Scilab Textbook Companion for A Textbook Of Engineering Mechanics by R.S. Khurmi[1]

Created by
Shubham Kumar Lala
B.TECH
Computer Engineering
VELLORE INSTITUTE OF TECHNOLOGY, CHENNAI CAMPUS
College Teacher
None
Cross-Checked by
None

January 20, 2020

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

Book Description

Title: A Textbook Of Engineering Mechanics

Author: R.S. Khurmi

Publisher: S. Chand & Company Ltd., Ram Nagar, New Delhi

Edition: 20

Year: 2014

ISBN: 8121926165

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

Exa Example (Solved example)

Eqn Equation (Particular equation of the above book)

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

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

Contents

List of Scilab Codes

Chapter 2

Composition and Resolution of Forces

Scilab code Exa 2.1 To find the resultant of two forces

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 F1 = 100;       //(N(newton))
6 F2 = 150;       //(N(newton))
7 theta = 45;          //(degree)
8
9 //R = (F1^2 + F2^2 + 2*F1*F2*cosX)^(1/2))
10 Resultant = ((F1^2 + F2^2 + 2*F1*F2*cosd(theta))
    ^(1/2));
11 printf("Resultant Force = %.2f N",Resultant);....//
    The answers vary due to round off error
```

Scilab code Exa 2.2 To find the smaller force

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  AOC = 120;              //(degrees)
6  F1 = 40;               //(N)(bigger force)
7  BOC = 90;             //(degrees)(angle between
       resultant and smaller force(F2))
8  //From geometry
9  AOB = AOC - BOC;        //(degrees)
10
11 //Also, tan(AOB) = (F2*sin(AOC))/(F1 + F2*cos(AOC))
12 //After simplification
13 //F2 = (tand(AOB)*F1)/(sind(AOC) - tand(AOB)*cosd(
       AOC))
14 F2 = (tand(AOB)*F1)/(sind(AOC) - tand(AOB)*cosd(AOC)
       )
15
16 printf("Smaller force = %.2f N",F2);
```

Scilab code Exa 2.3 To find the magnitude of two force

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  //Let F1 an F2 be two forces
6  //If two force act at right angles:
7  //Resultant = (F1^2 + F2^2)^(1/2)
8  //(F1)^2 + (F2)^2 = 10;
9
10 //when angle between two forces is 60 degrees
11 //Resultant = (F1^2 + F2^2 + 2*F1*F2*cosd(60))*(1/2)
12 //Simplifying:
13 //F1*F2 = 3;
```

```
14
15 //X = (F1 + F2)^2 = 16;
16 X = 16;
17 //Y = (F1 - F2)^2 = 4;
18 Y = 4;
19
20 x = X^(1/2);
21 y = Y^(1/2);
22
23 //2*F1 = x + y;
24 F1 = (x + y)/2;
25 F2 = (x - y)/2;
26
27 printf("F1 = %.2f N\n",F1);
28 printf("F2 = %.2f N",F2);
```

Scilab code Exa 2.4 To calculate tensions in the ropes

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 W = 1000;          //(newton)(weight)
6
7 //Resolving the forces horizontally and equating the
    same;
8 //T1*cosd(60) - T2*cosd(45) = 0....(i)
9
10 //Resolvong the forces vertically:
11 ///T1*sind(60) + T2*sind(45) - 1000 = 0....(ii)
12
13 //From (i) and (ii):
14 A = [cosd(60), -cosd(45); sind(60), sind(45)];    //
    Coefficient matrix
15 C = [0; -1000];    //Constant term matrix
```

```
16
17 [x,nsA] = linsolve(A,C);
18 T1 = x(1);
19 T2 = x(2);
20
21 printf("T1 = %.2f N\n",T1);     //The answers vary
      due to round off error
22 printf("T2 = %.2f N",T2);     //The answers vary due
      to round off error
```

Scilab code Exa 2.5 To calculate the magnitude of resultant

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 //From figure....
6 ABC = 90;        //(degrees)
7 BC = 50;         //(newton)
8 AB = 40;         //(newton)
9 CA = 30;         //(newton)
10
11 theta = asind(CA/BC);
12
13 //Resolving the forces horizontally;
14 H = AB - CA*cosd(theta);              //(newton)
15 V = BC - CA*sind(theta);              //(newton)
16
17 Resultant = sqrt(H^2 + V^2);
18 printf("Resultant Force = %.2f N",Resultant);
      //The answers vary due to round off error
```

Scilab code Exa 2.6 To find the magnitude and direction of force

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
F1 = 50;        //(kN)
F2 = 25;        //(kN)
F3 = 35;        //(kN)
F4 = 20;        //(kN)

//Resolving forces horizontally:
H = F2 - F4;            //(kN)

//Resolving the forces vertically:
V = (-F1) + (-F3);      //(kN)

//Magnitude of resultant force:
Resultant = (H^2 + V^2)^(1/2);

printf("Resultant Force = %.2f kN\n",Resultant);

//Direction of resultant force:
//Let theta = angle of resultant with horizontal

//tan(theta) = V/H;
theta = atand(abs(V/H));

//Since H is positive and V is negative, therefore
   resultant lies between 270 degrees and 360
   degrees.
//Actual angle of resultant force:
Angle = 360 - theta;
printf("Angle of Resultant force = %.2f degrees",
   Angle);      //The answers vary due to round off
   error
```

Scilab code Exa 2.7 To find the magnitude and direction of the resultant force

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5
6  F1 = 60;          //(N)
7  F2 = 50;          //(N)
8  F3 = 40;          //(N)
9  F4 = 30;          //(N)
10 F5 = 20;          //(N)
11 alpha = 30;       //(degrees)(Angle between two
       adjacent forces)
12
13 //Resolving all the forces horizontally:
14 H = F5*cosd(0) + F4*cosd(alpha) + F3*cosd(2*alpha) +
       F2*cosd(3*alpha) + F1*cosd(4*alpha);
15
16 //Resolving all the forces vertically:
17 V = F5*sind(0) + F4*sind(alpha) + F3*sind(2*alpha) +
       F2*sind(3*alpha) + F1*sind(4*alpha);
18
19 Resultant = (H^2 + V^2)^(1/2);
20 printf("Resultant force = %.2f N\n",Resultant);
21
22 //Let theta = angle of resultant with horizontal
23 //tan(theta) = V/H;
24 theta = atand(abs(V/H));
25
26 //Since both H and V are positive, therefore actual
       angle of resultant lies between 0 and 90.
27 printf("Angle of resultant force = %.2f degrees",
       theta);
```

Scilab code Exa 2.8 To find the magnitude and direction of the resultant force

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5
6 F1 = 35;          //(N)
7 F2 = 30;          //(N)
8 F3 = 25;          //(N)
9 F4 = 20;          //(N)
10
11 alpha = 30;      //(degrees)
12 theta = 45;      //(degrees)
13 beta = 40;       //(degrees)
14
15 //Resolving the forces horizontally:
16 H = F4*cosd(alpha) + F3*cosd(90) + F2*cosd(180-theta
      ) + F1*cosd(180+beta);              //(N)
17
18 //Resolving the forces vertically:
19 V = F4*sind(alpha) + F3*sind(90) + F2*sind(180-theta
      ) + F1*sind(180+beta);         //(N)
20
21 Resultant = (H^2 + V^2)^(1/2);
22 printf("Resultant force = %.2f N\n",Resultant);
23
24 //Let theta = angle of resultant with horizontal;
25 //tan(theta) = V/H;
26 theta = atand(abs(V/H));
27
28 //Since H is negative and V positive, therefore the
      resultant lies between 90 degrees and 180 degrees
      .
29
30 Angle = 180 - theta;
31 printf("Angle of resultant = %.2f degrees",Angle);
```

Scilab code Exa 2.9 To find the magnitude direction and position of the resultant

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 FP = 1000;     //(N)(Force at P)
6 FQ = 1500;     //(N)(Force at Q)
7 FR = 1000;     //(N)(Force at R)
8 FS = 500;      //(N)(Force at S)
9
10 alpha = 90;    //(degrees)(Angle at P)
11 beta = 60;     //(degrees)(Angle at Q)
12 phi = 45;      //(degrees)(Angle at R)
13 psi = 30;    //(degrees)(Angle at S)
14
15 //Resolving all the forces horizontally:
16 H = FP*cosd(alpha) + FQ*cosd(beta) + FR*cosd(phi) +
      FS*cosd(psi);              //(N)
17
18 //Resolving all the forces vertically:
19 V = FP*sind(alpha) + FQ*sind(beta) + FR*sind(phi) +
      FS*sind(psi);               //(N)
20
21 Resultant = (H^2 + V^2)^(1/2);                //(N)
22 printf("Magnitude of resultant = %.2f N\n",Resultant
      );      //The answers vary due to round off error
23
24 //Let theta be the angle of the resultant with PS.
25 //tan(theta) = V/H;
26 theta = atand(abs(V/H));
27 printf("Angle of resultant = %.2f degrees\n",theta);
28
29 //Since both H and V are positive the resultant lies
```

```
    between 0 degree and 9 degrees.
30
31 //Let x = distance between P and the line of action
    of the resultant force.
32 //Taking moments of the vertical components of the
    forcesand the resultant about P, and equating the
    same:
33
34 //3256 * x = (FP*0) + (FQ*0.866) + (FR*0.707)*8 + (
    FS * 0.5)*12 = 13852;
35 x = (FP*sind(alpha) + FQ*sind(beta)*4 + FR*sind(phi)
    *8 + FS*sind(psi)*12)/V;
36 printf("Position of the resultant = %.2f m",x);
    //(The answers vary due to round off error)
```

Chapter 3

Moments and their Applications

Scilab code Exa 3.1 To find the moment of the force

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 P = 15;           //(N)(Force applied)
6 l = 0.8;          //(m)(width of the door)
7 theta = 60;       //(degrees)(Angle of the force with
     horizontal)
8
9 //Moment when the force acts perpendicular to the
     door;
10 M1 = P * l;             //(N-m)
11 printf("Moment of the force about the hinge = %.2f N
     -m\n",M1);
12
13 //Moment when the force acts at an angle of 60
     degrees to the door;
14 //From geometry, we find that the perpendicular
     distance between the line of action of the force
```

```
    and hinge
15 OC = l * sind(theta);          //(m)
16 M2 = P * OC;                  //(N-m)
17
18 printf("Moment when the force acts at an angle of 60
     degrees to the door = %.2f N-m",M2);      //The
    answers vary due to round off error
```

Scilab code Exa 3.2 To find the maximum weight

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 w = 30;           //(N)(Weight of the plank)
6 L = 2;            //(m)(Length of the plank)
7 d = 1.4;          //(m)(Distance between end A and a
    point B on the plank)
8
9 //If the plank is not to topple, then the reaction
    at A should be zero for the maximum weight at C.
    Now taking moments about B and equating the same.
10 //30 * 0.4 = w * 0.6
11 x = L - d;       //(m)(Distance between B and C)
12 y = 0.4;         //(m)(Distance between B and the
    force)
13 W = (w * y)/(x);
14 printf("Maximum weight = %.2f N",W);
```

Scilab code Exa 3.4 To find the least pull and reaction on the block

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
```

```
clc
clear all
d = 600;          //(mm)(Diameter of wheel)
OR = 300;         //(mm)(Radius of the wheel)
OB = 150;         //(mm)(Height of the wheel)
w = 5;            //(kN)(Weight of the wheel)


//Let P = least pull required just to turn the wheel
    in kN
//From geometry:
//sin(theta) = 150/300;
theta = asind(OB/OR);

AB = sqrt(OR^2 - OB^2);          //(mm)

//Taking moments about A and equating the same.
//P * 300 = 5 * 260
P = (w * AB)/(OR);               //(kN)
printf("Least pull = %.2f kN\n",P);

//Let R = reaction on the block in kN;
//Resolving the forces horizontally and equating the
    same:
//R*cosd(30) = P*sind(30)
R = (P * sind(theta))/cosd(theta);          //(kN)
printf("Reaction on the block = %.2f kN",R);
```

Scilab code Exa 3.7 To find the magnitude direction and position of the resultant

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
BC = 20;     //(cm)
```

```
6  BE = 10;      //(cm)
7  FBC = 12;     //(kN)
8  FAB = 7;      //(kN)
9  FEC = 8;      //(kN)
10 FDE = 6;      //(kN)
11 FCA = 9;      //(kN)
12 FBD = 5;      //(kN)
13
14 //Magnitude of the resultant force:
15 alpha = atand(BC/BE);           //Angle BEC = alpha
16
17 //Resolving all the forces horizontally:
18 H = FEC*sind(alpha) + FBC + FBD*sind(45) - FCA*sind
      (45) - FDE*sind(alpha);              //(kN)
19
20 //Resolving all the forces vertically:
21 V = FAB + FEC*cosd(alpha) - FBD*cosd(45) - FCA*cosd
      (45) + FDE*cosd(alpha);              //(kN)
22
23 Resultant = sqrt(H^2 + V^2);        //(N)
24 printf("Magnitude of the resultant force = %.2f kN\n
      ",Resultant);
25
26 //Direction of resultant force:
27 //Let theta = Angle, which the resultant force makes
       with the horizontal:
28 theta = atand(V/H);
29 printf("Angle of resultant with the horizontal = %.2
      f degrees\n",theta);
30
31 //Position of  the resultant force:
32 //l = Perpendicular distance between the point E and
       the line of action of the resultant force.
33 //Taking moments about E and equating the same:
34 x = ((FAB*0) + (FEC*0) + (FBC*BE) + (FBD*sind(45)) +
       (FCA*sind(45)) + (FDE*0))/(Resultant);
35 printf("Perpendicular distance between the point E
      and the line of action of the resultant force = %
```

```
.2f cm",x);      //The answers vary due to round
    off error
```

Scilab code Exa 3.8 To find the magnitude and direction of the resultant force

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  FAB = 4;     //(kN)
6  FAD = 10;    //(kN)
7  FAC = 8;     //(kN)
8  //Resultant force in magnitude and direction:
9  //Resolving the forces horizontally:
10 H = FAB + FAC*cosd(45);          //(units)
11
12 //Resolving the forces vertically:
13 V = -FAD + (-FAC*cosd(45));      //(units)
14
15 R = sqrt(H^2 + V^2);      //(units)
16 printf("Magnitude of the resultant force = %.2f
       units\n",R);     //The answers vary due to round
       off error
17
18 //Let theta = Angle  which the resultant force makes
       with the horizontal:
19 theta = atand(abs(V/H));          //(degrees)
20
21 //Since, H is positive and V is negative, therefore
       the resultant lies between 270 degrees and 360
       degrees;
22 Angle = 360 - theta;        //(degrees)
23 printf("Angle of resultant = %.2f degrees\n",Angle);
          //The answers vary due to round off error
24
```

```
25 //Magnitude and sense of force along AJ and JH:
26 //P1 = force along AH and,
27 //P2 = force along AJ.
28
29 //From geometry:
30 X1 = atand(0.5/1);              //(degrees)
31 X2 = atand(0.5/1);             //(degrees)
32
33 //Resolving the forces P1 and P2 horizontally:
34 //P1*cosd(X1) + P2*sind(X2) - 9.656 = 0;     (1)
35
36 //Resolving the forces P1 and P2 vertically and
      equating them with the vertical component of thhe
       resultant force:
37 //P1*sind(X1) + P2*cosd(X2) - 15.656 = 0;    (2)
38
39 //Solving equation (1) and (2):
40 A = [cosd(X1), sind(X2); sind(X1), cosd(X2)];     //(
      Coefficient matrix)
41 C = [-9.656; -15.656];     //(Constant matrix)
42 [x,nsA]=linsolve(A,C);
43 P1 = x(1);
44 P2 = x(2);
45
46 printf("Value of P1 = %.2f units and value of P2 = %
      .2f units",P1,P2);     //(The answers vary due to
      round off error)
```

Scilab code Exa 3.9 To find the tension

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 W = 150;          //(N)(Weight of signal arm)
```

```
d1 = 175;        //(mm)(Distance between F1 and 150N)
d2 = 75;         //(mm)(Distance between F1 and B)

//First of all, taking moment about the fulcrum(F1)
   of the signal and equating the same;
//P * 75 = 150 * 175;
P = (W * d1)/d2;              //(N)

//Now, taking moment about the fulcrum(F2) of the
   operating wire and equating the same;
//T * 75 = 350 * 150;
T = (P * W)/d2;             //(N)
printf("Required tension = %.2f N",T);
```

Scilab code Exa 3.11 To find the tension

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
W = 0.2;           //(N/mm)(Weight of lever)
P = 100;           //(N)(Force applied on the effort
   arm)
a = 200;           //(mm)(Length of the effort arm)
b = 100;           //(mm)(Length of the load arm)

WAB = a * W;        //(N)(Weight of lever AB)

//Taking moments about the hinge B and equating the
   same:
//T * 100 = (100 * 200) + (40 * 100);
T = ((P * a) + (WAB * b))/(b);          //(N)
printf("Required tension = %.2f N",T);       //The
   answer provided in the textbook is wrong
```

Scilab code Exa 3.12 To find the magnitude and direction of reaction

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 VF = 200;          //(N)(Vertical force at C)
6 HF = 300;          //(N)(Horizontal force at C)
7
8 AB = 10;           //(cm)
9 BC = 12;           //(cm)
10
11 alpha = 30;       //(degrees)
12 beta = 40;        //(degrees)
13 phi = 60;         //(degrees)
14 //Magnitude of the force(P):
15 //Taking moments about the hinge B and equating the
      same:
16 //P * 10*sind(60) = 200 * 12*cosd(30) + 300 * 12*
      cosd(60);
17 P = (VF * BC*cosd(alpha) + HF * BC*cosd(phi))/(AB*
      sind(phi));                //(N)
18 printf("Magnitude of the force = %.2f N\n",P);
19
20 //Magnitude of the reaction at B:
21
22 //Resolving the forces horizontally:
23 H = HF + P*cosd(20);                //(N)
24
25 //Resolving the forces vertically:
26 V = VF - P*sind(20);                //(N)
27
28 Reaction = (H^2 + V^2)^(1/2);              //(N)
29 printf("Magnitude of reaction at B = %.2f N\n",
```

```
    Reaction);
30
31 //Direction of reaction at B:
32 theta = atand(V/H);              //(degrees)
33
34 //SInce both the values of H and V are positive,
    therefore resultant lies between 0 degrees and 90
     degrees:
35
36 printf("Direction of reaction at B = %.2f degrees",
    theta);
```

Scilab code Exa 3.13 To find the value of W

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 P = 100;          //(N)(Effort)
6 d1 = 175;         //(Distance between A and F1)
7 d2 = 275;         //(Distance between C and D)
8 d3 = 25;
9 d4 = 125;         //(Distance between B and C)
10
11 Lup = d1/d3;              //leverage of the upper
    lever AB
12 Llow = (d2 + d3)/d3;      //leverage of the lower
    lever CF2
13
14 TotalLeverage = Lup * Llow;
15
16 //Total Leverage = W/P;
17 W = [P * TotalLeverage]/1000;          //(kN)
18 printf("W = %.2f kN",W);
```

Chapter 4

Parallel Forces and Couples

Scilab code Exa 4.1 To find the magnitude of the resultant force

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 FA = 50;     //(N)(Force at A)
6 FB = 100;    //(N)(Force at B)
7 AB = 360;    //(mm)(Length of rod)
8 //Magnitude of the resultant force:
9 //Since the given forces are like and parallel,
     therefore magnitude of the resultant force:
10 R = FA + FB;           //(N)
11 printf("Magnitude of the resultant force = %.2f N\n"
     ,R);
12
13 //Point where the resultant force acts
14 //Let x = Distance between the line of action of the
      resultant force (R) and A (i.e., AC) in mm;
15 //Taling clockwise and anticlockwise moments of the
     forces about C and equating the same;
16 x = (FB * (AB))/(FA + FB);
17 printf("Distance between the line of action of the
```

```
resultant force (R) and A (i.e., AC) = %.2f mm",x
);
```

Scilab code Exa 4.2 To find the distance of the body

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5
6 FD = 200;      //(N)(Force at D)
7 AB = 3;        //(m)
8 MT = 350;      //(N)(Maximum tension)
9 FC = 400;      //(N)(Force at C)
10
11 //x = Distance between the body of weight 200 N and
   support A.
12 //Now taking clockwise and anticlockwise moments
   about B and equating the same;
13 //350 * 3 = 200 * (3 - x) + 400 * 1.5;
14 x = ((FD*AB + FC*AB/2)-(MT*AB))/FD;            //(m)
15 printf("Distance between the body of weight 200 N
   and support A = %.2f m",x);
```

Scilab code Exa 4.3 To find the magnitude and point of the resultant force

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5
6 FA = 400;        //(N)(Force at A)
7 FB = 100;        //(N)(Force at B)
```

```
AB = 150;          //(mm)
//Magnitude of the resultant force:
//Since the forces are unlike and parallel, therefore magnitude of the resultant force;
R = FA - FB;               //(N)
printf("Magnitude of the resultant force = %.2f N\n",R);

//x = Distance between the lines of action of the resultant force and A in mm.
//Taking clockwise and anticlockwise moments about A and equating the same;
//300 * x = 100 * 150;
x = (FB*AB)/R;          //(mm)
printf("Distance between the lines of action of the resultant force and A = %.2f mm",x);
```

Scilab code Exa 4.4 To find the point of beam support

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
W = 100;       //(N)(Weight of rod AB)
l = 6;         //(mm)(Length of rod AB)
w1 = 60;       //(N)(Weight of the body supported at A)
w2 = 80;       //(N)(Weight of the body supported at B)

//Let x = Distance between B and the point where the beam should be supported;
//Taking moments of the weights about D and equating the same;
//80 * x = 60 * (6 - x) + 100 * (3 - x);
```

```
13 x = (w1*l + W*l/2)/(w2+w1+W);                    //(m)
14 printf("Distance between B and the point where the
      beam should be supported = %.2f m",x);
```

Scilab code Exa 4.7 To find the unknown values of P and Q

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  FAC = 100;      //(N)(Force along AC)
6  FBC = 200;      //(N)(Force along BC)
7  FAB = 100;      //(N)(Force along AB)
8  AB = 1;         //(m)
9  alpha = 45;     //(Degrees)
10
11 //Resolving the forces horizontally:
12 //100 - 100 * cosd(45) - P = 0
13 P = FAB - FAC*cosd(alpha);          //(N)
14 printf("Value of P = %.2f N\n",P);      //The answers
      vary due to round off error
15
16 //Resolving the forces vertically:
17 //200 - 100 * sin(45) - Q = 0
18 Q = FBC - (FAC * sind(alpha));       //(N)
19 printf("Value of Q = %.2f N\n",Q);     //The answers
      vary due to round off error
20
21 //Magnitude of the couple:
22 //Taking moments about A:
23 M = (-FBC * AB) + (-P * AB);          //(N-m)
24 printf("Magnitude of the couple = %.2f N-m",M);
      //The answers vary due to round off error
```

Scilab code Exa 4.10 To find the weight supported by each man

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5
6 AB = 2.5;      //(m)(Length of machine component)
7 H = 1;         //(m)(Height of the component)
8 theta = 30;       //(degrees)(Inclination)
9 W = 100;       //(N)(Weight of the component)
10
11 //P = Weight supported by the man at A
12 //Q = Weight supported by the man at B
13 //C = Point where the vertical line through the
      centre of gravity cuts the lower face
14 //Joining G(centre of gravity) with M(mid-point of
      AB)
15
16 //From geometry:
17 GM = 1/2;         //(m)
18 AM = 2.5/2;       //(m)
19 AC = AM - GM * tand(theta);
20 CB = AB - AC;    //(m)
21
22 //P + Q = 100                (1)
23 //And P * CA = Q * CB        (2)
24 //From (1) and (2):
25 P = (W*CB)/(CB+AC);                  //(N)
26 Q = W - P;              //(N)
27
28 printf("Weight supported by the man at A = %.2f N\n"
      ,P);     //(The answers vary due to round off
      error)
```

```
29 printf("Weight supported by the man at B = %.2f N",Q
   );
```

Chapter 5

Equilibrium of Forces

Scilab code Exa 5.1 To find the force in the strings

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 W = 15;     //(N)(Weight at C)
6 BAC = 60;          //(degrees)
7 CBX = 45;          //(degrees)(Exterior angle at B)
8 CBA = 90 - CBX;   //(degrees)
9 BCA = 180 - BAC - CBA
10 alpha = 180 - CBX;     //(degrees)(Angle between TBC
     and 15 N)
11 theta = 360 - alpha - BCA;     //(degrees)(Angle
     between TAC and 15 N)
12
13 delta = 360 - (alpha + theta);     //(Angle between
     TAC and TBC)
14
15 //Let TAC = Force in the string AC, and
16 //TBC = Force in the string BC.
17 //Applying Lami's equation at C,
18 //   15/sind(75) = TAC/sind(135) = TBC/sind(150)
```

```
19 TAC = (W * sind(alpha))/sind(delta);
20 TBC = (W * sind(theta))/sind(delta);
21
22 printf("Force in the string AC = %.2f N\n",TAC);
23 printf("Force in the string BC = %.2f N",TBC);
```

Scilab code Exa 5.2 To find the tension in the strings

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 FB = 1000;     //(N)(Force at joint B)
6 FC = 1000;     //(N)(Force at joint C)
7 alpha = 30;    //(degrees)
8 beta = 60;     //(degrees)
9 theta = 120;   //(degrees)
10
11 //Let TAB = Tension in the portion AB of the string,
12 //TBC = Tension in the portion BC of the string, and
13 //TCD = Tension in the portion CD of the string
14
15 //Applying Lami's equation at joint B:
16 // TAB/sind(60) = TBC/sind(150) = 1000/sind(150)
17
18 TAB = (FC * sind(beta))/sind(alpha + theta);       //
   (N)
19 TBC = (FC * sind(alpha))/sind(alpha);        //(N)
20
21 printf("Tension in the portion AB of the string = %
   .2f N\n",TAB);     //(The answers vary due to
   round off error)
22 printf("Tension in the portion BC of the string = %
   .2f N\n",TBC);
23
```

```
24 //Applying Lami's equation at joint C:
25 //TBC/sind(120) = TCD/sind(120) = 1000/sind(120)
26 TCD = (FB * sind(theta))/sind(theta);      //(N)
27
28 printf("Tension in the portion CD of the string = %
      .2f N",TCD);
```

Scilab code Exa 5.3 To find the tension and magnitudes of W1 and W2

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  //(i) Tensions in the portion AB, BC and CD of the
      string:
6  //TAB = Tension in the portion AB,
7  //TBC = Tension in the portion BC.
8  //TCD = TDE = 300 N
9  TCD = 300;          //(N)
10 alpha = 150;        //(degrees)
11 beta = 120;         //(degrees)
12
13 //Applying Lami's equation at C:
14 // TBC/sind(150) = W2/sind(120) = 300/sind(90)
15 TBC = TCD * sind(alpha);           //(N)
16 W2 = TCD * sind(beta);            //(N)
17
18 //Applying Lami's equation at B:
19 //TAB/sind(90) = W1/sind(60\\150) = TBC/sind(120)
20 TAB = TBC/sind(beta);             //(N)
21 W1 = (TBC * sind(alpha))/sind(beta);      //(N)
22
23 //(i)
24 printf("Tension in the portion AB = %.2f N\n",TAB);
25 printf("Tension in the portion BC = %.2f N\n",TBC);
```

```
26 printf("Tension in the portion CD = %.2f N\n",TCD);
27
28 //(ii)
29 printf("Magnitude of W1 = %.2f N\n",W1);
30 printf("Magnitude of W2 = %.2f N",W2);
```

Scilab code Exa 5.5 To find the reaction at the surface of contact

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 W = 100;     //(N)(Weight of cylinder)
6 alpha = 40;     //(degrees)
7 beta = 15;      //(degrees)
8
9 //Let RA = Reaction at A,
10 //RB = Reaction at B.
11 //Applying Lami's equation at O,
12 // RA/sind(180 - 40) = RB/sind(180 - 15) = 100/sind
   (15 + 40)
13 RA = (W * sind(alpha))/sind(alpha + beta);       //(N
   )
14 RB = (W * sind(beta))/sind(alpha + beta);       //(N)
15
16 printf("Reaction at A = %.2f N\n",RA);      //The
   answers vary due to round off error
17 printf("Reaction at B = %.2f N",RB);
```

Scilab code Exa 5.6 To find the pressure at the points of contact

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
```

```
3  clc
4  clear all
5  WP = 200;              //(N)(Weight of the cylinder P)
6  WQ = 500;              //(N)(Weight of the cylinder Q)
7  ED = 100/2;            //(Radius of cylinder P)
8  BF = 180/2;            //(Radius of cylinder Q)
9  BCF = 60;              //(degrees)
10 CF = BF*cotd(BCF);     //(mm)
11 FE = 180 - (CF + ED);     //(mm)
12 BG = FE;
13 AB = ED + BF;          //(mm)
14 ABG = acosd(BG/AB);    //(degrees)
15 //Applying Lami's equation at A:
16 //R1/sind(90 + ABG) = R2/sind(90) = WP/sind(180 -
      ABG)
17 R1 = (WP * cosd(ABG))/sind(ABG);      //(N)
18 R2 = WP/sind(ABG);                     //(N)
19
20 printf("Reaction of the cylinder P at the vertical
      side = %.2f N\n",R1);      //The answers vary due
      to round off error
21 printf("Reaction of the cylinder P at the point of
      contact with the cylinder Q = %.2f N\n",R2);
      //The answers vary due to round off error
22
23 //Applying Lami's equation at B:
24 //R3/sind(90 + ABG) = R2/sind(60) = (R4 - WQ)/sind
      (180 + 30 - 56.1)
25 R3 = (R2 * cosd(ABG))/sind(BCF);       //(N)
26 R4 = WQ + (R2 * sind(180 + 30 - ABG))/sind(BCF);
      //(N)
27
28 printf("Reaction of the cylinder Q on the inclined
      surface = %.2f N\n",R3);     //The answers vary
      due to round off error
29 printf("Reaction of the cylinder Q on the base of
      the channel = %.2f N",R4);     //The answers vary
      due to round off error
```

Scilab code Exa 5.7 To determine the pressures at different portions

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 W = 100;     //(N)(Weight of cylinder)
6 alpha = 38.7;     //(degrees)(Angle POS)
7 beta = 77.4;      //(degrees)(Angle between R1 and R2
    )
8 //(i)
9 //Applying Lami's equation at O:
10 //R1/sind(141.3) = R2/sind(141.3) = 100/sind(77.4)
11 R1 = (W * sind(alpha))/sind(beta);      //(N)
12 R2 = (W * sind(alpha))/sind(beta);      //(N)
13
14 printf("Reaction R1 of the cylinder B on the
    cylinder A = %.2f N\n",R1);      //The answers
    vary due to round off error
15 printf("Reaction R1 of the cylinder C on the
    cylinder A = %.2f N\n",R2);      //The answers
    vary due to round off error
16
17 //(ii)
18 //Applying Lami's equation at P:
19 //64/sind(90) = R3/sind(180 - 38.7) = (R4 - 100)/
    sind(9 + 38.7)
20 R4 = W + R2 * cosd(alpha);        //(N)
21 printf("Reaction of the cylinder B on the base of
    the channel = %.2f N\n",R4);      //The answers
    vary due to round off error
22
23 //(iii)
24 R3 = R2 * sind(alpha);         //(N)
```

```
25 printf("Reaction of the cylinder B on the vertical
     side of the channel = %.2f N",R3);     //The
     answers vary due to round off error
```

Scilab code Exa 5.9 To find the inclination of the rod

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 //From geometry:
6 //tan(X) = (r * (2*cos(X) - 1.5))/2*r*sin(X)
7 //Solving:
8 //4*cos(X)^2 - 1.5*cos(X) - 2 = 0
9 //Let cos(X) = x
10 a = 4;     //(coefficient of cos(X)^2)
11 b = -1.5; //(coefficient of cos(x))
12 c = -2;    //(constant term)
13 x = poly(0,"x");
14 p = a*x^2 + b*x + c;
15 x = roots(p);
16 x = real(x(1));
17 theta = acosd(x);
18 printf("The inclination of the rod with the
     horizontal = %.2f degrees",theta);     //Answer in
      textbook is wrong
```

Scilab code Exa 5.10 To find the forces in the members of the crane

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
```

```
FB = 250;      //(kN)(Force at B)
BC = 20;       //(m)
BE = BC;       //(m)
CE = 10;       //(m)
alpha = asind((CE/2)/BC);      //(degrees)
BD = sqrt(BC^2 - (CE/2)^2);    //(m)
DF = 8;                        //(m)
AB = 25;       //(m)
beta = asind(DF/BD);           //(degrees)
BF = sqrt(BD^2 - DF^2);        //(m)
ABF = acosd(BF/AB);            //(degrees)
theta = ABF - beta;

//Applying Lami's equation at B:
//T/sind(180 - 24.4) = 1.936*P/sind(45.1) = 250/sind
   (180 - 20.7)
T = (FB * sind(beta))/sind(theta);              //(kN)
P = (FB * sind(ABF))/(2*cosd(alpha) * sind(theta));
   //(kN)

printf("Force in the member AB = %.2f kN\n",T);
   //The answers vary due to round off error
printf("Force in each members BC and BE = %.2f kN",P
   );
```

Scilab code Exa 5.13 To determine the reactions

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
P = 4;              //(kN)
W = 2;              //(kN)
a = 3.0;            //(m)
b = 0.9;            //(m)
```

```
c = 1.8;            //(m)

//Using the conditions of equilibrium one by one:
VC = P + W;          //(kN)
RB = (P*a + W*b)/c;       //(kN)
HC = RB;             //(kN)
RC = sqrt(HC^2 + VC^2);          //(kN)

printf("Rb = %.2f kN\n",RB);
printf("Rc = %.2f kN",RC);
```

Chapter 6

Centre of Gravity

Scilab code Exa 6.1 To find the centre of gravity

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 //As the section is symmetricsl about Y-Y axis,
     bisecting the web, therefore its centre of
     gravity will lie on this axis. Split up the
     section into two rectangles ABCH and DEFG as
     shown in Fig. 6.10.
6 //Let bottom of the web FE be the axis of reference:
7 //(i) Rectangle ABCH:
8 AB = 100;                  //(mm)
9 BC = 30;                   //(mm)
10 AF = 150;                 //(mm)
11 a1 = AB * BC;             //(mm^2)
12 y1 = (AF - BC/2);         //(mm)
13
14 //(ii) Rectangle DEFG:
15 DE = AF - BC;             //(mm)
16 FE = 30;                  //(mm)
17 a2 = DE * FE;            //(mm^2)
```

```
y2 = DE/2;                  //(mm)

//Distance between centre of gravity of the section
   and bottom of the flange FE:
y_bar = ((a1 * y1) + (a2 * y2))/(a1 + a2);      //(mm
   )
printf("Distance between centre of gravity of the
   section and bottom of the flange FE = %.2f mm",
   y_bar);       //The answers vary due to round off
   error
```

Scilab code Exa 6.2 To find the centre of gravity

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
//As the section is symmetrical about X-X axis,
   therefore its centre of gravity will lie on this
   axis. Now split up the whole section into three
   rectangles ABFJ, EGKJ and CDHK as shown in Fig.
   6.11.
//Let the face AC be the axis of reference :
//(i) Rectangle ABFJ:
AB = 50;                    //(mm)
AJ = 15;                    //(mm)
a1 = AB * AJ;               //(mm^2)
x1 = AB/2;                  //(mm)

//Rectangle EGKJ:
AC = 100;                   //(mm)
HD = 15;                    //(mm)
KG = 15;                    //(mm)
a2 = (AC - (HD + AJ)) * KG;         //(mm^2)
x2 = KG/2;                      //(mm)
```

```
19
20 //Rectangle CDHK:
21 CD = 50;                    //(mm)
22 a3 = CD * KG;                      //(mm^2)
23 x3 = CD/2;                         //(mm)
24
25 //Distance between the centre of gravity of the
      section and left face of the section AC:
26 x_bar = ((a1 * x1) + (a2 * x2) + (a3 * x3))/(a1 + a2
       + a3);              //(mm)
27 printf("Distance between the centre of gravity of
      the section and left face of the section AC = %.2
      f mm",x_bar);       //The answers vary due to round
       off error
```

Scilab code Exa 6.3 To find the position of the centre of gravity

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 //As the section is symmetrical about Y-Y axis,
      bisecting the web, therefore its centre of
      gravity will lie on this axis. Now split up the
      section into three rectangles as shown in Fig.
      6.12.
6 //Let the bottom of the bottom flange be the axis of
       reference:
7 //(i) Bottom flange:
8 l1 = 300;                          //(mm)
9 h1 = 100;                          //(mm)
10 a1 = l1 * h1;                     //(mm^2)
11 y1 = h1/2;                        //(mm)
12
13 //(ii) Web:
```

```
14 l2 = 50;                          //(mm)
15 h2 = 300;                         //(mm)
16 a2 = l2 * h2;                     //(mm^2)
17 y2 = h1 + h2/2;                   //(mm)
18
19 //(iii) Top flange:
20 l3 = 150;                        //(mm)
21 h3 = 50;                         //(mm)
22 a3 = l3 * h3;                    //(mm^2)
23 y3 = h1 + h2 + h3/2;            //(mm)
24
25 //The distance between centre of gravity of the
      section and bottom of the flange:
26 y_bar = ((a1 * y1) + (a2 * y2) + (a3 * y3))/(a1 + a2
       + a3);                 //(mm)
27 printf("The distance between centre of gravity of
      the section and bottom of the flange = %.2f mm",
      y_bar);
```

Scilab code Exa 6.4 To find the centroid of an unequal angle section

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 //As the section is not symmetrical about any axis,
      therefore we have to find out the values of x and
       y for the angle section. Split up the section
      into two rectangles as shown in Fig. 6.13.
6 //Let the face of the vertical section and bottom
      face of the horizontal section be axes of
      reference:
7 //(i) Rectangle 1:
8 h1 = 100;                        //(mm)
9 l1 = 20;                         //(mm)
```

```
10 a1 = h1 * l1;            //(mm^2)
11 x1 = l1/2;               //(mm)
12 y1 = h1/2;               //(mm)
13
14 //Rectangle 2:
15 l2 = 80;                 //(mm)
16 h2 = 20;                 //(mm)
17 a2 = (l2 - l1) * h2;     //(mm^2)
18 x2 = h2 + (l2 - l1)/2;   //(mm)
19 y2 = h2/2;               //(mm)
20
21 //The distance between centre of gravity of the
      section and left face:
22 x_bar = ((a1 * x1) + (a2 * x2))/(a1 + a2);    //(mm)
23 printf("The distance between centre of gravity of
      the section and left face = %.2f mm\n",x_bar);
24
25 //Distance between centre of gravity of the section
      and bottom face:
26 y_bar = ((a1 * y1) + (a2 * y2))/(a1 + a2);    //(mm)
27 printf("Distance between centre of gravity of the
      section and bottom face = %.2f mm",y_bar);
```

Scilab code Exa 6.5 To find the centre of gravity

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 //As the section is not symmetrical about any axis,
      therefore we have to find out the values of both
      x and y for the lamina:
6 //Let left edge of circular portion and bottom face
      rectangular portion be the axes of reference:
7 //(i) Rectangular portion:
```

```
l1 = 100;                  //(mm)
h1 = 50;                   //(mm)
a1 = l1 * h1;             //(mm^2)
x1 = 25 + l1/2;           //(mm)
y1 = h1/2;                 //(mm)

//(ii) Semicircular portion:
r = 25;                    //(mm)
a2 = (%pi/2) * (r^2);          //(mm^2)
x2 = 25 - (4 * r)/(3 * %pi);           //(mm)
y2 = 50/2;                        //(mm)

//(iii) Triangular portion:
a3 = (l1/2 * l1/2)/2;                 //(mm^2)
x3 = r + l1/2 + l1/4;                //(mm)
y3 = l1/2 + (l1/2)/3;                    //(mm)

//The distance between centre of gravity of the
   section and left edge of the circular portion:
x_bar = ((a1 * x1) + (a2 * x2) + (a3 * x3))/(a1 + a2
    + a3);
printf("The distance between centre of gravity of
   the section and left edge of the circular portion
    = %.2f mm\n",x_bar);      //The answers vary due
   to round off error

//Distance between centre of gravity of the section
   and bottom face of the rectangular portion:
y_bar = ((a1 * y1) + (a2 * y2) + (a3 * y3))/(a1 + a2
    +a3);           //(mm)
printf("Distance between centre of gravity of the
   section and bottom face of the rectangular
   portion = %.2f mm",y_bar);     //The answers vary
   due to round off error
```

Scilab code Exa 6.6 To find the centre of gravity from a point

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 //As the lamina is symmetrical about y-y axis,
    bisecting the lamina, therefore its centre of
    gravity lies on this axis. Let O be the reference
     point. From the geometry of the lamina. Semi-
    vertical angle of the lamina:
6 alpha = %pi/6;          //(degrees)
7 r = 220;                //(mm)(Radius)
8
9 //The distance between the reference point O and
    centre of gravity of the lamina:
10 y_bar = ((2*r)/3)*(sin(alpha)/alpha);
11 printf("The distance between the reference point O
    and centre of gravity of the lamina = %.2f mm",
    y_bar);     //The answers vary due to round off
    error
```

Scilab code Exa 6.7 To calculate the distance of the centre of mass

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 h = 30;       //(mm)
6 H = 120;      //(mm)
7 //(i) Cylinder:
8 //v1 = %pi * r^2 * 30;        (mm^3)
9 y1 = h/2;                   //(mm)
10
11 //(ii) Right circular cone:
```

```
12 //v2 = (%pi/3) * r^2 * h;        //(mm^3)
13 y2 = h + H/4;                    //(mm)
14
15 //Distance between centre of gravity of the section
      and base of the cylinder:
16 //v1 = 30*%pi*r^2
17 //v2 = 40*%pi*r^2
18 V1 = 30*%pi;
19 V2 = 40*%pi;
20 //y_bar = (v1*y1 + v2*y2)/(v1 + v2);
21 y_bar = ((V1*y1) + (V2*y2))/(V1 + V2);     //(mm)
22 printf("Distance between centre of gravity of the
      section and base of the cylinder = %.2f mm",y_bar
      );
```

