in cm

printf ("\n\n Rate of heat energy pass through the
    insulation H = \n\n %3i W" ,H);
printf ("\n\n Additional insulation required to
    reduce heat transfer rate by half r2_dash = \n\n
    %2i cm" ,r2_dash);
```

Scilab code Exa 23.3 C23P3

```
clear
clc
//to find final equilibrium temperature of the
    system

//Given:
//mass of copper cube
mc = 75//in gram
//temperature of oven
T0 = 312//in degree celsius
//mass of water
mw = 220//in gram
```

```
//heat capacity of beaker
Cb = 190//in J/K
//intial temperature of water and beaker
Ti = 12.0//in degree celsius
//heat capacity of water
Cw = 4190//in J/Kg.K
//heat capacity of copper cube
Cc = 387//in J/Kg.K

//Solution:
//applying laws of thermodynamics
//for equilibrium condition
//final equilibrium temperature of the system
Tf = (((mw*10^-3)*Cw*Ti)+(Cb*Ti)+((mc*10^-3)*Cc*T0))/(((mw*10^-3)*Cw)+(Cb)+((mc*10^-3)*Cc))//in degree celsius //taking masses in Kg
//heat transfer for water
Qw = (mw*10^-3)*Cw*(Tf-Ti)//in J
//heat transfer for beaker
Qb = Cb*(Tf-Ti)//in J
//heat transfer for copper
Qc = (mc*10^-3)*Cc*(Tf-T0)//in J
Qw = nearfloat("pred",7011)
Qb = nearfloat("pred",1441)
Qc = nearfloat("pred",-8450)

printf ("\n\n Final equilibrium temperature of the system Tf = \n\n %.1f degree celsius" ,Tf);
printf ("\n\n Heat transfer for water Qw = \n\n %4i J" ,Qw);
printf ("\n\n Heat transfer for beaker Qb = \n\n %4i J" ,Qb);
printf ("\n\n Heat transfer for copper Qc = \n\n %4i J" ,Qc);
```

Scilab code Exa 23.4 C23P4

```
clear
clc
//to find work done by three different paths

//Given:
//refer to figure 23-17 from page no. 529
//final volume
vf = 1.0//in m^3
//initial volume
vi = 4.0//in m^3
//final pressure
pf = 40//in Pa
//initialvolume
pi = 10//in Pa

//Solution:
//applying laws of thermodynamics
//work done by constant pressure in path 1
W = -pi*(vf-vi)//in J
//work done in constant volume in path 1
w = 0//in J
//work done by path 1
W1 = W+w//in J
//work done by path 2
W2 = -pi*vi*(log(vf/vi))//in J
//work done by path 3
W3 = 0-(pf*(vf-vi))//in J

printf ("\n\n Work done by constant pressure in path
    1 W = \n\n %2i J" ,W);
printf ("\n\n Work done by path 1 W1 = \n\n %2i J" ,
    W1);
printf ("\n\n Work done by path 2 W2 = \n\n %2i J" ,
    W2);
printf ("\n\n Work done by path 3 W3 = \n\n %2i J" ,
    W3);
```

Scilab code Exa 23.5 C23P5

```
clear
clc
//to find speed of sound in the gas

//Given:
//room temperature
T = 20//in degree celsius
//parameter gama for air
gama = 1.4
//molar gas constant
R = 8.31//in J/mol.K
//molar mass for air
M = 0.0290//in Kg/mol

//Solution:
//applying laws of thermodynamics
//speed of sound in the gas
v = sqrt((gama*R*(T+273))/M)//in m/s
v = round(v)

printf ("\n\n Speed of sound in the gas v = \n\n %3i
    m/s" ,v);
```

Scilab code Exa 23.6 C23P6

```
clear
clc
//to find time required for room temperature to be
   21 degree celsius
```

```
//Given:
//room temperature
T = 0//in degree celsius
//length of room
l = 6//in meters
//breadth of room
b = 4//in meters
//height of room
h = 3//in meters
//power of heater
p = 2//in KW
//final air temperature
T1 = 21//in degree celsius