Scilab code Exa 6.8 To find the position of the centre of gravity

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 //As the body is symmetrical about Y-Y axis,
      therefore its centre of gravity will lie on this
      axis as shown in Fig. 6.18. Let bottom of the
      hemisphere (D) be the point of reference.
6 //(i) Hemisphere:
7 r = 30;                          //(mm)(Radius)
8 v1 = 2*%pi/3 * r^3;              //(mm^3)
9 y1 = 5*r / 8;                    //(mm)
10
11 //(ii) Right circular cone:
12 h = 40;                           //(mm)(Height)
13 v2 = %pi/3 * r^2 * h;             //(mm^3)
14 y2 = r + h/4;                     //(mm)
15
```

```
16 //The distance between centre of gravity of the body
       and bottom of hemisphere D:
17 y_bar = ((v1 * y1) + (v2 * y2))/(v1 + v2);       //(
       mm)
18 printf("The distance between centre of gravity of
       the body and bottom of hemisphere D = %.2f mm",
       y_bar);      //The answers vary due to round off
       error
```

Scilab code Exa 6.10 To find the greatest height of the cylinder

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 //Let h = Height of the cylinder in mm.
6
7 //(i) Right circular cylinder:
8 //w1 = rho1 * (%pi/4) * d^2 * h;
9 //y1 = 60 + 0.5*h           (mm)
10
11 //(ii) Hemisphere:
12 //w2 = rho2 * (2*%pi/3) * r^3;
13 r = 60;                    //(mm)
14 y2 = (5 * r)/8;           //(mm)
15
16 //Distance between centre of gravity of the combine
       body from P(y):
17 // 60 = (w1*y1 + w2*y2)/(w1 + w2);
18 //Solving:
19 h = sqrt(6480000/1800);      //(mm)
20 printf("Height of the cylinder in mm = %.2f mm",h);
```

Scilab code Exa 6.11 To find the centre of gravity

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 r = 60;          //(mm)(Radius)
6 h = 30;          //(mm)(Height)
7 //Let O be the centre of the given sphere and ABC is
      the segment of this sphere as shown in Fig.
     6.21.
8 //As the section is symmetrical about X-X axis,
     therefore its centre of gravity lies on this axis
     .
9 //Let O be the reference point.
10 //We know that centre of gravity of the segment of
     sphere:
11 x_bar = (3 * (2*r - h)^2)/(4 * (3*r - h));      //(mm
     )
12 printf("Centre of gravity = %.2f mm",x_bar);
```

Scilab code Exa 6.13 To find the position of the centre of gravity

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 //(i) Trapezium ABCD:
6 DC = 200;          //(mm)
7 AB = 300;          //(mm)
8 h = 120;           //(mm)(height)
9 a1 = h * ((DC + AB)/2);           //(mm^2)
10 y1 = h/3 * ((AB + 2*DC)/(AB + DC));     //(mm)
11
12 //(ii) Semicircle:
```

```
13 r = 90;              //(mm)(Radius)
14 a2 = 1/2 * %pi * r^2;        //(mm^2)
15 y2 = (4*r) / (3*%pi);        //(mm)
16
17 //The distance between centre of gravity of the
      section and AB:
18 y_bar = ((a1 * y1) - (a2 * y2))/(a1 - a2);      //(mm
      )
19 printf("The distance between centre of gravity of
      the section and AB = %.2f mm",y_bar);
```

Scilab code Exa 6.14 To determine the centroid of the remaining area

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 //(i) Rectangle:
6 l = 40 + 40;          //(mm)
7 b = 30;               //(mm)
8 a1 = l * b;          //(mm^2)
9 x1 = l/2;             //(mm)
10 y1 = b/2;            //(mm)
11 r = l/4;             //(mm)
12
13 //(ii) Triangle:
14 a2 = l*(60 - b) / 2;       //(mm^2)
15 x2 = l*2 / 3;        //(mm)
16 y2 = b + b/3;       //(mm)
17
18 //(iii) Semicircle:
19 a3 = %pi/2 * (r)^2;        //(mm^2)
20 x3 = l/2 + r;           //(mm)
21 y3 = (4*(r))/(3*%pi);     //(mm)
22
```

```
23 //The distance between centre of gravity of the area
       and left face of trapezium:
24 x_bar = (a1*x1 + a2*x2 - a3*x3)/(a1 + a2 - a3);    //
       (mm)
25 printf("The distance between centre of gravity of
       the area and left face of trapezium = %.2f mm\n",
       x_bar);
26
27 //The distance between centre of gravity of the area
       and base of the trapezium:
28 y_bar = (a1*y1 + a2*y2 - a3*y3)/(a1 + a2 - a3);
       //(mm)
29 printf("The distance between centre of gravity of
       the area and base of the trapezium = %.2f mm",
       y_bar);     //The answers vary due to round off
       error
```

Scilab code Exa 6.16 To find the centre of gravity

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 //(i) Hemisphere ADE:
6 r = 60;                          //(mm)(Radius)
7 v1 = 2*%pi / 3 * r^3;            //(mm^3)
8 x1 = 5*r / 8;                    //(mm)
9
10 //(ii) Right circular cylinder ABCD:
11 h = 150;                          //(mm)(Height)
12 v2 = %pi * r^2 * h;             //(mm^3)
13 x2 = r + h/2;                   //(mm)
14
15 //(iii) Cone BCF:
16 v3 = %pi/3 * r^2 * h;           //(mm^3)
```

```
17 x3 = r + h*3/4;                    //(mm)
18
19 //The distance between centre of gravity of the
      solid and left edge of the hemisphere(E):
20 x_bar = (v1*x1 + v2*x2 - v3*x3)/(v1 + v2 - v3);
          //(mm)
21 printf("The distance between centre of gravity of
      the solid and left edge of the hemisphere(E) = %
      .2f mm",x_bar);
```

Scilab code Exa 6.17 To find the centre of gravity

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  //(i) Right circular cone OCD:
6  R = 1;                                //(m)(Radius)
7  H = 4;                                //(m)(Height)
8  v1 = %pi/3 * R^2 * H;                 //(m^3)
9  y1 = 4/4;                               //(m)
10
11 //(ii) Right circular cone OAB:
12 r = 0.5;                               //(m)(Radius)
13 h = 2;                                 //(m)(Height)
14 v2 = %pi/3 * (r)^2 * h;                //(m)
15 y2 = h + h/4;                          //(m)
16
17 //(iii) Circular hole:
18 d = 0.5;                               //(m)(radius)
19 v3 = %pi/4 * d^2 * h;              //(m^3)
20 y3 = h/2;                                 //(m)
21
22 //The distance between centre of gravity of the body
      and the base of the cone:
```

```
23 y_bar = (v1*y1 - v2*y2 - v3*y3)/(v1 - v2 - v3);
          //(m)
24 printf("The distance between centre of gravity of
     the body and the base of the cone = %.2f m",y_bar
     );
```

Scilab code Exa 6.18 To find the depth of the section plane

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 r = 20;       //(mm)(Radius)
6 h = 60;       //(mm)(Height)
7 //Centre of gravity of the composite section:
8 //(i) Right circular cone:
9 v1 = (%pi/3) * r^2 * h;       //(mm^3)
10 y1 = h * (3/4);              //(mm)
11
12 //(ii) Hemisphere:
13 v2 = (2*%pi/3) * r^3;         //(mm^3)
14 y2 = h + (3 * r)/8;          //(mm)
15
16 //Distance between centre of gravity of the body and
     apex of the cone:
17 y_bar = (v1*y1 + v2*y2)/(v1 + v2);      //(mm)
18 printf("Distance between centre of gravity of the
     body and apex of the cone = %.2f mm\n",y_bar);
19
20 //Depth of the section plane below the apex:
21 //Radius of the cut out cone:
22 //r = h/3
23
24 //Volume of the cut out cone:
25 //v3 = (%pi/3) * r^2 * h = 0.1164 * h^2     (mm^3)
```

```
26
27 //Distance between centre of gravity of the cut out
      cone and its apex:
28 //y3 = 0.75*h
29
30 ////Distance between the centre of gravity of the
      body and apex of the cone (54 + 5 = 59 mm):
31 // y_bar = (v1*y1 + v2*y2 - v3*y3)/(v1 + v2 - v3);
32
33 //Solving:
34 //h^4 - 78.67*h^3 = -2399200      -(i)
35
36 h = poly(0,"h");
37 p = h^4 - 78.67*h^3 + 2399200;
38 x = roots(p);
39 printf("Depth of the section plane below the apex =
      %.2f mm",x(2));     //(The answers vary due to
      round off error)
```

Chapter 7

Moment of Inertia

Scilab code Exa 7.1 To find the moment of inertia of a rectangular section

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 b = 30;              //(mm)(Width of the section)
6 d = 40;              //(mm)(Depth of the section)
7
8 //Moment of inertia of the section about an axis
     passing through its centre of gravity and
     parallel to X-axis
9
10 Ixx = (b*d^3)/12;          //(mm^4)
11
12 //Moment of inertia of the section about an axis
     passing through its centre of gravity and
     parallel to Y-axis
13 Iyy = (d*b^3)/12;          //(mm^4)
14
15 printf("Moment of Inertia about X-X axis = %.2f mm
     ^4\n",Ixx);
16 printf("Moment of Inertia about Y-Y axis = %.2f mm^4
```

```
",Iyy);
```

Scilab code Exa 7.2 To find the moment of inertia of a hollow rectangular section

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
b = 60;        //(mm)(External breadth)
d = 80;        //(mm)(External depth)
b1 = 30;       //(mm)(Internal breadth)
d1 = 40;       //(mm)(Internal depth)

Ixx = (b*d^3)/12 - (b1*d1^3)/12;        //(mm^4)
Iyy = (d*b^3)/12 - (d1*b1^3)/12;        //(mm^3)

printf("Ixx = %.2f mm^4\n",Ixx);
printf("Iyy = %.2f mm^4",Iyy);
```

Scilab code Exa 7.3 To find the moment of inertia of a circular section

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
d = 50;          //(mm)(Diameter)
Ixx = (%pi/64) * (d^4);          //(mm^4)
printf("Moment of Inertia = %.2f mm^4",Ixx);   //The
   answers vary due to round off error
```

Scilab code Exa 7.4 To find the moment of inertia of a hollow circular section

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 D = 80;                  //(mm)(External diameter)
6 d = 60;                  //(mm)(Internal diameter)
7 Ixx = (%pi/64)*(D^4 - d^4);              //(mm^4)
8 printf("Moment of Inertia = %.2f mm^4",Ixx);     //
    The answers vary due to round off error
```

Scilab code Exa 7.5 To find the moment of inertia of isosceles triangular section

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 b = 80;          //(mm)(Base width)
6 h = 60;          //(mm)(Height)
7
8 //Moment of inertia about the centre of gravity:
9 Ig = (b*h^3)/36;           //(mm^4)
10
11 //Moment of inertia about the base BC:
12 Ibc = (b*h^3)/12;         //(mm^4)
13
14 printf("Moment of inertia about the centre of
    gravity = %.2f mm^4\n",Ig);
15 printf("Moment of inertia about the base BC = %.2f
   mm^4",Ibc);
```

Scilab code Exa 7.6 To find the moment of inertia of a hollow triangular section

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 B = 180;          //(mm)(Base width of main triangle)
6 b = 120;          //(mm)(Base width of cut out
     triangle)
7 H = 100;          //(mm)(Height of main triangle)
8 h = 60;           //(mm)(Height of cut out triangle)
9
10 //Moment of Inertia of the triangular section about
     the base BC:
11 Ibc = (B*H^3)/12 - (b*h^3)/12;        //(mm^4)
12 printf("Moment of Inertia of the triangular section
     about the base BC = %.2f mm^4",Ibc);
```

Scilab code Exa 7.7 To find the moment of inertia of a semicircular section

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 d = 100;          //(mm)(Diameter of the section)
6 r = 50;           //(mm)(Radius)
7
8 //Moment of Inertia of the section about its centre
     of gravity and parallel to X-X  axis:
9 Ixx = 0.11 * r^4;       //(mm^4)
10
11 //Moment of Inertia of the section about its centre
     of gravity and parallel to Y-Y axis:
12 Iyy = 0.393 * r^4;      //(mm^4)
13
14 printf("Moment of Inertia of the section about its
     centre of gravity and parallel to X-X  axis = %.2
```

```
    f mm^4\n",Ixx);
15 printf("Moment of Inertia of the section about its
    centre of gravity and parallel to Y-Y axis = %.2f
    mm^4",Iyy);     //(The answers vary due to round
    off error)
```

Scilab code Exa 7.8 To find the moment of inertia of a hollow semicircular section

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  D = 200;          //(mm)(Outer diameter)
6  R = 100;          //(mm)(Outer radius)
7  d = 120;          //(mm)(Inner diameter)
8  r = 60;           //(mm)(Inner radius)
9
10 //Moment of Inertia about the base AB:
11 Iab = 0.393 * (R^4 - r^4);        //(mm^4)
12 printf("Moment of Inertia about the base AB = %.2f
    mm^4",Iab);     //The answers vary due to round
    off error
```

Scilab code Exa 7.9 To find the moment of inertia of an area

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  ED = 120;            //(mm)
6  FE = 40;             //(mm)
7  KA = 100;            //(mm)
8  CB = 40;             //(mm)
```

```
AB = 240;                    //(mm)
//Splitting the area into two rectangles 1 and 2 as
   shown in Fig.7.13.
//Moment of inertia of section(1) about its centre
   of gravity and parallel to axis K-K:
Ig1 = (ED*(FE)^3)/12;                //(mm^4)

//distance between centre of gravity of section (1)
   and axis K-K:
h1 = KA + FE/2;                      //(mm)

//Moment of inertia of section(1) about axis K-K:
I1 = Ig1 + (ED*FE)*(ED)^2;           //(mm^2)

//Moment of inertia of section(2) about its centre
   of gravity and parallel to axis K-K:
Ig2 = CB*(AB)^3 / 12;                 //(mm^4)

//distance between centre of gravity of section(2)
   and axis K-K:
h2 = KA + AB/2;                      //(mm)

//Moment of inertia of section(2) about the axis K-K
   :
I2 = Ig2 + (AB*CB)*(h2)^2;           //(mm^4)

//Moment of inertia of the whole area about the axis
    K-K:
IKK = I1 + I2;                       //(mm^4)

printf("Moment of inertia of the whole area about
   the axis K-K = %.2f mm^4",IKK);
```

Scilab code Exa 7.10 To find the moment of inertia of a T section

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  l1 = 150;     //(mm)
6  b1 = 50;      //(mm)
7  l2 = 50;     //(mm)
8  b2 = 150;      //(mm)
9
10 //Splitting up the whole section into two rectangles
       1 and 2:
11 //(i) Rectangle(1):
12 a1 = l1*b1;                //(mm^2)
13 y1 = l1 + b1/2;            //(mm)
14
15 //(ii) Rectangle(2):
16 a2 = l2*b2;                //(mm^2)
17 y2 = b2/2;                 //(mm)
18
19 //The distance between centre of gravity of the
      section and bottom of the web:
20 y_bar = (a1*y1 + a2*y2)/(a1 + a2);          //(mm)
21
22 //Moment of inertia about X-X axis:
23 Ig1 = l1*(b1)^3 / 12;                  //(mm^4)
24
25 //Distance between centre of gravity of rectangle(1)
       and X-X axis:
26 h1 = y1 - y_bar;                          //(mm)
27
28 //Moment of inertia of rectangle(1) about X-X axis:
29 Ix1 = Ig1 + (a1*(h1)^2);          //(mm^4)
30
31 //Moment of inertia of rectangle(2) about an axis
      through its centre of gravity and parallel to X-X
       axis:
32 Ig2 = h1*(b2)^3 / 12;            //(mm^4)
33
```

```
34 //Distance between centre of gravity of rectangle(2)
       and X-X axis:
35 h2 = y_bar - y2;                          //(mm)
36
37 //Moment of inertia of rectangle(2) about X-X axis:
38 Ix2 = Ig2 + (a2*(h1)^2);                  //(mm^4)
39
40 //Moment of Inertia of the whole section about X-X
       axis:
41 Ixx = Ix1 + Ix2;                          //(mm^4)
42
43 //Moment of inertia about Y-Y axis:
44 //M.I. of rectangle(1) about Y-Y axis:
45 Iy1 = h1*(l1)^3 / 12;                     //(mm^4)
46
47 //M.I. of rectangle(2) about Y-Y axis:
48 Iy2 = l1*(h1)^3 / 12;                     //(mm^4)
49
50 //Moment of Inertia of the whole section about Y-Y
       axis:
51 Iyy = Iy1 + Iy2;                          //(mm^4)
52
53 printf("Moment of Inertia of the whole section about
       X-X axis = %.2f mm^4\n",Ixx);
54 printf("Moment of Inertia of the whole section about
       Y-Y axis = %.2f mm^4",Iyy);
```

Scilab code Exa 7.11 To find the moment of inertia of an I section

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 //Splitting up the whole section into three
      rectangles 1, 2 and 3 as shown in Fig. 7.15.
```

```
6  //(i)Rectangle(1):
7  l1 = 60;                          //(mm)
8  h1 = 20;                          //(mm)
9  h2 = 100;                         //(mm)
10 a1 = l1*h1;                       //(mm^2)
11 y1 = h1 + h2 + h1/2;              //(mm)
12
13 //(ii)Rectangle(2):
14 l2 = 20;                          //(mm)
15 a2 = h2 * l2;                      //(mm^2)
16 y2 = l2 + h2/2;                   //(mm)
17
18 //(iii)Rectangle(3):
19 l3 = 100;                         //(mm)
20 h3 = 20;                          //(mm)
21 a3 = l3 * h3;                      //(mm^2)
22 y3 = h3/2;                        //(mm)
23
24 //Distance between centre of gravity of the section
      and bottom face:
25 y_bar = (a1*y1 + a2*y2 + a3*y3)/(a1 + a2 + a3);
              //(mm)
26
27 //Moment of inertia of rectangle(1) about an axis
      through its centre of gravity and parallel to X-X
       axis:
28 Ig1 = l1*(h1)^3 / 12;                 //(mm^4)
29
30 //Distance between centre of gravity of rectangle(1)
       and X-X axis:
31 hh1 = 130 - y_bar;                      //(mm)
32
33 //Moment of Inertia of rectangle(1) about X-X axis:
34 Ix1 = Ig1 + (a1*(hh1)^2);          //(mm^4)
35
36 //Moment of inertia of rectangle(2) about an axis
      through its centre of gravity and parallel to X-X
       axis:
```

```
Ig2 = l2*(h2)^3 / 12;                    //(mm^4)

//Distance between centre of gravity of rectangle(2)
    and Y-Y axis:
hh2 = 70 - y_bar;                        //(mm)

//Moment of inertia of rectangle(2) about X-X axis:
Ix2 = Ig2 + (a2*(hh2)^2);               //(mm^4)

//Moment of inertia of rectangle(3) about an axis
   through its centre of gravity and parallel to X-X
    axis:
Ig3 = l3*(h3)^3 / 12;                    //(mm^4)

//Distance between centre of gravity of rectangle(3)
    and X-X axis:
hh3 = y_bar - y3;                        //(mm)

//Moment of inertia of rectangle(3) about X-X axis:
Ix3 = Ig3 + (a3*(hh3)^2);               //(mm^4)

//Moment of Inertia of the whole section about X-X
   axis:
Ixx = Ix1 + Ix2 + Ix3;                   //(mm^4)
printf("Moment of Inertia of the whole section about
    X-X axis = %.2f mm^4",Ixx);          //(The answers
   vary due to round off error)
```

Scilab code Exa 7.12 To find the moment of inertia of an angle section

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
//Splitting up the section into two rectangles 1 and
```

```
    2 as shown in Fig.7.16.
6
7 //Moment of Inertia about centroidal X-X axis:
8
9 //Rectangle(1):
10 l1 = 20;                  //(mm)
11 h1 = 100;                 //(mm)
12 a1 = h1 * l1;              //(mm^2)
13 y1 = h1/2;               //(mm)
14
15 //Rectangle(2):
16 l2 = 80;                 //(mm)
17 h2 = 20;                 //(mm)
18 a2 = (l2 - l1)*h2;       //(mm)
19 y2 = h2/2;               //(mm)
20
21 //The distance between the centre of gravity of the
     section and bottom face:
22 y_bar = (a1*y1 + a2*y2)/(a1 + a2);       //(mm)
23
24 //Moment of inertia of rectangle(1) about an axis
     through its centre of gravity and parallel to X-X
     axisL
25 Ig1 = l1*(h1)^3 / 12;              //(mm^4)
26
27 //Distance of centre of gravity of rectangle(1) from
     X-X axis:
28 hh1 = y1 - y_bar;                          //(mm)
29
30 //Moment of inertia of rectangle(1) about X-X axis:
31 Ix1 = Ig1 + (a1*(hh1)^2);          //(mm^4)
32
33 //Moment of inertia of rectangle(2) about an axis
     through its centre of gravity and parallel to X-X
     axis:
34 Ig2 = (l2 - l1)*(h2)^3 / 12;               //(mm^4)
35
36 //Distance of centre of gravity of rectangle(2) from
```

```
    X-X axis:
hh2 = y_bar - y2;                          //(mm)

//Moment of inertia of rectangle(2) about X-X axis:
Ix2 = Ig2 + (a2*(hh2)^2);                  //(mm^4)

//Moment of Inertia of whole section about X-X axis:
Ixx = Ix1 + Ix2;                           //(mm^4)
printf("Moment of Inertia of whole section about X-X
    axis = %.2f mm^4\n",Ixx);              //(Answer rounded
    off in book)

//Moment of Inertia about centroidal Y-Y axis:

//Rectangle(1):
x1 = y1;                    //(mm)

//Rectangle(2):
x2 = y2;              //(mm)

//Distance between the centre of gravity of the
   section and left face:
x_bar = (a1*x1 + a2*x2)/(a1 + a2);              //(mm)

//Moment of inertia of rectangle(1) about an axis
   through its centre of gravity and parallel to Y-Y
    axis:
Ig1 = h1*(l1)^3 / 12;                      //(mm^4)

//Distance of centre of gravity of rectangle(1) from
    Y-Y axis:
hh1 = x_bar - x1;                               //(mm)

//Moment of Inertia of rectangle(1) about Y-Y axis:
Iy1 = Ig1 + (a1*(hh1^2));                  //(mm^4)

//Moment of inertia of rectangle(2) about an axis
   through its centre of gravity and parallel to Y-Y
```

```
    axis:
Ig2 = h2*(l2 - l1)^3 / 12;                    //(mm^4)

//Distance of centre of gravity of rectangle(2) from
    Y-Y axis:
hh2 = x_bar - x2;                             //(mm)

//Moment of Inertia of rectangle(2) about Y-Y axis:
Iy2 = Ig2 + (a2*(hh2)^2);                     //(mm^4)

//Moment of Inertia of the whole section about Y-Y
    axis:
Iyy = Iy1 + Iy2;                              //(mm^4)

printf("Moment of Inertia of the whole section about
    Y-Y axis = %.2f mm^4",Iyy);               //The answers
    vary due to round off error
```

Scilab code Exa 7.13 To find the moment of inertia of a cast iron beam

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
//Moment of inertia of the section about horizontal
    axis passing through the centroid of the section:

//Moment of inertia of the rectangular section about
    its horizontal axis passing through its centre
    of gravity:
b = 120;                                      //(mm)(Breadth
    )
d = 150;                                      //(mm)(Depth)
Ix1 = b*(d)^3 / 12;                           //(mm^4)

```

```
12 //Moment of inertia of the circular section about a
     horizontal axis passing through its centre of
     gravity:
13 r = 50;                                    //(mm)(Radius)
14 Ix2 = %pi/4 * (r)^4;                      //(mm^4)
15
16 //Moment of inertia of the whole section about
     horizontal axis passing through the centroid of
     the section:
17 Ixx = Ix1 - Ix2;                          //(mm^4)
18 printf("Moment of inertia of the whole section about
     horizontal axis passing through the centroid of
     the section = %.2f mm^4\n",Ixx);
19
20 //Moment of inertia of the section about vertical
     axis passing through its centre of gravity:
21 Ig1 = d*(b)^3 / 12;                  //(mm^4)
22
23 //Area of one semicircular section with 50 mm radius
     :
24 a = %pi*(r)^2 / 2;                     //(mm^2)
25
26 //Moment of inertia of a semicircular section about
     a vertical axis passing through its centre of
     gravity:
27 Ig2 = 0.11*(r)^4;                      //(mm^4)
28
29 //Distance between centre of gravity of the
     semicircular section and its base:
30 h = (4*r) / (3*%pi);                       //(mm)
31
32 //Distance between centre of gravity of the
     semicircular section and centre of gravity of the
     whole section:
33 h2 = 60 - h;                        //(mm)
34 I = Ig2 + (a*(h2)^2);              //(mm^4)
35
36 //Moment of inertia of both the semicircular
```

```
    sections about centre of gravity of the whole
    section:
37 II = 2*I;                                //(mm^4)
38
39 //Moment of Inertia of the whole section about a
    vertical axis passing through the centroid of the
     section:
40 MI = Ig1 - II;                           //(mm^4)
41 printf("Moment of Inertia of the whole section about
     a vertical axis passing through the centroid of
    the section = %.2f mm^4",MI);     //(The answers
    vary due to round off error)
```

Scilab code Exa 7.14 To find the moment of inertia of a hollow section

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  //Let y be the distance between centre of gravity of
       the section from the bottom face:
6  //(i)Rectangle:
7  h1 = 300;                    //(mm)(Height)
8  l1 = 200;                    //(mm)(Length)
9  a1 = h1 * l1;                //(mm^2)
10 y1 = h1/2;                  //(mm)
11
12 //(ii)Circular hole:
13 d = 150;                     //(mm)(Diameter)
14 x = 100;                     //(mm)(Distance from top
    to centre of circle)
15 a2 = %pi/4 * (d)^2;        //(mm^2)
16 y2 = h1 - x;               //(mm)
17
18 //Distance between the centre of gravity of the
```

```
    section and its bottom face:
y = (a1*y1 - a2*y2)/(a1 - a2);          //(mm)

//Moment of inertia of rectangular section about an
    axis through its centre of gravity and parallel
    to X-X axis:
Ig1 = l1*(h1)^3 / 12;                   //(mm^4)

//Distance of centre of gravity of rectangular
    section and X-X axis:
hh1 = (h1/2) - y;                       //(mm)

//Moment of inertia of rectangle about X-X axis:
I = Ig1 + (h1*l1)*(hh1)^2;              //(mm^2)

//Moment of inertia of circular section about an
    axis through its centre of gravity and parallel
    to X-X axis:
Ig2 = %pi/64 *(d)^4;                    //(mm^4)

//Distance between centre of gravity of the circular
     section and X-X axis:
hh2 = (h1 - x) - y;                     //(mm)

//Moment of inertia of the circular section about X-
    X axis:
II = Ig2 + (a2 * (hh2)^2);              //(mm^4)

//Moment of Inertia of the whole section about X-X
    axis:
MI = I - II;                            //(mm^2)
printf("Moment of Inertia of the whole section about
    X-X axis = %.2f mm^4",MI);     //The answers vary
    due to round off error
```

Scilab code Exa 7.15 To find the moment of inertia of a triangular section with a

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 //Let y be the distance between the centre of
     gravity of the section and the base BC.
6 //(i)Triangular section:
7 b1 = 100;                          //(mm)(Base)
8 d1 = 90;                           //(mm)(Height)
9 a1 = (b1*d1) / 2;                  //(mm^2)
10 y1 = d1/3;                        //(mm)
11
12 //(ii)Rectangular hole:
13 b2 = 20;                          //(mm)(Breadth)
14 d2 = 30;                          //(mm)(Height)
15 a2 = b2*d2;                       //(mm^2)
16 y2 = d2 + d2/2;                   //(mm)
17
18 //Distance between the centre of gravity of the
     section and base BC of the triangle:
19 y = (a1*y1 - a2*y2)/(a1 - a2);       //(mm)
20
21 //Moment of inertia of the section about X-X axis:
22
23 //Moment of inertia of the triangular section
     through its centre of gravity and parallel to X-X
      axis:
24 b = b1;
25 d = d1;
26 Ig1 = b*(d)^3 / 36;                  //(mm^4)
27
28 //Distance between the centre of gravity of the
     section and X-X axis:
29 hh1 = 30 - y;                        //(mm)
30
31 //Moment of inertia of the triangular section about
```

```
    X-X axis:
I = Ig1 + (a1*(hh1)^2);            //(mm^4)

//Moment of inertia of the rectangular hole through
    its centre of gravity and parallel to the X-X
    axis:
b = b2;
d = d2;
Ig2 = b*d^3 / 12;                  //(mm^4)

//Distance between the centre of gravity of the
    section and X-X axis:
hh2 = 45 - y;                      //(mm)

//Moment of inertia of rectangular section about X-X
     axis:
II = Ig2 + (a2*(hh2)^2);           //(mm^4)

//Moment of inertia of the whole section about X-X
    axis:
Ixx = I - II;                      //(mm^4)

//Moment of inertia of the section about the base BC
    :

//Moment of inertia of the triangular section about
    the base BC:
Ig1 = b1*(d1)^3 / 12;              //(mm^4)

//Moment of inertia of the rectangular hole through
    its centre of gravity and parallel to X-X axis:
Ig2 = b2*d2^3 / 12;                //(mm^4)

//Distance between the centre of gravity of the
    section about the base BC:
h2 = d2 + d2/2;                    //(mm)

//Moment of inertia of rectangular section about the
```

```
    base BC:
60 i = Ig2 + (a2*(h2)^2);                //(mm^4)
61
62 //Moment of Inertia of the whole section about the
    base BC:
63 Ibc = Ig1 - i;                          //(mm^4)
64
65 printf("Moment of inertia of the whole section about
    X-X axis = %.2f mm^4\n",Ixx);
66 printf("Moment of Inertia of the whole section about
    the base BC = %.2f mm^4",Ibc);
```

Scilab code Exa 7.16 To find the moment of inertia of a compound beam section

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  l = 160;                  //(mm)(Length)
6  b = 12;                   //(mm)(Breadth of rod)
7  h = 300;                  //(mm)(Height of the stand)
8  MI_ISLB = 73.329;           //(mm^4)(Moment of inertia
    of ISLB 300 section about X - X axis)
9  //Moment of inertia of one steel plate section about
    an axis passing through its centre of gravity
    and parallel to X-X axis:
10 Ig = l*((b)^3)/12;              //(mm^4)
11
12 //Distance between the centre of gravity of the
    plate section and X-X axis:
13 h = (h/2) + b/2;                      //(mm)
14
15 //Moment of inertia of one plate section about X-X
    axis:
16 a = l*b;
```

```
17 I = Ig + (a)*(h^2);          //(mm^2)
18
19 //Moment of inertia of the compound beam section
      about X-X axis:
20 Ixx = (MI_ISLB * 10^6) + 2*(I);      //(mm^4)
21
22 printf("Moment of inertia of the compound beam
      section about X-X axis = %.2f mm^4",Ixx);     //
      The answers vary due to round off error
```

Scilab code Exa 7.17 To find the moment of inertia of a compound section

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  l = 200;          //(mm)(Total length)
6  b = 15;           //(mm)(Breadth)
7  h = 200;          //(mm)(Height)
8  x = 60;           //(mm)(Length of upper part)
9  MI_ISJB = 7.807 * 10^6;     //(Moment of inertia of
      ISJB section about X - X axis)
10 //Moment of inertia of one plate section about an
      ais passing through its centre of gravity and
      parallel to X-X axis:
11 Ig = l*(b)^3 / 12;                  //(mm^4)
12
13 //Distance between the centre of gravity of the
      plate section and X-X axis:
14 h1 = h/2 + b/2;                        //(mm)
15
16 //Moment of inertia of the plate section about X-X
      axis:
17 a = l*b;
18 I = Ig + (a*(h1)^2);          //(mm^4)
```

```
19
20 //Moment of inertia of the compound about X-X axis:
21 Ixx = 2*(MI_ISJB) + 2*(I);      //(mm^4)
22
23 printf("Moment of inertia of the compound about X-X
       axis = %.2f mm^4",Ixx);
```

Scilab code Exa 7.18 To find the moment of inertia of a build up section

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  l = 90;                  //(mm)(Length)
6  b1 = 10;                 //(mm)(Breadth)
7  b2 = 10;                 //(mm)(Breadth)
8  d = 20;                  //(mm)(Distance between the two
       structures)
9  b3 = 10;                 //(mm)(Breadth of the rod)
10 h = 130;                 //(mm)(Height of the structure)
11 //Moment of inertia of one top or bottom plate about
       an axis through its centre and parallel to X-X
       axis:
12 Ig1 = l*(b1)^3 / 12;                //(mm^4)
13
14 //Distance between centre of gravity of the plates
       from X-X axis:
15 h1 = 65 - 5;                        //(mm)
16
17 //Moment of inertia of top and bottom plates about X
       -X axis:
18 a1 = l*b2;
19 I = 2*(Ig1 + (a1)*(h1)^2);          //(mm^2)
20
21 //Moment of inertia of part(1) of one channel
```

```
   section about an axis through its centre of
   gravity and parallel to X-X axis:
y = 90 - 20 - 20 - 10 - 10;          //(mm)(Length
   of top of the structure)

Ig2 = (y)*(b2)^3 / 12;                //(mm^4)

//Distance of centre of gravity of this part from X-
   X axis:
h2 = 55 - 5;                          //(mm)

//Moment of Inertia of part(1) about X-X axis:
a2 = y*b2;
II = 4*(Ig2 + (a2)*(h2)^2);          //(mm^4)

//Moment of inertia of part(2) of the channel about
   an axis through its centre of gravity and
   parallel to X-X axis:
Ig3 = 2*(b1*(l)^3/12);               //(mm^4)

//Moment of Inertia of the whole built-up section
   about an axis through its centre of gravity
   parallel to X-X axis:
Ixx = I + II + Ig3;                   //(mm^4)

printf("Moment of Inertia of the whole built-up
   section about an axis through its centre of
   gravity parallel to X-X axis = %.2f mm^4",Ixx);
         //The answer provided in the textbook is
   wrong
```

Chapter 8

Principles of Friction

Scilab code Exa 8.1 To find the magnitude of force

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 W = 300;                          //(N)(Weight of the body
    )
6 Mu = 0.3;                          //(Coefficient of
    friction)
7 alpha = 25;                           //(degrees)(Angle
    made by the force with the horizontal)
8 //P = Magnitude of the force, which can move the
    body,
9 //F = Force of friction
10
11 //Resolving the forces horizontally:
12 //F = P*cos(I)      -(i)
13
14 //Resolving the forces vertically:
15 //R = W - P*sin(I)  -(ii)
16
17 //From (i) and (ii)-
```

```
18 //F = Au * R
19 //P*cos(I) - Au * W + P * Au * sin(I) = 0
20 x = Mu * sind(alpha) + cosd(alpha);      //(
      Coefficient of P)
21 y = -Mu * W;                  //(Constant term)
22 A = [x];
23 c = [y];
24 [P,nsA] = linsolve(A,c);
25 printf("Magnitude of the force, which can move the
      body = %.2f N",P);
```

Scilab code Exa 8.2 To determine the weight of the body and coefficient of frictio

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5
6 //For pull of 180 N:
7 P1 = 180;        //(N)(Pull)
8 alpha = 30;    //(degrees)(Angle of pull)
9 //Resolving the forces horizontally:
10 F1 = P1*cosd(alpha);              //(N)
11
12 //Resolving the forces vertically:
13 //R1 = W - 180*sind(30)
14
15 //F1 = u*R1;
16 //155.9 = u*(W-90)                  -(1)
17
18 //For push of 220 N:
19 P2 = 220;         //(N)(Push)
20 alpha = 30;     //(degrees)(Angle of push)
21 //Resolving the forces horizontally:
22 F2 = P2*cosd(alpha);             //(N)
```

```
23
24 //Resolving the forces vertically:
25 //R2 = W + 220*sind(30)
26
27 //F2 = u*R2;
28 //190.5 = u*(W + 110)              -(2)
29
30 //Eqn (1) / (2)
31 //Solving:
32 // (F2 - F1) * W - (110 * F1) - (90 * F2) = 0
33 A = [(F2 - F1)];
34 c = [(-(110*F1)-(90*F2))];
35 [x,nsA] = linsolve(A,c);
36 W = x;
37 printf("Weight of the body = %.2f N\n",W); //The
      answers vary due to round off error
38
39 //Substituting W in (1):
40 mu = F1/(W - P1*sind(alpha));
41 printf("Coefficient of friction = %.2f",mu);
```

Scilab code Exa 8.3 To find the force required to move the block

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 //Considering the forces acting on block A:
6 //Resolving the forces vertically:
7
8 mu = 0.3;                  //(Coefficient of friction)
9 alpha = 30;                //(degrees)
10 //T*sind(30) = 1 - R1;          -(1)
11
12 //Resolving the forces horizontally:
```

```
13
14 //T*cosd(30) = 0.3*R1;
15
16 //Eqn (1) / (2):
17 R1 = 1/(mu*tand(alpha) + 1);        //(kN)
18
19 F1 = mu*R1;                         //(kN)
20
21 //Considering block B:
22 //Resolving the forces vertically:
23 R2 = 2 + R1;                        //(kN)
24 F2 = mu*R2;                         //(kN)
25
26 //Resolving the forces horizontally:
27 P = F1 + F2;                       //(kN)
28 printf("Force required to move the block(P) = %.2f
   kN",P);
```

Scilab code Exa 8.4 To find the normal and tangential force

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 P = 5;                    //(kN)
6 mu = 0.2;                  //(Coefficient of friction
   at C)
7 AB = 1.5;                 //(m)(Total length)
8 AC = 1;                   //(m)(Length till pulley)
9 r1 = 50;                  //(mm)(Inner radius of pulley
   )
10 r2 = 75;                 //(mm)(Outer radius of pulley
   )
11 //Let R = Normal reaction of the pulley on the beam
   at C.
```

```
12 //Taking moments about the hinge A and equating the
      same:
13 R = (P * AB)/AC;          //(kN)
14
15 //Maximum force of friction at C:
16 MF = mu*R;                 //(kN)
17
18 //Taking moments about the centre of the pulley and
      equating the same:
19 W = (AB * r2)/r1;         //(kN)
20 printf("Maximum load(W) = %.2f kN\n",W);
21
22 //Normal and tangential forces transmitted at C:
23 //R1 = Normal force or normal reaction at C;
24 //F1 = Tangential force at C.
25 W = 3;                    //(kN)
26 P = 4.5;                  //(kN)
27 //Taking moments about the hinge A and equating the
      same:
28 R1 = (P * AB)/AC;          //(kN)
29 printf("Normal reaction at C = %.2f kN\n",R1);
30
31 //Taking moments about the centre of the pulley and
      equating the same:
32 F1 = (W * r1)/r2;              //(kN)
33 printf("Tangential force at C = %.2f kN",F1);
```

Scilab code Exa 8.5 To determine the minimum and maximum value of force

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 W = 500;                    //(N)(Weight of the body)
6 alpha = 25;                     //(degrees)(Angle at
```

```
    which plane is inclined)
phi = 20;                    //(degrees)(Angle of
    friction)
//Minimum value of P:
//For the minimum value of P, the body is at the
    point of sliding downwards:
P1 = W * (sind(alpha - phi)/cosd(phi));        //(N)

//Maximum value of P:
//For the maximum value of P, the body is sliding
    upwards:
P2 = W * (sind(alpha + phi)/cosd(phi));        //(N)

printf("Minimum value of P = %.2f N\n",P1);     //The
    answers vary due to round off error
printf("Maximum value of P = %.2f N",P2);
```

Scilab code Exa 8.6 To find the minimum force required

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
W = 400;              //(N)(Weight of the body)
u = 0.3;              //(Coefficient of friction)
b = 2.4;              //(m)
h = 1.2;              //(m)
//Whether it is necessary to push the body down the
    plane or hold it back from sliding down:
alpha = atand(h/b);                //(degrees)

//Normal reaction:
R = W * cosd(alpha);                    //(N)

//Force of friction:
```

```
16 F = u * R;                              //(N)
17
18 //Resolving the 400 N force along the plane:
19 F1 = W * sind(alpha);                    //(N)
20
21 //The body will slide down:
22 //Minimum force required parallel to the plane:
23 P = F1 - F;                              //(N)
24
25 printf("The force along the plane (which is
      responsible for sliding the body) is more than
      the force of friction, therefore the body will
      slide down. Or in other words, it is not
      necessary to push the body down the plane, rather
       it is necessary to hold it back from sliding
      down.\n");
26 printf("Minimum force required parallel to the plane
      = %.2f N",P);     //The answers vary due to round
       off error
```

Scilab code Exa 8.7 To find the weight of body and coefficient of friction

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 P1 = 200;                 //(N)
6 alpha = 15;                 //(degrees)
7 P2 = 230;                 //(N)
8 beta = 20;                  //(degrees)
9
10 //Resolving the forces at right angles to the plane:
11 //R1 = W * cosd(15);                    -(1)
12
13 //Resolving the forces along the plane:
```

```
14 //200 = W * (u * cosd(15) + sind(15));        -(2)
15
16 //Resolving the forces at right angles to the plane:
17 //R2 = W * cosd(20);                  -(3)
18
19 //Resolving the forces at right angles to the plane:
20 //230 = W * (u * cosd(20) + sind(20));      -(4)
21
22 //Coefficient of friction:
23 //Dividing (4) by (2):
24 mu = ((P1 * sind(beta)) - (P2 * sind(alpha)))/((P2 *
       cosd(alpha)) - (P1 * cosd(beta)));
25 printf("Coefficient of friction = %.2f\n",mu);    //
     The answers vary due to round off error
26
27 //Weight of the body:
28 //Substituting the value of u in equation (2):
29 W = P1/(mu * cosd(alpha) + sind(alpha));
30 printf("Weight of the body = %.2f N",W);     //(The
     answers vary due to round off error)
```

Scilab code Exa 8.8 To determine the minimum magnitude of the force

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 W = 100;              //(N)
6 alpha = 30;               //(degrees)(Angle at which
     the plane is inclined)
7 mu = 0.25;             //(Coefficient of friction)
8 phi = atand(mu);
9
10 //Magnitude of force to keep the object in position:
11 F = W * tand(alpha - phi);             //(N)
```

```
12 printf("Magnitude of force to keep the object in
      position = %.2f N",F);     //The answers vary due
      to round off error
```

Scilab code Exa 8.9 To find the inclination of plane and coefficient of friction

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  W = 1.5;            //(kN)(Load)
6  P1 = 2;             //(kN)(Horizontal effort)
7  P2 = 1.25;          //(kN)(Effort parallel to the
      inclined plane)
8
9  //Inclination of the plane:
10 //alpha = Inclination of the plane;
11 //phi = Angle of friction;
12
13 //P = W * tand(alpha + phi);
14 //alpha + phi = atand(2/1.5);          //(degrees)
15
16 //Also:
17 //P = W * (sind(alpha + phi)/cos(phi));
18 alpha_plus_phi = atand(P1/W);
19 phi = acosd(W*sind(alpha_plus_phi)/P2);          //(
      degrees)
20 alpha = atand(P1/W) - phi;          //(degrees)
21 printf("Inclination of the plane = %.2f degrees\n",
      alpha);     //The answers vary due to round off
      error
22
23 //Coefficient of friction:
24 mu = tand(phi);
25 printf("Coefficient of friction = %.2f",mu);
```

Scilab code Exa 8.10 To find the smallest weight of block A

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5
6 //Doing it by first method
7 mu_A = 0.3;                //(Coefficient of friction
     between block A and horizontal surface)
8 mu_B = 0.4;                //(Coefficient of friction
     between block B and the inclined surface)
9
10 WB = 100;                 //(N)(Weight of the block)
11
12 alpha = 60;                //(degrees)(Angle of
     inclination)
13 //Let WA = Smallest weight of block A
14 //P = uA * WA;
15
16 phi = atand(mu_B);             //(degrees)
17
18 //Force that will hold the system in equilirium:
19 WA = (WB * tand(alpha - phi))/mu_A;          //(N)
20 printf("Force that will hold the system in
     equilirium = %.2f N",WA);    //The answers vary
     due to round off error
```

Scilab code Exa 8.11 To find the value of force

```
1 //OS-version - Windows 10
```

```
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5
6 //Solving by 1st method
7
8 W = 300;                  //(N)(Load)
9 P1 = 60;                  //(N)(Force)
10 theta = 30;                  //(degrees)(Angle at which
     force is inclined)
11 //Let A = Angle of inclination of the plane
12 //Smooth plane:
13 mu = 0;
14 Phi = 0;                  //(degrees)
15
16 //Resolving the forces:
17 alpha = asind(P1*cosd(theta) / W);            //(
     degrees)
18
19 //Rough plane:
20 //Force required to move the load up the plane:
21 mu = 0.3;
22 Phi = atand(mu);                              //(degrees
     )
23 P = W * (sind(alpha + Phi)/cosd(Phi));        //(N)
24
25 printf("Force required to move the load up the plane
     = %.2f N",P);     //The answers vary due to round
     off error
```

Scilab code Exa 8.13 To find the minimum and maximum value of W2

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
```

```
4 clear all
5 W1 = 1;                    //(kN)(First Load)
6 alpha1 = 45;                //(degrees)(Angle made by
     inclined plane OA with the horizontal)
7
8 alpha2 = 30;                //(degrees)(Angle made by
     inclined plane OB with the horizontal)
9
10 Phi = 20;                  //(degrees)(Angle of friction
     for both the planes)
11
12 //Maximum value of W2:
13 P = W1 * tand(alpha1 + Phi);                  //(kN)
14 //E2 = W2 * tand(alpha2 - Phi)
15
16 //Equating E1 and E2:
17 W2 = P/(tand(alpha2 - Phi));
18 printf("Maximum value of W2 = %.2f kN\n",W2);
19
20 //Minimum value of W2:
21 P = W1 * tand(alpha1 - Phi);                 //(kN)
22 //E2 = W2 * tand(alpha2 + Phi);
23
24 //Equating E1 and E2:
25 W2 = [P/(tand(alpha2 + Phi))]*1000;          //(N)
26 printf("Minimum value of W2 = %.2f N",W2);     //The
     answers vary due to round off error
```

Scilab code Exa 8.14 To determine the horizontal force P

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 WA = 1;                    //(kN)(Weight of block A)
```

```
WB = 3;                 //(kN)(Weight of block B)
alpha = 45;                 //(degrees)(Angle of
    inclination of plane with horizontal)
theta = 30 - alpha;                 //(degrees)
mu = tand(theta);           //(Coefficient of
    friction)
phi = 15;           //(degrees)(Angle of limiting
    friction)
T = WA * (sind(alpha + phi) / cosd(theta - phi));
          //(kN)

beta = 30;      //(degrees)(Angle of 1kN force with
    horizontal)

F = 3;     //(N)(Force acting downwards)
//Resolving the forces horizontally:
RB = F + WA * sind(beta);           //(kN)
FB = WA*cosd(beta);                 //(kN)
//Resolving the forces horizontally:
P = WA * cosd(beta) + FB;           //(kN)
printf("Horizontal force(P) = %.2f kN",P);     //The
    answers vary due to round off error
```

Chapter 9

Applications of Friction

Scilab code Exa 9.1 To find the frictional force acting on the ladder

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 l = 3.25;              //(m)(Length of the ladder)
6 w = 250;               //(N)(Weight of the ladder)
7 d = 1.25;              //(m)(Distance between the lower
      end of the ladder and the wall)
8 mu_f = 0.3;             //(Coefficient of friction
     between the ladder and the floor)
9
10 //Ff = Frictional force acting on the ladder at the
     Point of contact between the ladder and floor
11 //Rf = Normal reaction at the floor
12
13 //Resolving the forces vertically:
14 Rf = 250;                //(N)
15
16 //From geometry:
17 BC = sqrt((l)^2 - (d)^2);        //(m)
18
```

```
19 //Taking moments about B and equating the same:
20 Ff = ((Rf * d) - (w * d/2))/3;      //(N)
21
22 printf("Frictional Force = %.2f N\n",Ff);     //The
      answers vary due to round off error
23
24 //Equilibrium of the ladder:
25 E = mu_f * Rf;                  //(N)
26
27 printf("Since E = %.2f N > Ff = %.2f N the ladder
      will remain in an equilibrium position.",E,Ff);
```

Scilab code Exa 9.2 To calculate the coefficient of friction between the ladder an

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 l = 5;                  //(m)(Length of the ladder)
6 alpha = 70;                  //(degrees)(Angle which the
      ladder makes with the horizontal)
7 w1 = 900;               //(N)(Weight of the ladder)
8 w2 = 750;               //(N)(Weight of man)
9 d = 1.5;                //(m)(Distance between the man
      and bottom of ladder)
10
11 //Let uf = Coefficient of friction between ladder
      and floor
12 //Rf = Normal reaction at the floor
13
14 //Resolving the forces vertically:
15 Rf = w1 + w2;                  //(N)
16
17 //Force of friction at A:
18 //Ff = uf * Rf
```

```
19
20 //Taking moments about B and equating the same:
21 //Rf * 5 * sind(20) = (uf * 1650 * 5 * cosd(20)) +
      4875 * sind(20)
22 //Substituting values of Rf anf Ff and dividing both
      sides by 5 * sind(20)
23 mu_f = (Rf - (w2*3.5*sind(20)))/(1650*cotd(20));
24 printf("Coefficient of friction between the ladder
      and the floor = %.2f",mu_f);     //The answers
      vary due to round off error
```

Scilab code Exa 9.4 To calculate the force and inclination of the ladder

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 l = 3;     //(m)(Length of the ladder)
6 W = 200;   //(N)(Weight of the ladder)
7 uw = 0.25;     //(Coefficient of friction between the
      wall and the ladder)
8 uf = 0.35;     //(Coefficient of friction between the
      floor and ladder)
9 alpha = 60;      //(degrees)(Angle at A)
10
11 //(i) Horizontal force (P) applied to the ladder at
      floor level to prevent slipping:
12
13 //Resolving the forces horizontally:
14 //P + Ff = Rw                              -(i)
15
16 Rf1 = 1000;           //(N)(Top reaction force)
17 Fw1 = 200;            //(N)(Force at centre)
18
19 //Resolving the forces vertically:
```

```
20 //Rf + Fw = 1000 + 200 = 1200          -(ii)
21
22 //Taking moments about A and equating the same:
23 //1100 = Fw + Rw*tand(60)               -(iii)
24
25 //Also: Fw = uw * Rw = 0.25*Rw
26 //Substituting this in equation (iii):
27 Rw = 1100/(0.25 + tand(alpha));       //(N)
28
29 Fw = 0.25*Rw;          //(N)
30
31 //Substituting Fw in (ii):
32 Rf = (Rf1 + Fw1) - Fw;         //(N)
33 Ff = uf * Rf;          //(N)
34
35 //Substituting Ff in equation (i):
36 P = Rw - Ff;          //(N)
37 printf("The horizontal force P to be applied to
      ladder at the floor level to prevent slipping = %
      .2f N\n",P);
38
39 //(ii) Inclination of the ladder with the horizontal
       for no slipping:
40 //Resolving the forces horizontally:
41 //Rw = Ff = uf * Rf;       -(iv)
42
43 //Resolving the forces vertically:
44 //Rf + Fw = 1000 + 200 = 1200       (N)
45
46 //Fw = uw * Rw = 0.09*Rf
47 Rf = (Rf1 + Fw1)/(1 + uf * uw);      //(N)
48
49 Rw = uf * Rf;      //(N)
50 Fw = (uf * uw) * Rf;      //(N)
51
52 //Taking moments about A and equating the same:
53 alpha = atand((1100 - Fw)/Rw);    //(Degrees)
54 printf("Inclination of the ladder with the
```

```
horizontal for no slipping = %.2f degrees",alpha)
;
```

Scilab code Exa 9.6 To determine the minimum horizontal force

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5
6  //Solving by analytical method
7  W = 1500;     //(N)(Weight of the block)
8  alpha = 10;   //(Degrees)(Angle of the wedge)
9  mu = 0.3;      //(Coefficient of friction between all
       the four surfaces of contact)
10 phi = atand(mu);     //(Degrees)
11
12 //Let P = Minimum horizontal force required to raise
       the block.
13
14 //Considering the equilibrium of the block:
15 //Resolving the forces horizontally:
16 //R2 = 2.132*R1
17
18 //Resolving the forces vertically:
19 alpha = 16.7;      //(degrees)
20 theta = 26.7;      //(degrees)
21 R1 = W/((cosd(alpha)/sind(theta))*cosd(theta) - sind
       (alpha));      //(N)
22 R2 = (cosd(alpha)/sind(theta)) * R1;
                      //(N)
23
24 //Considering the equilibrium of the wedge:
25 //Resolving the forces vertically:
26 R3 = (R2 * cosd(theta))/cosd(alpha);     //(N)
```

```
27
28 //Resolving the forces horizontally:
29 P = R2 * sind(theta) + R3 * sind(alpha);     //(N)
30 printf("Minimum horizontal force required to raise
      the block = %.2f N",P);     //The answers vary due
       to round off error
```

Scilab code Exa 9.8 To find the effort required at the end of the handle

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 d = 50;           //(mm)(Mean diameter of screw jack)
6 r = 25;           //(mm)(Radius)
7 p = 10;           //(mm)(Pitch of screw)
8 u = 0.15;         //(Coefficient of friction between
      screw and nut)
9 phi = 8.5;          //(degrees)
10 l = 700;         //(mm)(Length of the handle)
11 W = 10;          //(kN)(Load to be raised)
12
13 //Let P1 = Effort required at the end of 700 mm long
       handle to raise the load.
14 //alpha = Helix angle;
15 alpha = atand((p)/(%pi * d));          //(degrees)
16
17 //Effort required at mean radius of the screw to
      raise the load:
18 P = W * tand(alpha + phi);       //(kN)
19
20 //Effort required at the end of the handle may be
      obtained from the relation:
21 P1 = (P * r)/l * 1000;            //(N)
22
```

```
23 printf("Effort required at the end of the handle = %
   .2f N",P1);     //The answers vary due to round
   off error
```

Scilab code Exa 9.9 To find the effort to be applied at the end of the lever

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  r = 2.5;          //(cm)(Mean radius of the screw)
6  p = .75;          //(cm)(Pitch of the thread)
7  mu = 0.12;        //(Coefficient of friction)
8  l = 60;           //(cm)(Length of the lever)
9  W = 2000;         //(N)(Weight to be raised)
10 phi = atand(mu);  //(degrees)
11
12 //Let P1 = Effort required at the end of the 60 cm
   long handle to raise the weight;
13 //alpha = Helix angle;
14 alpha = atand(p/(2*%pi*r));     //(degrees)
15
16 //Effort required at mean radius of the screw to
   raise the weight:
17 P = W * tand(alpha + phi);         //(N)
18
19 //Effort applied at the end of the lever:
20 P1 = (P * r)/(l);                  //(N)
21
22 printf("Effort applied at the end of the lever = %.2
   f N",P1);     //The answers vary due to round off
   error
```

Scilab code Exa 9.10 To find the torque

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 n = 2;                    //(No. of threads)
6 p = 4;                    //(mm)(Pitch)
7 r = 25;                   //(mm)(Mean radius)
8 mu = 0.3;                  //(Coefficient of friction)
9 phi = atand(mu);           //(degrees)
10 W = 500;                  //(N)(Pressure)
11
12 alpha = atand((n * p)/(2 * %pi * r));        //(
      degrees)
13
14 //Effort required at the mean radius of the screw to
       press the books:
15 P = W * tand(alpha + phi);           //(N)
16
17 //Torque required to press the books:
18 T = P * r;                 //(N-mm)
19 printf("Torque required to press the books = %.2f N-
      mm",T);                          //The answers vary
      due to round off error
```

Scilab code Exa 9.11 To find the force

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 d = 75;                   //(mm)(Mean diameter of thread)
6 r = 37.5;                 //(mm)(Radius)
7 p = 15;                   //(mm)(Pitch of thread)
```

```
l = 500;                //(mm)(Length of lever)
W = 25000;                 //(N)(Load to be lowered)
mu = 0.05;               //(Coefficient of friction
   between the screw and thread)
phi = atand(mu);         //(Degrees)

//alpha = Helix angle
alpha = atand(p/(%pi * d));              //(Degrees)

//Effort required at the mean radius of the screw to
    lower the load:
P = W * tand(alpha - phi);          //(N)

//Effort required at the end of the handle:
P1 = (P * r)/l;                          //(N)
printf("Effort required at the end of the handle = %
   .2f N",P1);                 //The answers vary due
   to round off error
```

Scilab code Exa 9.12 To find the force

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
D = 5;             //(cm)(Outer diameter of the screw)
mu = 0.1;           //(Coefficient of friction)
phi = atand(mu);       //(Degrees)
l = 70;            //(cm)(Length of the lever)
W = 4000;          //(N)(Load to be lifted)
p = 1/2;           //(cm)(Pitch of the screw)
d = 5 - (2 * 0.5);          //(cm)(Internal diameter
   of the screw)
md = (D + d)/2;         //(cm)(Mean diameter of the
   screw)
```

```
13
14 //Let alpha = Helix angle
15 alpha = atand(p/(%pi * md));       //(Degrees)
16
17 //(i)Force required at the end of 70 cm long lever
      to lift the load:
18
19 //Force required to be applied at the mean radius to
       lift the load:
20 P = W * tand(alpha + phi);        //(N)
21
22 //Force required at the end of the lever:
23 P1 = (P * md/2)/l;          //(N)
24
25 //(ii)Force required at the end of 70 cm long lever
      to lower the load:
26
27 //Force required at the mean radius to lower the
      load:
28 P = W * tand(phi - alpha);        //(N)
29
30 //Force required at the end of the lever:
31 P2 = (P * md/2)/l;          //(N)
32
33 printf("Force required at the end of 70 cm long
      lever to lift the load = %.2f N\n",P1);     //The
      answers vary due to round off error
34 printf("Force required at the end of 70 cm long
      lever to lower the load = %.2f N",P2);     //The
      answers vary due to round off error
```

Scilab code Exa 9.13 To find the efficiency of the screw jack

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
```

```
clc
clear all
W = 2.5;          //(kN)
d = 75;           //(mm)(Mean diameter of the screw)
p = 12;           //(mm)(Pitch of the screen)
mu = 0.075;        //(Coefficient of friction between
    the screw and nut)
phi = atand(mu);          //(Degrees)

alpha = atand(p/(%pi * d));       //(Degrees)

//Efficiency of the screw jack:
eta = tand(alpha)/tand(alpha + phi);
printf("Efficiency of the screw = %.2f %%",eta *
   100);    //The answers vary due to round off
   error
```

Scilab code Exa 9.14 To find the torque

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
d = 75;             //(mm)(Mean diameter of square
   thread)
r = 37.5;           //(mm)(Mean radius)
p = 15;             //(mm)(Pitch)
mu = 0.05;           //(Coefficient of friction)
phi = atand(mu);     //(Degrees)
Re = 360;           //(mm)(Radius of effort arm)
W = 6000;           //(N)(Load lifted)

//(i)Tangential force to be applied at the jack:
alpha = atand(p/(%pi * d));     //(Degrees)

```

```
16 //Tangential force required at the mean radius to
     lift the load:
17 P = W * tand(alpha + phi);            //(N)
18
19 //The effort applied at a radius of 36 cm:
20 P1 = (P * r)/Re;           //(N)
21 printf("The effort applied at a radius of 36 cm = %
     .2f N\n",P1);     //The answers vary due to round
     off error
22
23 //(ii)Self-locking of the screw:
24
25 //Efficiency of the screw jack:
26 eta = tand(alpha)/tand(alpha + phi);
27 printf("Efficiency of the screw jack = %.2f %%\n",
     eta * 100);     //The answers vary due to round
     off error
28 printf("Since efficiency of the screw jack is more
     than 50 %%, therefore it is not self-locking\n");
29
30 //Torque, which must be applied to keep the load
     from descending:
31 //Force that must be applied at the mean radius to
     keep the load from descending:
32 P2 = W * tand(alpha - phi);          //(N)
33
34 //Torque, which must be applied to keep the load
     from descending:
35 T = P2 * r;              //(N-m)
36 printf("Torque, which must be applied to keep the
     load from descending = %.2f N-mm",T);       //(N-
     mm)(Answer in textbook is wrong)
```

Chapter 10

Principles of Lifting Machines

Scilab code Exa 10.1 To find the mechanical advantage velocity ratio and efficienc

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 W = 1000;                          //(N)(Weight)
6 P = 25;                            //(N)(Effort)
7 x = 0.1;                           //(m)(Distance through
     which the weight is moved)
8 y = 8;                             //(m)(Distance through
     which the effort is moved)
9
10 //Mechanical advantage of the machine:
11 MA = W/P;
12 printf("Mechanical advantage of the machine = %.2f\n
     ",MA);
13
14 //Velocity ratio of the machine:
15 VR = y/x;
16 printf("Velocity ratio of the machine = %.2f\n",VR);
17
18 //Efficiency of the machine:
```

```
19 eta = MA/VR;
20 printf("Efficiency of the machine = %.2f %%",eta *
      100);
```

Scilab code Exa 10.2 To determine if the machine is reversible

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  VR = 30;                    //(Velocity Ratio)
6  W = 1500;                   //(N)(Load)
7  P = 125;                    //(N)(Effort)
8
9  MA = W/P;                  //(Mechanical Advantage)
10 eta = MA/VR;                 //(Efficiency)
11
12 printf("Efficiency = %.2f %%\n",eta * 100);
13 printf("Since efficiency of the machine is less than
       50 %%, therefore the machine is non-reversible."
      );
```

Scilab code Exa 10.3 To find the value of effort

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  VR = 50;                    //(Velocity Ratio)
6  P = 100;                    //(N)(Effort)
7  W = 4000;                   //(N)(Load)
8
9  //Reversibility of the machine:
```

```
10 MA = W/P;                    //(Mechanical Advantage)
11 eta = MA/VR;                  //(Efficiency)
12
13 printf("Efficiency = %.2f %%\n",eta * 100);
14 printf("Since efficiency of the machine is more than
       50 %%, therefore the machine is reversible.\n");
15
16 //Effort to be applied:
17 //Let P1 = Effort required to lift a load of 4000 N
      when the machine is at the point of reversing.
18
19 //MA = W/P1 = 4000/P1;
20 //Efficiency = MA/VR = (4000/P1)/50;
21 eta = 0.5;      //(Efficiency)
22 P1 = W/(VR * eta);              //(N)
23 printf("Effort required to lift a load of 4000 N
      when the machine is at the point of reversing = %
      .2f N",P1);
```

Scilab code Exa 10.4 To calculate the efficiency and friction

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 P = 100;                  //(N)(Effort)
6 W = 840;                  //(N)(Load)
7 VR = 10;                  //(Velocity ratio)
8
9 MA = W/P;                   //(Mechanical advantage)
10 n = MA/VR;                 //(Efficiency)
11 printf("Efficiency = %.2f %%\n",n * 100);
12
13 //Friction of the machine:
14 Feffort = P - W/VR;           //(N)
```

```
15
16 //Friction of the machine in terms of load:
17 Fload = (P * VR) - W;       //(N)
18
19 printf("Friction of the machine = %.2f N\n",Feffort)
      ;
20 printf("Friction of the machine in terms of load = %
      .2f N",Fload);
```

Scilab code Exa 10.5 To determine the load and the law of the machine

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 P = 120;                  //(N)(Effort)
6 VR = 18;                  //(Velocity ratio)
7 n = 0.6;                  //(Efficiency)
8 L = 3500;                 //(N)(Load to be lifted by
      machine)
9
10 //Load lifted by the machine:
11 //MA = W/P = W/120;
12 //n = MA/VR = (W/120)/18;
13 W = n * VR * P;              //(N)
14 printf("Load lifted by the machine = %.2f N\n",W);
15
16 //Law of the machine:
17 //In the second case:
18 P = 200;                     //(N)
19 W = 2600;                    //(N)
20
21 //Substituting the two values of P and W in the law
      of machine, i.e., P = m*W + C
22 //120 = m*1296 + C;                -(1)
```

```
23 //200 = m*2600 + C;                -(2)
24 //(2)-(1)
25 m = 80/1304;
26
27 //Substituting m in eqn (2):
28 C = P - m*W;
29
30 printf("Law of the machine: P = %.2f W + %.2f\n",m,C
   );     //(The answers vary due to round off error)
31
32 //Effort required to run the machine at a load of
   3.5 kN:
33 P = (m * L) + C;                //(N)
34 printf("Effort required to run the machine at a load
    of 3.5 kN = %.2f N",P);       //The answers vary
   due to round off error
```

Scilab code Exa 10.6 To find the effort required

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 P1 = 40;                  //(N)(Effort)
6 W1 = 1000;                //(N)(Load)
7 n = 0.5;                  //(N)(Efficiency)
8 P2 = 74;                  //(N)(Effort)
9 W2 = 2000;                //(N)(Load)
10 L = 5000;                //(N)
11
12 //Velocity ratio when the efficiency is 0.5:
13 MA = W1/P1;              //(Mechanical Advantage)
14
15 //n = MA/VR;
16 VR = MA/n;              //(Velocity ratio)
```

```
17 printf("Velocity ratio when the efficiency is 0.5 =
     %.2f\n",VR);
18
19 //Efficiency when P is 74 N and W is 2000 N:
20 MA = W2/P2;          //(Mechanical Advantage)
21 n = MA/VR;           //(Efficiency)
22
23 printf("Efficiency when P is 74 N and W is 2000 N =
     %.2f o/o\n",n * 100);
24
25 //Effort required to raise a load 5000 N
26 //Substituting the two values of P and W in the law
     if the machine i.e., P = m*W + C:
27 //40 = m * 1000 + C          -(1)
28 //74 = m * 2000 + C          -(2)
29 //(2)-(1):
30 m = 34/1000;
31
32 //Substituting this value of m ineqn (1):
33 C = P1 - m*W1;
34
35 printf("Law of machine: P = %.2f W + %.2f \n",m,C);
36
37 //Effort required to raise a load of 5000 N:
38 P = (m * L) + C;                //(N)
39 printf("Effort required to raise a load of 5000 N =
     %.2f N",P);
```

Scilab code Exa 10.7 To find the law of the machine and the effort required

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 P = 12;           //(N)(Effort)
```

```
6  VR = 18;          //(Velocity Ratio)
7  n = 0.6;          //(Efficiency)
8
9  //Load lifted by the machine:
10 //MA = W/P = W/12;
11 //n = MA/VR = (W/12)/18;
12 W = n * VR * P;          //(N)
13 printf("Load lifted by the machine = %.2f N\n",W);
14
15 //Law of the machine:
16 Feffort = P -  (W/VR);      //(N)
17 C = Feffort;
18
19 //Substituting the values of P and C in the law of
      machine:
20 m = (P - C)/W;
21 printf("Law of machine: P = %.2f W + %.2f\n",m,C);
22
23 //Effort required to run the machine at no load:
24 //Substituting the value of W = 0 in the law of
      machine:
25 P = 4.8;          //(N)
26 printf("Effort required to run the machine at no
      load = %.2f N\n",P);
27
28 //Effort required to run the machine at a load of
      900 N:
29 //Substituting the value of W = 900 in the law of
      machine:
30 W = 900;
31 P = m * W + C;
32 printf("Effort required to run the machine at a load
      of 900 N = %.2f N",P);
```

Scilab code Exa 10.8 To find the mechanical advantage velocity ratio efficiency of

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
W = 2000;                //(N)(Load)
VR = 40;                 //(Velocity ratio)
m = 0.04;
C = 7.5;

//Mechanical advantage:
//Substituting the value of W in the law of machine:
P = m*W + C;             //(N)
MA = W/P;                //(Mechanical advantage)
printf("Mechanical advantage = %.2f N\n",MA);     //
   The answers vary due to round off error

//Efficiency of the machine:
n = MA/VR;               //(Efficiency)
printf("Efficiency = %.2f o/o\n",n * 100);     //The
   answers vary due to round off error

//Maximum efficiency of the machine:
Maxn = 1/(m * VR);           //(Maximum efficiency)
printf("Maximum efficiency of the machine = %.2f o/o
   \n",Maxn * 100);

//Relation between F and W:
//Feffort = W*(m - 1/VR) + C;
printf("Feffort = %.2f W + %.2f\n",(m - 1/VR),C);

//Value of F when W is 2 kN:
W = 2000;        //(N)
F = ((m - 1/VR)*W) + C;          //(N)
printf("Value of F when W is 2 kN = %.2f N",F);
```

Scilab code Exa 10.9 To find the maximum possible mechanical advantage and efficie

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 VR = 100;                  //(Velocity ratio)
6 W = 600;                   //(N)(Load)
7
8 //Maximum possible mechanical advantage:
9 //Comparing the law of machine equation:
10 m = 0.02;
11 C = 8;
12 MaxMA = 1/m;
13 printf("Maximum possible mechanical advantage = %.2f
      \n",MaxMA);
14
15 //Maximum possible efficiency:
16 Maxn = 1/(m * VR);
17 printf("Maximum possible efficiency = %.2f %%\n",
      Maxn * 100);
18
19 //Effort required to lift a load of 600 N:
20 P = m * W + C;
21 Feffort = P - W/VR;              //(N)
22 printf("Effort required to lift a load of 600 N = %
      .2f N\n",Feffort);
23
24 //Efficiency of the machine:
25 MA = W/P;                  //(Mechanical advantage)
26 eta = MA/VR;                 //(Efficiency)
27 printf("Efficiency of the machine = %.2f %%",eta *
      100);
```

Scilab code Exa 10.10 To find the effort required and the efficiency of the machin

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
//From geometry of graph:
E1 = 142.5;         //(N)(Effort)
E2 = 42.5;          //(N)(Effort)
L1 = 1500;          //(N)(Load)
L2 = 250;           //(N)(Load)
m = (E1 - E2)/(L1 - L2);
C = 22.5;
VR = 18;              //(Velocity ratio)
printf("Law of the machine: P = %.2f W + %.2f\n",m,C);

//Effort required when the load is 2 kN:
L = 2000;                          //(N)(Load)
P = (m * L) + C;              //(N)
printf("Effort required when the load is 2 kN = %.2f N\n",P);

//Efficiency of the machine when the load is 2 kN:
W = 2000;                 //(N)(Load)
MA = W/P;                 //(Mechanical advantage)
eta = MA/VR;                //(Efficiency)
printf("Efficiency of the machine when the load is 2 kN = %.2f %%\n",eta * 100);     //The answers vary due to round off error

//Maximum efficiency the machine can attain:
Maxn = 1/(m * VR);              //(Maximum efficiency)
printf("Maximum efficiency the machine can attain = %.2f %%",Maxn * 100);
```

Chapter 11

Simple Lifting Machines

Scilab code Exa 11.1 To find the efficiency of the machine

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 D = 300;                  //(mm)(Diameter of wheel)
6 d = 30;                   //(mm)(Diameter of axle)
7 W = 900;                  //(N)(Load lifted by the
     machine)
8 P = 100;                  //(N)(Effort applied to lift
     the load)
9
10 VR = D/d;                //(Velocity ratio)
11 MA = W/P;                //(Mechanical advantage)
12 eta = MA/VR;               //(Efficiency)
13 printf("Efficiency = %.2f %%",eta * 100);
```

Scilab code Exa 11.2 To find the mechanical advantage velocity ratio and efficienc

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
W = 60 + 420;           //(N)(Total load to be lifted
    )
d = 100;                //(mm)(Diameter of the load
    axle)
D = 500;                //(mm)(Diameter of effort
    wheel)
P = 120;                //(N)(Effort)

//Mechanical advantage:
MA = W/P;
printf("Mechanical advantage = %.2f\n",MA);

//Velocity ratio:
VR = D/d;
printf("Velocity ratio = %.2f\n",VR);

//Efficiency of the machine:
eta = MA/VR;
printf("Efficiency = %.2f %%\n",eta * 100);

//If weight of water is only considered:
printf("If weight of water is only considered:\n");

MA = 420/P;             //(Mechanical advantage)
printf("Mechanical advantage = %.2f\n",MA);

eta = MA/VR;                    //(Efficiency)
printf("Efficiency = %.2f %%",eta * 100);
```

Scilab code Exa 11.3 To find the efficiency and the frictional effort loss

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 d1 = 80;              //(mm)(Larger diameter of the
     wheel)
6 d2 = 70;              //(mm)(Smaller diameter of the
     wheel)
7 D = 250;              //(mm)(Diameter of the effort
     wheel)
8 W = 1050;             //(N)(Load lifted)
9 P = 25;               //(N)(Effort)
10
11 //Velocity ratio:
12 VR = (2*D)/(d1 - d2);
13 printf("Velocity ratio = %.2f\n",VR);        //The
     answer provided in the textbook is wrong
14
15 //Efficiency:
16 MA = W/P;               //(Mechanical advantage)
17 eta = MA/VR;             //(Efficiency)
18 printf("Efficiency = %.2f %%\n",eta * 100);
19
20 //Frictional effort lost:
21 Feffort = P - W/VR;       //(N)
22 printf("Frictional Effort lost = %.2f N",Feffort);
```

Scilab code Exa 11.4 To determine the diameter of each axle

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 P = 6;                  //(N)(Effort)
6 W = 60;                 //(N)(Load raised)
```

```
7 n = 0.8;                  //(Efficiency)
8 D = 300;                  //(mm)(Diameter of effort wheel
    )
9 d = 280;                  //(mm)(Sum of the diameters of
    axles)
10
11 //Velocity ratio of the machine:
12 MA = W/P;                //(Mechanical advantage)
13 //0.8 = MA/VR = 10/VR;
14 VR = MA/n;               //(Velocity ratio)
15 printf("Velocity ratio = %.2f\n",VR);
16
17 //Difference between the diameters of the axles:
18 //Let d = d1 - d2;
19 d = (2 * D)/VR;                    //(mm)
20 printf("Difference between the diameters of the
    axles = %.2f mm\n",d);
21
22 //Diameter of each axle:
23 //Solving d1 - d2 = 48 and d1 + d2 = 280;
24 d1 = 164;          //(mm)
25 d2 = 116;          //(mm)
26 printf("d1 = %.2f mm\n",d1);
27 printf("d2 = %.2f mm",d2);
```

Scilab code Exa 11.5 To find the load

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 de = 400;             //(mm)(Diameter of effort wheel)
6 r1 = 150;             //(mm)(Radius of the axles)
7 r2 = 100;             //(mm)(Radius of the axles)
8 dr = 10;              //(mm)(Diameter of the rope)
```

```
d_a1 = 300;          //(mm)(Diameter of axle)
d_a2 = 200;          //(mm)(Diameter of axle)
P = 25;              //(N)(Effort)
eta = 0.84;            //(Efficiency)

//Let W = Load that can be lifted by the machine
//Effective diameter of the effort wheel:
D = de + dr;        //(mm)(Effective diameter of the
   wheel)

//Effective diameter of the axles:
d1 = d_a1 + dr;      //(mm)(Effective diameter of
   axle)
d2 = d_a2 + dr;      //(mm)(Effective diameter of
   axle)

//Velocity ratio of the differential wheel and axle:
VR = (2 * D)/(d1 - d2);       //(Velocity ratio)

//MA = W/P = W/25;

//n = MA/VR = (W/25)/8.2;
W = eta * VR * P;          //(N)
printf("Load that can be lifted by the machine = %.2
   f N",W);
```

Scilab code Exa 11.6 To calculate the effort

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
R = 125;              //(mm)(Radius of larger groove)
r = 115;              //(mm)(Radius of smaller groove
   )
```

```
7  eta = 0.8;                  //(Efficiency)
8  W = 1500;                  //(N)(Load to be raised)
9
10 //Let P = Effort required to raise the load
11 //Velocity ratio:
12 VR = (2 * R)/(R - r);           //(Velocity ratio)
13
14 //MA = W/P = 1500/P;            //(Mechanical
       advantage)
15
16 //n = MA/VR = (1500/P)/VR;
17 P = W/(VR * eta);                //(Effort required to
       raise the load)
18 printf("Effort required to raise the load = %.2f N",
       P);
```

Scilab code Exa 11.7 To find the mechanical advantage velocity ratio and efficienc

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  W = 1800;                  //(N)(Load)
6  P = 100;                   //(N)(Effort applied)
7  T1 = 12;                   //(Number of teeth on the larger
       block)
8  T2 = 11;                   //(Number of teeth on the
       smaller block)
9
10 //Velocity ratio:
11 VR = (2 * T1)/(T1 - T2);       //(Velocity ratio)
12 printf("Velocity ratio = %.2f\n",VR);
13
14 //Mechanical advantage:
15 MA = W/P;                        //(Mechanical advantage
```

```
    )
16 printf("Mechanical advantage = %.2f\n",MA);
17
18 //Efficiency of the machine:
19 eta = MA/VR;                    //(Efficiency)
20 printf("Efficiency = %.2f %%",eta * 100);
```

Scilab code Exa 11.8 To find the maximum load

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  T1 = 90;              //(No. of cogs on the effort
       wheel)
6  T2 = 25;              //(No. of teeth on the pinion)
7  T3 = 40;              //(No. of teeth on the spur wheel
       )
8  T4 = 8;               //(No. of cogs on the load wheel)
9  P = 50;               //(N)(Effort)
10 eta = 0.75;             //(Efficiency)
11
12 VR = (T1/T2) * (T3/T4);     //(Velocity ratio)
13 //MA = W/P = W/50;           //(Mechanical advantage)
14
15 //eta = MA/VR = (W/50)/18;
16 W = eta * P * VR;                //(Maximum load that
       can be lifted)
17 printf("Maximum load that can be lifted = %.2f N",W)
       ;
```

Scilab code Exa 11.9 To find the efficiency of the machine

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
T = 40;      //(No. of teeth on the worm wheel)
D = 300;     //(mm)(Diameter of the effort wheel)
r = 50;      //(mm)(Radius of load drum)
W = 1800;    //(N)(Load lifted)
P = 24;      //(N)(Effort)

VR = (D * T)/(2 * r);     //(Velocity ratio)
MA = W/P;                 //(MEchanical advantage)
eta = MA/VR;                //(Efficiency)
printf("Efficiency = %.2f %%",eta * 100);
```

Scilab code Exa 11.10 To determine the effort

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
n = 2;           //(No. of threads)
T = 60;           //(No. of teeth on the worm wheel)
D = 250;          //(mm)(Diamter of the effort wheel)
r = 50;           //(mm)(Radius of the load drum)
eta = 0.5;          //(Efficiency)
W = 300;          //(N)(Load to be lifted)

VR = (D * T)/(2 * n * r);     //(Velocity ratio)

//MA = W/P = 300/P;             //(Mechanical
   advantage)
//n = MA/VR = (300/P)/75;       //(Efficiency)
P = W/(VR * eta);                 //(Effort required
   to lift the load)
```

```
17 printf("Velocity ratio of the machine = %.2f\n",VR);
18 printf("Effort required to lift the load = %.2f N",P
   );
```

Scilab code Exa 11.11 To find the efficiency of the block

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 D = 200;          //(mm)(Diameter of effort wheel)
6 T = 60;           //(No. of teeth in worm wheel)
7 r = 40;           //(mm)(Radius of load drum)
8 W = 9000;         //(N)(Load to be lifted)
9 P = 75;           //(N)(Effort)
10
11 VR = (D * T)/r;       //(Velocity ratio)
12 MA = W/P;             //(Mechanical advantage)
13 eta = MA/VR;            //(Efficiency)
14 printf("Efficiency = %.2f %%",eta * 100);
```

Scilab code Exa 11.12 To find the load that can be lifted

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 n = 2;            //(No. of threads)
6 D = 400;           //(mm)(Diameter of the effort
     wheel)
7 r = 50;            //(mm)(Radius of load drum)
8 eta = 0.35;          //(Efficiency)
9 P = 80;            //(N)(Effort)
```

```
10 T = 50;              //(No. of teeth in worm wheel)
11
12 //Let W = Load which can be lifted by the machine
13 VR = (D * T)/(n * r);          //(Velocity ratio)
14
15 //MA = W/P = W/80;              //(Mechanical
      advantage)
16 ///eta = MA/VR = (W/80)/200;     //(Efficiency)
17 W = eta * P * VR;                //(Load which can be
       lifted by the machine)
18 printf("Load which can be lifted by the machine = %
      .2f N",W);
```

Scilab code Exa 11.13 To find the efficiency of the machine and the effect of fric

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 T2 = 25;          //(No. of teeth on pinion)
6 T1 = 100;         //(No. of teeth on spur wheel)
7 r = 50;           //(mm)(Radius of drum)
8 l = 300;          //(mm)(Length of the handle)
9 P = 20;           //(N)(Effort)
10 W = 300;         //(N)(Load lifted)
11
12 VR = (l/r) * (T1/T2);          //(Velocity ratio)
13 MA = W/P;                      //(Mechanical advantage
      )
14 eta = MA/VR;                     //(Efficiency)
15 printf("Efficiency = %.2f %%\n",eta * 100);
16
17 //Effect of friction:
18 Fload = (P * VR) - W;             //(N)
19 Feffort = P - (W/VR);             //(N)
```

```
20 printf("It means that if the machine would have been
       ideal(i.e. without friction) then it could lift
     an extraload of %.2f N with the same effort of 20
      N. Or it could have required %.2f N less force
     to lift the same load of 300 N.",Fload,Feffort);
```

Scilab code Exa 11.14 To find the law of the machine and the efficiency of the mac

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 l = 700;          //(mm)(Length of the lever)
6 T2 = 12;          //(No. of pinion teeth)
7 T1 = 96;          //(No. of spur teeth)
8 r = 100;          //(mm)(Radius of load axle)
9
10 //(i)Law of the machine:
11 P1 = 60;         //(N)
12 W1 = 1800;       //(N)
13 P2 = 120;        //(N)
14 W2 = 3960;       //(N)
15
16 //Substituing the values of P and W in the law of
      the machine: P = m*W + C
17 //60 = (m*1800) + C          -(1)
18 //120 = (m*3960) + C         -(2)
19 //(2)-(1)
20 m = P1/2160;
21 //Substituting this value of m in eqn (1)
22 C = P1 - m*W1;
23
24 printf("Law of the machine: P = %.2f W + %.2f\n",m,C
      );
25
```

```
26 //(ii)Efficiencies of the machine in both the cases:
27 VR = (l/r) * (T1/T2);     //(Velocity ratio)
28 //First case:
29
30 MA1 = W1/P1;                //(Mechanical advantage)
31 eta1 = MA1/VR;               //(Efficiency)
32 printf("Efficiency of the machine in first case = %
      .2f %%\n",eta1 * 100);     //The answers vary due
      to round off error
33
34 //Second case:
35
36 MA2 = W2/P2;               //(Mechanical advantage)
37 eta2 = MA2/VR;               //(Efficiency)
38 printf("Efficiency of the machine in second case = %
      .2f %%",eta2 * 100);
```

Scilab code Exa 11.15 To find the effort required

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 T2 = 20;          //(No. of teeth of pinion)
6 T4 = 25;
7 T1 = 50;          //(No. of teeth of spur wheel)
8 T3 = 60;
9 l = 0.5;          //(m)(Length of the handle)
10 r = 0.25;        //(m)(Radius of the load drum)
11 eta = 0.6;         //(Efficiency)
12 W = 720;         //(N)(Load to be lifted)
13
14 //Let P = Effort required in newton to lift the load
15 VR = (l/r)*(T1/T2)*(T3/T4);    //(Velocity ratio)
16 //MA = W/P = 720/P;            //(Mechanical
```

```
    advantage)
17 //eta = (720/P)/12;                    //(Efficiency)
18 P = W/(VR * eta);                     //(N)(Effort required
       in newton to lift the load)
19 printf("Effort required in newton to lift the load =
       %.2f N",P);
```