//Solution:
//applying laws of thermodynamics
//volume of room
V = (l*b*h)*1000//in L
//number of moles of gas
n = V/22.4//in mol //since 1 mol occupies 22.4L of volume
//refer to table 23-4
//molar heat capacity
Cv = 20.8//in J/mol.K
//using relation of heat capacity
//absorbtion of heat take place
Q = n*Cv*(T1-T)//in J
//time required for room temperature to be 21 degree celsius
t = Q/(p*10^3)//in seconds //taking power in W
t = nearfloat("pred",701)

printf ("\n\n Volume of room V = \n\n %5i L" ,V);
printf ("\n\n Number of moles of gas n  = \n\n %.1e mol" ,n);
printf ("\n\n Absorbtion of heat take place Q = \n\n %.1e J" ,Q);
```

```
printf ("\n\n Time required for room temperature to be 21 degree celsius t = \n\n %3i seconds" ,t);
```

Scilab code Exa 23.7 C23P7

```
clear
clc
//to find change in internal energy

//Given:
//refer to figure 23-17 from page no. 529
//refer to problem 23-4
//final volume
vf = 1.0//in m^3
//initial volume
vi = 4.0//in m^3
//initialvolume
pi = 10//in Pa
//value of constant for monoatomic gas
gama = 1.66
//number of moles of ideal gas
n = 0.11//in mol
//molar gas constant
R = 8.31//in J/mol.K

//Solution:
//applying laws of thermodynamics
//applying adiabatic relationship
//final pressure of gas
pf = (pi*(vi^gama))/(vf^gama)//in Pa
//initial temperature of gas
Ti = (pi*vi)/(n*R)//in K
//final temperature of gas
Tf = (pf*vf)/(n*R)//in K
//applying internal energy formula
```

```
//change in internal energy
delta_Eint = (3/2)*(n*R*(Tf-Ti))//in J
pf = round(pf)
Ti = round(Ti)

printf ("\n\n Final pressure of gas pf = \n\n %3i Pa
    " ,pf);
printf ("\n\n Initial temperature of gas Ti = \n\n
    %2i K" ,Ti);
printf ("\n\n Final temperature of gas Tf = \n\n %3i
     K" ,Tf);
printf ("\n\n Change in internal energy delta_Eint =
     \n\n %2i J" ,delta_Eint);
```

Scilab code Exa 23.8 C23P8

```
clear
clc
//to find work done on the system
//to find heat added to the system
//to find change in internal energy of the system

//Given:
//refer to figure 23-23 from page no. 535
//mass of water
m = 1.00//in Kg
//initial volume of liquid
vi = 1.00e-3//in m^3
//final volume of steam
vf = 1.671//in m^3
//atmospheric pressure
p = 1.01e5//in Pa
//molar gas constant
R = 8.31//in J/mol.K

```

```
//Solution:
//applying laws of thermodynamics
//applying constant pressure relationship
//work done on the system
W = (-p*(vf-vi))//in KJ
//latent heat of vaporization
L = 2256//in KJ/Kg
//heat added to the system
Q = L*m//in KJ
//change in internal energy of the system
delta_Eint = Q+W//in KJ

printf ("\n\n Work done on the system W = \n\n %.2e J" ,W);
//answer of Q and delta_Eint slightli changes.But answer by scilab program is same as that of calculator answer
printf ("\n\n Heat added to the system Q = \n\n %4i KJ" ,Q);
printf ("\n\n Change in internal energy of the system delta_Eint = \n\n %4i KJ" ,delta_Eint);
```

Scilab code Exa 23.9 C23P9

```
clear
clc
//to find work done on the system
//to find heat added to the system
//to find change in internal energy of the system

//Given:
//refer to figure 23-21 from page no. 534
//number of moles
n = 0.75//in mol
//pressures at corresponding points
```

```
PA = 3.2e3//in Pa
PB = 1.2e3//in Pa
//volume at corresponding point
VA = 0.21//in m^3
//molar gas constant
R = 8.31//in J/mol.K
//value of constants
Cv = 20.8//in J/mol.K
Cp = 29.1//in J/mol.K