Scilab code Exa 11.16 To find the efficiency and the law of the machine

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  r = 80;          //(mmm)(Radius of the load drum)
6  l = 360;         //(mm)(Length of the handle)
7  T2 = 20;         //(No. of teeth on pinions)
8  T4 = 30;
9  T1 = 75;         //(No. of teeth on spur wheels)
10 T3 = 90;
11 P1 = 90;         //(N)
12 W1 = 1800;       //(N)
13 P2 = 135;        //(N)
14 W2 = 3150;       //(N)
15
16 //(a)Law of the machine:
17 //Substituting the values of P and W in the law of
       the machine i.e., P = m*W + C
18 //90 = m*1800 + C          -(1)
19 //135 = m*3150 + C         -(2)
20 //(2)-(1)
21 m = (P2-T3)/(W2-W1);
22 //Substituting this value of m in eqn (1):
23 C = T3 - m*W1;
24 printf("Law of the machine: P = %.2f W + %.2f\n",m,C
       );
```

```
25
26 //(b)Effort to lift a load of 4500 N:
27 W = 4500;                    //(N)
28 //Substituing the value of W= 4500 N in the law of
      the machine:
29 P = (m * W) + C;          //(N)
30 printf("Effort to lift a load of 4500 N = %.2f N\n",
      P);
31
32 //(c)Efficiency of the machine in the above case:
33 VR = (1/r)*(T1/T2)*(T3/T4);     //(Velocity ratio)
34 MA = W/P;                        //(Mechanical
      advantage)
35 eta = MA/VR;                      //(Efficiency)
36 printf("Efficiency = %.2f o/o\n",eta * 100);     //
      The answers vary due to round off error
37
38 //(d)Maximum efficiency of the machine:
39 eta_max = 1/(m * VR);      //(Maximum efficiency of
      the machine)
40 printf("Maximum efficiency of the machine = %.2f o/o
      ",eta_max * 100);     //The answers vary due to
      round off error
```

Scilab code Exa 11.17 To find the efficiency of the machine and the amount of fric

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 n = 3;            //(No. of pulleys)
6 W = 320;          //(N)(Weight lifted)
7 P = 50;           //(N)(Effort)
8 //Efficiency of the machine:
9 VR = 2^(n);       //(Velocity ratio of first system
```

```
    of pulleys)
MA = W/P;          //(Mechanical advantage)

eta = MA/VR;         //(Efficiency)
printf("Efficiency = %.2f %%\n",eta * 100);

//Amount of friction:
Fload = (P * VR) - W;          //(N)
printf("Amount of friction = %.2f N\n",Fload);

//Amount of friction in terms of effort:
Feffort = P - (W/VR);         //(N)
printf("Amount of friction in terms of effort = %.2f
    N",Feffort);
```

Scilab code Exa 11.18 To calculate the amount of effort wasted in friction and the

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
W = 1000;                  //(N)(Weight lifted)
P = 125;                   //(N)(Effort)
n = 2 * 5;                 //(No. of pulleys)

//Amount of effort wasted in friction:
VR = n;              //(Velocity ratio)

//Amount of effort wasted in friction:
Feffort = P - (W/VR);          //(N)
printf("Amount of effort wasted in friction = %.2f N
    \n",Feffort);

//Amount of frictional load:
Fload = (P * VR) - W;          //(N)
```

```
18 printf("Amount of frictional load = %.2f N",Fload);
       //The answer provided in the textbook is
   wrong
```

Scilab code Exa 11.19 To calculate the amount of effort wasted in friction

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 n = 4;          //(No. of pulleys)
6 W = 1800;       //(Load lifted)
7 eta = 0.75;       //(Efficiency)
8
9 //Effort required to lift the load:
10 VR = 2^n - 1;            //(Velocity ratio of third
   system of pulleys)
11 //MA = W/P = 1800/P;
12 //n = MA/VR;
13 P = W/(eta * VR);                //(Effort required to
   lift the load)
14 printf("Effort required to lift the load = %.2f N\n"
   ,P);
15
16 //Effort wasted in friction:
17 Feffort = P - (W/VR);          //(N)
18 printf("Effort wasted in friction = %.2f N",Feffort)
   ;
```

Scilab code Exa 11.20 To find the effort

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
```

```
clc
clear all
p = 10;          //(mm)(Pitch of thread)
l = 400;         //(mm)(Length of the handle)
W = 2000;        //(N)(Load lifted)
eta = 0.45;        //(Efficiency)
VR = (2 * %pi * l)/p;        //(Velocity ratio)
//MA = W/P = 2000/P;         //(Mechanical advantage)

//n = MA/VR;
P = W/(VR * eta);              //(N)(Effort applied)
printf("Effort applied = %.2f N",P);     //The
   answers vary due to round off error
```

Scilab code Exa 11.21 To find the efficiency of the machine

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
p1 = 12;            //(mm)(Pitch of the screw)
p2 = 10;            //(mm)
l = 300;            //(mm)(Arm length of screw jack)
W = 7500;           //(N)(Load lifted)
P = 30;             //(N)(Effort)

VR = (2 * %pi * l)/(p1 - p2);      //(Velocity ratio)
MA = W/P;                          //(Mechanical
   advanyage)
eta = MA/VR;                         //(Efficiency)
printf("Efficiency = %.2f o/o",eta * 100);
```

Scilab code Exa 11.22 To find the effort required

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 p1 = 10;          //(mm)(Pitch of the screw jack)
6 p2 = 7;           //(mm)
7 eta = 0.28;         //(Efficiency)
8 l = 360;          //(mm)(Arm length of screw jack)
9 W = 5000;         //(N)(Load lifted)
10 VR = (2 * %pi * l)/(p1 - p2);      //(Velocity ratio)
11 //MA = W/P = 5000/P;
12 //n = MA/VR;
13 P = W/(VR * eta);       //(Effort required to lift
    the load)
14 printf("Effort required to lift the load = %.2f N",P
    );    //The answers vary due to round off error
```

Scilab code Exa 11.23 To find the efficiency of the jack

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 l = 300;             //(mm)
6 T = 50;              //(No. of teeth in the worm wheel
    )
7 p = 10;              //(mm)(Pitch of screw)
8 P = 100;             //(N)(Effort)
9 W = 100000;          //(N)(Load lifted)
10 n = 2;             //(No. of threads)
11
12 VR = (2 * %pi * l * T)/(n * p);  //(Velocity ratio)
13 MA = W/P;                         //(MEchanical
    advantage)
14 eta = MA/VR;                        //(Efficiency)
```

```
15 printf("Efficiency = %.2f %%",eta * 100);
```

Chapter 12

Support Reactions

Scilab code Exa 12.1 To find the reactions at A and B

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 l = 5;                    //(m)(Span)
6 F1 = 3;                   //(kN)(Force at 2 m from A)
7 F2 = 4;                   //(kN)(Force at 3 m from A)
8 F3 = 5;                   //(kN)(Force at 4 m from A)
9 x = 2;                    //(m)(Distance from A to 3 kN
     force)
10 y = 3;                   //(m)(Distance from A to 4 kN
     force)
11 z = 4;                   //(m)(Distance from A to 5 kN
     force)
12 l = 5;                   //(m)(Total length)
13 //Let RA = Reation at A,
14 //RB = Reaction at B
15
16 //Anticlockwise moment due to RB about A
17 // M1 = l * RB = 5 * RB;          //(kN-m)          -(1)
18
```

```
19 //Sum of the clockwise moments about A:
20 M2 = (F1 * x) + (F2 * y) + (F3 * z);      //(kN-m)
      -(2)
21
22 //Equating (1) and (2):
23 RB = M2/l;                              //(kN)
24 RA = (F1 + F2 + F3) - RB;               //(kN)
25 printf("Reaction at A = %.2f kN\n",RA);
26 printf("Reaction at B = %.2f kN",RB);
```

Scilab code Exa 12.2 To find the reactions RA and RB

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 l = 6;          //(m)(Span)
6 //Let RA = Reaction at A
7 //RB = Reaction at B
8 //Anticlockwise moment due to the reaction RB about
     A:
9 //M1 = RB * l = 6 * RB;      (kN-m)     -(1)
10
11 FC = 4;      //(kN)(Force at C)
12 FD = 1.5;    //(kN)(Force at D)
13 F = 2;       //(kN/m)(Force per meter between C and E)
14 d = 1.5;     //(m)(Distance between the partitions)
15
16 //Sum of clockwise moments about A:
17 M2 = (FC * d) + (F * d)*d*d + (FD * 3*d);     //(kN-m
     )    -(2)
18
19 //Equating (1) and (2):
20 RB = M2/6;        //(kN)
21 RA = FC + (F * d) + FD - RB;       //(kN)
```

```
22 printf("Reaction at RA = %.2f kN\n",RA);
23 printf("Reaction at RB = %.2f kN",RB);
```

Scilab code Exa 12.3 To find the support reactions at A and B

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 l = 4.5;              //(m)(Span)
6 //Let RA = Reaction at A.
7 //RB = Reaction at B.
8 //Anticlockwise moment due to RB about A:
9 //M1 = RB * l = 4.5 * RB;      (kN-m)     -(1)
10 FA = 1;     //(kN/m)(Force per meter on side A)
11 FB = 2;     //(kN/m)(Force per meter on side B)
12
13 //Sum of clockwise moments due to uniformly varying
      load about A:
14 x = 2.25;     //(m)
15 M2 = (FA * l * x) + (x * 3);     //(kN-m)     -(2)
16
17 //Equating (1) and (2):
18 RB = M2/l;          //(kN)
19 RA = (FA * l) + (l * 1/2) - RB;     //(kN)
20 printf("Reaction at A = %.2f kN\n",RA);
21 printf("Reaction at B = %.2f kN",RB);
```

Scilab code Exa 12.4 To find the support reactions at A and B

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
```

```
4  clear all
5  l = 6;          //(m)(Span)
6  //Let RA = Reaction at A;
7  //RB = Reaction at B.
8  //Anticlockwise moment due to RB about A:
9  //M1 = l * RB = 6 * RB;          //(kN-m)     -(1)
10 FC = 4;      //(kN)(Force at C)
11 FD = 4;      //(kN)(Force at D)
12 FCD = 2;       //(kN/m)(Force per metre between C and
       D)
13 FB = 2;        //(kN/m)(Force per meter along B)
14
15 d1 = 1;      //(m)(Distance between the small
       partitions)
16 d2 = 3;      //(m)(Distance between the big partition)
17
18 //Sum of clockwise moments due to loads about A:
19 M2 = (FC * d1) + (FCD * d1)*1.5 + (FD * 2*d1) + FB
       /2*d2*5;   //(kN-m)     -(2)
20
21 //Equating (1) and (2):
22 RB = M2/l;                          //(kN)
23 RA = (FC + FCD + FD + 3) - RB;      //(kN)
24
25 printf("Reaction at A = %.2f kN\n",RA);
26 printf("Reaction at B = %.2f kN",RB);
```

Scilab code Exa 12.5 To determine the reactions at the supports A and B

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  l = 3;          //(m)(Span)
6  //Let RA = Reaction at A,
```

```
7 //RB = Reaction at B.
8 //Anticlockwise moment due to RB and load at C about
      A:
9 //M1 = RB * l + (1 * 1.5) = 3*RB + 1.5     (kN-m)
     -(1)
10 CA = 1.5;       //(m)(Distance between C and A)
11 AE = 2;         //(m)(Distance between A and E)
12 AB = 3;         //(m)(Distance between A and B)
13 BD = 1;         //(m)(Distance between B and D)
14 FAE = 2;        //(kN/m)(Force per meter between A and
      E)
15 FE = 3;         //(kN)(Force at E)
16 FBD = 2;        //(kN/m)(Force per meter between B and
      D)
17 FC = 1;         //(kN)(Force at C)
18
19 //Sum of clockwise moments due to loads about A:
20 M2 = (FAE * AE)*1 + (FE * AE) + (FC * BD)*3.5;  //(
     kN-m)    -(2)
21
22 //Equating (1) and (2):
23 RB = (M2 - CA)/l;                          //(kN)
24 RA = FC + (FAE * AE) + FE + (1 * BD) - RB;       //(
     kN)
25
26 printf("Reaction at A = %.2f kN\n",RA);
27 printf("Reaction at B = %.2f kN",RB);
```

Scilab code Exa 12.6 To determine the location of the two supports

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5
```

```
6 //Solved by using alternative method
7 L = 5;       //(m)(Length of the beam)
8 l = 3;       //(m)(Span)
9
10 //Let RC = Reaction at C,
11 //RC = Reaction at D,
12 //x = Distance of the support C from the left hand
      end;
13
14 //Sum of the anticlockwise moments due to RD and
      point load at A about the support C:
15 //M1 = (5.5 * 3) + (3 * x) = 16.5 + 3 *x;     -(1)
16
17 //Sum of clockwise moments due to the loads about C:
18 //M2 = 5*(5 - x) + 0.6 * 5*(2.5 - x) = 32.5 - 8*x;
        -(2)
19
20 //Equating (1) and (2):
21 //16.5 + 3*x = 32.5 - 8*x
22 x1 = 8;     //(Coefficient of x R.H.S)
23 x2 = 3;     //(Coefficient of x L.H.S)
24
25 c1 = 16.5;     //(Constant L.H.S)
26 c2 = 32.5;     //(Constant R.H.S)
27 A = [x1+x2];
28 c = [c1 - c2];
29 [x,nsA] = linsolve(A,c);
30 printf("It is thus obvious that the first support
      will be located at distance of %.2f from A and
      second support at a distance of %.2f m from A.",x
      ,x+l);
```

Scilab code Exa 12.7 To determine the reactions at A and B

```
1 //OS-version - Windows 10
```

```
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5
6  //Solving by analytical method
7  l = 6;          //(m)(Span)
8  AC = 2;         //(m)
9  CD = 2;         //(m)
10 DB = 2;         //(m)
11 F1 = 5;         //(kN)(Force at C)
12 F2 = 4;         //(kN)(Force at D)
13 F3 = 1.5;       //(kN/m)(Force per metre between C and
       D)
14 alpha = 135;   //(degrees)(Angle made at BD bt 4 kN
      force)
15 //Let RA = Reaction at A,
16 //RB = Reaction at B.
17 //Resolving the 4 kN load at D vertically:
18 RD = F2 * sind(180 - alpha);       //(kN)
19
20 //Resolving it horizontally:
21 RH = F2 * cosd(180 - alpha);       //(kN)
22
23 //Anticlockwise moment due to RB about A:
24 //M1 = RB * 6 = 6 * RB;        //(kN-m)    -(1)
25
26 //Sum of clockwise moments due to loads about A:
27 M2 = (F1 * AC) + (F3 * AC)*3 + RD*(AC + CD);     //(
      kN-m)    -(2)
28
29 //Equating (1) and (2):
30 RB = M2/l;         //(kN)
31
32 //Vertical component of the reaction RA:
33 VRA = F1 + (F3 * AC) + RD - RB;       //(kN)
34
35 RA = sqrt(VRA^2 + RH^2);     //(kN)
36
```

```
37 theta = atand(RH/VRA);           //(Degrees)(Angle
      which the reaction at A makes with vertical)
38
39 printf("Reaction at A = %.2f kN\n",RA);
40 printf("Reaction at B = %.2f kN\n",RB);
41 printf("Angle which the reaction at A makes with
      vertical = %.2f degrees",theta);     //The answers
       vary due to round off error
```

Scilab code Exa 12.8 To determine the reactions at A and B

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  l = 8.5;     //(m)(Total length)
6  F1 = 5;      //(kN)(Force at C)
7  F2 = 4;      //(kN)(Force at D)
8  F3 = 5;      //(kN)(Force at E)
9  AC = 2;      //(m)
10 CD = 2;      //(m)
11 DE = 3;      //(m)
12 EB = 1.5;    //(m)
13 phi = 45; //(degrees)(Angle at 4 kN)
14 alpha = 30; //(degrees)(Angle at B)
15 //Resolving the 4 kN load at D vertically:
16 RDV = F2*sind(phi);          //(kN)
17
18 //Resolving it horizontally:
19 RDH = F2*cosd(phi);          //(kN)
20
21 //Vertical component of reaction RB:
22 //VRB = RB * cosd(30) = 0.866*RB;
23
24 //Anticlockwise moment due to vertical component of
```

```
   reaction RB about A:
//M1 = 0.866*RB*8.5 = 7.361 * RB;     -(1)

//Sum of the clockwise moments due to loads about A:
M2 = (F1 * AC) + (RDV * (AC + CD)) + (F3 * (AC + CD
   + DE));      //(kN-m)    -(2)

///Equating (1) and (2):
RB = M2/(cosd(alpha)*l);          //(kN)

//Vertical component of the reaction RB:
VRB = cosd(alpha) * RB;        //(kN)

//Horizontal component of reaction RB:
HRB = RB * sind(alpha);    //(kN)

//Vertical component of reaction RA:
VRA = (F1 + F2*cosd(phi) + F3) - VRB;      //(kN)

//Horizontal component of reaction RA:
HRA = HRB - RDV;                 //(kN)

RA = sqrt(HRA^2 + VRA^2);        //(kN)

//Angle which the reaction at A makes with the
   vertical:
theta = atand(HRA/VRA);      //(Degrees)

printf("Reaction at A = %.2f kN\n",RA);
printf("Reaction at B = %.2f kN\n",RB);
printf("Angle which the reaction at A makes with the
    vertical = %.2f degrees",theta);
```

Scilab code Exa 12.9 To determine the reactions at A and B

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  l = 9;        //(m)(Span)
6  DC = 3;       //(m)
7  CA = 3;       //(m)
8  AE = 3;       //(m)
9  EF = 3;       //(m)
10 FB = 3;       //(m)
11 P1 = 3;       //(kN)(Force at D)
12 P2 = 12;      //(kN)(Force at E)
13 P3 = 9;       //(kN)(Force at F)
14 alpha = 30; //(degrees)(Angle at 12 kN)
15 beta = 30; //(degrees)(Angle at B)
16 //Let RA = Reaction at A,
17 //RB = Reaction at B.
18
19 //Resolving the 12 kN load at E vertically:
20 EV = P2 * sind(alpha);          //(kN)
21
22 //Resolving it horizontally:
23 EH = P2 * cosd(alpha);          //(kN)
24
25 //Vertical component of reaction RB:
26 // VRB = RB*cosd(30) = 0.866*RB;
27
28 //Anticlockwise moment due to vertical component of
      reaction RB about A:
29 //M1 = 0.866*RB * 9 = 7.794*RB;       -(1)
30
31 //Sum of clockwise moments due to loads about A:
32 M2 = (EV * AE) + (P3 * (AE + EF)) + (P1 * DC);    //
      (kN-m)     -(2)
33
34 //Equating (1) and (2):
35 RB = M2/(cosd(alpha)*P3);      //(kN)
36
```

```
37 //Vertical component of reaction RB:
38 VRB = RB*cosd(beta);          //(kN)
39
40 //Horizontal component of reaction RB:
41 HRB = RB*sind(beta);          //(kN)
42
43 //Vertical component of reaction RA:
44 VRA = (EV + VRB) - P3;          //(kN)
45
46 //Horizontal component of reaction RA:
47 HRA = (P1 + EH) - HRB;     //(kN)
48
49 RA = sqrt(HRA^2 + VRA^2);  //(kN)
50
51 //Angle which the reaction at A makes with the
      vertical:
52 theta = atand(HRA/VRA);        //(Degrees)
53
54 printf("Reaction at A = %.2f kN\n",RA);
55 printf("Reaction at B = %.2f kN\n",RB);     //The
      answers vary due to round off error
56 printf("Angle which the reaction at A makes with the
       vertical = %.2f degrees",theta);     //(The
      answers vary due to round off error)
```

Scilab code Exa 12.10 To determine the direction and the magnitude of the resultan

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 l = 9;          //(m)(Span)
6 FC = 6;         //(kN)(Force at C)
7 FB = 9;         //(kN-m)
8 d = 3;          //(m)(Distance between partitions)
```

```
alpha = 45;    //(degrees)

//Let RA = Reaction at A,
//RD = Reaction at D.

//Vertical component of reaction RD:
//VRD = RD*cosd(45) = 0.707*RD;

//Anticlockwise moment due to the vertical component
    of reaction RD about A:
//M1 = 6.363*RD;        -(1)

//Sum of clockwise moments due to moment at B and
   Load at C about A:
M2 = FB + (FC * 2*d);          //(kN-m)     -(2)

//Equating (1) and (2):
RD = M2/(cosd(alpha)*FB);             //(kN)

//Vertical component reaction RD:
VRD = RD*cosd(alpha);     //(kN)

//Horizontal component of RD:
HRD = RD*sind(alpha);     //(kN)

//Vertical component reaction of RA:
VRA = 6 - 5;              //(kN)

RA = sqrt(HRD^2 + VRA^2);         //(kN)

//Angle which the reaction at A makes with the
   vertical:
T = atand(HRD/VRA);               //(Degrees)

printf("Reaction at A = %.2f kN\n",RA);
printf("Angle which the reaction at A makes with the
    vertical = %.2f degrees",T);     //The answer
   vary due to round off error
```

Scilab code Exa 12.11 To find the reactions at the two supports

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  l = 9;          //(m)(Span)
6  //Let RA = Reaction at A,
7  //RB = Reaction ata B.
8
9  FC = 1;      //(kN)(Force at C)
10 FD = 2;      //(kN)(Force at D)
11 FE = 1;      //(kN)(Force at E)
12 FF = 2;      //(kN)(Force at F)
13 FG = 5;      //(kN)(Force at G)
14
15 AC = 2.25;    //(m)(Perpendicular distance between A
      and the lines of action of the loads at C)
16 AD = 4.5;     //(m)(Perpendicular distance between A
      and the lines of action of the loads at D)
17 AE = 6.75;    //(m)(Perpendicular distance between A
      and the lines of action of the loads at CE)
18 d = 3;        //(m)(Distance between the forces at
      the base)
19 //Equating the anticlockwise and clockwise moments
      about A:
20 RB = [(FC * AC) + (FD * AD) + (FE * AE) + (FF * d) +
     (FG * 2*d)]/9;          //(kN)
21 RA = (FC+FD+FE+FF+FG) - RB;      //(kN)
22
23 printf("Reaction at A = %.2f kN\n",RA);
24 printf("Reaction at B = %.2f kN",RB);
```

Scilab code Exa 12.12 To determine the reactions at A and C

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 l = 4;          //(m)(Span)
6 //Let RA = Reaction at A,
7 //RC = Reaction at C.
8 FB = 8;         //(kN)(Force at B)
9 FD = 12;        //(kN)(Force at D)
10 h = 1.5;       //(m)(Height of the triangle formed)
11 d = l/2;
12 //Taking moments about A and equating the same:
13 RC = [(FB * h) + (FD * d)]/4;      //(kN)
14
15 //Vertical component of reaction RA:
16 VA = FD - RC;       //(kN)
17
18 //Horizontal reaction at the left hand support A:
19 HA = 8;           //(kN)
20
21 //Reaction at A:
22 RA = sqrt(HA^2 + VA^2);     //(kN)
23
24 //Angle which the reaction RA makes with the
    vertical:
25 theta = atand(HA/VA);           //(Degrees)
26
27 printf("Reaction at A = %.2f kN\n",RA);
28 printf("Reaction at C = %.2f kN\n",RC);
29 printf("Angle which the reaction RA makes with the
    vertical = %.2f degrees",theta);
```

Scilab code Exa 12.14 To find the reactions at A and E

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 l = 8;          //(m)(Span)
6 //Let RA = Reaction at A,
7 //RE = Reaction at E.
8
9 FB = 1;       //(kN)(Force at B)
10 FC = 1;      //(kN)(Force at C)
11 FD = 4;      //(kN)(Force at D)
12 x = 1.5;     //(m)(Distance between the horizontal
      strips)
13 y = 2;       //(m)(Distance between the vertical
      strips)
14
15 //Taking moments about A and equating the same:
16 RE = [(FB * y) + (FC * 2*y) + (FD * 3*x)]/8;     //(
      kN)
17
18 //Vertical component of reaction:
19 VA = RE - 2;          //(kN)
20
21 //Horizontal reaction at the left hand support:
22 HA = 4;              //(kN)
23
24 //Reaction at A:
25 RA = sqrt(HA^2 + VA^2);      //(kN)
26
27 //Angle which the reaction at A makes with the
      vertical:
28 theta = atand(HA/VA);              //(Degrees)
```

```
29
30 printf("Reaction at A = %.2f kN\n",RA);
31 printf("Reaction at E = %.2f kN\n",RE);
32 printf("Angle which the reaction at A makes with the
       vertical = %.2f degrees",theta);
```

Scilab code Exa 12.15 To determine the reactions at the two supports

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5
6 //Solving by analytical method
7 l = 10;        //(m)(Span)
8 alpha = 30;    //(degrees)(Angle at A)
9 F1 = 1;        //(kN)(Force at A)
10 F2 = 2;       //(kN)(Force at C)
11 F3 = 1;       //(kN)(Force at D)
12 //Let RA = Reaction at A,
13 //RB = Reaction at B.
14
15 //From the geometry of the figure, the distance
      between the support A and the line of action of
      the load at D:
16 d1 = (l/2)/cosd(alpha);     //(m)
17
18 //Perpendicular distance between the support A and
      the line of action of the load at C:
19 d2 = d1/2;            //(m)
20
21 //Equating the anticlockwise moments and the
      clockwise moments about A:
22 RB = [(F2 * d2) + (F3 * d1)]/l;      //(kN)
23
```

```
24 W = F1 + F2 + F3;        //(Total wind load)
25
26 HW = W*cosd(60);     //(Horizontal component of the
      total wind load)
27
28 VW = W*sind(60);     //(Vertical component of the
      total wind load)
29
30 //Balance vertical reaction at A:
31 BA = VW - RB;         //(kN)
32
33 //Reaction at A:
34 RA = sqrt(HW^2 + BA^2);      //(kN)
35
36 //Angle which the reaction RA makes with the
      vertical:
37 theta = atand(HW/BA);             //(Degrees)
38
39 printf("Reaction at A = %.2f kN\n",RA);
40 printf("Reaction at B = %.2f kN\n",RB);
41 printf("Angle which the reaction RA makes with the
      vertical = %.2f degrees",theta);     //The answers
       vary due to round off error
```

Scilab code Exa 12.16 To find the reaction at P and Q

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 l = 6.92;         //(m)
6 FR = 1;           //(kN)(Force at R)
7 FS = 2;           //(kN)(Force at S)
8 FQ = 1;           //(kN)(Force at Q)
9 d = 3;            //(m)(Distance between adjacent
```

```
    forces)
alpha = 60;      //(degrees)

//Let RP = Reaction at P,
//RQ = Reaction at Q.

//Taking moments about Q and equating the same:
RP = [(FS * d) + (FQ * 2*d)]/1;      //(kN)

W = FR + FS + FQ;      //(kN)(Total wind load)

//Horizontal component of total wind load:
HW = W*cosd(alpha);     //(kN)

//Vertical component of total wind load:
VW = W*sind(alpha);     //(kN)

//Balance vertical reaction at Q:
VQ = VW - RP;       //(kN)

//Reaction at Q:
RQ = sqrt(HW^2 + VQ^2);      //(kN)

//Angle which the reaction RQ makes with the
    vertical:
theta = atand(HW/VQ);           //(Degrees)

printf("Reaction at P = %.2f kN\n",RP);
printf("Reaction at Q = %.2f kN\n",RQ);
printf("Angle which the reaction RQ makes with the
    vertical = %.2f degrees",theta);
```

Scilab code Exa 12.17 To find the reactions at A and D

```
//OS-version - Windows 10
```

```
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  AB = 3;       //(m)
6  BC = 3;       //(m)
7  FE = BC;      //(m)
8  CE = 4;       //(m)
9  ED = 3;       //(m)
10 F1 = 5;       //(kN)(Force at F)
11 F2 = 2;       //(kN)(Force at E)
12 theta = 45; //(degrees)(Angle at D)
13
14 //Let RA = Reaction at A,
15 //RD = Reaction at D.
16
17 //Horizontal component of reaction at D:
18 //RDH = RDV = 0.707*RD;
19
20 //Now, taking moments about A and equating the same:
21 //RDV*9 - RDH*4 = (5 * 3) + (2 * 6)
22 RD = (F1 * FE + F2 * (FE + ED))/((9 - 4)*cosd(theta)
      );        //(kN)
23
24 RDH = cosd(theta)*RD;
25 RDV = cosd(theta)*RD;
26
27 //Vertical component of reaction at A:
28 RAV = (F1 + F2) - cosd(theta)*RD;      //(kN)
29
30 //Horizontal component of reaction at A:
31 RAH = RDH;
32
33 RA = sqrt(RAV^2 + RAH^2);      //(kN)
34
35 printf("Reaction at A = %.2f kN\n",RA);
36 printf("Reaction at D = %.2f kN",RD);
```

Scilab code Exa 12.18 To find the reactions at the supports

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 l = 8;          //(m)(Span of the truss)
6 F1 = 1;         //(kN)(Force at A)
7 F2 = 2;         //(kN)(Force at C)
8 F3 = 1;         //(kN)(Force at D)
9 l = 8;          //(m)(Total length)
10 theta = 30;    //(degrees)(Angle at A, B and angle
    formed at C and E)
11 //Let RA = Reaction at the left support A,
12 //RB = Reaction at the right support B.
13
14 //Equating the anticlockwise and clockwise moments
    about A:
15 RB = [2*2/cosd(theta) + 1*4/cosd(theta)]/(l * sind
    (2*theta));   //(kN)
16 RA = (F1 + F2 + F3) - RB;          //(kN)
17
18
19 //Total horizontal component of the loads:
20 H_Sum = F1*cosd(2*theta) + F2*cosd(2*theta) + F3*
    cosd(2*theta);   //(kN)
21
22 //Horizontal thrust on each support:
23 RAH = 1;          //(kN)
24 RBH = 1;          //(kN)
25
26 //Equating the anticlockwise and clockwise moments
    about A:
27 RBV = [2*2/cosd(theta) + 1*4/cosd(theta)]/l;      //(
```

```
    kN)
28 RAV = [F1*sind(2*theta) + F2*sind(2*theta) + F3*sind
    (2*theta)] - RBV;     //(kN)
29
30 //Reaction at A:
31 RA = sqrt(RAH^2 + RAV^2);      //(kN)
32
33 //Angle which the reaction RA makes with the
    vertical:
34 theta_A = atand(RAV/RAH);           //(Degrees)
35
36 //Reaction at B:
37 RB = sqrt(RBH^2 + RBV^2);      //(kN)
38
39 //Angle which the reaction RB makes with the
    vertical:
40 theta_B = atand(RBV/RBH);           //(Degrees)
41
42 printf("Reaction at A = %.2f kN\n",RA);
43 printf("Angle which the reaction RA makes with the
    vertical = %.2f degrees\n",theta_A);     //The
    answers vary due to round off error
44 printf("Reaction at B = %.2f kN\n",RB);     //The
    answers vary due to round off error
45 printf("Angle which the reaction RB makes with the
    vertical = %.2f degrees",theta_B);     //The
    answers vary due to round off error
```

Chapter 13

Analysis of Perfect Frames Analytical Method

Scilab code Exa 13.1 To find the forces in the members

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 F = 10;      //(kN)(Force acting at A)
6 FB = 7.5;    //(kN)(Force acting at B)
7 FC = 2.5;    //(kN)(Force acting at C)
8 alpha = 60; //(degrees)(Angle at B)
9 theta = 30; //(degrees)(Anlge at C)
10
11 //From geometry:
12 BC = 5;      //(m)
13 AB = 1.25;   //(m)
14 AC = 3.75;   //(m)
15
16 //Taking moments about B and equating the same:
17 RC = (F * AB)/BC;       //(kN)
18 RB = F - RC;            //(kN)
19
```

```
20 //Method of Joints:
21 //Resolving the forces vertically and equating the
      same:
22 PAB = RB/sind(alpha);       //(kN)(Compression)
23
24 //Resolving the forces horizontally and equating the
      same:
25 PBC = PAB * cosd(alpha);      //(kN)(Tension)
26
27 //Resolving the forces vertically and equating the
      same:
28 PAC = RC/sind(theta);       //(kN)(Compression)
29 printf("Method of joints :\n")
30 printf("Force in the member AB = %.2f kN(Compression
      )\n",PAB);
31 printf("Force in the member AC = %.2f kN(Compression
      )\n",PAC);
32 printf("Force in the member BC = %.2f kN(Tension)\n\
      n",PBC);
33
34 //Method of Sections
35
36 PAB = FB/sind(alpha);       //(kN)(Compression)
37 PBC = FB/tand(alpha);       //(kN)(Tension)
38 PAC = FC/sind(theta);       //(kN)(Compression)
39 printf("Method of sections :\n")
40 printf("Force in the member AB = %.2f kN(Compression
      )\n",PAB);
41 printf("Force in the member AC = %.2f kN(Compression
      )\n",PAC);
42 printf("Force in the member BC = %.2f kN(Tension)\n\
      n",PBC);
```

Scilab code Exa 13.2 To find the forces in the members

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
F1 = 2;       //(kN)(Force at B)
F2 = 4;       //(kN)(Force at C)
AD = 6;       //(m)
AE = 3;       //(m)
ED = 3;       //(m)
theta = 60;   //(degrees)(Angles at the base of the triangles)

//Taking moments about A and equating the same:
RD = [(F1 * AE/2) + (F2 * (AE + ED/2))]/AD;      //(kN)
RA = (F1 + F2) - RD;                    //(kN)

//Method of joints:
//Resolving the forces vertically and equating the same:
PAB = RA/sind(theta);     //(kN)(Compression)

//Resolving the forces horizontally and equating the same:
PAE = PAB * cosd(theta);     //(kN)(Tension)

//Resolving the forces vertically and equating the same:
PCD = RD/sind(theta);       //(kN)(Compression)

//Resolving the forces horizontally and equating the same:
PDE = PCD * cosd(theta);      //(kN)(Tension)

//Resolve the forces vertically and equating the same:
PBE = [PAB*sind(theta) - PDE]/sind(theta);      //(kN)(Tension)
```

```
31
32 //Resolving the forces horizontally and equating the
       same:
33 PBC = PAB*cosd(theta) + PBE*cosd(theta);       //(kN)
     (Compression)
34
35 //Resolving the forces horizontally and equating the
       same:
36 PCE = (F2 - PCD*sind(theta))/sind(theta);      //(kN)
     (Compression)
37
38 printf("Force in the member AB = %.2f kN(Compression
     )\n",PAB);
39 printf("Force in the member AE = %.2f kN(Tension)\n"
     ,PAE);
40 printf("Force in the member CD = %.2f kN(Compression
     )\n",PCD);
41 printf("Force in the member DE = %.2f kN(Tension)\n"
     ,PDE);
42 printf("Force in the member BE = %.2f kN(Tension)\n"
     ,PBE);
43 printf("Force in the member BC = %.2f kN(Compression
     )\n",PBC);
44 printf("Force in the member CE = %.2f kN(Compression
     )",PCE);
45 //The answers vary due to round off error
```

Scilab code Exa 13.3 To determine the nature and magnitude of the forces

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 FD = 1500;     //(N)(Force at D)
6 //Taking moments about A and equating the same:
```

```
7  n = 4;              //(No. of partitions)
8  VB = FD/n;          //(N)
9  VA = FD -VB;        //(N)
10 //h = 2.25*a;
11 H = 2.25;
12
13 //From geometry:
14 //y = 2.25a;
15 //x = 3a;
16 X = 3;
17 //theta = atand(y/x);
18 theta = atand(H/X);        //(Degrees)
19
20 //Taking moments about joint M and equating the same
   :
21 P1 = VB/sind(theta);        //(N)(Compression)
22
23 //Taking moments about joint A and equating the same
   :
24 P2 = FD/2;              //(N)(Tension)
25
26 //Taking moments about the joint L, and equating the
    same:
27 P3 = P2/(X/2);              //(N)(Tension)
28
29 printf("Force in the member 1 = %.2f N(Compression)\
   n",P1);
30 printf("Force in the member 2 = %.2f N(Tension)\n",
   P2);
31 printf("Force in the member 3 = %.2f N(Tension)\n",
   P3);
```

Scilab code Exa 13.4 To determine the nature and magnitude of the forces

```
1 //OS-version - Windows 10
```

```
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  FB = 8;      //(kN)(Force at B)
6  FG = 6;      //(kN)(Force at G)
7  FF = 12;     //(kN)(Force at F)
8  AB = 1.5;    //(m)
9  d = 2;       //(m)(Distance between the partitions)
10 alpha = 60;    //(Degrees)(Angles at A and E)
11 theta = 30;    //(Degrees)(Angles at partitions)
12
13 //Taking moments about A and equating the same:
14 RE = [(FB * AB) + (FG * d) + (FF * 2*d)]/(3*d);
          //(kN)
15 RA = (FB + FG + FF) - RE;                          //(
   kN)
16
17 //Taking moments about the joint G and equating the
   same:
18 PBC = [(RA * d) - (FB * 0.5)]/(d*sind(theta));    //
   (kN)(Compression)
19
20 //Taking moments about the joint B and equating the
   same:
21 PGC = [(RA * AB) + (FG * 0.5)]/((d/2)*cosd(theta));
        //(kN)(Compression)
22
23 //Taking moments about the joint C and equating the
   same:
24 PGF = [(RA * 2*AB) - (FG * (d/2))]/((d + d/2)*tand(
   theta));     //(kN)(Tension)
25
26 printf("Force in the member BC = %.2f kN(Compression
   )\n",PBC);
27 printf("Force in the member GC = %.2f kN(Compression
   )\n",PGC);
28 printf("Force in the member GF = %.2f kN(Tension)",
   PGF);    //THe answers vary due to round off
```

error

Scilab code Exa 13.5 To determine the magnitude and nature of forces

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 F1 = 10;     //(kN)(Force at top)
6 //Reaction against A:
7 RA = 5;      //(kN)
8 RB = 5;      //(kN)
9 alpha = 45;     //(degrees)
10
11 theta = atand(3/3);           //(Degrees)
12 alpha = atand(6/3);       //(Degrees)
13
14 //Resolving the forces horizontally and equating the
     same:
15 //PAC = PAD*cosd(45)/cosd(alpha) = 1.58*PAD;
16
17 //Resolving the forces vertically and equating the
     same:
18 //1.58*PAD * 0.8941 = 5 + PAD*0.707;
19 PAD = RA/0.7056;          //(kN)(Tension)
20 PAC = 1.58 * PAD;       //(kN)(Compression)
21 PBD = PAD;
22 //Resolving the forces vertically and equating the
     same:
23 PCD = PAD*sind(alpha) + PBD*sind(alpha);     //(kN)(
     Tension)
24
25 printf("Force in the member AD and DB = %.2f kN(
     Tension)\n",PAD);
26 printf("Force in the member AC and CB = %.2f kN(
```

```
    Compression)\n",PAC);     //The answers vary due
    to round off error
27 printf("Force in the member CD = %.2f kN(Tension)",
    PCD);     //The answers vary due to round off
    error
```

Scilab code Exa 13.6 To find the forces in the members

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  FA = 10;      //(kN)(Force at A)
6  CD = 3;       //(m)
7  AD = 3;       //(m)
8  alpha = 60;   //(degrees)(Angles at A and D)
9  //Method of Joints:
10 //Resolving the forces vertically and equating the
    same:
11 PAB = FA/sind(alpha);      //(kN)(Tension)
12
13 //Resolving the forces horizontally and equating the
    same:
14 PAD = PAB * cosd(alpha);        //(kN)(Compression)
15
16 //Resolving the forces vertically and equating the
    same:
17 PBD = (PAB * sind(alpha))/sind(alpha);       //(kN)(
    Compression)
18
19 //Resolving the forces horizontally and equating the
    same:
20 PBC = PAB * cosd(alpha) + PBD * cosd(alpha);     //(
    kN)(Tension)
21
```

```
22 printf("Solved using method of joints.\n");
23 printf("Force in the member AB = %.2f kN(Tension)\n"
   ,PAB);
24 printf("Force in the member AD = %.2f kN(Compression
   )\n",PAD);
25 printf("Force in the member BD = %.2f kN(Compression
   )\n",PBD);
26 printf("Force in the member BC = %.2f kN(Tension)\n\
   n",PBC);
27
28 //Method of sections
29 //Taking moments of the forces acting on right part
   of truss only, about the joint D and equating the
    same
30 PAB = (FA * CD)/(AD * sind(alpha));
31
32 //Now taking moments of the forces in the right part
    of the truss only about the joint B and equating
    the same
33 PAD = (FA * AD/2)/(CD * sind(alpha));
34
35 //taking moments of the forces in the right part of
   the truss only about the joint D and equating the
    same
36 PBC = (FA * CD)/(AD * sind(alpha));
37
38 //Now taking moments of the forces in the right part
    of the truss only about the joint C and equating
    the same
39 PBD = ((FA * CD) - (PAD * AD * sind(alpha)))/(AD/2 *
    sind(alpha));
40
41 printf("Solved using method of sections.\n");
42 printf("Force in the member AB = %.2f kN(Tension)\n"
   ,PAB);
43 printf("Force in the member AD = %.2f kN(Compression
   )\n",PAD);
44 printf("Force in the member BD = %.2f kN(Compression
```

```
    )\n",PBD);
45 printf("Force in the member BC = %.2f kN(Tension)",
    PBC);
```

Scilab code Exa 13.7 To find the value of W

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  F = 15;      //(kN)(Force to be produced)
6  //Solving by method of section:
7  //Taking moments of the forces in the right part of
       the truss only, about the joint E and equating
       the same:
8  //PAB = [(W*1.5) + (W*4.5)]/2;
9  //The value of W which would produce the force of 15
       kN in the member AB = W/3W * 15
10 //PAB = 3*W
11 x = 3;
12 W = F/x;       //(kN)
13 printf("The value of W which would produce the force
       of 15 kN in the member AB = %.2f kN",W);
```

Scilab code Exa 13.8 To find the forces in all the members of the truss

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  AG = 1.5;      //(m)
6  GF = 1.5;      //(m)
7  FE = 1.5;      //(m)
```

```
ED = 1.5;       //(m)
FB = 1000;      //(N)(Force at B)
FC = 1000;      //(N)(Force at C)
FD = 500;       //(N)(Force at D)

//From geometry:
CDE = atand(AG/(GF + FE + ED));     //(Degrees)

//Resolving the forces vertically at D:
PCD = FD/sind(CDE);      //(N)(Tension)

//Resoving the forces horizontally at D:
PDE = PCD*cosd(CDE);      //(N)(Compression)

//Resolving the forces horizontally and equating the
    same:
//PBC = 1584 + PFC;              -(1)

//Resolving the forces vertically and equating the
   same:
//1000 + 1584*sind(18.4) = PFC*sind(18.4) + PBC*sind
   (18.4);
PFC = FC/(2*sind(CDE));         //(N)(Compression)

//Substituting the value of PFC in equation (1):
PBC = PCD + PFC;          //(N)(Tension)

//Resolving the forces horizontally:
PGF = PCD + PCD*cosd(CDE);      //(N)(Compression)

//Resolving the forces vertically and equating the
   same:
PBF = PCD*sind(CDE);              //(N)(Tension)

//From geometry:
GBF = atand(GF/1);                //(Degrees)

//Resolving the forces horizontally at B and
```

```
   equating the same:
//0.3156*PAB = 0.2773*PBG + 1000;      -(2)

//Resolving the forces vertically at B and equating
   the same:
//0.3156*PAB + 0.5548*PBG = 2500;      -(3)
//(3)-(2):
PBG = 1500/0.8321;            //(N)(Compression)

//Substituting PBG in (2):
PAB = 1500/0.3156;           //(N)(Tension)

PCE = 0;
PFE = PDE;

printf("Force in the member AB = %.2f kN(Tension)\n"
   ,PAB);
printf("Force in the member BC = %.2f kN(Tension)\n"
   ,PBC);
printf("Force in the member CD = %.2f kN(Tension)\n"
   ,PCD);
printf("Force in the member DE = %.2f kN(Compression
   )\n",PDE);
printf("Force in the member CE = %.2f kN\n",PCE);
printf("Force in the member FE = %.2f kN(Compression
   )\n",PFE);
printf("Force in the member FC = %.2f kN(Compression
   )\n",PFC);
printf("Force in the member BF = %.2f kN(Tension)\n"
   ,PBF);
printf("Force in the member GF = %.2f kN(Compression
   )\n",PGF);
printf("Force in the member BG = %.2f kN(Compression
   )",PBG);
//The answers vary due to round off errors
```

Scilab code Exa 13.9 To find the forces in the members

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 FE = 5;     //(kN)(Force at E)
6 x = 2;      //(m)(Distance between the horizontal
     partitions)
7 y = 4;      //(m)(Distance between the vertical
     partitions)
8
9 //Taking moments about the joint D and equating the
     same:
10 PCE = (FE * y)/x;          //(kN)(Tension)
11
12 //Taking moments about the joint B and equating the
     same:
13 PCD = [(FE*2*y) - (PCE*2)]/y;      //(kN)(
     Compression)
14
15 //Taking moments about the joint C and equating the
     same:
16 PBD = (FE * y)/x;          //(kN)(Tension)
17
18 printf("Force in the member CE = %.2f kN(Tension)\n"
     ,PCE);
19 printf("Force in the member CD = %.2f kN(Compression
     )\n",PCD);
20 printf("Force in the member BD = %.2f kN(Tension)",
     PBD);
```

Scilab code Exa 13.10 To determine the forces in the members

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  FB = 3;              //(kN)(Force at B)
6  FC = 4;              //(kN)(Force at C)
7  FD = 2;              //(kN)(Force at D)
8
9  AB = 2;       //(m)
10 BC = 4;       //(m)
11 CD = 4;       //(m)
12 AE = 4;       //(m)
13 AP = 8;       //(m)
14 PG = 2;       //(m)
15 AD = 10;      //(m)
16 CP = 2;       //(m)
17 PO = 8;       //(m)
18 PD = 2;       //(m)
19
20 //Triangles OPG and OAE similar:
21 AO = (AP * AE)/PG;         //(m)
22 DO = AO - AD;       //(m)
23 OC = 10;            //(m)
24
25 //In triangle CGP:
26 GCP = atand(PG/CP);          //(Degrees)
27 COR = 90 - GCP;              //(Degrees)
28 OR = OC*cosd(COR);           //(m)
29
30 //Geometry in the triangle OCQ:
31 GOP = atand(PG/PO);          //(Degrees)
32
33 //In triangle OCQ:
34 CQ = OC*sind(GOP);           //(m)
35
36 //Taking moments of the forces acting on right part
```

```
    of the frame only, about the joint G and equating
     the same:
37 PCD = (FD * PD)/CP;          //(kN)(Tension)
38
39 //Taking moments of the forces acting in the right
    part of the truss only about the imaginary joint
    O and equating the same:
40 PCG = (FD * DO)/OR;       //(kN)(Tension)
41
42 //Taking moments of the forces acting in the right
    part of the truss only about the joint C and
    equating the same:
43 PFG = (FD * CD)/CQ;        //(kN)(Compression)
44
45 printf("Force in the member CD = %.2f kN(Tension)\n"
    ,PCD);
46 printf("Force in the member CG = %.2f kN(Tension)\n"
    ,PCG);
47 printf("Force in the member FG = %.2f kN(Compression
    )",PFG);
```

Scilab code Exa 13.11 To find the forces in the members of the structure

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 F1 = 8;       //(kN)(Force at B)
6 F2 = 12;      //(kN)(Force at D)
7 BD = 1.5;     //(m)
8 DC = 2;       //(m)
9 AD = 2;       //(m)
10 l = 4;       //(m)(Total length of base)
11
12 //Taking moments about A and equating the same:
```

```
13 VC = [(F1 * BD) + (F2 * AD)]/l;       //(kN)
14 VA = F2 - VC;                          //(kN)
15 HA = 8;                                //(kN)
16
17 //From geometry:
18 theta = atand(BD/AD);       //(Degrees)
19
20 //Solving by method of joints:
21 //Resolving the forces vertically and equating the
      same:
22 PAB = VA/sind(theta);       //(kN)(Compression)
23
24 //Resolving the forces horizontally and equating the
      same:
25 PAD = F1 + PAB * cosd(theta);       //(kN)(Tension)
26
27 //Considering joint C:
28 //Resolving the forces vertically and equating the
      same:
29 PBC = VC/sind(theta);            //(kN)(Compression)
30
31 //Resolving the forces horizontally and equating the
      same:
32 PCD = PBC * cosd(theta);        //(kN)(Tension)
33
34 PBD = 12;                       //(kN)(Tension)
35
36 printf("Force in the member AB = %.2f kN(Compression
      )\n",PAB);
37 printf("Force in the member AD = %.2f kN(Tension)\n"
      ,PAD);
38 printf("Force in the member BC = %.2f kN(Compression
      )\n",PBC);     //The answers vary due to round off
       error
39 printf("Force in the member CD = %.2f kN(Tension)\n"
      ,PCD);     //The answers vary due to round off
      error
40 printf("Force in the member BD = %.2f kN(Tension)",
```

```
PBD);
```

Scilab code Exa 13.13 To find the forces in the members

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
l = 8;          //(m)(Total length)
F1 = 3;         //(kN)(Force at B)
F2 = 4;         //(kN)(Force at D)
d1 = 2;         //(m)(Distance between subsequent slots
   lower part)
d2 = 1.5;       //(m)(Distance between subsequent slots
   upper part)
//Taking moments about the joint A and equating the
   same:
RE = [(F1 * d1) + (F2 * d2 * 3)]/l;       //(kN)

//From geometry:
theta = atand(3/4);              //(Degrees)
alpha = atand(3*d2/d1);         //(Degrees)

//Resolving the forces horizontally and equating the
    same:
//PDE * cosd(66) = PHE * cosd(36.9)    -(1)

//Resolving the forces vertically and equating the
   same:
//PDE * sind(66) = PHE * sind(36.9) + 3     -(2)

//From (1) and (2):
PHE = F1/1.2;           //(kN)(Tension)
PDE = 1.97 * PHE;       //(kN)(Compression)

```

```
27 PDH = 0;                //(kN)(As there is no other
      member at joint H to balance the component of
      this force)
28
29 printf("Force in the member HE = %.2f kN(Tension)\n"
      ,PHE);
30 printf("Force in the member DE = %.2f kN(Compression
      )\n",PDE);
31 printf("Force in the member DH = %.2f kN",PDH);
```

Scilab code Exa 13.14 To find the forces in the members

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  FR = 10;      //(kN)(Force at R)
6  FS = 20;      //(kN)(Force at S)
7  FQ = 10;      //(kN)(Force at Q)
8  d = 3;        //(m)(Distance between forces)
9  PQ = 6.92;    //(m)
10 alpha = 60;  //(degrees)
11 theta = 30;  //(degrees)
12
13 //Taking moments about Q and equating the same:
14 VP = [(FS * d) + (FR * 2*d)]/PQ;      //(kN)
15
16 //Total wind loads on the truss:
17 W = FR + FS + FQ;       //(kN)
18
19 //Horizontal component of wind load:
20 HCQ = W * cosd(alpha);      //(kN)
21
22 //Vertical component of the wind load:
23 VCQ = W * sind(alpha);      //(kN)
```

```
24
25 //Vertical reaction at Q:
26 VQ = VCQ - VP;           //(kN)
27
28 //Resolving the forces vertically and equating the
      same:
29 PPR = VQ/sind(alpha);     //(kN)(Compression)
30
31 //Resolving the forces horizontally and equating the
      same:
32 PPT = PPR * cosd(alpha);    //(kN)(Tension)
33
34 //Considering joint Q:
35 //Resolving the forces vertically and equating the
      same:
36 PSQ = [VQ - FQ*cosd(theta)]/sind(theta);      //(kN)(
      Compression)
37
38 //Resolving the forces horizontally and equating the
      same:
39 PQT = PSQ*cosd(theta) + FS - FQ*sind(theta);     //(
      kN)(Tension)
40 PRS = PSQ;
41
42 //Considering joint T:
43 PST = 20;                //(kN)(Compression)
44
45 //Resolving the forces vertically and equating the
      same:
46 PRT = [PST*sind(alpha)]/sind(alpha);        //(kN)(
      Tension)
47
48 printf("Force in the member PR = %.2f kN(Compression
      )\n",PPR);
49 printf("Force in the member PT = %.2f kN(Tension)\n"
      ,PPT);
50 printf("Force in the member SQ = %.2f kN(Compression
      )\n",PSQ);
```

```
51 printf("Force in the member QT = %.2f kN(Tension)\n"
   ,PQT);
52 printf("Force in the member ST = %.2f kN(Compression
   )\n",PST);
53 printf("Force in the member RS = %.2f kN(Compression
   )\n",PRS);
54 printf("Force in the member RT = %.2f kN(Tension)",
   PRT);     //The answers vary due to round off
   error
```

Scilab code Exa 13.15 To find the forces in the members

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  l = 12;          //(m)(Total length)
6  F1 = 10;         //(kN)(Force at D)
7  F2 = 20;         //(kN)(Force at F)
8  F3 = 10;         //(kN)(Force at G)
9  AC = 4;          //(m)
10 CE = 4;          //(m)
11 EG = 4;          //(m)
12 theta = 30;      //(degrees)(Angle at A)
13 alpha = 30;      //(degrees)(Angle at G)
14
15 //Taking moments about G and equating the same:
16 VA = [(F1 * AC) + (F2 * EG * cosd(alpha)) + (F1*(AC
   + CE)*cosd(alpha))]/l;     //(kN)
17
18 //Taking moments about the the joint C and equating
   the same:
19 PBD = (VA * AC)/2;          //(kN)(Compression)
20
21 //Taking moments about the joint D and equating the
```

```
    same:
22 PCE = (VA * 6)/(6 * tand(alpha));          //(kN)(
    Tension)
23
24 PCD = 0;
25
26 printf("Force in the member BD = %.2f kN(Compression
    )\n",PBD);     //(The answers vary due to round
    off error)
27 printf("Force in the member CE = %.2f kN(Tension)\n"
    ,PCE);     //(The answers vary due to round off
    error)
28 printf("Force in the member CD = %.2f kN",PCD);
```

Scilab code Exa 13.16 To find the forces in the members

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 AB = 3;      //(m)
6 BC = 3;      //(m)
7 CE = 4;      //(m)
8 BF = 4;     //(m)
9 FF = 5;     //(kN)(Force at F)
10 FE = 2;     //(kN)(Force at E)
11 alpha = 45;     //(degrees)(Angle at D)
12
13 //Taking moments about A and equating the same:
14 //RDV = RDH
15 RDH = [(FF * AB) + (FE * (AB+BC))]/5;     //(kN)
16 RDV = RDH;                          //(kN)
17
18 //Taking moments about the joint F and equating the
    same:
```

```
19 PBC = [(RDV * 2*BC) - (FE * BC)]/BF;     //(kN)(
     Compression)
20
21 //Taking moment about the joint C and equating the
     same:
22 PFE = [(RDV * BF) - (RDV * BC)]/BF;       //(kN)(
     Compression)
23
24 //Taking moments about the joint B and equating the
     same:
25 PFC = [(PFE * BF) - (FE * BC) + (RDV * 2*BC) - (RDV
     * BF)]/2.4;          //(kN)(Tension)
26
27 printf("Force in the member BC = %.2f kN(Compression
     )\n",PBC);
28 printf("Force in the member FE = %.2f kN(Compression
     )\n",PFE);
29 printf("Force in the member FC = %.2f kN(Tension)",
     PFC);
```

Scilab code Exa 13.18 To determine the pull in the chain and the forces in the mem

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 F1 = 2;     //(kN)(Force at D)
6 AD = 1.2;   //(m)
7 DC = 0.9;   //(m)
8 alphaA = 30;      //(degrees)(Angle at A)
9 alphaDi = 60;     //(degrees)(Angle at D interior)
10 alphaDe = 45;    //(degrees)(Angle at D exterior)
11 //Taking moments about the joint A and equating the
     same:
12 P = (F1*cosd(alphaDe)*AD)/DC;         //(kN)
```

```
13 printf("Pull in the chain = %.2f kN\n",P);
14
15 //Force in each member:
16 //Horizontal reaction at A:
17 HA = P - (F1*cosd(alphaDe));          //(kN)
18
19 //Vertical reaction at A:
20 VA = F1*sind(alphaDe);                    //(kN)
21
22 //Considering joint A:
23 //Resolving the forces vertically and equating the
     same:
24 PAB = VA/sind(alphaA);                //(kN)(
     Compression)
25
26 //Resolving the forces horizontally and equating the
     same:
27 PAD = PAB*cosd(alphaA) - HA;          //(kN)(Tension
     )
28
29 //Considering joint D:
30 //Resolving the forces horizontally and equating the
     same:
31 PBD = (PAD - F1*cosd(alphaDe))/cosd(alphaDi);
     //(kN)(Compression)
32
33 //Resolving the forces vertically and equating the
     same:
34 PCD = PBD*sind(alphaDi) + F1*sind(alphaDe);
     //(kN)(Tension)
35
36 //From geometry:
37 BE = 0.3;       //(m)
38 CE = 0.38;      //(m)
39
40 theta = atand(BE/CE);
41 //Resolving the forces horizontally at C and
     equating the same:
```

```
42 PBC = P/sind(theta);          //(kN)(Compression)
43
44 printf("Force in the  member AB = %.2f kN(
      Compression)\n",PAB);
45 printf("Force in the  member AD = %.2f kN(Tension)\n
      ",PAD);
46 printf("Force in the  member BD = %.2f kN(
      Compression)\n",PBD);
47 printf("Force in the  member CD = %.2f kN(Tension)\n
      ",PCD);
48 printf("Force in the  member BC = %.2f kN(
      Compression)\n",PBC);
49 //The answers vary due to round off error
```

Scilab code Exa 13.19 To determine the tension in the cable

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 F1 = 1;       //(kN)(Force at 2)
6 F2 = 1;       //(kN)(Force at 3)
7 alpha = 30;     //(degrees)(Angle at 1)
8 theta = 60;     //(degrees)(Angle at 5)
9
10 //(a)Tension in the cable:
11 //T = Tension in the cable,
12 //a = Length of each side of the equilateral
      triangle.
13 //Taking moments about the joint 5 and equating the
      same:
14 T = F1/cosd(theta);              //(kN)
15 printf("Tension in the cable = %.2f kN\n",T);
16
17 //(b)Nature and magnitude of the force in each bar:
```

```
18 //Considering joint 1:
19 //Resolving the forces vertically and equating the
     same:
20 P12 = (2*sind(alpha))/sind(theta);     //(kN)(Tension
     )
21
22 //Resolving the forces horizontally and equating the
      same:
23 P14 = 2*cosd(alpha) + P12*cosd(theta);          //(
     kN)(Compression)
24
25 //Considering joint 2:
26 //Resolving the forces vertically and equating the
     same:
27 P24 = [F1 - P12*cosd(theta)]/sind(theta);
28
29 //Resolving the forces horizontally and equating the
      same:
30 P23 = P12*cosd(theta);              //(kN)(Tension)
31
32 //Considering joint 4:
33 P34 = 0;
34 P24 = 0;
35 P45 = P14;                          //(kN)(Compression)
36 P24 = 0;
37 P25 = 0;
38
39 //Considering joint 3:
40 //Resolving the forces vertically and equating the
     same:
41 P35 = F1/cosd(alpha);             //(kN)(Tension)
42
43 printf("Force in the member 1-2 = %.2f kN(Tension)\n
     ",P12);
44 printf("Force in the member 1-4 = %.2f kN(
     Compression)\n",P14);
45 printf("Force in the member 2-4 = %.2f kN\n",P24);
46 printf("Force in the member 2-3 = %.2f kN(Tension)\n
```

```
    ",P23);
47 printf("Force in the member 3-4 = %.2f kN\n",P34);
48 printf("Force in the member 4-5 = %.2f kN(
    Compression)\n",P45);
49 printf("Force in the member 3-5 = %.2f kN(Tension)\n
    ",P35);
50
51 //The answers vary due to round off error
52
53 //(c)Reaction at the wall:
54 R = sqrt(P45^2 + P35^2 + 2*P45*P35*cosd(2*theta));
     //(kN)
55 printf("Reaction at the wall = %.2f kN",R);
```

Scilab code Exa 13.20 To determine the forces in the members

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5
6  //Horizontal component of 15 kN load is zero:
7  PEF = 0;
8  FF = 15;        //(kN)(Force at F)
9  d1 = 1;         //(m)(Distance between the vertical
       partitions)
10 d2 = 1.5;       //(m)(Distance between the horizontal
       partitions)
11 //Taking moments of the forces acting on the upper
       portion of the frame about the joint A and
       equating the same:
12 PCD = (FF * 2*d1)/(2*d2);       //(kN)
13
14 //Taking moments of the forces about the joint D and
     equating the same:
```

```
15 PAB = (FF * d1)/(2*d2);        //(kN)
16
17 printf("Force in the member AB = %.2f kN\n",PAB);
18 printf("Force in the member CD = %.2f kN\n",PCD);
19 printf("Force in the member EF = %.2f kN\n",PEF);
```

Chapter 15

Equilibrium of Strings

Scilab code Exa 15.1 To determine the vertical reactions horizontal thrusts sag po

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 AB = 12;          //(m)(Span)
6 FC = 10;          //(kN)(Force at C)
7 FD = 15;          //(kN)(Force at D)
8 FE = 20;          //(kN)(Force at E)
9 x = 3;            //(m)(Distance between line AB and
     point D)
10 y = 3;           //(m)(Distance between each point
     division)
11 //(i) Vertical reactions at A and B:
12 //Taking moments about A and equating the same:
13 VB = [(FC * y) + (FD * 2*y) + (FE * 3*y)]/AB;
     //(kN)
14 VA = (FC + FD + FE) - VB;                          //(kN
     )
15
16 printf("Vertical reaction at A = %.2f kN\n",VA);
17 printf("Vertical reaction at B = %.2f kN\n",VB);
```

```
18
19 //(ii)Horizontal thrusts at A and B:
20 //Taking moments of the forces acting in the string
     ACD about D and equating the same:
21 H = [(FE * 2*y) - (FC * y)]/y;            //(kN)
22 printf("Horizontal thrust at A and B = %.2f kN\n",H)
     ;
23
24 //(iii)Sag of points C and E:
25 //Taking moments of the forces acting in the string
     AC about C and equating the same:
26 yC = (FE * y)/H;      //(m)Sag of point of C)
27
28 //Taking moments of the forces acting in the string
     EB about E and equating the same:
29 yE = (VB * y)/H;      //(m)(Sag of point of E)
30
31 printf("Sag of point of C = %.2f m\n",yC);
32 printf("Sag of point of E = %.2f m\n",yE);
33
34 //(iv)Tensions in all the segments of the string:
35 //Vector diagram is drawn and following measurements
     are found out from it.
36 TAC = 36;          //(kN)(Tension in AC)
37 TCD = 31.6;        //(kN)(Tension in CD)
38 TDE = 30.4;        //(kN)(Tension in DE)
39 TEB = 39.1;        //(kN)(Tension in EB)
40
41 printf("Tension in AC = %.2f kN\n",TAC);
42 printf("Tension in CD = %.2f kN\n",TCD);
43 printf("Tension in DE = %.2f kN\n",TDE);
44 printf("Tension in EB = %.2f kN",TEB);
```

Scilab code Exa 15.2 To find the maximum tension in the cable

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 l = 30;              //(m)(Span)
6 yc = 3;              //(m)(Maximum dip)
7 w = 10;              //(kN/m)(Uniformly distributed
     load)
8
9 //Vertical reaction at the supports:
10 V = (w*l)/2;        //(kN)
11
12 //Horizontal thrust in the cable:
13 H = (w*l^2)/(8*yc);      //(kN)
14
15 //Maximum tension in the cable:
16 Tmax = sqrt(V^2 + H^2);       //(kN)
17 printf("Maximum tension in the cable = %.2f kN",Tmax
     );     //The answers vary due to round off error
```

Scilab code Exa 15.3 To find the cross sectional area of the cable

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 l = 40;           //(m)(Span)
6 d = 1.5;          //(m)(Width of platform)
7 Wp = 20;           //(kN/m^2)(Load on platform)
8 yc = 5;           //(Central dip)
9 f = 1050;         //(N/mm^2)(Maximum permissible
     stress in the cable)
10
11 //Load per metre lenght of the cable:
12 W = d * Wp;          //(kN/m)
```

```
13
14 //Load on each cable:
15 w = W/2;                  //(kN/m)
16
17 //Maximum tension in the cable:
18 Tmax = ((w*l)/2) * sqrt(1 + (l^2/(16 * yc^2))) *
      1000;    //(N)
19
20 //Necessary cross-sectional area of the cable:
21 A = Tmax/f;               //(mm^2)
22 printf("Necessary cross-section area of the cable =
      %.2f mm^2",A);     //(The answers vary due to
      round off error)
```

Scilab code Exa 15.4 To find the horizontal thrust and maximum tension in the cabl

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 l = 120;                 //(m)(Span)
6 d = 3;                   //(m)(Difference between the
      levels of supports)
7 w = 5;                   //(kN/m)(Uniformly distributed
      load)
8 yc = 5 - 3;              //(m)(Sag of the cable)
9
10 //Let l1 = Horizontal length of AC,
11 //l2 = Horizontal length of CB.
12
13 //l1/l2 = sqrt((yc + d)/yc) = 1.58;
14 //l1 = 1.58*l2;              -(1)
15
16 //l1 + l2 = 120;             -(2)
17
```

```
18 //From (1) and (2):
19 l2 = 120/2.58;              //(m)
20 l1 = 120 - l2;              //(m)
21
22 //Horizontal thrust in the cable:
23 H = (w*l1^2)/(2*(yc + d));          //(kN)
24 printf("Horizontal thrust in the cable = %.2f kN\n",
      H);          //(Answer rounded off in book)
25
26 //Maximum tension in the cable:
27 //Vertical reaction at A:
28 VA = (w*l)/2 + (H*d)/l;               //(kN)
29
30 Tmax = sqrt(VA^2 + H^2);           //(kN)(Maximum
      tension in the cable)
31 printf("Maximum tension in the cable = %.2f kN",Tmax
      );          //The answers vary due to round off
      error
```

Scilab code Exa 15.5 To find the length of the cable

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 l = 80;          //(m)(Span)
6 yc = 6;          //(m)(Central dip)
7
8 //Length of each cable:
9 L = l + ((8*yc^2)/(3*l));              //(m)
10 printf("Length of each cable = %.2f m",L);
```

Scilab code Exa 15.7 To find the length of the cable

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  l = 60;          //(m)(Span)
6  yc = 3;          //(m)(Depth of the lowest point from
       the support B)
7  d = 12 - 3;     //(m)(Difference between the levels of
       the supports)
8
9  //Let l1 = Horizontal length AC,
10 //l2 = Horizontal length CB.
11
12 //l1/l2 = 2;
13 //l1 = 2*l2;          -(1)
14
15 //l1 + l2 = 100;     -(2)
16
17 //From (1) and (2):
18 l2 = l/3;           //(m)
19 l1 = 2*l2;          //(m)
20
21 //Length of the cable:
22 L = l + (2*(yc + d)^2)/(3*l1);      //(m)
23 printf("Length of the cable = %.2f m",L);     //The
       answers vary due to round off error
```

Scilab code Exa 15.8 To compute the length of the cable horizontal component of te

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  l = 45;              //(m)(Span)
6  yc = 2;              //(m)(Depth of the lowest point
```

```
    from support A)
d = 8 - 2;          //(m)(Difference between the
    levels of the supports)
w = 20;             //(kN/m)(Uniformly distributed
    load over the span)

//(a) Length of the cable:
//Let l1 = Horizontal length of the CB,
//l2 = Horizontal length of AC.

//l1/l2 = 2;

//l1 = 2*l2;            -(1)
//l1 + l2 = 45;         -(2)

//From (1) and (2):
l2 = 45/3;          //(m)
l1 = 2*l2;          //(m)

//Length of the cable:
L = l + (2*(yc + d)^2)/(3*l1) + 2*yc^2/(3*l2);    //(
    m)
printf("Length of the cable = %.2f m\n",L);

//(b) Horizontal component of tension in the cable:
H = (w*l1^2)/(2*(yc + d));          //(kN)
printf("Horizontal component of tension in the cable
     = %.2f kN\n",H);

//(c) Magnitude and position of maximum tension
    occuring in the cable:
//Maximum tension will occur at the support B:
VB = (w*l)/2 + (H*d)/l;             //(kN)

//Maximum tension:
Tmax = sqrt(VB^2 + H^2);            //(kN)
printf("Magnitude and position of maximum tension
    occuring in the cable = %.2f kN",Tmax);
```

Scilab code Exa 15.10 To find the distance between the two supports and the maximu

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 L = 20;       //(m)(Length of the cable)
6 w = 25;       //(N/m)(Weight of the cable)
7 yc = 5;       //(m)(Central dip)
8
9 //Distance between the two supports:
10 //D = Distance between the two supports.
11 //psi = Angle which the tangent at B makes with the
     X-X axis:
12 //c = Parameter of the catenary.
13
14 //For the support B:
15 s = 10;          //(m)
16 //y = (5 + c);
17 //Substituting the above values in the general
     equation of the catenary:
18 //y^2 = c^2 + s^2;
19 c = 7.5;         //(m)
20
21 //Horizontal pull at C:
22 H = w*c;        //(N)
23
24 //Vertical reaction at B:
25 VB = w*s;       //(N)
26
27 psi = atand((w*s)/H);     //(Degrees)
28
29 //Distance between the two supports:
30 D = 2*[2.3*c*log10(secd(psi) + tand(psi))];      //(m
```

```
    )
31 printf("Distance between the two supports = %.2f m\n
    ",D);
32
33 //Maximum tension in the cable:
34 //Tension in the cable A:
35 T = w*(yc + c);        //(N)
36 printf("Maximum tension in the cable = %.2f N",T);
```

Chapter 17

Linear Motion

Scilab code Exa 17.1 To find the distance covered

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 u = 0;              //(Initial velocity)
6 a = 0.4;            //(m/s^2)(Acceleration)
7 t = 20;             //(s)(Time taken)
8
9 //Distance covered:
10 s = u*t + (1/2)*a*t^2;        //(m)
11 printf("Distance covered = %.2f m",s);
```

Scilab code Exa 17.2 To find the distance travelled

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
```

```
5 u = 7.5;        //(m/s)(Initial velocity)
6 a = 0.5;        //(m/s^2)(Acceleration)
7 t = 12;         //(s)(Time taken)
8
9 //Distance travelled:
10 s = u*t + (1/2)*a*t^2;            //(m)
11 printf("Distance travelled by the train = %.2f m",s)
    ;
```

Scilab code Exa 17.4 To calculate retardation and time required to stop the car

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 u = 20;             //(m/s)(Initial velocity)
6 v = 0;              //(Final velocity)
7 s = 50 - 10;        //(m)(Distance travelled by the
    car)
8
9 //(i) Retardation:
10 //Let a = acceleration of the motorist.
11 //v^2 = u^2 + 2*a*s;
12 a = -(u^2)/(2*s);                //(Retardation)
13 printf("Retardation = %.2f m/s^2\n",abs(a));
14
15 //(ii)Time required to stop the car:
16 //t = Time required to stop the car in seconds.
17 //v = u + a*t;
18 t = -u/a;                    //(s)
19 printf("Time required to stop the car = %.2f s",t);
```

Scilab code Exa 17.5 To find the uniform acceleration and the velocity of the car

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  t1 = 10;            //(s)
6  s1 = 30;            //(m)
7  t2 = 12;            //(s)
8  s2 = 42;            //(m)
9
10 //Uniform acceleration:
11 //u = Initial velocity of the car,
12 //a = Uniform acceleration.
13
14 //30 = u*t + (1/2)*a*t^2 = 10*u + 50*a
15 //Multiplying it by 6:
16 //180 = 60*u + 300*a      -(1)
17
18 //Distance travelled by the car in 12 seconds:
19 //42 = 12*u + 72*a
20 //Multiplying it by 5:
21 //210 = 60*u + 360*a      -(2)
22
23 //(2)-(1):
24 a = 30/60;                //(m/s^2)
25 printf("Uniform acceleration = %.2f m/s^2\n",a);
26
27 //Velocity at the end of 15 seconds:
28 t = 15;                   //(s)
29 //Substituting 'a' in equation (1):
30 u = (180 - 150)/60;              //(m/s)
31
32 v = u + a*t;              //(m.s)
33 printf("Velocity at the end of 15 seconds = %.2f m/s
   ",v);
```

Scilab code Exa 17.7 To calculate velocity and time

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5
6 //Considering the motion of the train between the
     first and second kilometre stones:
7
8 s = 1000;          //(m)(Distance)
9 u1 = 5;             //(m/s)(Initial velocity)
10 v1 = 10;           //(m/s)(Final velocity)
11
12 //Velocity with which the train passes the third km
     stone:
13 //v^2 = u^2 + 2*a*s;
14 a = (v1^2 - u1^2)/(2*s);        //(m/s^2)
15
16 //Considering the motion of the train between the
     second and third kilometre stones:
17 s = 1000;        //(m)(Distance)
18 u2 = 10;          //(m/s)(Initial velocity)
19 v2 = sqrt(u2^2 + 2*a*s);           //(m/s)
20 V = v2*(3600/1000);                //(km.p.h)(
     Converting from m/s to km.p.h)
21
22 printf("Velocity when it passes the third kilometre
     stone = %.2f km.p.h\n",V);     //The answers vary
     due to round off error
23
24 //Time taken for each of the two intervals of one
     kilometre:
25 //t1 = Time taken by the train to travel tthe first
     one kilometre,
26 //t2 = Time taken by the train to travel the second
     kilometre.
27
```

```
28 t1 = (v1 - u1)/a;        //(s)
29 t2 = (v2 - u2)/a;     //(s)
30
31 printf("Time taken by the train to travel the first
      one kilometre = %.2f s\n",t1);
32 printf("Time taken by the train to travel the second
       kilometre = %.2f s",t2);
33 //The answers vary due to round off error
```

Scilab code Exa 17.8 To find the time when train B will overtake train A

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  uA = 0;          //(Initial velocity of train A)
6  aA = 0.2;        //(m/s^2)(Uniform acceleration of
      train A)
7  vA = 12.5;       //(m/s)(Final velocity of train A)
8
9  uB = 0;          //(Initial velocity of train B)
10 aB = 0.4;        //(m/s^2)(Uniform acceleration of
      train B)
11 vB = 20;         //(m/s)(Final velocity of train B)
12
13 //tA = Time taken by the train A to attain a speed
      of 12.5 m/s,
14 //T = Time in second when the train B will overtake
      the train A from its start.
15
16 tA = vA/aA;            //(s)
17
18 //Distance travelled by the train A to attain the
      speed:
19 s1 = uA*tA + (1/2)*aA*tA^2;          //(m)
```

```
20
21 tB = vB/aB;                    //(s)
22
23 //Distance travelled by the train B to attain this
      speed:
24 s2 = uB*tB + (1/2)*aB*tB^2;           //(m)
25
26 //Train A has travelled for (T + 60) seconds. Total
      distance travelled by the train A during this
      time:
27 //sA = 390.6 + 12.5*[(T + 60) - 62.5]        (m)
      -(1)
28
29 //Total distance travelled by the Train B:
30 //sB = 500 + 20*(T - 50)      (m)        -(2)
31
32 //(1) = (2):
33 T = 859.3/7.5;          //(s)
34 printf("Train B will overtake train A at %.2f s",T);
         //The answers vary due to round off error
```

Scilab code Exa 17.9 To find the height of the bridge

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 u = -4.9;          //(m/s)(Initial velocity)
6 t = 2;             //(s)(Time taken)
7 g = 9.8;           //(m/s^2)(Accn due to gravity)
8
9 //Height of the bridge:
10 h = u*t + (1/2)*g*t^2;            //(m)
11
12 printf("Height of the bridge = %.2f m",h);
```

Scilab code Exa 17.10 To calculate the velocity of the packet

```
1 ////OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 u = -12;        //(m/s)(Velocity of the balloon when
     the packet is dropped)
6 t = 2;          //(s)(Time taken)
7 g = 9.8;        //(m/s^2)(Accn due to Gravity)
8
9 //Velocity of the packet after 2 sec:
10 v = u + g*t;       //(m/s)
11 printf("Velocity of the packet after 2 sec = %.2f m/
     s",v);
```

Scilab code Exa 17.11 To find the height of the building

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 u = 0;          //(Initial velocity)
6 t = 2.8;        //(s)(Time taken)
7 g = 9.8;        //(m/s^2)(Accn due to Gravity)
8
9 //Height of the building:
10 s = u*t + (1/2)*g*t^2;       //(m)
11 printf("Height of the building = %.2f m",s);
```

Scilab code Exa 17.12 To find the velocity with which it will hit the ground

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 u = 0;        //(Initial velocity)
6 s = 65;       //(m)(Height of the building)
7 g = 9.8;      //(m/s^2)(Accn due to Gravity)
8
9 //Let v = Final velocity of the stone with which it
     will hit the ground.
10 v = sqrt(u^2 + 2*g*s);        //(m/s)
11 printf("Final velocity of the stone with which it
     will hit the ground = %.2f m/s",v);     //The
     answers vary due to round off error
```

Scilab code Exa 17.13 To find the distance

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 u = -28;        //(m/s)(Initial velocity)
6 t = 2;          //(s)(Time)
7 g = 9.8;        //(m/s^2)(Accn due to Gravity)
8
9 //s = u*t + (1/2)*g*t^2;
10 s = u*t + (1/2)*g*t^2;      //(m)
11 printf("Distance it will cover in 2 seconds = %.2f m
     ",s);
```

Scilab code Exa 17.14 To find the height to which the bullet will rise

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 u = -80;       //(m/s)(Initial velocity)
6 v = 0;         //(Final velocity)
7 g = 9.8;       //(m/s^2)(Accn due to Gravity)
8
9 s = (v^2 - u^2)/(2*g);       //(m)
10 printf("Height the bullet will rise = %.2f m",s);
```

Scilab code Exa 17.15 To find the separation between the bodies

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 u1 = 0;          //(Initial velocity of body 1)
6 u2 = 0;          //(Initial velocity of body 2)
7 t1 = 3;          //(s)(Time taken by the first body)
8 t2 = 3 - 1;      //(s)(Time taken by the second body)
9 g = 9.8;         //(m/s^2)(Accn due to Gravity)
10
11 //Distance covered by the first body in 3 seconds:
12 h1 = u1*t1 + (1/2)*g*t1^2;       //(m)
13
14 //Distance covered by the second body in 2 seconds:
15 h2 = u2*t2 + (1/2)*g*t2^2;       //(m)
16
17 //Separation between the bodies:
18 S = h1 - h2;          //(m)
19 printf("Separation between the bodies = %.2f m",S);
```

Scilab code Exa 17.16 To find the total time taken by the stone

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 u = -29.4;          //(m/s)(Initial velocity)
6 h = 34.3;           //(m)(Height of tower)
7 g = 9.8;            //(m/s^2)(Accn due to Gravity)
8
9 //Let t = Time taken by the stone to reach the foot
     of the tower.
10
11 //Using s = u*t + (1/2)*g*t^2;
12 //t^2 - 6*t - 7 = 0;
13 //Solving the quadratic equation(using positive
     value):
14 p = poly([-7 -6 1],'t','c');
15 t = roots(p);
16 printf("Time taken by the stone to reach the foot of
      the tower = %.2f s",t(1));
17 //Positive root
```

Scilab code Exa 17.17 To find the velocity of the second stone

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 //Considering the upward motion of the first stone:
6 u = -49;    //(m/s)(Initial velocity)
7 v = 0;      //(Final velocity)
8 g = 9.8;    //(m/s^2)(Accn due to Gravity)
9
10 //t = Time taken by the stone to reach the maximum
```

```
    height;
t = (v-u)/(g);       //(s)

//Total time of flight:
T = 2*t;             //(s)

//Considering the second stone:
//Time taken by the second for going upwards and
    coming back to the earth:
t2 = T - 2;          //(s)

//Time taken by the stone to reach the maximum
    height:
T2 = t2/2;           //(s)

//Considering the upward motion of the second stone:
u = g*T2;            //(m/s)
printf("The velocity with which the second stone was
    thrown upwards = %.2f m/s",u);
```

Scilab code Exa 17.17 To find the velocity of the second stone

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
//Considering the upward motion of the first stone:
u = -49;    //(m/s)(Initial velocity)
v = 0;      //(Final velocity)
g = 9.8;    //(m/s^2)(Accn due to Gravity)

//t = Time taken by the stone to reach the maximum
    height;
t = (v-u)/(g);       //(s)

```

```
13 //Total time of flight:
14 T = 2*t;                    //(s)
15
16 //Considering the second stone:
17 //Time taken by the second for going upwards and
      coming back to the earth:
18 t2 = T - 2;                 //(s)
19
20 //Time taken by the stone to reach the maximum
      height:
21 T2 = t2/2;                  //(s)
22
23 //Considering the upward motion of the second stone:
24 u = g*T2;                   //(m/s)
25 printf("The velocity with which the second stone was
       thrown upwards = %.2f m/s",u);
```

Scilab code Exa 17.18 To find when and where the two stones cross each other

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 H = 50;                      //(m) Height of the tower
6
7 //Time taken by the stone to cross each other:
8 // t = Time taken by the stones to cross each other,
9
10 //Top stone:
11 //s = u*t + (1/2)*g*t^2 = 0.5*g*t^2          -(1)
12
13 //Second stone:
14 //50 - s = u*t + (1/2)*g*t^2 = -25*t + 0.5*g*t^2
      -(2)
15
```

```
16 //(1) + (2):
17 t = H/25;        //(s)
18
19 //Substituting the value of t = 2 in equation (1):
20 s = 0.5 * g * t^2;        //(m)
21 printf("Time when the two stones cross each other =
      %.2f s\n",t);
22 printf("Point where the stones cross each other = %
      .2f m",s);
```

Scilab code Exa 17.19 To determine the height of the tower and the initial velocit

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 T1 = 5.64;          //(s)(Total time taken by first
      stone)
6 T2 = 3.6;           //(s)(Time taken by second stone)
7
8 g = 9.8;            //(m/s^2)(Accn due to Gravity)
9
10 //Let u = Initial velocity of the stones.
11 //Considering the upward motion of the first stone
      from the top of the tower:
12 //Time taken by the first stone to reach maximum
      height and then to reach the top of the tower:
13 TT1 = T1 - T2;         //(s)
14
15 //Time taken by the second stone to reach maximum
      height:
16 TT2 = TT1/2;             //(s)
17
18 //Final velocity of the stone:
19 v = g*TT2;              //(m/s)
```