//Solution:
//applying laws of thermodynamics
//using ideal gas law
//temperature at A
TA = (PA*VA)/(n*R)//in K
////temperature at B
TB = (PB*VA)/(n*R)//in K //since VA=VB
//volume at C
VC = (n*R*TA)/(PB)//in m^3 //since TC = TA and PC = PB
//during process A-B
//applying constant volume relationship
//heat added to the system
//redefining TA AND TB
TA = 108//in K
TB = 40//in K
Q1 = n*Cv*(TB-TA)//in J
//work done on the system
W1 = 0//in J
//change in internal energy of the system
delta_Eint1 = Q1+W1//in J

//during process B-C
//applying constant pressure relationship
//heat added to the system
Q2 = n*Cp*(TA-TB)//in J //since TC = TA
//work done on the system
W2 = -PB*(VC-VA)//in J //since VB = VA
```

```
//change in internal energy of the system
delta_Eint2 = Q2+W2//in J

//during process C-A
//applying isothermal relationship
//work done on the system
W3 = -n*R*TA*(log(VA/VC))//in J
//change in internal energy of the system
delta_Eint3 = 0//in J
//heat added to the system
Q3 = delta_Eint3-W3 //in J
//delta_Eint1 = nearfloat("succ",-1061)
//Q2 = nearfloat("succ",1480)
//delta_Eint2 = nearfloat("succ",1060)
//W3 = nearfloat("succ",660)
//Q3 = nearfloat("succ",-661)
//totol work done during process
W = W1+W2+W3//in J
//total change in internal energy during process
delta_Eint = delta_Eint1+delta_Eint2+delta_Eint3//in J
TA = round(TA)
//value of Q2,delta_Eint2 ,delta_E slightly varies than book.But answer by scilab is same as that of calculator answer

printf ("\n\n Temperature at A TA = \n\n %3i K" ,TA);
printf ("\n\n Temperature at B TB = \n\n %3i K" ,TB);
printf ("\n\n Volume at C VC = \n\n %.2f m^3" ,VC);
printf ("\n\n During process A-B");
printf ("\n\n Heat added to the system Q1 = \n\n %4i J" ,Q1);
printf ("\n\n Work done on the system W1 = \n\n %3i J" ,W1);
printf ("\n\n Change in internal energy of the system delta_Eint1 = \n\n %4i J" ,delta_Eint1);
```

```
printf("\n\n During process B-C");
printf("\n\n Heat added to the system Q2 = \n\n %4i J",Q2);
printf("\n\n Work done on the system W2 = \n\n %3i J",W2);
printf("\n\n Change in internal energy of the system delta_Eint2 = \n\n %4i J",delta_Eint2);
printf("\n\n During process C-A");
printf("\n\n Heat added to the system Q3 = \n\n %4i J",Q3);
printf("\n\n Work done on the system W3 = \n\n %3i J",W3);
printf("\n\n Change in internal energy of the system delta_Eint3 = \n\n %4i J",delta_Eint3);
printf("\n\n Totol work done during process W = \n\n %3i J",W);
printf("\n\n Total change in internal energy during process delta_Eint = \n\n %4i J",delta_Eint);
```

Chapter 24

ENTROPY AND THE SECOND LAW OF THERMODYNAMICS

Scilab code Exa 24.1 C24P1

```
clear
clc
//to find entropy change of water during process

//Given:
//mass of water
m = 1.8//in Kg
//initial temperature of water and hot plate
Ti = 20//in degree celsius
//final temperature of hot plate
Tf = 100//in degree celsius
//heat capacity of water
c = 4190//in J/Kg.K

//Solution:
//applying laws of thermodynamics
//applying formula for entropy change
```

```
//entropy change of water during process
delta_S = m*c*(log((Tf+273)/(Ti+273)))//in J/K //
    taking temperatures in K

printf ("\n\n Entropy change of water during process
     delta_S = \n\n %4i J/K" ,delta_S);
```

Scilab code Exa 24.2 C24P2

```
clear
clc
//to find temperature rise of system water+stone
//to find entropy change of system
//to find entropy change of reverse process

//Given:
//refer to figure 24-1 from page no. 548
//mass of stone
ms = 1.5//in Kg
//mass of water
mw = 4.5//in Kg
//vertical height
h = 2.5//in meters
//initial temperature of water and stone
T = 300//in K
//specific heat capacity of water
cw = 4190//in J/Kg.K
//specific heat capacity of stone material
cs = 790//in J/Kg.K
//acceleration due to gravity
g = 9.8//in m/s^2

//Solution:
//applying laws of thermodynamics
//applying formula for entropy change for
```

```
irreversible process
//heat transfer
Q = mw*g*h//in J
//temperature rise of system water+stone
delta_T = Q/((mw*cw)+(ms*cs))//in K
//entropy change of system
delta_S = Q/(T)//on J/K
//entropy change of reverse process
delta_s = -Q/T//in J/k //since heat is extracted from system

printf ("\n\n Heat transfer Q = \n\n %3i J" ,Q);
printf ("\n\n Temperature rise of system water+stone delta_T = \n\n %.1e K" ,delta_T);
printf ("\n\n Entropy change of system delta_S = \n\n %.2f J/k" ,delta_S);
printf ("\n\n Entropy change of reverse process delta_s = \n\n %.2f J/K" ,delta_s);
```

Scilab code Exa 24.3 C24P3

```
clear
clc
//to find net entropy change of irreverse process

//Given:
//refer to figure 24-3(a)and (b) from page no. 549
//mass of hot water
m = 0.57//in Kg
//initial temperature of hot water
TiH = 363//in K
//initial temperature of cold water
TiC = 283//in K
//equilibrium temperature
Tf = 323//in K
```

```
//specific heat capacity of water
c = 4190//in J/Kg.K

//Solution:
//applying laws of thermodynamics
//applying formula for entropy change for
   irreversible process
//entropy change of hot water
delta_SH = m*c*log(Tf/TiH)//in J/K
//entropy change of cold water
delta_SC = m*c*log(Tf/TiC)//in J/K
//net entropy change of irreverse process
delta_S = delta_SH+delta_SC//in J/K
delta_SH  = round(delta_SH)
delta_SC  = round(delta_SC)
delta_S = round(delta_S)

printf ("\n\n Entropy change of hot water delta_SH =
    \n\n %3i J/K" ,delta_SH);
printf ("\n\n Entropy change of cold water delta_SC
   = \n\n %3i J/K" ,delta_SC);
printf ("\n\n Net entropy change of irreverse
   process delta_S = \n\n %3i J/K" ,delta_S);
```

Scilab code Exa 24.4 C24P4

```
clear
clc
//to find net entropy change of the gas for
   irreverse process

//Given:
//refer to figure 24-5(a)and (b) from page no. 550
//number of moles
n = 0.55//in mol
```

```
//room temperature
T = 293//in K
//molar gas constant
R = 8.31//in J/mol.K

//Solution:
//applying laws of thermodynamics
//applying formula for entropy change for isothermal
    expansion
//ratio of final to initial volumes //since both
   chamber are of same volumes
Vf_by_Vi = 2
//entropy change of the gas for irreverse process
delta_S = n*R*log(Vf_by_Vi)//in J/K

printf ("\n\n Ratio of final to initial volumes VF/
   Vi = \n\n %1i" ,Vf_by_Vi);
printf ("\n\n Entropy change of the gas for
   irreverse process delta_S = \n\n %.2f J/K" ,
   delta_S);
```

Scilab code Exa 24.5 C24P5

```
clear
clc
//to find maximum possible efficiency of turbine

//Given:
//temperature of steam in boiler
TH = 520//in degree celsius
////temperature of steam in condenser
TL = 100//in degree celsius

//Solution:
//applying laws of thermodynamics
```

```
//applying formula for carnot cycle
//maximum possible efficiency of turbine
Emax = 1-((TL+273)/(TH+273))
Emax1 = Emax*100//in percent
Emax1 = round(Emax1)

printf("\n\n Maximum possible efficiency of turbine
    Emax = \n\n %.2f",Emax);
printf("\n\n Maximum possible efficiency of turbine
    Emax = \n\n %2i percent",Emax1);
```

Scilab code Exa 24.6 C24P6