```
20
21 //Height of the tower:
22 u = 10;          //(m/s)(Initial velocity)
23 t = 3.6;         //(s)(Time)
24 s = u*t + (1/2)*g*T2^2;       //(m)
25 printf("Initial velocity of the stones = %.2f m/s\n"
      ,u);
26 printf("Height of the tower = %.2f m",s);
```

Scilab code Exa 17.20 To find the velocity with which the stone will strike the gr

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 //Considering upward motion of the stone:
6 u1 = -20;          //(m/s)(Initial velocity)
7 v1 = 0;            //(Final velocity)
8 g = 9.8;           //(m/s^2)(Accn due to Gravity)
9
10 //Let s1 = Maximum height reached by the stone.
11 s1 = (v1^2 - u1^2)/(2*g);        //(m)
12
13 //Considering downward motion of the stone up to the
      glass pan:
14 u2 = 0;           //(Initial velocity)
15 s2 = s1/2;        //(m)(Distance covered by the stone)
16
17 //Let v2 = Final velocity of the stone, witb which
     it strikes the pan:
18 v2 = sqrt(abs(u2^2 + 2*g*s2));     //(m/s)
19
20 //Considering the motion of the stone after breaking
      the glass pan:
21 u3 = v2/2;         //(m/s)
```

```
22 s3 = abs(s1) - abs(s2);        //(m)
23
24 //Let v3 = Final velocity of the stone, with which
      it strikes the ground.
25 v3 = sqrt(u3^2 + 2*g*s3);          //(m/s)
26 printf("Final velocity of the stone, with which it
      strikes the ground = %.2f m/s",v3);
```

Scilab code Exa 17.21 To find the height from where the body started to fall

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5
6 //Solving by 1st method
7 //Considering the motion between A and B:
8 s = 10;      //(m)(Distance travelled)
9 t = 0.2;     //(s)(Time taken)
10 g = 9.8;    //(m/s^2)(Accn due to Gravity)
11
12 //Let u = Initial velocity of the body at A.
13 u = (s - (1/2)*s*t^2)/t;     //(m/s)
14
15 //Considering the motion from O to A:
16 u = 0;      //(Initial velocity)
17 v = 49;     //(m/s)(Final velocity)
18
19 s = (v^2 - u^2)/(2*g);      //(m)
20 printf("The height above the higher point from where
     it started to fall = %.2f m",s);
```

Scilab code Exa 17.22 To find the depth of the well

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 //Considering the downward motion of the stone:
6 u = 0;        //(Initial velocity)
7
8 //Let t = Time taken by the stone to reach the
     bottom of the well.
9 //Depth of the well:
10 //s = u*t + (1/2)*g*t^2 = 4.9*t^2;    -(1)
11
12 //Time taken by sound to t=reach the top:
13 //t = (Depth of well)/(Velocity of sound) = 4.9*t
     ^2/350;              -(2)
14
15 //Since total time = 4 seconds
16 //4.9*t^2 + 350*t - 1400 = 0
17 //Taking positive value:
18 p = poly([-1400 350 4.9],'t','c');
19 t = roots(p);
20 t = t(2);        //(Taking positive value for time)
21
22 //Substituting the value of t in equation (1):
23 s = 4.9*t^2;          //(m)
24
25 printf("Depth of the well = %.2f m",s);     //The
     answers vary due to round off error
```

Scilab code Exa 17.23 To find the time required

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
```

```
5 s = 20;          //(m)(Distance travelled by particle)
6 t = 1;           //(s)(Time)
7 g = 9.8;         //(m/s^2)(Accn due to Gravity)
8
9 //Let u = Initial velocity of particle at the time
     of starting.
10 //20 = u*t + (1/2)*g*t^2;
11 u = (s - (1/2)*g*t^2)/t;      //(m/s)
12
13 //Velocity of the particle after covering 20 metres:
14 v = u + g*t;            //(m/s)
15
16 //Let t be the time required to cover a distance of
     20 metres when the particle has initial velocity:
17 //Distance covered by the particle in this time:
18 //20 = u*t + (1/2)*g*t^2;
19 //4.9*t^2 + 24.9*t - 20 = 0;
20
21 //Taking positive value:
22 p = poly([-20 24.9 4.9],'t','c');
23 t = roots(p);
24 t = t(2);      //(Taking positive value for time)
25
26 printf("Time required to cover next 20 metres = %.2f
     s",t);
```

Scilab code Exa 17.24 To compare the times of falls of the particles

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 //Considering motion of the stone A(i.e., form A to
     C).
6 u1 = 0;       //(Initial velocity)
```

```
7  s1 = 100;      //(m)(Distance)
8  g = 9.8;       //(m/s^2)(Accn due to Gravity)
9
10 //Let t1 = Time taken by the particle A to reach C.
11
12 //Distance travelled by the stone A(s1):
13 t1 = sqrt(s1/((1/2)*g));    //(s)
14
15 //Considering motion of the stone B(first from B to
     D):
16 u2 = 0;           //(Initial velocity)
17 s2 = 50;          //(m)(Distance)
18 //Let t2 = Time taken by the particle B to reach D.
19
20 t2 = sqrt(s2/((1/2)*g));       //(s)
21
22 //Considering motion of stone B(from D or E to F):
23 //Time taken by the particle B to reach F:
24 T = 2*t2;      //(s)
25
26 printf("Ratio of the two times = %.2f",T/t1);
```

Scilab code Exa 17.25 To determine the time taken by the stone and the distance tr

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  //Considering the motion of the stone:
6  u = 0;         //(Initial velocity)
7  g = 9.8;       //(m/s^2)(Accn due to Gravity)
8
9  //(a)Time taken by the stone to hit the cage:
10 //t = Time taken by the stone to hit the cage.
11
```

```
12 //Distance travelled by the stone before the impact:
13 //s = u*t + (1/2)*g*t^2 = 4.9*t^2;       -(1)
14
15 //Considering motion of the cage for the first 25
      metres:
16 //Let t = Time taken by the cage to travel 25 m.
17 u = 0;          //(Initial velocity)
18 a = 0.5;        //(m/s^2)(Acceleration)
19 s = 25;         //(m)(Distance)
20 t = sqrt(s/(2*a));      //(s)
21
22 //Total time taken by the cage before impact = (10 +
      t) s
23 //Distance travelled by the cage in (10 + t) s:
24 // s = 0.25*(10 + t)^2;          -(2)
25
26 //(1) = (2):
27 //4.65*t^2 - 5*t - 25 = 0;
28 //Taking positive value:
29 p = poly([-25 -5 4.65],'t','c');
30 t = roots(p);
31 t = t(1);          //(Taking positive value for time)
32 printf("Time taken by the stone to hit the cage = %
      .2f s\n",t);
33
34 //(b) Distance travelled by the cage before impact:
35 //Substituting the value of t in equation (1):
36 s = (1/2)*a*(10 + t)^2;      //(m)
37 printf("Distance travelled by the cage before impact
      = %.2f m",s);     //The answers vary due to round
      off error
```

Scilab code Exa 17.26 To find the initial velocity of the body

```
1 //OS-version - Windows 10
```

```
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 //Solving by 1st method
6 s = 40;      //(m)(Disatnce traversed)
7 n = 4;       //(No. of second)
8 g = 9.8;     //(m/s^2)(Accn due to Gravity)
9
10 //Let u = Initial velocity of the body.
11 //Using s = u + (g/2)*(2*n - 1);
12 u = s - (g/2)*(2*n -1);     //(m/s)
13 printf("Initial velocity of the body = %.2f m/s",u);
```

Scilab code Exa 17.27 To find the ratio of the distances covered by the particle i

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 u = 0;       //(Initial velocity)
6 n1 = 3;      //(Initial no. of second)
7 n2 = 5;      //(Final no. of second)
8
9 //s3 = u + (a/2)*(2*n1 - 1);
10 //s5 = u + (a/2)*(2*n2 - 1);
11
12 //Ratio of distances covered:
13 //Ratio = s3/s5;
14 Ratio = [(1/2)*(2*n1 - 1)]/[(1/2)*(2*n2 - 1)];
15 printf("The ratio of distances covered by it in the
   3 rd and 5 th seconds of its motion = %.2f : %.2f
   ",[(1/2)*(2*n1 - 1)]*2,[(1/2)*(2*n2 - 1)]*2);
```

Scilab code Exa 17.28 To find the time of travel of the body

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 u = 0;            //(Initial velocity)
6 g = 9.8;          //(m/s^2)(Accn due to Gravity)
7
8 //Distance travelled in first three seconds =
     Distance travelled in nth second;
9
10 //Let n = Time of travel for the body;
11
12 //Distance travelled in first 3 s:
13 t = 3;      //(s)
14 s = u*t + (1/2)*g*t^2;     //(1)
15
16 //Distance travelled in the nth second after it
     starts:
17 //sn = u + (g/2)*(2*n - 1) = (g/2)*(2*n - 1);
     -(2)
18
19 //Equating (1) and (2):
20 //n - [(s*2)/g + 1]/2 = 0;
21 A = [1];
22 c = [-[(s*2)/g + 1]/2];
23 [n,nsA] = linsolve(A,c);
24
25 printf("Time of travel of the body = %.2f s",n);
```

Scilab code Exa 17.30 To find the initial velocity of the ball

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
```

```
clc
clear all
n1 = 5;     //(Initial no. of second)
n2 = 6;     //(Final no. of second)
g = 9.8;    //(m/s^2)(Accn due to Gravity)

//Let u = Initial velocity of the ball.

//Distance covere by the ball in the 5 th second
   after it starts:
//s5 =u + 45;     -(1)

//Distance covere by the ball in the 6 th second
   after it starts:
//s6 = -u + 55

//Since the distance covered by the ball in the 5 th
    second is twice the distance covered by it in
   the 6 th second:
//s5 = 2*s6
u = 110 - 45;      //(ms)
printf("Initial velocity = %.2f m/s",u);
```

Scilab code Exa 17.31 To find the maximum velocity of the lift acceleration of the

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
h = 900;        //(m)(Height)
hh = 300;       //(m)(Next height traversed)
//(i) Maximum velocity of the lift:
//t1 = Time of acceleration
//t2 = Time of retardation
//v = Maximum velocity of the lift
```

```
11
12 //Considering the motion from O to A:
13 //s1 = Area of the triangle OAL
14 //900 = (1/2)*t1*v        -(1)
15
16 //Considering the motion of the lift from A to B:
17 //s2 = Area of the triangle ALB
18 //300 = (1/2)*t2*v        -(2)
19
20 //(1) / (2):
21 //t1 - 3*t2 = 0;          -(3)
22
23 //t1 + t2 - 30 = 0;       -(4)
24
25 A = [1, -3; 1, 1];
26 c = [0; -30];
27 [x0,nsA] = linsolve(A,c);
28
29 t1 = x0(1);          //(s)
30 t2 = x0(2);          //(s)
31
32 //Substituting t1 in equation 1:
33 v = (h * 2)/t1;       //(m/s)
34 printf("Maximum velocity of the lift = %.2f m/s\n",v
      );
35
36 //(ii) Acceleration of the lift:
37 a1 = v/t1;             //(m/s^2)
38 printf("Acceleration of the lift = %.2f m/s^2\n",a1)
      ;
39
40 //(iii) Retardation of the lift:
41 a2 = 3*a1;               //(m/s^2)
42 printf("Retardation of the lift = %.2f m/s^2",a2);
43 //The answers vary due to round off error
```

Scilab code Exa 17.32 To find the time lost in the journey

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 //Considering the motion of the train from A to B:
6 s1 = 1;          //(km)(Distance travelled)
7 u1 = 30;         //(km.p.h.)(Initial velocity)
8 v1 = 15;         //(km.p.h.)(Final velocity)
9
10 //Let t1 = Time taken by the train to move from A to
     B.
11 t1 = 2/(u1 + v1) * 60;     //(min)(Converting from hr
     to min)
12
13 //Considering motion of the train from B to C:
14 s2 = 0.5;         //(km)(Distance travelled)
15 u2 = 15;         //(km.p.h.)(Initial velocity)
16 v2 = 30;         //(km.p.h)(Final velocity)
17
18 //Let t2 = Time taken by the train to move from B to
     C.
19
20 t2 = (1/(u1 + v1))*60;     //(min)(Converting from hr
     to min)
21
22 //Total time:
23 T = t1 + t2;          //(min)
24
25 //If the train had moved uniformly with a velocity
    of 30 km/hr, then the time required to cover 1.5
    km:
26 s = 30;                   //(km/hr)
```

```
d = 1.5;                        //(km)
t = (60/s)*d;           //(min)

//Time lost:
TL = T - t;             //(min)
printf("Time lost = %.2f min",TL);
```

Scilab code Exa 17.33 To find the uniform speed of the cage

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
s = 750;          //(m)(Total distance travelled)

//Considering the motion of cage from O to A:
u1 = 0;           //(Initial velocity)
s1 = s/4;       //(m)(Distance travelled)

//Let a1 = Constant acceleration of the cage,
//v1 = Uniform velocity of the cage (AD or BE)

//Area of the triangle OAD(s1):
//t1 * v1 = 187.5 * 2;      -(1)

//Considering the motion of the cage from A to B:
s2 = s - (2 * s1);        //(m)

//Area of the triangle ABED(s2):
//t2 * v2 = 375;            -(2)

//From equation (1) and (2):
//t1 = t2 = t3;
//Since t1 + t2 + t3 = 45;
t1 = 15;      //(s)
```

```
27 t2 = 15;     //(s)
28 t3 = 15;     //(s)
29
30 //Considering the motion of the  cage from O to A:
31 u1 = 0;          //(Initial velocity)
32 s1 = 187.5;      //(m)(Distance travelled)
33 t1 = 15;         //(s)(Time)
34
35 a1 = (s1*2)/(t1^2);        //(m/s^2)
36
37 //Speed of the cage while traversing the central
      portion of the shaft:
38 v1 = u + a1*t1;      //(m/s)
39 u = 0;
40 printf("Speed of the cage while traversing the
      central portion of the shaft = %.2f m/s",v1);
```

Chapter 18

Motion Under Variable Acceleration

Scilab code Exa 18.1 To find the velocity and acceleration of the body

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 t=poly(0,"t");
6 //Equation of displacement:
7 s = t^3 - 2*t^2 + 3;                    //-(i)
8
9 //Velocity after 5 seconds:
10 //Differentiating (i) w.r.t. t:
11 v = derivat(s);                        //-(ii)
12 a = derivat(v);                        //-(iii)
13 //Substituting t = 5 in (ii):
14 v = pol2str(v);
15 t = 5;
16 v = evstr(v);     //(m/s)
17 printf("Velocity after 5 seconds = %.2f m/s\n",v);
18
19 //Acceleration after 5 seconds:
```

```
20
21 //Substituting t = 5 in equating (iii):
22 a = pol2str(a);
23 t = 5;
24 a = evstr(a);              //(m/s^2)
25 printf("Acceleration after 5 seconds = %.2f m/s^2",a
   );
```

Scilab code Exa 18.2 To find the velocity and acceleration at the start and accele

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 t=poly(0,"t");
6 //Equation of displacement:
7 s = 12*t + 3*t^2 - 2*t^3;           //-(i)
8
9 //Velocity at start:
10 //Differentiating (i) w.r.t. t:
11 v = derivat(s);                    //-(ii)
12 a = derivat(v);                    //-(iii)
13
14 //Substituting t = 0 in (ii):
15 v = pol2str(v);
16 t = 0;
17 v = evstr(v);     //(m/s)
18 printf("Velocity at start = %.2f m/s\n",v);
19
20 //Acceleration at start:
21 //Differentiating (ii) w.r.t. t:
22
23 //Substituting t = 0 in (iii):
24 a = pol2str(a);
25 t = 0;
```

```
a = evstr(a);                    //(m/s^2)
printf("Acceleration at start = %.2f m/s^2\n",a);

//Acceleration, when the velocity is zero:
//Substituting (ii) = 0:
//12 + 6*t - 6*t^2 = 0
q = poly([12 6 -6],'t','c');
x = roots(q);
t = x(1);                //t=2

//Substituting t = 2 in (iii):
a = derivat(derivat(s));
a = pol2str(a);
a = evstr(a);          //(m/s^2)
printf("Acceleration, when the velocity is zero = %
   .2f m/s^2",a);
```

Scilab code Exa 18.3 To find the velocity and acceleration at the start time when

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
t=poly(0,"t");
//Equation of displacement:
s = 18*t + 3*t^2 - 2*t^3;              //-(i)

//(1) Velocity and acceleration at start:
//Differentiating (i) w.r.t. t:
v = derivat(s);                        //-(ii)

//Substituting t = 0 in (ii):
t = 0;
v = pol2str(v);
v = evstr(v);          //(m/s)
```

```
17 printf("Velocity at start = %.2f m/s\n",v);
18
19 //Differentiating (ii) w.r.t. t:
20 a = derivat(derivat(s))                    //-(iv)
21 //Substituting t = 0 in (iv):
22 t = 0;
23 a = pol2str(a);
24 a = evstr(a);          //(m/s^2)
25 printf("Acceleration at start = %.2f m/s^2\n",a);
26
27 //(2) Time, when the particle reaches its maximum
      velocity:
28 //Equating (iii) to 0;
29 //6 - 12t = 0                          (v)
30 A = [-12];
31 c = [6];
32 [x,nsA] = linsolve(A,c);
33 t = x;          //(s)
34 printf("Time, when the particle reaches its maximum
      velocity = %.2f s\n",t);
35
36 //(3) Maximum velocity of the particle:
37 //Substituting t = 0.5 in (ii):
38 v = derivat(s);
39 v = pol2str(v);
40 v = evstr(v);
41 printf("Maximum velocity of the particle = %.2f m/s"
      ,v);
```

Scilab code Exa 18.4 To calculate the displacement and velocity

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
```

```
5  t=poly(0,"t");
6
7  //Equation of acceleration:
8  a = t^3 - 3*t^2 + 5;                          //-(i)
9  //Integrating (i):
10 //v = t^4/4 - t^3 + 5*t + C1                   -(ii)
11 v = t^4/4 - t^3 + 5*t;
12 //Substituting t = 1 and v = 6.25 in (ii):
13 t = 1;
14 v = pol2str(v)
15 x = evstr(v);
16 C1 = 6.25 - x;
17
18 //v = t^4/4 - t^3 + 5*t + 2                    -(iii)
19 //Substituting t = 2;
20 v = integrate('t^3 - 3*t^2 + 5','t',0,2) + C1;
             //(m/s)
21 printf("Velocity at 2 seconds = %.2f m/s\n",v);
22
23 //Displacement at t = 2 seconds:
24 //Integrating (iii) w.r.t. t:
25 //s = t^5/20 - t^4/4 + (5*t^2)/2 + 2*t + C2      -(v)
26 s = t^5/20 - t^4/4 + 5 * t^2/2 + 2*t;
27 //Substituting t = 1 and s = 8.8 in (v):
28 t = 1;
29 x = evstr(s);
30 C2 = 8.8 - x;
31
32 //s = t^5/20 - t^4/4 + (5*t^2)/2 + 2*t + 4.5      -(
      vi)
33
34 //Substituting t = 2 in (vi):
35 s = integrate('t^4/4 - t^3 + 5*t + 2','t',0,2) + C2;
         //(m)
36 printf("Displacement at 2 seconds = %.2f m",s);
```

Scilab code Exa 18.5 To find the distance

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 //Equation of acceleration: a = 10/(v + 1)
6 //Rewriting the equation:
7 //(v^2 + v) dv = 10 ds                -(i)
8 //Integrating (i):
9 //v^3/3 + v^2/2 = 10*s + C1           -(ii)
10 v = poly(0,"v");
11 C1 = v^3/3 + v^2/2;
12 //Substituting s = 0 and v = 0 in (ii):
13 s = 0;
14 v = 0;
15 C1 = pol2str(C1);
16 C1 = evstr(C1);
17
18 //2*v^3 + 3*v^2 = 60*s                -(iii)
19
20 //Substituting v = 10 m/s in (iii):
21 s = (integrate('v^2 + v','v',0,10))/10;          //(m)
22 printf("Distance in which the train will attain 35
     km.p.h = %.2f m",s);
```

Scilab code Exa 18.6 To find the velocity and the distance travelled by the partic

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
```

```
s = poly(0,"s");
//Equation of acceleration:
a = 10 - 0.006*s^2;

//Rewriting the given equation:
//v dv = (10 - 0.006*s^2) ds          -(i)

//(a) Velocity of the particle, when it has
   travelled 50 metres:
//Integrating (i):
//v^2 = 20*s - 0.004*s^3 + 2*C1       -(ii)


//Substituting s = 0 and v = 0 in (ii):
C1 = 0;
//v^2 = 20*s - 0.004*s^3              -(iii)

//Substituting s = 50 in (iii):
vsquare = 2*(integrate('10 - 0.006*s^2','s',0,50));
v = sqrt(vsquare);    //(m/s)
printf("Velocity of the particle, when it has
   travelled 50 metres = %.2f m/s\n",v);

//(b) Distance travelled by the particle, when it
   comes to rest:
//Substituting v = 0 in equation (iii):
p = 20*s - 0.004*s^3;     //(m)
x = roots(p);
x = x(1);
printf("Distance travelled by the particle, when it
   comes to rest = %.2f m",x);
```

Scilab code Exa 18.7 To find the acceleration and velocity at the time of start di

```
//OS-version - Windows 10
```

```
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  t = poly(0,"t");
6  //Equation of acceleration:
7  a = 2 - 3*t;                    //-(ii)
8
9  //(a) Acceleration and velocity at the time of start
       :
10 //Substituting t = 0 in (i):
11 a = pol2str(a);
12 t = 0;
13 a = evstr(a);     //(m/s^2)
14 printf("Acceleration at the time of start = %.2f m/s
       ^2\n",a);
15
16 //Rewriting equation (i):
17 //dv = (2 - 3*t) dt                  -(ii)
18
19 //Integrating (ii):
20 //v = 2*t - (3*t^2)/2 + C1           -(iii)
21
22 //Substituting t = 5 and v = 20 in (iii):
23 t = 5;
24 v = 2*t - 3*t^2/2;
25 x = evstr(v);
26 C1 = 20 - x;
27 //v = 2*t - (3*t^2)/2 + C1           -(iv)
28
29 //Substituting t = 0 in (iv):
30 v = integrate('2 - 3*t','t',0,0) + C1;              //(
       m/s)
31 printf("Velocity at the time of start = %.2f m/s\n",
       v);
32
33 //(b) Distance from the origin at the start of
       observation:
34 //Rewriting (iv):
```

```
35 //ds = (2*t - (3*t^2)/2 + 47.5) dt
36 //Integrating:
37
38 //s = t^2 - t^3/2 + 47.5*t + C2          -(v)
39 //Substituting t = 10 and s = 85;
40 t = 10;
41 s = t^2 - t^3/2 + 47.5*t;
42 x = evstr(s);
43 C2 = 85 - x;
44
45 //s = t^2 - t^3/2 + 4.75*t + 10
46
47 //Substituting t = 0 in the above equation:
48 s = C2;      //(m)
49 printf("Distance from the origin at the start of
      observation = %.2f m\n",s);
50
51 //(c) Time after start of observations in which the
      velocity becomes zero:
52 //Substituting v = 0 in (iv):
53 //3*t^2 - 4*t - 95 = 0;
54 p = poly([-95 -4 3],'t','c');
55 x = roots(p);
56 x = x(1);
57 printf("Time after start of observations in which
      the velocity becomes zero = %.2f s",x);
```

Chapter 19

Relative Velocity

Scilab code Exa 19.1 To find the actual velocity of the rain

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 v = 48;       //(km.p.h)(Velocity of train)
6 theta = 60;   //(degrees)(Angle at C)
7
8 //In triangle OBC:
9 //tand(60) = v/OC;          //(OC = Actual velocity
    of the train)
10 OC = (v/tand(theta));     //(km.p.h.)
11 printf("Actual velocity of the train = %.2f km.p.h."
    ,OC);
```

Scilab code Exa 19.2 To find the actual direction and velocity of the wind

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
```

```
clc
clear all
OA = 6;      //(km.p.h.)(East)
AB = OA;     //(km)
CA = 6;      //(km)
theta = atand(CA/AB);     //(Degrees)
printf("Actual direction = %.2f degrees\n",theta);

//In triangle OAC:
CO = CA/sind(theta);       //(km.p.h)
printf("Actual velocity = %.2f km.p.h",CO);
```

Scilab code Exa 19.3 To find the actual velocity in magnitude and direction of the

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
v1 = 20;       //(km.p.h)(Velocity 1)
t1 = 45;       //(Degrees)(Direction 1)

v2 = 12;       //(km.p.h)(Velocity 2)
t2 = 30;       //(Degrees)(Direction 2)

//From triangle ACD:
//y = (20 - x)/tand(45) = 20 - x      -(1)

//In triangle BCD:
//y = (12 - x)/tand(30) = 12 - x;     -(2)

//Equating (1) and (2):
x = (v1*tand(t2) - v2)/(tand(t2) - 1);          //(
   km)

//Substituting x in (1):
```

```
21 y = 20 - x;                  //(km)
22
23 alpha = atand(x/y);          //(Degrees)(Actual
      direction)
24 printf("Actual direction = %.2f degrees\n",alpha);
         //The answers vary due to round off error
25
26 //In triangle OCD:
27 CO = sqrt(x^2 + y^2);         //(km.p.h.)
28 printf("Actual velocity = %.2f km.p.h.",CO);
29 //The answers vary due to round off error
```

Scilab code Exa 19.4 To find the direction and velocity of the person

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 //In triangle OBC:
6 OB = 4;          //(m)
7 BC = 5;          //(m)
8 theta = acosd(OB/BC);       //(Degrees)
9 printf("Direction in which he may run = %.2f degrees
     \n",theta);
10
11 OC = sqrt(BC^2 - OB^2);     //(m/s)
12 printf("Velocity with which he should enter = %.2f m
     /s",OC);
```

Scilab code Exa 19.5 To find the velocity of the second ship relative to the first

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
```

```
clc
clear all
v1 = 32;          //(km.p.h.)(Velocity of first ship)
v2 = 24;          //(km.p.ph.)(Velocity of second ship)
theta = 40 + 45;      //(degrees)(Total angle)
//(a)
//In parallelogram OBRC:
OR = sqrt(v1^2 + v2^2 + 2*v1*v2*cosd(180 - theta));
     //(km.p.h.)
printf("Velocity of the second ship relative to the
   first = %.2f km.p.h.\n",OR);     //The answers
   vary due to round off error

//(b)
d = 160;              //(km)(Distance between the ships)
t = d/OR;         //(hrs)(Time when the two ships will
   be 160 km apart)
printf("Time when the two ships will be 160 km apart
    = %.2f hrs",t);
```

Chapter 20

Projectiles

Scilab code Exa 20.1 To calculate the horizontal distance

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 V = 30;        //(m/s)(Horizontal velocity of
     aircraft)
6 g = 9.8;       //(m/s^2)(Accn due to Gravity)
7
8 //Horizontal distance of the aircraft from the
     target when it released the bomb:
9 //Considering vertical motion of the bomb due to
     gravitational acceleration only;
10 u = 0;        //(Initial velocity)
11 s = 1000;     //(m)(Distance covered)
12
13 //Let t = Time required by bomb to reach the ground.
14
15 //Using s = u*t + (1/2)*g*t^2;
16 t = sqrt(s/((1/2)*g));        //(s)
17
18 H = V * t;     //(m)(Horizontal distance of the
```

```
    aircraft from the target when it released the
    bomb)
19 printf("Horizontal distance of the aircraft from the
     target when it released the bomb = %.2f m\n",H);
        //The answers vary due to round off error
20
21 //Direction and velocity with which the bomb hits
    the target:
22
23 v = u + g*t;       //(m/s)(Final velocity)
24
25 //Angle which the bomb makes with vertical:
26 theta = atand(V/v);        //(Degrees)
27 printf("Angle which the bomb makes with vertical
    when it hits the target = %.2f degrees\n",theta);
        //The answers vary due to round off error
28
29 //Resultant velocity:
30 R = sqrt(v^2 + V^2);        //(m/s)
31 printf("Resultant velocity with which the bomb hits
    the target = %.2f m/s",R);     //The answers vary
    due to round off error
```

Scilab code Exa 20.2 To find the minimum velocity of the bike and also the angle o

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 x = 4;      //(m)(Width of ditch)
6 s = 2;      //(m)(Vertical distance between A and B)
7 g = 9.8;    //(m/s^2)(Accn due to Gravity)
8 u = 0;      //(Initial velocity)
9 //Minimum velocity of motor cycle at A:
10 //Using s = u*t + (1/2)*g*t^2;
```

```
11 t = sqrt(s/((1/2)*g));     //(s)
12
13 //Minimum velocity of the motorcycle at A:
14 v = (x/t);       //(m/s)
15 printf("Minimum velocity of the motorcycle at A = %
      .2f km.p.h\n",v*(3600/1000));       //(Converting
      m/s to km.p.h)
16
17 //Inclination and magnitude of the velocity of motor
       vehicle just after clearing the ditch(i.e., at B
      ):
18 V = u + g*t      //(Final velocity)
19
20 theta = atand(v/V);      //(Degrees)
21 printf("Inclination of the velocity with the
      vertical = %.2f degrees\n",theta);     //The
      answers vary due to round off error
22
23 //Magnitude of velocity:
24 R = sqrt(v^2 + V^2);      //(m/s)
25 printf("Magnitude of the velocity of the motor cycle
       just after clearing the ditch = %.2f km.p.h.",R
      *(3600/1000));
```

Scilab code Exa 20.3 To find inclination time and horizontal distance

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 v = 55.56;     //(m/s)(Aeroplane velocity)
6 h = 1000;      //(m)(Height of plane)
7 u = 300;       //(m/s)(Velocity of shell)
8 g = 9.8;       //(m/s^2)(Accn due to Gravity)
9
```

```
10 //Inclination of the gun:
11 //Let alpha = Inclination of gun with the horizontal
   ,
12 //t = Time taken by the shell to hit the plane.
13
14 //AB = 55.56*t;       -(1)
15
16 //and horizontal distance:
17 //AB = 300*cosd(alpha)*t;          -(2)
18 //Equating (1) and (2):
19 alpha = acosd(v/u);      //(Degrees)
20
21 printf("Inclination of gun with the horizontal = %.2
   f degrees\n",alpha);
22
23 //Vertical componnent of shell velocity:
24 uy = u*sind(alpha);      //(m/s)
25
26 //Vertical distance:(s = u*t + (1/2)*g*t^2)
27 //1000 = 295*t - 4.9*t^2;
28 //4.9*t^2 - 295*t + 1000 = 0;
29 a = 4.9;
30 b = -295;
31 c = 1000;
32 //Solving quadratic equation(taking lesser value
   since the shell can hit the plane twice, so least
    time is considered):
33 t = poly(0,"t");
34 p = a*t^2 + b*t + c;
35 t = roots(p);
36 printf("Time after firing the shell will hit the
   plane = %.2f s\n",t(2));     //The answers vary
   due to round off error
37
38 //Horizontal distance of the plane from the gun:
39 AB = v*t(2);           //(m)
40 printf("Horizontal distance of the plane from the
   gun = %.2f m",AB);     //The answers vary due to
```

```
    round off error
```

Scilab code Exa 20.4 To find the time taken by the projectile to reach the ground

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 alpha = 30;       //(Degrees)(Angle of projection
     with the horizontal)
6 u = 40;           //(m/s)(Velocity of projection)
7 g = 9.8;          //(m/s^2)(Accn due to Gravity)
8
9 t = (2*u*sind(alpha))/g;      //(s)
10 printf("Time taken by the projectile to reach the
     ground after the instant of firing = %.2f s",t);
```

Scilab code Exa 20.5 To find the horizontal range

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 u = 15;          //(m/s)(Velocity of projection)
6 alpha = 25;      //(Degrees)(Angle of projection with
     the horizontal)
7 g = 9.8;         //(m/s^2)(Accn due to Gravity)
8
9 R = (u^2 * sind(2*alpha))/g;       //(m)(Horizontal
     range of the ball)
10 printf("Horizontal range of the ball = %.2f m",R);
       //The answers vary due to round off error
```

Scilab code Exa 20.6 To find the height risen by the bullet

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 u = 100;          //(m/s)(Velocity of projection)
6 alpha = 45;       //(Degrees)(Angle of projection with
     the horizontal)
7 g = 9.8;          //(m/s^2)(Accn due to Gravity)
8
9 H = (u^2 * (sind(alpha))^2)/(2*g);     //(m)
10 printf("Maximum height to which the bullet will rise
     = %.2f m",H);
```

Scilab code Exa 20.7 To find the angle of projection and the greatest possible ran

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 H = 5;     //(m)(Height of the tunnel)
6 u = 60;    //(m/s)(Velocity of projection)
7 g = 9.8;   //(m/s^2)(Accn due to Gravity)
8
9 //Let alpha = Angle of projection
10 //Using H = (u^2 * (sind(alpha))^2)/(2*g);
11 alpha = asind(sqrt((H*2*g)/(u^2)));
12 printf("Angle of projection = %.2f degrees\n",alpha)
     ;
13
14 //Greatest possible range:
```

```
15 R = (u^2 * sind(2*alpha))/g;     //(m)
16 printf("Greatest possible range = %.2f m",R);     //
   The answers vary due to round off error
```

Scilab code Exa 20.9 To find the velocity of the particle

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  u1 = 5;     //(m/s)(Velocity of projection of first
      particle)
6  alpha1 = 60;     //(Degrees)(Angle of projection of
      first particle with the horizontal)
7  alpha2 = 45;    //(Degrees)(Angle of projection of
      second particle with the horizontal)
8  g = 9.8;        //(m/s^2)(Accn due to Gravity)
9
10 //Let u2 = Velocity of projection of the second
      particle.
11 //(a)Velocity of the second particle for equal
      horizontal range:
12 //Using R = (u^2 * sind(2*alpha))/g;
13 //For equal horizontal range:
14 //(u1^2*sind(2*alpha1))/g = (u2^2*sind(2*alpha2))/g
15 u2 = sqrt(u1^2 * (sind(2*alpha1)/sind(2*alpha2)));
      //(m/s)
16 printf("Velocity of projection of the second
      particle = %.2f m/s\n",u2);
17
18 //(b)Velocity of the second particle for equal
      maximum height:
19 //H = (u^2 * (sind(alpha))^2)/(2*g);
20 //For equal maximum height:
21 //(u1^2 * (sind(alpha1))^2)/(2*g) = (u2^2 * (sind(
```

```
    alpha2))^2)/(2*g);
22 u2 = sqrt(u1^2 * (((sind(alpha1))^2)/((sind(alpha2))
    ^2)));
23 printf("Velocity of the second particle for equal
    maximum height = %.2f m/s\n",u2);
24
25 //(c)Velocity of the second particle for equal time
    of flight:
26 //t = (2*u*sind(alpha))/g;     (Time of flight)
27
28 //For equal time of flight:
29 //(2*u1*sind(alpha1))/g = (2*u2*sind(alpha2))/g;
30 u2 = u1 * (sind(alpha1)/sind(alpha2));      //(m/s)
31 printf("Velocity of the second particle for equal
    time of flight = %.2f m/s",u2);
```

Scilab code Exa 20.11 To find the angle of projection

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 alpha1 = 15;      //(Degrees)(Angle of projection
    with the horizontal)
6 //R1 = R - 12;     //(m)(Horizontal range)
7
8 alpha2 = 45;      //(Angle of projection with the
    horizontal)
9 g = 9.8;          //(m/s^2)(Accn due to Gravity)
10 //R2 = R + 24;    //(m)(Horizontal range)
11
12 //Let u = Velocity of projection,
13 //alpha = Angle of projection to hit the mark.
14
15 //(1):
```

```
// R - 12 = (u^2 * sind(30))/g;

//(2):
// R + 24 = (u^2 * sind(90))/g;

//(1)/(2):
R = 24/0.5;          //(m)

//Substituting R in (1):
u = sqrt(((R - 12)*g)*2);       //(m/s)

//H = (u^2 * sind(2*alpha))/g;     (Horizontal
   distance)
H = 48;
alpha = asind(((H * g)/u^2))/2;        //(Degrees)
printf("Angle of projection to hit the mark = %.2f
   degrees",alpha);
```

Scilab code Exa 20.12 To find the greatest elevation and horizontal distance

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
h = 150;       //(m)(Height of the cliff)
u = 180;       //(m/s)(Velocity of projection)
alpha = 30;    //(Degrees)(Angle of projection with
   the horizontal)
g = 9.8;       //(m/s^2)(Accn due to Gravity)

//1. The greatest elevation above the ground reached
    by the projectile:
H = (u^2 * (sind(alpha)^2))/(2*g);     //(m)

//Greatest elevation above the ground reached by the
```

```
    projectile:
s = H + h;          //(m)
printf("Greatest elevation above the ground reached
   by the projectile = %.2f m\n",s);    //The
   answers vary due to round off error

//2. The horizontal distance from the gun to the
   point, where the projectile strikes the ground:

//Time taken by the projectile to reach the maximum
   height:
t1 = (u * sind(alpha))/g;       //(s)

//t2 = Time taken by the projectile to reach the
   ground from the maximum height.
//Vertical distance: ( s = u*t + (1/2)*g*t^2 )

t2 = sqrt(s/((1/2)*g));      //(s)

//Total time:
t = t1 + t2;        //(s)

//Horizontal distance:
R = (180 * cosd(30)) * t;      //(km)
printf("Horizontal distance from the gun to the
   point, where the projectile strikes the ground =
   %.2f km",R/1000);
```

Scilab code Exa 20.13 To calculate the actual velocity of the bullet

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
alpha = 30;      //(Degrees)(Angle of projection with
```

```
     the horizontal)
u1 = 100;        //(m/s)(Initial velocity of
   projection)
g = 9.8;        //(m/s^2)(Accn due to Gravity)

//Maximum height:
H = (u1^2 * (sind(alpha))^2)/(2*g);       //(m)

//Time taken:
t1 = (u1*sind(alpha))/g;                  //(s)
   -(1)

//Considering the vertical motion of the bullet from
     the maximum height to the target due to
   gravitational acceleration only:
u = 0;           //(Initial velocity)
s = H + 80;      //(Total distance)

//Let t2 = Time taken by the bullet to reach the
   target from the maximum height.
//Using s = u*t2 + (1/2)*g*t2^2;
t2 = sqrt(s/((1/2)*g));      //(s)     -(2)

//Total time required:
T = t1 + t2;        //(s)

//Final velocity of the bullet in the vertical
   direction:
v = u + g*t2;          //(m/s)

//Horizontal component of the initial velocity:
vh = u1*cosd(alpha);    //(m/s)

//Actual velocity with which the bullet will strike
   the target:
V = sqrt(v^2 + vh^2);     //(m/s)
printf("Actual velocity with which the bullet will
   strike the target = %.2f m/s",V);
```

Scilab code Exa 20.14 To find the angle of projection

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 u = 30;       //(m/s)(Initial velocity)
6 OB = 15;      //(m)(Distance of point of projection
     from wall)
7 h = 6;        //(m)(Height of the wall)
8
9 //Let alpha = Angle of projection
10 //Vertical component = 30 * cosd(alpha)
11
12 //Vertical distance travelled by the shot:
13 //6 = (30 * sind(alpha))*t - 4.9*t^2      -(1)
14
15
16 //In order to enable the shot just to clear the top
     of the wall, it must traverse 15 m in tseconds:
17 //t = 15/(30 * cosd(alpha));      -(2)
18
19 //Substituting (2) in (1):
20 //1.225*tand(alpha)^2 - 15*tand(alpha) + 7.225 = 0;
21 //Let tand(alpha) = y
22
23 p = poly([7.225 -15 1.225],'y','c');
24 y = roots(p);
25
26 a1 = real(y(1));
27 a2 = real(y(2));
28 alpha1 = atand(a1);
29 alpha2 = atand(a2);
30 printf("Angle of projection: %.2f degrees or %.2f
```

```
    degrees",alpha1,alpha2);
31 //The answers vary due to round off error
```

Scilab code Exa 20.15 To find the least initial velocity of the projectile

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  h = 3.6;      //(m)(Height of wall)
6  OB = 4.8;     //(m)(Distance of the wall fom the
       point of projection)
7  BC = 3.6;     //(m)(Distance of strike point from the
        foot of the wall)
8  g = 9.8;      //(m/s^2)(Accn due to Gravity)
9
10 //Let u = Initial velocity of projection,
11 //alpha = Angle of projection.
12
13 //Range:
14 //4.8 + 3.6 = (u^2 * sind(2*alpha))/(2*g);
15 //u^2 = (8.4*g)/(2*sind(alpha)*cosd(alpha))      -(1)
16
17 //Equation of the path of trajectory:
18 //3.6 = 4.8*tand(alpha) - (g*(4.8)^2)/(2*u^2*cosd(
       alpha)^2);
19 //Substituting u^2 in equation (1):
20 alpha = atand(h/(OB-2.74));       //(Degrees)
21
22 //Substituting alpha in equation (1):
23 u = sqrt((4.2 * g)/(sind(alpha)*cosd(alpha)));    //(
       m/s)
24
25 printf("Least initial velocity = %.2f m/s",u);
```

Scilab code Exa 20.16 To find when and where the two bullets will meet each other

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 alpha = 30;       //(Degrees)(Angle of projection of
     both the guns)
6 vA = 350;         //(m/s)(Velocity of projection of
     first gun)
7 vB = 300;         //(m/s)(Velocity of projection of
     second gun)
8 d = 30;           //(m)(Distance between the muzzles)
9
10 //Let t = Time in seconds, when the two shots meet
     after they leave the guns,
11 //x = Horizontal distance between A and C
12 //y = Vertical distance between A and C
13
14 AD = d * cosd(alpha);       //(m)
15
16 //x = 350*cosd(30)*t = 175*sqrt(3)*t;    -(1)
17
18 //(15*sqrt(3)) = 300*cosd(30)*t = 150*sqrt(3)*t;
     -(2)
19
20 //(1) + (2):
21 t = (d*cosd(alpha))/(vA*cosd(alpha) + vB*cosd(alpha)
     );        //(s)
22 printf("Time in seconds, when the two shots meet
     after they leave the guns = %.2f s\n",t);     //
     The answers vary due to roun off error
23
24 //Point where the two shots meet:
```

```
25 //Substituting t in (1):
26 x = vA*cosd(alpha)*t;      //(m)
27
28 u = vA*sind(alpha);        //(m/s)(Vertical
      component of vA)
29
30 //Vertical distance between A and C:
31 y = u*t - (1/2)*g*t^2;      //(m)
32
33 printf("Points where the two shots meet: x = %.2f m
      and y = %.2f m",x,y);
34 //The answers vary due to roun off error
```

Scilab code Exa 20.17 To find the velocity and direction of the projectile

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2 firing.
3 clc
4 clear all
5 u = 80;      //(m/s)(Initial velocity of projection)
6 alpha = 65;    //(Degrees)(Angle of projection with
      the horizontal)
7 t = 5;        //(s)(Time)
8
9 //Velocity of the projectile:
10 V = sqrt(u^2 + g^2*t^2 - 2*u*sind(alpha)*g*t);     //
      (m/s)
11
12 //Direction of the projectile:
13 //Let theta = Angle which the projectile makes with
      the horizontal
14 theta = atand((u*sind(alpha) - g*t)/(u*cosd(alpha)))
      ; //(Degrees)
15
16 printf("Velocity of the projectile = %.2f m/s\n",V);
```

```
        //The answers vary due to round off error
17 printf("Angle which the projectile makes with the
    horizontal = %.2f degrees",theta);
```

Scilab code Exa 20.18 To find time and distance

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  u = 100;       //(m/s)(Initial velocity of the
       projectile)
6  alpha = 45;    //(Degrees)(Angle of projection)
7  theta = 30;    //(Degrees)(Angle of projection at
       point P)
8  g = 9.8;       //(m/s^2)(Accn due to Gravity)
9
10 //Let t = Time for the particle to reach the point P
       from O.
11
12 //tand(theta) = (u*sind(alpha) - g*t)/(u*cosd(alpha)
       )
13 t = -((tand(theta)*u*cosd(alpha)) - (u*sind(alpha)))
       /g;     //(s)
14 printf("Time for the particle to reach the point P
       from O = %.2f s\n",t);
15
16 //Distance OP:
17 x = u*cosd(alpha) * t;     //(m)
18
19 //Vertical component of velocity:
20 uy = u*sind(alpha);        //(m)
21
22 //Vertical distance AP:
23 y = uy*t - (1/2)*g*t^2;    //(m)
```

```
24
25 OP = sqrt(x^2 + y^2);        //(m)
26 printf("Distance OP = %.2f m",OP);
```

Scilab code Exa 20.19 To find the velocity and direction of the projectile

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 u = 500;      //(m/s)(Velocity of projection)
6 alpha = 35;     //(Degrees)(Angle of projection)
7 g = 9.8;      //(m/s^2)(Accn due to Gravity)
8
9 //Velocity of the projectile after 29 seconds:
10 t = 29;        //(s)
11 v29 = sqrt(u^2 + g^2*t^2 - 2*u*sind(alpha)*g*t);
      //(m/s)
12
13 //Velocity of the particle after 30 seconds:
14 t = 30;       //(s)
15 v30 = sqrt(u^2 + g^2*t^2 - 2*u*sind(alpha)*g*t);
      //(m/s)
16
17 printf("Velocity of the projectile after 29 seconds
      = %.2f m/s\n",v29);
18 printf("Velocity of the particle after 30 seconds =
      %.2f m/s\n",v30);
19
20 //Direction of the projectile after 29 and 30
      seconds:
21 t = 29;       //(s)
22 theta29 = atand((u*sind(alpha) - g*t)/(u*cosd(alpha)
      ));      //(Degrees)
23
```

```
24 t = 30;      //(s)
25 theta30 = atand((u*sind(alpha) - g*t)/(u*cosd(alpha)
   ));      //(Degrees)
26
27 printf("Direction of the projectile after 29 seconds
    = %.2f degrees\n",theta29);
28 printf("Direction of the projectile after 30 seconds
    = %.2f degrees",theta30);
```

Scilab code Exa 20.20 To find time when the particle will move perpendicular to it

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 u = 100;       //(m/s)(Initial velocity of projection
   )
6 alpha = 30;    //(Degrees)(Angle of projection with
   the horizontal)
7 g= 9.8;        //(m/s^2)(Accn due to Gravity)
8
9 //Let t = Time from the instant of projection, when
   the particle will move perpendicular to its
   initial direction.
10 theta = -60;     //(Degrees)(Actual angle)
11
12 //tand(theta) = (u*sind(alpha) - g*t)/(u*cosd(alpha)
   );
13 t = -((tand(theta)*u*cosd(alpha)) - u*sind(alpha))/g
   ;    //(s)
14 printf("Time when the particle will move
   perpendicular to its initial direction = %.2f s",
   t);
```

Scilab code Exa 20.21 To find the velocity and direction of a body

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 u = 50;          //(m/s)(Initial velocity)
6 alpha = 50;      //(Degrees)(Angle of projection)
7 h = 30;          //(m)(Height)
8 g = 9.8;         //(m/s^2)(Accn due to Gravity)
9
10 //(i) Velocity of the projectile:
11 v = sqrt(u^2 - 2*g*h);      //(m/s)
12 printf("Velocity of the projectile = %.2f m/s\n",v);
13
14 //(ii) Direction of the projectile:
15 theta = atand((sqrt(u^2*sind(alpha)^2 - 2*g*h))/(u*
      cosd(alpha)));
16 printf("Direction of the projectile = %.2f degrees",
      theta);     //The answers vary due to round off
      error
```

Scilab code Exa 20.22 To find the angle of projection

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 //Given: velocity of the particle at its greatest
      height = sqrt(2/5) * velocityat half of it
      greatest height.
6
```

```
7 //Let u = Initial velocity of projection,
8 //alpha = Angle of projection with the horizontal.
9
10 //Velocity of projectile at greatest height:
11 //v = u*cosd(alpha)    -(1)
12
13 //Half of greatest height:
14 //h = (1/2)*((u^2*sind(alpha)^2)/(2*g));
15
16 //Velocity of projectile at half of the greatest
   height:
17 //V = sqrt(u^2 - 2*g*h)    -(2)
18
19 //Substituting (1) and (2) in the given equation:
20 alpha = asind(sqrt(3/4));     //(Degrees)
21 printf("Angle of projection with the horizontal = %
   .2f degrees",alpha);
```

Scilab code Exa 20.23 To find the time of flight of the ball

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 u = 10;          //(m/s)(Velocity of projection)
6 alpha = 35;      //(Degrees)(Angle of projection with
   the horizontal)
7 Beta = 15;         //(Degrees)(Inclination of the
   plane)
8
9 //Time of flight when the ball is projected upwards:
10 t1 = (2*u*sind(alpha - Beta))/(g*cosd(Beta));     //
   (s)
11 printf("Time of flight when the ball is projected
   upwards = %.2f s\n",t1);
```

```
12
13 //Time of flight when the ball is projected
      downwards:
14 t2 = (2*u*sind(alpha + Beta))/(g*cosd(Beta));
      //(s)
15 printf("Time of flight when the ball is projected
      downwards = %.2f s",t2);
```

Scilab code Exa 20.24 To find the range and time of flight of the particle

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 u = 30;          //(m/s)(Velocity of projection)
6 alpha = 55;      //(Degrees)(Angle of projection with
      the horizontal)
7 Beta = 20;          //(Degrees)(Angle of plane)
8 g = 9.8;         //(m/s^2)(Accn due to Gravity)
9
10 //Maximum range:
11 //For maximu range, angle of projection:
12 alpha = 180/4 + Beta/2;     //(Degrees)
13
14 printf("alpha = %2.f degrees\n",alpha);
15 printf("Since the given angle of projection is 55
      degrees, therefore the range up the plane is
      maximum one for the given plane.\n")
16
17 //Range of the projectile:
18 R = (u^2)/(g * cosd(Beta)^2) * [ sind(2*alpha - Beta
      ) - sind(Beta)];        //(m)
19 printf("Range of the projectile = %.2f m\n",R);
20
21 //Time of flight:
```

```
22 t = (2*u*sind(alpha - Beta))/(g * cosd(Beta));
     //(s)
23 printf("Time of flight = %.2f s",t);
```

Scilab code Exa 20.25 To find the range of the plane

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  Beta = atand(5/12);      //(Degrees)
6  u = 200;        //(m/s)(Velocity of projection with
       the horizontal)
7  alpha = 30;           //(Degrees)(Angle of projection
       )
8  g = 9.8;       //(m/s^2)(Accn due to Gravity)
9
10 //(a) Range of the plane, when the shot is fired up
       the plane:
11 R1 = (u^2)/(g*cosd(Beta)^2) * [sind(2*alpha - Beta)
       - sind(Beta)];        //(m)
12 printf("Range of the plane, when the shot is fired
       up the plane = %.2f m\n",R1);     //The answers
       vary due to round off error
13
14 //(b) Range of the plane, when the shot is fired
       down the plane:
15 R2 = (u^2)/(g*cosd(Beta)^2) * [sind(2*alpha + Beta)
       + sind(Beta)];        //(m)
16 printf("Range of the plane, when the shot is fired
       down the plan = %.2f m",R2);     //The answers
       vary due to round off error
```

Chapter 21

Motion of Rotation

Scilab code Exa 21.1 To find the angular velocity and angular displacement

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 omega0 = 0;     //(Initial angular velocity)
6 alpha = 0.5;      //(rad/sec^2)(Angular acceleration)
7 t = 10;     //(sec)(Time)
8
9 //Angular velocity of the flywheel:
10 omega = omega0 + alpha*t;      //(rad/sec)
11 printf("Angular velocity of the flywheel = %.2f rad/
   sec\n",omega);
12
13 //Angular displacement of the flywheel:
14 theta = omega0*t + (1/2)*alpha*t^2;          //(rad)
15 printf("Angular displacement of the flywheel = %.2f
   rad",theta);
```

Scilab code Exa 21.2 To find angular acceleration and number of revolutions of the

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 omega0 = 1.5*%pi;     //(rad/sec)(Initial angular
     velocity)
6 omega = 3*%pi;        //(rad/sec)(Final angular
     velocity)
7 t = 30;           //(sec)
8
9 //(a) Angular acceleration of the wheel:
10 //Let alpha = Angular acceleration of the wheel
11 //Using W = W0 + alpha * t;
12 alpha = ((omega - omega0)/t);       //(rad/sec^2)
13 printf("Angular acceleration of the wheel = %.2f pi
     rad/sec^2\n",alpha/%pi);
14
15 //(b) No. of revolutions made by the wheel in 30
     seconds:
16 //Using theta = W0*t + (1/2)*(alpha)*t^2;
17 theta = omega0*t + (1/2)*(alpha)*t^2;      //(rad)
18 rev = theta/(2*%pi);        //(rev)
19 printf("No. of revolutions made by the wheel in 30
     seconds = %.2f rev",rev);
```

Scilab code Exa 21.3 To find the number of revolutions

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 omega0 = 6*%pi;      //(rad/sec)(Initial angular
     velocity)
6 omega = 4*%pi;       //(rad/sec)(Final angular
     velocity)
```

```
7 t = 20;          //(sec)(Time)
8
9 //Revolutions of the wheel, before it stops:
10 //Let alpha = Uniform angular acceleration,
11 //theta = Angular displacement of the flywheel
      before coming to rest.
12
13 //Using W = W0 + alpha * t;
14 alpha = (omega - omega0)/t;      //(rad/sec^2)
15
16 //Using W^2 = W0^2 + 2*(alpha)*theta
17 theta = omega0^2/(2*alpha);
18 rev = theta/(2*%pi);     //(rev)
19 printf("Revolutions of the wheel, before it stops =
      %.2f rev\n",abs(rev));
20
21 //Time in which the wheel will come to rest:
22 //Using W = W0 + alpha * t;
23 omega = 0;       //(Final velocity)
24 t = (omega - omega0)/alpha;       //(sec)
25 printf("Time in which the wheel will come to rest =
      %.2f min",t/60);
```

Scilab code Exa 21.4 To find the time when the pulley will come to rest

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 omega0 = 0;      //(Initial angular velocity)
6 alpha1 = 0.5;     //(rad/s^2)(Acceleration)
7 t1 = 120;         //(s)(Time)
8
9 omega = (omega0 + alpha1 * t1)/(2*%pi) * 60;      //(
      r.p.m.)
```

```
10 printf("Angulae speed at the end of 2 minutes = %.2f
       r.p.m.\n",omega);     //The answers vary due to
   round off error
11
12 //Let t2 = Time in which the pulley will come to
   rest.
13 omega0 = 60;     //(rad/sec)(Initial angular velocity
   )
14 omega = 0;       //(Final angular velocity)
15 alpha2 = -0.3;     //(rad/s^2)(Retardation)
16 t2 = -omega0/alpha2;     //(sec)
17 printf("Time in which the pulley will come to rest =
    %.2f sec",t2);
```

Scilab code Exa 21.6 To find the total time taken to complete 400 revolutions

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5
6 //For first 4 seconds:
7 omega0 = 20*%pi;      //(rad/s)(Initial angular
   velocity)
8 omega = 36*%pi;       //(rad/s)(Final angular
   velocity)
9 t1 = 4;       //(sec)(Time taken during constant
   accelration)
10 t2 = 8;       //(sec)(Time taken during uniform
   angular velocity)
11 theta = 800*%pi;    //(rad)(Total angular
   displacement)
12
13 //Let alpha = Angular acceleration of the shaft.
14 //Using W = W0 + alpha*t1;
```

```
15 alpha = (omega - omega0)/t1;      //(rad/s^2)
      -(1)
16
17 //Angular displacement:
18 theta1 = omega0*t1 + (1/2)*alpha*t1^2;     //(rad)
       -(2)
19
20 //For the next 8 seconds:
21 omega0 = 36*%pi;      //(rad/s)(Initial velocity)
22 alpha = 4*%pi;    //(rad/s^2)(Angular acceleration)
23
24 omega = omega0 + alpha*t2;      //(rad/s)       -(3)
25 theta2 = omega0*t2 + (1/2)*(alpha)*(t2^2);     //(rad
      )    -(4)
26
27 //For constant angular velocity of 68*pi rad/s:
28 theta3 = theta - theta1 - theta2;          //(rad)
29
30 //Time taken:
31 t3 = theta3/(68*%pi);          //(s)
32
33 //Total time to complete 400 revolution or 800*pi
      rad:
34 T = t1 + t2 + t3;     //(s)
35 printf("Total time to complete 400 revolution or
      800*pi rad = %.2f sec",T);
```

Scilab code Exa 21.7 To find the angular acceleration maximum angular velocity and

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 theta = %pi/2;      //(rad)(Angular displacement)
6 T = 120;       //(s)(Total time)
```

```
t1 = 40;        //(sec)(Time for acceleration)
t2 = 60;        //(sec)(Time for uniform velocity)
t3 = 20;        //(sec)(Time for retardation)

//(i) Angular acceleration of the bridge:
//alpha1 = Angular acceleration of the bridge,
//alpha2 = Angular retardation of the bridge.

//For first 40 sec:
omega1 = 0;      //(Initial angular velocity)
t1 = 40;     //(sec)(Time)
//omega = omega1 + alpha1*t1 = 40*alpha1;    -(1)

//theta1 = W0*t1 + (1/2)*alpha1*t1^2 = 800*alpha1;
     -(2)

//For next 60 seconds:
t2 = 60;      //(s)
//theta2 = 2400*alpha1              -(iii)

//For last 20 seconds:
omega = 0;
t3 = 20;        //(s)
//alpha2 = 2*alpha1;

//Angular displacement:
//theta3 = 400*alpha1               -(iv)

//Total displacement:
//0.5*pi = theta1 + theta2 + theta3;
alpha1 = (0.5*%pi)/(3600);     //(rad/s^2)
printf("Angular acceleration of the bridge = %.2f *
   10^(-3) rad/s^2\n",alpha1 * 1000);

//(ii)Maximum angular velocity of the bridge:
omega = 40*alpha1;                  //(rad/s)
printf("Maximum angular velocity of the bridge = %.2
   f * 10^(-3) rad/s\n",omega*1000);
```

```
42
43 //(iii)Angular retardation of the bridge:
44 alpha2 = 2*alpha1;              //(rad/s^2)
45 printf("Angular retardation of the bridge = %.2f *
      10^(-3) rad/s^2",alpha2*1000);
```

Scilab code Exa 21.8 To determine the value of constant retardation time taken in

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 t = 10 - 0;     //(sec)(Time interval)
6 theta = 600*%pi;      //(rad)(Angular displacement)
7
8 //When time:
9 t1 = 7.5;      //(sec)
10 omega = 40*%pi;     //(rad/sec)
11 //(i) Value of  constant retardation:
12 //Let alpha = Constant retardation,
13 //omega0 =Initial angular velocity.
14
15 //Considering motion from 0 to 10 seconds;
16 //Angular displacement:
17 //600*pi = 10*omega0 - 50*alpha;       -(1)
18
19 //Considering motion of flywheel 0 to 7.5 seconds:
20 //Angular velocity:
21 //alpha = (omega0 - 40*pi)/7.5;        //(rad/s^2)
       -(2)
22
23 //Substituting the value of alpha in equation (i):
24 omega0 = (theta - (t^2/2)*omega/t1)/(t - (t^2/2)/t1)
      ;       //(rad/sec)
25
```

```
26 //Substituting omega0 in equation (iii):
27 alpha = (omega0 - omega)/t1;            //(rad/sec^2)
28 printf("Constant retardation = %.2f rad/sec^2\n",
      alpha);
29
30 //(ii) Total time taken by the flywheel to come to
      rest:
31 omega0 = 100*%pi;          //(rad/sec)
32 omega = 0;                 //(rad/sec)
33 alpha = 8*%pi;         //(rad/sec^2)
34
35 //Using: omega = omega0 + alpha*t;
36 t = omega0/alpha;           //(s)
37 printf("Total time taken by the flywheel to come to
      rest = %.2f sec\n",t);
38
39 //(iii)Total revolutions made till it comes to rest:
40 t = 12.5;        //(s)
41 theta = (omega0*t - (1/2)*alpha*t^2)/(2*%pi);     //(
      rev)
42 printf("Total revolutions made till it comes to rest
      = %.2f rev",theta);
```

Scilab code Exa 21.9 To find the linear velocity of a point on the periphery of a

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 r = 0.6;        //(m)(Radius of wheel)
6 omega0 = 0;          //(Initial angular velocity)
7 alpha = 0.8;   //(rad/s^2)(Angular acceleration)
8 t = 5;          //(s)(Time)
9
10 //Angular velocity:
```

```
11 omega = omega0 + alpha*t;      //(rad/sec)
12
13 //Linear velocity:
14 v = r*omega;                  //(m/s)
15 printf("Linear velocity of a point on its periphery
      = %.2f m/s",v);
```

Scilab code Exa 21.10 To find the linear velocity of a point on the periphery of a

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 r = 1;        //(m)(Radius of the pulley)
6 N = 240;      //(r.p.m.)(Angular frequency)
7
8 //Angular velocity of the pulley:
9 omega = (2 * %pi * N)/60;      //(rad/s)
10 printf("Angular velocity of the pulley = %.2f rad/
      sec\n",omega);
11
12 //Linear velocity of the particle on the periphery
      of the pulley:
13 v = r * omega;        //(m/s)
14 printf("Linear velocity of the particle on the
      periphery of the pulley = %.2f m/s",v);
```

Scilab code Exa 21.11 To find the angular retardation of the wheels

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
```

```
5 v = 20;      //(m/s)(Linear velocity)
6 r = 0.375;   //(m)(Radius)
7 s = 20;      //(m)(Distance travelled by the car)
8
9 //Angular retardation of the wheel:
10 omega = v/r;      //(rad/sec)
11
12 //Let a = Linear retardation of the wheel.
13 //v^2 = u^2 + 2*a*s;
14
15 a = -v^2/(2*s);      //(m/s^2)
16
17 //Angular retardation:
18 alpha = a/r;         //(rad/sec^2)
19 printf("Angular retardation of the wheel = %.2f rad/
   sec^2",alpha);     //The answers vary due to round
    off error
```

Scilab code Exa 21.12 To find the angular velocity displacement and acceleration c

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 t = poly(0,"t");
6 //Given:
7 theta = 2*t^3 + 0.5;          //-(i)
8 x = theta;
9 //Angular displacement after 2 seconds:
10 //Substituting t = 2 in equation (i):
11 theta = pol2str(theta);
12 t = 2;
13 theta = evstr(theta);     //(rad)
14 printf("Angular displacement after 2 seconds = %.2f
   rad\n",theta);
```

```
15
16 //Angular velocity after 2 seconds:
17 //Differentiating eqn (i) w.r.t. t;
18 omega = derivat(x);                    //-(ii)
19 y = omega;
20 //Substituting t = 2 in equation (ii):
21 omega = pol2str(omega);
22 t = 2;
23 omega = evstr(omega);          //(rad/sec)
24 printf("Angular velocity after 2 seconds = %.2f rad/
      sec\n",omega);
25
26 //Angular acceleration after 2 seconds:
27 //Differentiating eqn (ii) w.r.t. t;
28 alpha = derivat(y);                    //-(iii)
29 //Substituting t = 2 in eqn (iii):
30 alpha = pol2str(alpha);
31 t = 2;
32 alpha = evstr(alpha);       //(rad/sec^2)
33 printf("Angular acceleration after 2 seconds = %.2f
      rad/sec^2",alpha);
```

Scilab code Exa 21.13 To find the angular velocity and acceleration time and maxim

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 t = poly(0,"t");
6 //Given:
7 theta = 18*t + 3*t^2 - 2*t^3;          //-(i)
8
9 //(i) Angular velocity and acceleration at start:
10 //Differentiating equation (i) w.r.t. t;
11 omega = derivat(theta);                    //-(ii)
```

```
x = omega;
//Substituting t = 0 in equation (ii):
omega = pol2str(omega);
t = 0;
omega = evstr(omega);
printf("Angular velocity at start = %.2f rad/s\n",
   omega);

//Differentiating (ii) w.r.t. t:
alpha = derivat(x);                    //-(iii)
//Substituting t = 0 in equation (iii):
alpha = pol2str(alpha);
t = 0;
alpha = evstr(alpha);     //(rad/s^2)
printf("Angular acceleration at start = %.2f rad/s
   ^2\n",alpha);

//(ii) Time when the particle reaches maimum angular
    velocity:
//Equating eqn (iii) to 0;
//6 - 12*t = 0
t = 6/12;          //(sec)
printf("Time when the particle reaches maximum
   angular velocity = %.2f sec\n",t);

//(iii)Maximum angular velocity of the particle:
//Substituting t = 0.5 in equation (ii).
omega_max = omega + (alpha * t) - alpha * (t)^2;
   //(rad/sec)
printf("Maximum angular velocity of the particle = %
   .2f rad/s",omega_max);
```

Chapter 22

Combined Motion of Rotation and Translation

Scilab code Exa 22.1 To find the velocity of B

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 A = 60;      //(Degrees)(Inclination of the link with
      the horizontal)
6 vA = 2;      //(m/s)(Velocity of point A in
     horizontal direction)
7
8 //Let vB = Velocity of B in the vertical direction.
9 //vB/vA = cotd(60);
10 vB = vA * cotd(60);       //(m/s)
11 printf("Velocity of B in the vertical direction = %
     .2f m/s",vB);
```

Scilab code Exa 22.2 To find the velocity of C

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
vA = 5;      //(m/s)(Velocity of A)
OA = 2.6;    //(cm)
OC = 5.4;    //(cm)

//vC/vA = OC/OA;
vC = vA * (OC/OA);      //(m/s)
printf("Velocity of C = %.2f m/s",vC);     //The
   answers vary due to round off error
```

Scilab code Exa 22.3 To find the velocity of the piston

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
l = 1.125;      //(m)(Length of connecting rod)
r = 0.25;       //(m)(Length of crank)
N = 420;        //(r.p.m.)(Angular rotation of crank)
theta = 40;     //(Degrees)(Angle through which the
   crank has turned)

//Angular velocity of crank:
omega = (2 * %pi * N)/60;        //(rad/s)

//Velocity of B:
vB = omega * r;                  //(m/s)

OB = 1.45;        //(m)
OC = 1.1;         //(m)

//Velocity of C:
```

```
20 vC = (vB * OC)/OB;              //(m/s)
21 printf("Velocity of C = %.2f m/s",vC);
```

Scilab code Exa 22.4 To find the velocity of the piston

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 r = 0.3;        //(m)(Radius of the crank)
6 l = 1.2;        //(m)(Length if connecting rod)
7 N = 180;        //(r.p.m.)(Angular rotation of crank)
8 theta = 45;     //(Degrees)(Angle traversed by the
     crank)
9 AB = r;
10 BC = l;
11
12 //Angular velocity of the crank:
13 omega1 = (2 * %pi * N)/60;      //(rad/sec)
14
15 //From geometry:
16 phi = asind((AB * sind(45))/BC);      //(Degrees)
17
18 //Velocity of the piston:
19 Vc = omega1 * (l*sind(phi) + r*cosd(theta)*tand(phi)
     );    //(m/s)
20 printf("Velocity of the piston = %.2f m/s",Vc);
```

Scilab code Exa 22.5 To find the velocity of the piston

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
```

```
clear all
l = 1.125;      //(m)(Length of connecting rod)
r = 0.25;       //(m)(Length of crank)
N = 420;        //(r.p.m.)(Angular rotation of crank)
theta = 40;     //(Degrees)(Angle traversed by the
   crank)

//Angular velocity of crank:
omega = (2 * %pi * N)/60;          //(rad/s)

//Velocity of B:
vB = omega * r;                    //(m/s)

vC = 8.3;                          //(m/s)
printf("Velocity of C = %.2f m/s",vC);
```

Scilab code Exa 22.6 To find the velocity of the piston C angular velocity of the

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
N = 5;          //(rev/s)(Angular rotation of crank)
r = 0.3;        //(m)(Length of the crank)
l = 0.6;        //(m)(Length of the link CB)
BC = l;

//Angular velocity of crank:
omega = (2 * %pi * N);          //(rad/s)

//Velocity of B:
vB = omega * r;                 //(m/s)

vC = 6.65;          //(m/s)
printf("Velocity of piston = %.2f m/s\n",vC);
```

```
18
19 bc = 6.65;        //(m/s)
20
21 //Angular velocity of connecting rod:
22 omega = (bc/BC)/(2 * %pi) * 60;        //(r.p.m.)
23 printf("Angular velocity of connecting rod = %.2f r.
      p.m.\n",omega);     //The answers vary due to
      round off error
24
25 //Velocity of D:
26 vD = 7.43;             //(m/s)
27 printf("Velocity of D = %.2f m/s",vD);
```

Chapter 23

Simple Harmonic Motion

Scilab code Exa 23.1 To find the velocity and acceleration of the piston

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 N = 120;        //(r.p.m.)(Frequency of piston)
6 l = 2;          //(m)(Stroke length)
7 r = 1;          //(m)(Radius)
8 y = 0.75;       //(m)(Distance of piston from the
     centre)
9
10 //Angular velocity of piston:
11 omega = (2 * %pi * N)/60;          //(rad/sec)
12
13 //Velocity of piston:
14 v = omega * sqrt(r^2 - y^2);       //(m/s)
15 printf("Velocity of piston = %.2f m/s\n",v);
16
17 //Acceleration of piston:
18 a = omega^2 * y;                   //(m/s^2)
19 printf("Acceleration of piston = %.2f m/s^2",a);
```

Scilab code Exa 23.2 To find the velocity and acceleration of the body

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 r = 1;        //(m)(Amplitude)
6 T = 2;        //(s)(Periodic time)
7 t = 0.4;      //(s)(Time taken by the body from
     extreme position)
8
9 //Angular velocity of the body:
10 omega = (2 * %pi)/T;          //(rad/s)
11
12 //Displacement of the body after 0.4 sec from the
     extreme position(or 0.1 second from the mean
     position):
13 t = 0.1;                    //(s)
14 y = r * cos(omega*t);          //(m)
15
16 //Velocity of the body:
17 v = omega * sqrt(r^2 - y^2);      //(m/s)
18 printf("Velocity of the body = %.2f m/s\n",v);     //
     The answers vary due to round off error
19
20 //Acceleration of the body:
21 a = omega^2 * y;                  //(m/s^2)
22 printf("Acceleration of the body = %.2f m/s^2",a);
       //The answers vary due to round off error
```

Scilab code Exa 23.3 To find the amplitude and time period of a particle

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
v1 = 9;          //(m/s)(Velocity)
y1 = 2;          //(m)(Distance from centre)

v2 = 4;          //(m/s)(Velocity)
y2 = 3;          //(m)(Distance from centre)

//Velocity of the particle:
//v = omega * sqrt(r^2 - y^2);

//Substituting the values in this equation:
//9 = omega * sqrt(r^2 - (2)^2)        -(1)
//4 = omega * sqrt(r^2 - (3)^2)        -(2)

//(1)/(2):
r = sqrt(665/65);        //(m)

//Time-period of particle:
//Substituting this value of r in equation (1):
omega = 9/sqrt(r^2 - (2)^2);      //(rad/s)

T = (2 * %pi)/omega;         //(s)
printf("Time-period of the particle = %.2f s",T);
      //The answers vary due to round off error
```

Scilab code Exa 23.4 To find the time required by the particle in passing between

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
r = 5;     //(m)(Amplitude)
```

```
t = 4;     //(s)(Time taken)
y1 = 4;    //(m)(Distance of the point)
y2 = 2;    //(m)(Distance of the point)

//Angular velocity of the particle:
omega = (2 * %pi)/t * (180/%pi);     //(/s)

//Distance: y = r * sind(W * t)
//Substituting the value:
t1 = asind(y1/r)/omega;              //(s)
t2 = asind(y2/r)/omega;              //(s)

//Time required between the two points:
t = t1 - t2;                        //(s)
printf("Time required between the two points = %.2f
   s",t);
```

Scilab code Exa 23.5 To find the frequency amplitude and acceleration of the parti

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
v1 = 12;      //(m/s)(Velocity)
y1 = 0.05;    //(m)(Displacement)

v2 = 3;       //(m/s)(Velocity)
y2 = 0.1;     //(m)(Displacement)

//Amplitude of the motion:
//Let r = Amplitude of the motion,
//omega = Angular displacement of the body.

//Velocity of the body:
//v = omega * sqrt(r^2 - y^2)
```

```
17
18 //Substituting the values in the equation:
19 //12 = omega * sqrt(r^2 - y1^2)    -(1)
20 //3 = omega * sqrt(r^2 - y2^2)     -(2)
21
22 //Dividing (1) by (2):
23 r = sqrt(0.1575/15);          //(m)
24 printf("Amplitude of the motion = %.2f m\n",r);
25
26 //Frequency of the motion:
27 //Substituting the value of r in equation (1):
28 omega = 12/0.09;      //rad/sec)
29
30 //Frequency of motion:
31 N = omega/(2 * %pi);          //(Hz)
32 printf("Frequency of motion = %.2f Hz\n",N);
33
34 //Acceleration when the displacement is 75 mm:
35 y = 0.075;          //(m)
36 a = omega^2 * y;              //(m/s^2)
37 printf("Acceleration when the displacement is 75 mm
   = %.2f m/s^2",a);       //The answers vary due to
   round off error
```

Scilab code Exa 23.6 To calculate the maximum velocity and acceleration of a body

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 r = 0.1;          //(m)(Amplitude)
6 N = 2;            //(vib/sec)(Frequency of body)
7
8 //Angular velocity:
9 omega = (2 * %pi * N);          //(rad/s)
```

```
10
11 //Maximum velocity:
12 Vmax = r * omega;                    //(m/s)
13 printf("Maximum velocity = %.2f m/s\n",Vmax);     //
      The answers vary due to round off error
14
15 //Maximum acceleration:
16 Amax = omega^2 * r;                  //(m/s^2)
17 printf("Maximum acceleration = %.2f m/s^2",Amax);
```

Scilab code Exa 23.7 To find the amplitude maximum acceleration and speed of the p

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 N = 10;       //(No. of oscillation/min)
6 y = 6;        //(cm)(Displacement)
7 //v = 0.6*Vmax
8
9 //No. of oscillations/sec:
10 Nsec = 10/60;
11
12 //Time-period:
13 T = 1/Nsec;          //(s)
14
15 //Angular velocity:
16 omega = (2 * %pi)/T;          //(rad/sec)
17
18 //Linear velocity:
19 //v = W * sqrt(r^2 - y^2)
20 //0.6 * W * r = W * sqrt(r^2 - (8)^2)
21 r = sqrt(64/0.64);       //(cm)
22 printf("Amplitude = %.2f cm\n",r);
23
```

```
24 //Maximum acceleration of the particle:
25 Amax = omega^2 * r;              //(cm^2/s)
26 printf("Maximum acceleration of the particle = %.2f
      cm^2/s\n",Amax);
27
28 //Speed of the particle when it is 6 cm from the
      centre of oscillation:
29 v = omega * sqrt(r^2 - y^2);        //(cm/s)
30 printf("Speed of the particle when it is 6 cm from
      the centre of oscillation = %.2f cm/s",v);
```

Chapter 24

Laws of Motion

Scilab code Exa 24.1 To determine the force

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m = 100;      //(kg)(Mass of body)
6 a = 3.5;      //(m/s^2)(Acceleration)
7
8 //Force:
9 F = m * a;    //(N)
10 printf("Force = %.2f N",F);
```

Scilab code Exa 24.2 To find the weight of a body on earth moon and sun

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m = 50;       //(kg)(Mass of body)
```

```
ge = 9.8;     //(m/s^2)(Acceleration due to gravity
   on earth)
gm = 1.7;     //(m/s^2)(Acceleration due to gravity
   on moon)
gs = 270;     //(m/s^2)(Acceleration due to gravity
   on sun)

//(a) Weight of the body on the Earth:
F1 = m * ge;       //(N)
printf("Weight of the body on the Earth = %.2f N\n",
   F1);

//(b) Weight of the body on the moon:
F2 = m * gm;      //(N)
printf("Weight of the body on the moon = %.2f N\n",
   F2);

//(c) Weight of the body on the sun:
F3 = [m * gs]/1000;     //(N)
printf("Weight of the body on the sun = %.2f kN",F3)
   ;
```

Scilab code Exa 24.3 To determine the velocity of the body

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
m = 7.5;     //(kg)(Mass of body)
u = 1.2;     //(m/s)(Velocity)
F = 15;      //(N)(Force)
t = 2;       //(s)(Time)

//Acceleration of the body:
a = F/m;            //(m/s^2)
```

```
12
13 //Velocity of the body after 2 s:
14 v = u + a * t;      //(m/s)
15 printf("Velocity of the body after 2 s = %.2f m/s",v
   );
```

Scilab code Exa 24.4 To find the velocity of the vehicle

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m = 500;       //(kg)(Mass of vehicle)
6 u = 25;        //(m/s)(Initial velocity)
7 F = 200;       //(N)(Force)
8 t = 120;       //(s)(Time)
9
10 //(1) Velocity of vehicle when the force acts in the
    direction of motion:
11 a = F/m;      //(m/s^2)
12 v1 = u + a * t;        //(m/s)
13 printf("Velocity of vehicle when the force acts in
   the direction of motion = %.2f m/s\n",v1);
14
15 //(2) Velocity of the vehicle when the force acts in
    the opposite direction of motion:
16 a = -0.4;      //(m/s^2)
17 v2 = u + a * t;        //(m/s)
18 printf("Velocity of the vehicle when the force acts
   in the opposite direction of motion = %.2f m/s",
   v2);
```

Scilab code Exa 24.5 To find the time taken by the body to stop

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 F = 50;       //(N)(Retarding force)
6 m = 20;       //(kg)(Mass of the body)
7 u = 15;       //(m/s)(Initial velocity)
8 v = 0;        //(Final velocity)
9
10 //Let t = Time taken by the body to stop.
11
12 //Retardation:
13 a = -F/m;          //(m/s^2)
14
15 //Using: v = u + a * t;
16 t = (v - u)/a;    //(s)
17 printf("Time taken by the body to stop = %.2f s",t);
```

Scilab code Exa 24.6 To find the time taken by the car to increase its velocity

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m = 2.5;        //(tonnes)(Mass of the car)
6 F = 1;          //(kN)(Propelling force)
7 u = 10;         //(m/s)(Initial velocity)
8 v = 15;         //(m/s)(Final velocity)
9
10 //Let t = Time taken by the car to increase its
    speed.
11
12 //Acceleration of the car:
13 a = F/m;        //(m/s^2)
14
```

```
15 //Final velocity of the car:
16 //Using: v = u + a * t;
17 t = (v - u)/a;      //(s)
18 printf("Final velocity of the car = %.2f s",t);
```

Scilab code Exa 24.7 To find the time taken by the train to accelerate

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m = 800;        //(tonnes)(Mass of electric train)
6 Fr = 80;        //(kN)(Resistance to motion)
7 Ft = 200;       //(kN)(Transitive force)
8 v = 25;         //(m/s)(Final velocity)
9 u = 0;          //(m/s)(Initial velocity)
10
11 //Let t = Time taken by the electric train.
12
13 //Net force available to move the train:
14 F = Ft - Fr;      //(kN)
15
16 //Acceleration of the train:
17 a = F/m;          //(m/s^2)
18
19 //Using: v = u + a * t;
20 t = (v - u)/a;      //(s)
21 printf("Time taken by the electric train = %.2f s",t
   );     //The answers vary due to round off error
```

Scilab code Exa 24.8 To find the average resistance of water

```
1 //OS-version - Windows 10
```

```
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 //Considering the motion of the man from the top of
     the tower to the water surface:
6 m = 60;       //(kg)(Mass)
7 s = 20;       //(m)(Height of tower)
8 g = 9.8;      //(m/s^2)(Accn due to Gravity)
9
10 // v = Final velocity of the man when he reaches the
      water surface.
11 //Using: v^2 = u^2 + 2 * g * s;
12 v = sqrt(u^2 + 2 * g * s);
13
14 //Considering the motion of the man from the water
     surface up to the point in water from where he
     started rising.
15 u = v;      //(m/s)(Initial velocity)
16 v = 0;      //(Final velocity)
17 s = 2;      //(m)(Distance covered)
18
19 //Let a = Retardation due to water resistance.
20 //Using: v^2 = u^2 + 2 * a * s;
21 a = u^2/(2 * s);       //(m/s^2)
22
23 //Average resistance of water:
24 F = m * a;             //(N)
25 printf("Average resistance of water = %.2f N",F);
```

Scilab code Exa 24.9 To find the force that will stop the body

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
```

```
5  m = 10;      //(kg)(Mass of the body)
6  u = 20;      //(m/s)(Initial velocity)
7  v = 0;       //(Final velocity)
8  g = 9.8;     //(m/s^2)(Accn due to Gravity)
9
10 //(i) Force which will stop the body in 2 seconds:
11 t = 2;       //(s)(Time)
12 //Let a = Constant retardation.
13 //Using: v = u - a*t;
14 a1 = u/t;    //(m/s^2)
15
16 //Since body is falling under gravity:
17 a = g + a1;    //(m/s^2)
18
19 //Force applied to stop the body:
20 F = m * a;     //(N)
21 printf("Force applied to stop the body = %.2f N\n",F
      );
22
23 //(ii)Force which will stop the body in 2 metres:
24 //Using: v^2 = u^2 - 2 * a2 * s;
25 s = 2;         //(m)
26 a2 = u^2/(2 * s);              //(m/s^2)
27
28 //Since body is falling under gravity:
29 a = g + a2;    //(m/s^2)
30
31 //Force required to stop the body:
32 F = m * a;     //(N)
33 printf("Force required to stop the body = %.2f N",F)
      ;
```

Scilab code Exa 24.10 To find the magnitude of force responsible for motion

```
1  //OS-version - Windows 10
```

```
//Scilab-version - 6.0.2
clc
clear all
t = poly(0,"t");
//Given:
s = 5*t + 2*t^2;

m = 10;        //(kg)(Mass)

//Differentiating both sides w.r.t. t:
v = derivat(s);

//Again differentiating both sides w.r.t. t:
a = derivat(v);
a = coeff(a)
//Force:
F = m * a;
mprintf("Force responsible for the motion = %.2f N",
   F);
```

Scilab code Exa 24.11 To find the force exerted by the body on the lift floor

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
m = 50;      //(kg)(Mass of the body)
a = 1.2;     //(m/s^2)(Acceleration)
g = 9.8;     //(m/s^2)(Accn due to Gravity)

//Pressure exerted by the body on the floor:
F = m * (g + a);      //(N)
printf("Pressure exerted by the body on the floor =
   %.2f N",F);
```

Scilab code Exa 24.12 To find the force exerted by the body on the lift floor

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m = 100;      //(kg)(Mass of the body)
6 a = -0.8;     //(m/s^2)(Acceleration)
7 g = 9.8;      //(m/s^2)(Accn due to Gravity)
8
9 //(a) When the lift is moving upwards:
10 F1 = m * (g + a);       //(N)
11 printf("Force exerted by the body on the floor of
    the lift when the lift is moving upwards = %.2f N
    \n",F1);
12
13 //(b) When the lift is moving downwards:
14 F2 = m * (g - a);      //(N)
15 printf("Force exerted by the body on the floor of
    the lift when the lift is moving downwards = %.2f
     N",F2);
```

Scilab code Exa 24.13 To find the acceleration of the elevator

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m = 65;       //(kg)(Mass of the body)
6 F = 800;      //(N)(Force)
7 g = 9.8;      //(m/s^2)(Accn due to Gravity)
8
```