```
clear
clc
//to find work per cycle required to operate
   refrigerator
//to find heat per cycle discharged to the room

//Given:
//coefficient of performance of refrigerator
K = 4.7
//rate of heat extraction
QL = 250//in J/cycle

//Solution:
//applying laws of thermodynamics
//applying formula for refrigeration cycle
//work per cycle required to operate refrigerator
W = QL/K//in J/cycle
//heat per cycle discharged to the room
QH = W+QL//in J/cycle

printf("\n\n Work per cycle required to operate
   refrigerator W = \n\n %3i J/cycle",W);
```

```
printf ("\n\n Heat per cycle discharged to the room QH = \n\n %3i J/cycle",QH);
```

Scilab code Exa 24.7 C24P7

```
clear
clc
//to find minimum rate of energy to be supplied to the heat pump

//Given:
//outside temperature
TL = -10//in degree celsius
//interior temperature
TH = 22//in degree celsius
//heat transfer
QH = 16//in KW

//Solution:
//applying laws of thermodynamics
//applying formula for refrigeration cycle
//coefficient of performance
K = (TL+273)/((TH+273)-(TL+273))//taking temperature in K
//minimum rate of energy to be supplied to the heat pump
W_by_deltat = QH/(K+1)//in KW

printf ("\n\n Coefficient of performance K = \n\n %.2f",K);
printf ("\n\n Minimum rate of energy to be supplied to the heat pump W_by_deltat = \n\n %.1f KW",W_by_deltat);
```

Scilab code Exa 24.8 C24P8

```
clear
clc
//to find heat energy extracted from high
   temperature reservior per cycle
//to find heat energy discharge to low temperature
   reservior per cycle
//to find entropy change per cycle

//Given:
//work output
W = 120//in J per cycle
//efficiency
Ex = 75//in percent
//boiling point of water
TH = 100//in degree celsius
//freezing point of water
TL = 0//in degree celsius

//Solution:
//applying laws of thermodynamics
//applying formula for refrigeration cycle
//applying carnot cycle formula
//efficiency of carnot engine
Ec = 1-((273+TL)/(TH+273))//taking temperatures in K
Ec1 = Ec*100//in percent
//heat energy extracted from high temperature
   reservior per cycle
QH = W/(Ex*10^-2)//in J
//heat energy discharge to low temperature reservior
    per cycle
QL = QH-W//in J
delta_SH = -(QH)/(TH+273)//in J/K //taking
```

```
   temperatures in K
delta_SL = (QL)/(TL+273)//in J/K //taking
   temperatures in K
delta_SWS = 0//in J/K
//entropy change per cycle
delta_Sx = delta_SH+delta_SL+delta_SWS//in J/K
Ec1 = round(Ec1)

printf ("\n\n Efficiency of carnot engine Ec = \n\n
   %.3f" ,Ec);
printf ("\n\n Efficiency of carnot engine Ec = \n\n
   %2i percent" ,Ec1);
printf ("\n\n Heat energy extracted from high
   temperature reservior per cycle QH = \n\n %3i J"
   ,QH);
printf ("\n\n Heat energy discharge to low
   temperature reservior per cycle QL = \n\n %3i J"
   ,QL);
printf ("\n\n Entropy change per cycle delta_Sx = \n
   \n %.2f J/K" ,delta_Sx);
```

Scilab code Exa 24.9 C24P9

```
clear
clc
//to find number of independent ways
//to find number of microstates

//Given:
//number of molecules
N = 200//in molecules
//half number of molecules
N1 = 100//in molecules
//for 150 molecules in one box and 50 molecules in
   one box
```

```
n1 = 150
n2 = 50

//Solution:
//number of independent ways
w = factorial(N)/((factorial(N1))*(factorial(N1)))
//number of microstates
W = factorial(N)/((factorial(n1))*(factorial(n2)))
//answer is Nan.Because function factorial in scilab overflows as soon as N > 170 as here numerator is  N = 200 and answer of denominator is infinity

printf ("\n\n Number of independent ways w = \n\n %.2e" ,w);
printf ("\n\n Number of microstates W = \n\n %.2e" , W);
```