```
9 //Let a = Acceleration of the elevator.
10 //F = m * (g + a)
11
12 a = F/m - g;     //(m/s^2)
13 printf("Acceleration of the elevator = %.2f m/s^2",a
   );
```

Scilab code Exa 24.14 To find the total tension in the cables

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m1 = 500;      //(kg)(Mass of the elevator)
6 a = 3;         //(m/s^2)(Acceleration)
7 m2 = 70;       //(kg)(Mass of operator)
8 g = 9.8;       //(m/s^2)(Accn due to Gravity)
9
10 //Scale reading:
11 R1 = m2 * (g + a);                  //(N)
12 printf("Scale reading when the elevator is ascending
    = %.2f N\n",R1);
13
14 //Total tension in the cable of the elevator:
15 R2 = (m1 + m2)*(g + a);             //(N)
16 printf("Total tension in the cable = %.2f N",R2);
```

Scilab code Exa 24.15 To find the cable tension

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
```

```
5  m = 2500;       //(kg)(Mass of elevator)
6  u = 0;          //(Initial velocity)
7  s = 35;         //(m)(Distance travelled)
8  t = 10;         //(s)(Time)
9  g = 9.8;        //(m/s^2)(Accn due to Gravity)
10
11 //Cable tension:
12 //Let a = Constant acceleration of the elevator.
13 //Using s = u*t + (1/2)*a*t^2;
14 a = s/((1/2)*t^2);       //(m/s^2)
15
16 //Tension in the cable when the elevator is moving
      vertically downwards:
17 R = m * (g - a);         //(kN)
18 printf("Tension in the cable when the elevator is
      moving vertically downwards = %.2f kN\n",R/1000);
19
20 //Limits of cable tension:
21 //Cable tension when the elevator is moving
      vertically downwards with zero acceleration:
22 a = 0;          //(Acceleration)
23 R = m*(g - a);          //(kN)
24 printf("Cable tension when the elevator is moving
      vertically downwards with zero acceleration = %.2
      f kN\n",R/1000);
25
26 //Cable tension when the acceleration is maximum(i.e
      . 9.8 m/s^2):
27 a = 9.8;          //(m/s^2)
28 R = m * (g - a);       //(kN)
29 printf("Cable tension when the acceleration is
      maximum(i.e. 9.8 m/s^2) = %.2f",R/1000);
```

Scilab code Exa 24.16 To calculate the force transmitted by the man

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  m1 = 500;          //(kg)(Gross mass of elevator)
6  g = 9.8;           //(m/s^2)(Accn due to Gravity)
7
8  //Pull in the cable, during accelerated motion:
9  //Considering motion with constant acceleration:
10 u = 0;             //(Initial velocity)
11 v = 2;             //(m/s)(Final velocity)
12 s = 3;             //(m)(Distance travelled)
13
14 //Let a1 = Constant acceleration.
15 //Using: v^2 = u^2 + 2 * a1 * s;
16 a1 = v^2/(2*s);         //(m/s^2)
17
18 //Force required to produce this acceleration:
19 F1 = m1 * a1;           //(N)
20
21 //Total pull in the cable:
22 R = (m1 * g) + F1;          //(N)
23 printf("Total pull in the cable = %.2f N\n",R);
      //The answers vary due to round off error
24
25 //Force transmitted by the man during the
      decelerating motion:
26 u = 2;           //(m/s)(Initial velocity)
27 v = 0;           //(Final velocity)
28 t = 2;           //(s)(Time)
29 m2 = 75;         //(kg)(Mass of man)
30
31 //Let a2 = Constant deceleration
32
33 //Using: v = u + a2*t;
34
35 a2 = u/t;              //(m/s^2)
36
```

```
37 //Force transmitted by the man during deceleration
      motion:
38 R = m2 * (g - a2);              //(N)
39 printf("Force transmitted by the man during
      deceleration motion = %.2f N",R);
```

Scilab code Exa 24.17 To determine the acceleration of the two bodies and the tens

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  m1 = 80;        //(kg)(Mass of body A)
6  m2 = 20;        //(kg)(Mass of body B)
7  P = 400;        //(N)(Force applied on first body)
8  mu = 0.3;        //(Coefficient of friction)
9  g = 9.8;         //(Acceleration due to gravity)
10 R1 = m1*g;       //(N)
11 F1 = mu*R1;      //(N)
12 //Acceleration of the two bodies:
13 //Let a = Acceleration of the bodies,
14 //T = Tension in the thread.
15
16 //Resultant horizontal force on body A:
17 //P1 = 164.8 - T(towards left)
18
19 //Force causing acceleration to the body A:
20 //F = m1 * a = 80 * a;
21 //According to D' Alembert's principle:
22 //T = 164.8 - 80*a          -(1)
23
24 //Resultant horizontal force on body B:
25 //P2 = T - 58.8
26
27 R2 = m2*g;          //(N)
```

```
28 F2 = mu*R2;          //(N)
29 //Force causing acceleration to the body B:
30 //F = m2 * a = 20 * a;
31 //According to D' Alembert's principle:
32 //T = 58.8 + 20*a          -(2)
33
34 //Equating (1) and (2):
35 a = ((P - F1)-(F2))/(m1 + m2);          //(m/s^2)
36 printf("Acceleration of the bodies = %.2f m/s^2\n",a
      );
37
38 //Tension in the thread:
39 //Substituting a in equation (ii):
40 T = F2 + (m2 * a);          //(N)
41 printf("Tension in the thread = %.2f N",T);
```

Scilab code Exa 24.18 To find the velocity with which the machine gun will recoil

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 M = 25;          //(kg)(Mass of the machine gun)
6 m = 0.03;        //(kg)(Mass of the bullet)
7 v = 250;         //(Velocity of firing)
8
9 //Let V = Velocity with which the machine gun will
      recoil.
10 //MV = mv          //(Conservation of Energy)
11 V = (m * v)/M;     //(m/s)
12 printf("Velocity with which the machine gun will
      recoil = %.2f m/s",V);
```

Scilab code Exa 24.19 To find the recoil velocity of the gun distance it takes to

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m = 0.02;          //(kg)Mass of the bullet)
6 v = 300;           //(m/s)(Velocity of the bullet)
7 M = 100;           //(kg)(Mass of the carriage with
     gun)
8 F = 20;            //(N)(Resistance to sliding)
9
10 //(a) Velocity, with which the gun will recoil:
11 //Let V = Velocity with which the gun will recoil.
12
13 //M*V = m*v          (Conservation of Energy)
14 V = (m * v)/M;          //(m/s)
15 printf("Velocity with which the gun will recoil = %
     .2f m/s\n",V);
16
17 //(b) Distance, in which the gun comes to rest:
18 u = 0.06;      //(m/s)(Initial velocity)
19 v = 0;         //(Final velocity)
20
21 //a = Retardation of the gun,
22 //s = Distance in which the gun comes to rest.
23
24 a = F/M;      //(m/s^2)
25
26 //v^2 = u^2 - 2*a*s
27 s = u^2/(2 * a);       //(m)
28 printf("Distance in which the gun comes to rest = %
     .2f mm\n",s*1000);
29
30 //(c)Time taken by the gun in coming to rest:
31 //Using: v = u + a*t;
32 t = u/a;             //(s)
33 printf("Time taken by the gun in coming to rest = %
```

```
.2f s",t);
```

Scilab code Exa 24.20 To find the final velocity of the boat

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  v = 4.2;    //(m/s)
6  w = 80;     //(kg-wt)(Weight of each man)
7  W = 400;    //(kg-wt)(Weight of the boat)
8
9  //Let V = Velocity of the boat after the second man
       dives off the boat.
10
11 //Final momentum = 400 * V        (kg-m/s)    -(1)
12
13 //Momentum given by the first man to the boat:
14 M1 = w * v;         //(kg-m/s)
15
16 //Momentum given by the second man to the boat:
17 M2 = w * v;         //(kg-m/s)
18
19 //Final momentum:
20 M = M1 + M2;        //(kg-m/s)                  -(2)
21
22 //Equating (1) and (2):
23 V = M/W;            //(m/s)
24 printf("Velocity of the boat after the second man
       dives off the boat = %.2f m/s",V);
```

Scilab code Exa 24.21 To find the speed of the vehicle

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  m = 2000;          //(kg)(Mass of vehicle)
6  Fr = 50 * 2;       //(N)
7  u = 10;            //(m/s)(Initial velocity of the
       vehicle)
8  alpha = asind(1/80);     //(Degrees)(Angle of
       inclination)
9  t = 100;           //(s)(Time)
10 g = 9.8;           //(m/s^2)(Accn due to Gravity)
11
12 //Force due to inclination:
13 Fi = m * g * sind(alpha);       //(N)
14
15 //Net force available to move the vehicle:
16 F = Fi - Fr;           //(N)
17
18 a = F/m;      //(m/s^2)
19
20 //Speed of the train after running down the incline:
21 v = (u + a * t) * (3600/1000);      //(km.p.h.)
22
23 printf("Speed of the train after running down the
       incline = %.2f km.p.h.",v);
```

Scilab code Exa 24.22 To find the distance along the incline the body must travel

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  m = 200;          //(kg)(Mass of the body)
6  u = 0;            //(Initial velocity)
```

```
alpha = 15;     //(Degrees)(Inclination of the plane)
v = 10;         //(m/s)(Final velocity)
mu = 0.1;        //(Coefficient of friction)

//Let s = Distance through which the body will slide
    .

//Force responsible for sliding down the body:
Fs = m * g * sind(alpha);       //(N)

//Normal reaction:
R = m * g * cosd(alpha);        //(N)

//Force of friction:
Fr = mu * R;                     //(N)

//Net force available to move the body:
F = Fs - Fr;                    //(N)


a = F/m;                        //(m/s^2)

//Using: v^2 = u^2 + 2 * a * s;
s = v^2/(2 * a);                //(m)
printf("Distance through which the body will slide =
    %.2f m",s);
```

Scilab code Exa 24.23 To find the distance through which the wagon will travel

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
alpha = asind(0.05);      //(Degrees)
u = 10;                   //(m/s)(Initial velocity)
```

```
m = 50;                    //(tonnes)(Mass of the
    detached wagon)
v = 0;                     //(m/s)(Final velocity)
Tr = 5;                    //(kN)(Track resistance)
g = 9.8;                   //(m/s^2)(Accn due to
    Gravity)
//Let s = Distance through which the wagon will
    travel before coming to rest,
//a = Retardation of the rain.

//Resistance to the train due to upgrade:
Fr = m*g*sind(alpha);          //(N)

//Total resistance to the movement of the train:
F = Fr + Tr;           //(kN)

a = F/m;               //(m/s^2)

//Using: v^2 = u^2 + 2*a*s;
s = u^2/(2*a);         //(m)
printf("Distance through which the wagon will travel
    before coming to rest = %.2f m",s);
```

Scilab code Exa 24.24 To find the factor of safety against slipping

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
alpha = 10;            //(Degrees)
a = 1;                 //(m/s^2)(Acceleration)
m = 500;               //(kg)(Mass of the body placed
    on the truck)
mu = 0.4;               //(Coefficient of friction)

```

```
10 //Stability of the load:
11
12 //Force caused due to deceleration:
13 P1 = m * a;           //(N)
14
15 //Component of the load along the plane:
16 P2 = m*g*sind(alpha);          //(N)
17
18 //Total force, which will cause slipping:
19 P = P1 + P2;          //(N)
20
21 //Normal reaction:
22 R = m*g*cosd(alpha);       //(N)
23
24 //Force of friction:
25 F = mu*R;                   //(N)
26
27 printf("Since the force of friction %.2f is more
      than the force which will cause slipping %.2f ,
      therefore the load will not slip.\n",F,P);
28
29 //Factor of safety against slipping for this load:
30 Factor = F/P;
31 printf("Factor of safety against slipping for this
      load = %.2f",Factor);
```

Chapter 25

Motion of Connected Bodies

Scilab code Exa 25.1 To find the acceleration with which the mass comes down and t

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m1 = 45;          //(kg)(Mass of first body)
6 m2 = 30;          //(kg)(Mass of second body)
7 g = 9.8;          //(m/s^2)(Accn due to Gravity)
8
9 //Acceleration of the heavier mass:
10 a = g * [(m1 - m2)/(m1 + m2)];       //(m/s^2)
11 printf("Acceleration of the heavier mass = %.2f m/s
     ^2\n",a);
12
13 //Tension in the string:
14 T = [(2 * m1 * m2)/(m1 + m2)] * g;     //(N)
15 printf("Tension in the string = %.2f N",T);
```

Scilab code Exa 25.2 To determine the tensions in the string and accelerations of

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  m1 = 150;       //(kg)(Mass of the first block)
6  m2 = 50;        //(kg)(Mass of second block)
7  g = 9.8;        //(m/s^2)(Accn due to Gravity)
8
9  //Let a = Acceleration of the block,
10 //T = Tension in the string.
11
12 //Considering the motion of the 150 kg block:
13
14 //Resultant force = 150 * g - 2*T (downwards)
       -(1)
15
16 //Force acting on this block = 150 * a;          -(2)
17
18 //Equating (1) and (2):
19 //150 * g - 2 * T = 150 * a                      -(3)
20
21 //Considering the motion of 50 kg block:
22 //Resultant force = T - 50*g (upwards)           -(4)
23
24 //Force acting on this block = 50 * 2*a          -(5)
25 //Equating (4) and (5):
26 //T - 50*g = 100*a                               -(6)
27
28 //Solving (3) and (6):
29 a = (m2 * g)/(4*m2 + m1);                  //(m/s^2)
30 printf("Acceleration of block A = %.2f m/s^2\n",a);
31 printf("Acceleration of block B = %.2f m/s^2\n",2*a)
       ;
32
33 //Tension in the strings:
34 //Substituting the value of a in equation (3):
35 T = [(m1 * g) - (m1 * a)]/2;          //(N)
36 printf("Tension in the strings = %.2f N",T);
```

Scilab code Exa 25.3 To find the accelerations of the three masses

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m1 = 15;          //(kg)(First mass)
6 m2 = 6;           //(kg)(Second mass)
7 m3 = 4;           //(kg)(Third mass)
8 g = 9.8;          //(m/s^2)(Accn due to Gravity)
9
10 //Let a15 = Acceleration of 15 kg mass,
11 //a6 = Acceleration of 6 kg mass,
12 //a4 = Acceleration of 4 kg mass.
13
14 a15 = g * [(m1 - (m2 + m3))/(m1 + m2 + m3)];
      //(m/s^2)
15 a6 = g * [(m2 - m3)/(m2 + m3)];          //(m/s^2)
16 a4 = a6;                                 //(m/s^2)
17
18 printf("Acceleration of 15 kg mass = %.2f m/s^2\n",
      a15);
19 printf("Acceleration of 6 kg mass = %.2f m/s^2\n",0)
      ;
20 printf("Acceleration of 4 kg mass = %.2f m/s^2",a4 +
       a6);
```

Scilab code Exa 25.4 To find the acceleration of the body

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
```

```
clc
clear all
m2 = 10;          //(kg)(Mass of body A)
m1 = 5;           //(kg)(Mass of body B)
g = 9.8;          //(m/s^2)(Accn due to Gravity)

//Acceleration of body A:
a = m1/(m1 + m2) * g;          //(m/s^2)
printf("Acceleration of body A = %.2f m/s^2\n",a);

//Tension in the string:
T = (m1 * m2)/(m1 + m2) * g;     //(N)
printf("Tension in the string = %.2f N",T);     //The
    answers vary due to round off error
```

Scilab code Exa 25.5 To find the velocity acquired by the body B

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
m2 = 20;          //(kg)(Mass of block A)
m1 = 10;          //(kg)(Mass of block B)
mu = 0.25;        //(Coefficient of friction)
u = 0;            //(Initial velocity)
s = 1;            //(m)(Vertical distance)
g = 9.8;          //(m/s^2)(Accn due to Gravity)

//Let v = Final velocity of the block A.

//Acceleration of the block A:
a = g * [(m1 - mu*m2)/(m1 + m2)];          //(m/s^2)

//Using: v^2 = u^2 + 2 * a * s;
v = sqrt(u^2 + 2 * a * s);                 //(m/s)
```

```
19 printf("Final velocity of the block A = %.2f m/s",v)
   ;
```

Scilab code Exa 25.6 To determine the coefficient of friction between the block an

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  m2 = 10;          //(kg)(Mass of block A)
6  m1 = 5;           //(kg)(Mass of block B)
7  g = 9.8;          //(m/s^2)(Accn due to Gravity)
8
9  //Acceleration of block A:
10 acc_A = g/9;        //(m/s^2)
11
12 //Let T = Tension in the string in N,
13 //mu = Coefficient of friction between block and
      table.
14
15 //Normal reaction on the horizontal surface by the
      body of mass 10 kg:
16 R = m2*g;       //(N)
17
18 //Frictional force:
19 //F = mu * R;           //(m/s^2)
20
21 //Considering the motion of the block A:
22 //Resultant force = T - 10*mu*g;              -(1)
23 //Force acting on the block = 10 * acc_A;      -(2)
24
25 //Equating (1) and (2):
26 //T - 10*mu*g = 10/9 * g;
27
28 //Multiplying both sides by 2;
```

```
29 //2*T - 20*mu*g = 20/9 * g;                    -(3)
30
31 //Considering the motion of the block B:
32 //Resultant force = 5*g - 2*T                  -(4)
33 a = g/18;                  //(m/s^2)
34 //Force acting on it = ma;                     -(5)
35
36 //Equating (4) and (5):
37 //5*g - 2*T = 5/18 * g;                       -(6)
38
39 //Adding equations (3) and (6):
40 m = (m1*a + (2*m2*acc_A))/g
41 acc = m1 - m;      //(After solving 3 and 6)
42 mu = (acc)/(2*m2);
43 printf("Coefficient of friction between block and
      table = %.2f",mu);     //The answers vary due to
      round off error
```

Scilab code Exa 25.7 To find the acceleration with which the body will come down

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m2 = 30;     //(kg)(Mass of the body lying on smooth
      plane)
6 alpha = 15;      //(Degrees)(Inclination of the plane
      with horizontal)
7 m1 = 20;     //(kg)(Mass of the body which hangs
      freely beyond the pulley)
8
9 //Acceleration with which the body will come down:
10 a = g * [m1 - m2*sind(alpha)]/(m1 + m2);       //(m/s
      ^2)
11 printf("Acceleration with which the body will come
```

```
down = %.2f m/s^2",a);
```

Scilab code Exa 25.8 To find the tension in the rope acceleration and the distance

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  m2 = 150;       //(kg)(Mass of the body)
6  alpha = 10;     //(Degrees)(Inclination of the plane)
7  m1 = 80;        //(kg)(Mass of the man)
8  mu = 0.2;        //(Coefficient of friction)
9  g = 9.8;        //(m/s^2)(Accn due to Gravity)
10
11 //(i)Tension in the rope:
12 T = [(m1*m2*g)*(1 + sind(alpha) + mu*cosd(alpha))]/(
       m1 + m2);          //(N)
13 printf("Tension in the rope = %.2f N\n",T);     //The
       answers vary due to round off error
14
15 //(ii)Acceleration, with which the body moves up the
       plane:
16 a = [g * (m1 - m2*sind(alpha) - mu*m2*cosd(alpha))
       ]/(m1 + m2);          //(m/s^2)
17 printf("Acceleration, with which the body moves up
       the plane = %.2f m/s^2\n",a);
18
19 //(iii)Distance moved by the body in 4 sec starting
       from rest:
20 u = 0;            //(Initial velocity)
21 t = 4;            //(s)(Time)
22 a = 1.04;         //(m/s^2)(Acceleration)
23
24 //Distance moved by the body:
25 s = u*t + (1/2)*a*t^2;          //(m)
```

```
26 printf("Distance moved by the body in 4 sec starting
       from rest = %.2f m",s);
```

Scilab code Exa 25.9 To determine the velocity of the body A

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  m1 = 15;          //(kg)(Mass of body B)
6  m2 = 10;          //(kg)(Mass of body A)
7  alpha = 30;       //(Degrees)(Inclination of plane)
8  mu = 0.2;          //(Coefficient of friction)
9  g = 9.8;          //(m/s^2)(Accn due to Gravity)
10
11 //Let T = Tension in the string,
12 //a = Acceleration of the block A.
13
14 //Normal reaction:
15 R = m2*g*cosd(alpha);          //(N)
16
17 //Frictional force:
18 Fr = mu*R;                      //(N)
19
20 //Considering the motion of the block A:
21 //Resultant force = T - 68.3;              -(1)
22
23 //Force acting on it = 10*a;               -(2)
24
25 //Equating (1) and (2):
26 //T - 68.3 = 10*a
27 //2*T - 136.6 = 20*a                       -(3)
28
29 //Considering the motion of the body B:
30 //Resultant force = 147 - 2*T              -(4)
```

```
31
32 //Force acting on it = 7.5*a;                    -(5)
33
34 //Equating equations (4) and (5):
35 //147 - 2*T = 7.5*a                              -(6)
36
37 a = 10.4/27.5;                                  //(m/s^2)
38
39 //Velocity of the body A after 10 seconds, if the
      system starts from rest:
40 u = 0;          //(Initial velocity)
41 t = 10;         //(s)(Time)
42
43 v = u + a*t;          //(m/s)
44 printf("Velocity of the body A after 10 seconds, if
      the system starts from rest = %.2f m/s",v);     //
      The answers vary due to round off error
```

Scilab code Exa 25.10 To determine the acceleration of the body B and the tension

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 W1 = 500;        //(N)(Weight of body A)
6 W2 = 750;        //(N)(Weight of body B)
7 u = 0.2;         //(Coefficient of friction)
8
9 //Acceleration of the body B:
10 //Let T = Tension in the strings,
11 //a = Acceleration of the body B.
12
13 //From slope:
14 alpha = atand(3/4);
15
```

```
16 //Normal reaction:
17 R = W2 * cosd(alpha);          //(N)
18
19 //Frictional force:
20 Fr = u * R;                    //(N)
21
22 //Considering the motion of the body B:
23 //Resultant force = 2*T - 570;        (N)     -(1)
24
25 //Force acting on it = m2*g -W2/g * a = 76.5*a
      -(2)
26
27 //Equating (1) and (2):
28 //2*T - 570 = 76.5*a                          -(3)
29
30 //Considering the motion of the body A:
31 //Resultant force = 500 - T                   -(4)
32
33 //Force acting on it = m1 * 2*a = 102*a       -(5)
34
35 //Equating (4) and (5):
36 //500 - T = 102*a                             -(6)
37
38 //Solving (3) and (6):
39 a = 430/280.5;              //(m/s^2)
40
41 //Tension in the string supporting body A:
42 T = (1000 - 204*a)/2;          //(N)
43 printf("Tension in the string supporting body A = %
      .2f N",T);     //The answers vary due to round off
       error
```

Scilab code Exa 25.11 To find the tension in the string

```
1 //OS-version - Windows 10
```

```
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 alpha1 = 30;      //(Degrees)(Inclination of first
     plane)
6 alpha2 = 20;      //(Degrees)(Inclination of second
     plane)
7 m1 = 10;          //(kg)(Mass of first body)
8 m2 = 6;           //(kg)(Mass of second body)
9 g = 9.8;          //(m/s^2)(Accn due to Gravity)
10
11 //Tension in the string:
12 T = [(m1*m2*g)*(sind(alpha1) + sind(alpha2))]/(m1 +
     m2);
13 printf("Tension in the string = %.2f N",T);     //THe
     answers vary due to round off error
```

Scilab code Exa 25.12 To find the distance moved by the bodies

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m1 = 15;        //(kg)(Mass of body B)
6 m2 = 25;        //(kg)(Mass of body A)
7 alpha2 = 20;       //(Degrees)(Inclination of plane
     PQ with horizontal)
8 alpha1 = 60;       //(Degrees)(Inclination of plane
     QR with horizontal)
9 u = 0;             //(Initial velocity)
10 t = 3;            //(s)(Time)
11
12 //Acceleration of the body B:
13 //Let a = Acceleration of system,
14 //T = Tension in the cord.
```

```
15
16 //Considering the motion of the body B:
17 //Resultant force = 127.3 - T;        (N)     -(1)
18
19 //Force acting on it = 15*a;                  -(2)
20
21 //Equating (1) and (2):
22 //127.3 - T = 15*a                           -(3)
23
24 //Considering the motion of the body A:
25 //Resultant force = 83.8 + T                 -(4)
26
27 //Force acting on it = m2*a = 25*a           -(5)
28
29 //Equating (4) and (5):
30 //83.8 + T = 25*a                          -(6)
31
32 //Solving (3) and (6):
33 a = 211.1/40;              //(m/s^2)
34
35 //Distance moved by the body in 3 seconds:
36 s = u*t + (1/2)*a*t^2;          //(m)
37 printf("Distance moved by the body in 3 seconds = %
   .2f m",s);     //The answers vary due to round off
    error
```

Scilab code Exa 25.13 To find the fundamentals of resulting accelerations

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 alpha1 = 30;     //(Degrees)(Inclination of first
   plane)
6 alpha2 = 15;     //(Degrees)(Inclination of second
```

```
    plane)
m1 = 15;          //(kg)(Mass of first body)
m2 = 5;           //(kg)(Mass of second body)
mu = 0.3;          //(Coefficient of friction)
g = 9.8;          //(m/s^2)(Accn due to Gravity)

//Let a = Acceleration of the system,
//T = Tension in the string.

//Normal reaction on the plane AB:
R1 = m1*g*cosd(alpha1);              //(N)

//Frictional force:
Fr1 = mu*R1;                          //(N)

//Normal reaction on the plane BC:
R2 = m2*g*cosd(alpha2);             //(N)

//Frictional force:
Fr2 = mu*R2;                          //(N)

//Considering the motion of body 1:
//Resultant force = 35.3 - T            -(1)

//Force acting on it = 15*a             -(2)

//Equating (1) and (2):
//35.3 - T = 15*a                      -(3)

//Considering the motion of body 2:
//Resultant force = T - 27              -(4)

//Force acting on this body = 5*a      -(5)

//Equating (4) and (5):
//T - 27 = 5*a                        -(6)

//Solving (3) and (6):
```

```
44 a = ((m1*g*sind(alpha1) - Fr1) - (m2*g*sind(alpha2)
      + Fr2))/(m1 + m2);          //(m/s^2)
45 printf("Acceleration of the system = %.2f m/s^2",a);
```

Scilab code Exa 25.14 To find the acceleration of the masses and tension in the tw

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  mA = 4;          //(kg)(Mass of body A)
6  mB = 5;          //(kg)(Mass of body B)
7  mC = 15;         //(kg)(Mass of body C)
8  mu = 0.4;         //(Coefficient of friction)
9  g = 9.8;         //(m/s^2)(Accn due to Gravity)
10 alpha = 30;      //(Degrees)
11 //(i)Acceleration of the bodies:
12 //Let a = Acceleration of the bodies.
13 //TA = Tension in the string connected with body A.
14 //TB = Tension in the string connected with body B.
15
16 //Normal reaction on the horizontal surface due to
      body A:
17 RA = mA * g;          //(N)
18
19 //Frictional Force:
20 FrA = mu*RA;          //(N)
21
22 //Normal reaction on the inclined surface due to
      body B:
23 RB = mB * g * cosd(alpha);          //(N)
24
25 //Frictional Force:
26 FrB = mu*RB;                        //(N)
27
```

```
28 //Considering the motion of body A:
29 //Resultant force = TA - 15.68          -(1)
30
31 //Force acting on it = 4*a              -(2)
32
33 //Equating (1) and (2):
34 //TA = 4*a + 15.68                      -(3)
35
36 //Considering the motion of the body B:
37 //Resultant force = TB + 7.53           -(4)
38
39 //Force acting on it = 5*a              -(5)
40
41 //equating (4) and (5):
42 //TB = 5*a - 7.53                       -(6)
43
44 //Considering the motion of the body C:
45 //Resultant force = 147 - (TA + TB)     -(7)
46
47 //Force acting on it = 15*a             -(8)
48
49 //Equating (7) and (8):
50 //147 - (TA + TB) = 15*a
51 //Substituting the values of TA and TB   ;
52 a = 138.85/24;                 //(m/s^2)
53 printf("Acceleration of the bodies = %.2f m/s^2\n",a
      );     //The amswers vary due to round off error
54
55 //(ii)Tension in the two strings:
56 //Substituting the value of a in equation (3):
57 TA = 4*a + 15.68;              //(N)
58 printf("Tension in the string connected with body A
      = %.2f N\n",TA);      //The answers vary due to
      round off error
59
60 //Substituting the value of a in equation (6):
61 TB = 5*a - 7.53;               //(N)
62 printf("Tension in the string connected with body B
```

```
= %.2f N",TB);      //The answers vary due to round
  off error
```

Chapter 26

Helical Springs and Pendulums

Scilab code Exa 26.1 To determine the stiffness of the spring

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m = 4;          //(kg)(Mass)
6 n = 2;          //(Hz)(Frequency)
7 //Let s = Stiffness of the spring.
8
9 //Periodic time:
10 t = 1/n;          //(s)
11
12 //Also, periodic time:
13 //t = 2*%pi * sqrt(m/s);
14 s = ((2*%pi)^2 * m)/t^2;          //(N/m)
15 printf("Stiffness of the spring = %.2f N/m\n",s);
```

Scilab code Exa 26.2 To find the period of oscillation

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
M = 7;              //(kg)(Mass)
m = 0.5;            //(kg)(Mass)
delta = 0.01;          //(m)(Deflection when mass is m)
g = 9.8;            //(m/s^2)(Accn due to Gravity)

//Deflection of spring when mass = 7 kg;
Delta = (delta/m) * M;       //(m)

//Period of oscillation:
t = (2*%pi) * sqrt(Delta/g);    //(s)
printf("Period of oscillation = %.2f s",t);
```

Scilab code Exa 26.3 To determine the natural frequency of oscillation

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
delta = 0.0005;         //(m)(Deflection)
g = 9.8;            //(m/s^2)(Accn due to Gravity)

//Natural frequency of oscillation:
n = (1/(2*%pi)) * sqrt(g/delta);              //(Hz)
printf("Natural frequency of oscillation = %.2f Hz",
   n);     //The answers vary due to round off error
```

Scilab code Exa 26.4 To determine the number of oscillations of the body

```
//OS-version - Windows 10
```

```
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m = 3;          //(kg)(Mass of the body)
6 delta = 0.012;     //(m)(Deflection)
7 x = 0.025;     //(m)(Displacement)
8 g = 9.8;       //(m/s^2)(Accn due to Gravity)
9
10 //No. of oscillations of the body:
11 n = (1/(2*%pi)) * sqrt(g/delta);     //(Hz)
12 printf("No. of oscillations of the body = %.2f Hz\n"
      ,n);
13
14 //Maximum force in the spring:
15
16 //Angular velocity of the body:
17 omega = sqrt(g/delta);           //(rad/s)
18
19 //Maximum acceleration:
20 amax = omega^2 * x;          //(m/s^2)
21
22 //Maximum inertia force:
23 MIF = m * amax;         //(N)
24
25 //Maximum force in the spring:
26 MF = (m * g) + MIF;     //(N)
27 printf("Maximum force in the spring = %.2f N",MF);
```

Scilab code Exa 26.5 To calculate the natural period of vibrations and to calculat

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 w = 240;      //(kN)(Weight of empty wagon)
```

```
6 W = 320;      //(kN)(Load)
7 delta = 0.08;     //(m)(Deflection)
8 g = 9.8;      //(m/s^2)(Accn due to Gravity)
9
10 //(i) Period of vibrations when the wagon is empty:
11 delta_1 = (delta/W) * w;       //(m)
12
13 //Period of vibrations:
14 t1 = (2*%pi) * sqrt(delta_1/g);       //(s)
15 printf("Period of vibrations when the wagon is empty
    = %.2f s\n",t1);
16
17 //(ii) Period of vibrations when the wagon is loaded
   :
18 //Total load on the springs:
19 TL = w + W;      //(kN)
20
21 //Deflection of the spring when the wagon is loaded:
22 delta_2 = (delta/W) * TL;      //(m)
23
24 //Period of vibrations:
25 t2 = (2*%pi) * sqrt(delta_2/g);       //(s)
26 printf("Period of vibrations when the wagon is
   loaded = %.2f s\n",t2);
27
28 //(b) Velocity of the railway wagon when it is empty
   :
29 r = 0.1;     //(m)(Amplitude)
30 y = 0.04;    //(m)(Displacement)
31
32 //Angular velocity of the wagon:
33 omega = (2*%pi)/t1;          //(rad/s)
34
35 //Velocity:
36 v = omega * sqrt(r^2 - y^2);        //(m/s)
37 printf("Velocity of the railway wagon when it is
   empty = %.2f m/s",v);
```

Scilab code Exa 26.6 To determine the period of vibrations maximum velocity and ac

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m = 0.05;     //(tonnes)(Mass)
6 s1 = 4;        //(kN/m)(Stiffness of first spring)
7 s2 = 6;        //(kN/m)(Stiffness of second spring)
8 r = 0.04;      //(m)(Displacement)
9 g = 9.8;       //(m/s^2)(Accn due to Gravity)
10
11 //(a) When the springs are connected in series:
12 printf("When the springs are connected in series:\n")
   ;
13 //Spring constant of an equivalent spring:
14 s = (s1*s2)/(s1 + s2);      //(kN/m)
15
16 //Deflection of the spring:
17 delta = (m * g)/s;     //(m)
18
19 //Period of vibrations:
20 t = (2*%pi) * sqrt(delta/g);      //(s)
21 printf("Period of vibrations = %.2f s\n",t);     //
   The answers vary due to round off error
22
23 //Angular velocity of the block:
24 omega = (2*%pi)/t;        //(rad/s)
25
26 //Maximum velocity:
27 vmax = omega*r;          //(m/s)
28 printf("Maximum velocity = %.2f m/s\n",vmax);
29
30 //Maximum acceleration:
```

```
amax = omega^2 * r;      //(m/s^2)
printf("Maximum acceleration = %.2f m/s^2\n\n",amax)
   ;

//(b) When the springs are connected in parallel:
printf("When the springs are connected in parallel:\
   n");
//Spring constant equivalent:
s = s1 + s2;       //(kN/m)

//Deflection of the spring due to block of weight
   0.49 kN:
delta = (m * g)/s;                //(m)

//Period of vibrations:
t = (2*%pi) * sqrt(delta/g);     //(s)
printf("Period of vibrations = %.2f s\n",t);

//Angular velocity of the block:
omega = (2*%pi)/t;                //(rad/s)

//Maximum velocity:
vmax = omega * r;          //(m/s)
printf("Maximum velocity = %.2f m/s\n",vmax);

//Maximum acceleration:
amax = omega^2 * r;      //(m/s^2)
printf("Maximum acceleration = %.2f m/s^2",amax);
      //The answers vary due to round off error
```

Scilab code Exa 26.8 To find the length of the pendulum maximum acceleration of th

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
```

```
clear all
theta = (4*%pi)/180;          //(rad)(Angular amplitude
    )
n = 24;                       //(Hz)(Frequency)
t = 60;                       //(s)(Time)
g = 9.8;                      //(m/s^2)(Accn due to
    Gravity)

//(a) Length of the pendulum:
//Let l = length of the pendulum.

//Time period for one oscillation:
T = 60/n;             //(s)

//Also:
//T = (2*%pi)*sqrt(l/g);
l = ((T^2)*g)/((2*%pi)^2);              //(m)
printf("Length of the pendulum = %.2f m\n",l);

//(b) Maximum linear acceleration of the bob:
//Displacement of the bob:
AC = 1.55 * theta;            //(m)

//Angular velocity:
omega = (2*%pi)/T;                //(rad/s)
amax = omega^2 * AC;               //(m/s^2)
printf("Maximum linear acceleration of the bob = %.2
    f m/s^2\n",amax);

//(c) Maximum linear velocity of the bob:
vmax = omega * AC;                //(m/s)
printf("Maximum linear velocity of the bob = %.2f m/
    s\n",vmax);

//(d) Maximum angular velocity of the bob:
omega_max = vmax/l;             //(rad/s)
printf("Maximum angular velocity of the bob = %.2f
    rad/s",omega_max);     //The answers vary due to
```

round off error

Scilab code Exa 26.9 To find the angle which the cord makes with the vertical and

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 l = 0.6;          //(m)(Length of cord)
6 m = 2;            //(kg)(Mass of bob)
7 a = 3;            //(m/s^2)(Acceleration)
8 g = 9.8;          //(m/s^2)(Accn due to Gravity)
9
10 //Number of oscillations made by the bob per second:
11 //Time period for one oscillation:
12 t = (2*%pi)*sqrt(l/g);      //(s)
13
14 //No. of oscillations made by the bob per second:
15 n = 1/t;                    //(Hz)
16 printf("No. of oscillations made by the bob per
      second = %.2f Hz\n",n);     //The answers vary due
       to round off error
17
18 //Angle, which the cord will make with the vertical:
19 //Let theta = Angle which the cord will make with
      the vertical.
20
21 //Weight of the bob:
22 w = m*g;                     //(N)
23
24 //Inertial force acting on the bob:
25 IF = m*a;                    //(N)
26
27 theta = atand(IF/w);         //(Degrees)
28
```

```
29 //Tension in the cord:
30 T = sqrt(w^2 + IF^2);          //(N)
31 printf("Tension in the cord = %.2f N",T);
```

Scilab code Exa 26.10 To find the number of seconds a day will lose

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 gp = 301;              //(Gravity at pole)
6 ge = 300;              //(Gravity at the equator)
7
8 //Change in Gravity:
9 dg = ge - gp;
10
11 //Let dn = No. of seconds the pendulum will lose in
     one day.
12
13 //No. of seconds in one day or 24 hours:
14 n = 24 * 60 * 60;          //(s)
15
16 //Also,
17 //dn/n = dg/(2*g)
18
19 dn = (dg * n)/(2*gp);      //(s)
20 printf("No. of seconds the pendulum will lose in one
      day = %.2f s",dn);
```

Scilab code Exa 26.11 To find the length by which the pendulum should be shortened

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
```

```
clc
clear all
t = 1;          //(s)(Time)
dn = -5;        //(No. of second the pendulum loses
   in one day)
g = 9.8;        //(m/s^2)(Accn due to Gravity)

//Length of the pendulum which will have one beat
   per second:
//Let l = Length of the pendulum.

//Time for one beat:
//t = (%pi)*sqrt(l/g);

l = g/(t*(%pi)^2) * 1000;          //(mm)
printf("Length of the pendulum = %.2f mm\n",l);
   //(Answer rounded off in book)

//Length by which the pendulum should be shortened:
//Let dl = Length in mm by which the pendulum should
    be shortened, to keep the correct time.

//No. of seconds in one day or 24 hours:
n = 24 * 60 * 60;          //(s)

//Also,
//dn/n = -(dl/2*l);
dl = -(dn * 2 * l)/n;      //(mm)
printf("Length in mm by which the pendulum should be
    shortened, to keep the correct time = %.2f mm",
   dl);
```

Scilab code Exa 26.12 To find the approximate height of a mountain

```
//OS-version - Windows 10
```

```
//Scilab-version - 6.0.2
clc
clear all
dn = -20;       //(No. of seconds the pendulum loses
   in one day)
r = 6400;       //(km)(Radius of the earth)

//Let h = Height of mountain in km.

//No. of seconds in one day or 24 hours:
n = 24 * 60 * 60;       //(s)

//Also;
//dn/n = - h/r;
h = - (dn/n) * r;         //(km)
printf("Height of mountain in km = %.2f km",h);
```

Scilab code Exa 26.13 To find the period of small oscillation

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
l = 0.6;      //(m)(Length of rod)
m = 0.25;     //(kg)(Mass of rod)
h = 0.26;     //(m)(Distance between the point of
   suspension and the centre of gravity of the body)
g = 9.8;      //(m/s^2)(Accn due to Gravity)

//Mass of moment of inertia of the rod about the
   pivot G:
I0 = (m * (0.3)^2)/3 + m * h^2;     //(kg-m^2)

//Period of small oscillation:
t = (2*%pi) * sqrt(I0/(m*g*h));       //(s)
```

```
15 printf("Period of small oscillation = %.2f s",t);
```

Scilab code Exa 26.15 To find the frequency of oscillation

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 mr = 1;       //(kg)(Mass of rod)
6 mB = 2.5;     //(kg)(Mass at B)
7 l = 0.6;      //(m)(Length of rod)
8 s = 1800;      //(N/m)(Stiffness of spring)
9
10 //Let n = Frequency of oscillation,
11 //theta = Small angular displacement of the rod,
12 //alpha = Angular acceleration of the rod AB.
13
14 //Mass moment of inertia of the system about A:
15 IA = mr*(l)^2/3 + mB*(l)^2;      //(kg-m^2)
16
17 //Extension of the spring:
18 //d = 0.3*theta;
19
20 //Restoring force = s*d;
21
22 //Restoring moment about A = Restoring force * 0.3 =
    162*theta;      -(1)
23
24 //Disturbing moment about A = IA * alpha = 1.02 *
   alpha    N-m;      -(2)
25
26 //Equating (1) and (2):
27 //Let alpha/theta = AT;
28 AT = (s*l/2*l/2)/IA;
29
```

```
30 //Frequency of oscillation:
31 n = (1/(2*%pi))*sqrt(AT);          //(Hz)
32 printf("Frequency of oscillation = %.2f Hz",n);
```

Scilab code Exa 26.16 To find the length of equivalent simple pendulum

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m = 0.5;      //(kg)(Mass of the body)
6 h = 0.3;      //(m)(Distance of centre of oscillation
      from the centre of gravity)
7 IG = 0.125;  //(kg-m^2)(Moment of inertia about
     centroidal axis)
8
9 //Let kG = Radius of gyration about the centroidal
     axis.
10
11 //IG = m * kG^2;
12 kG = sqrt(IG/m);
13
14 //Length of equivalent simple pendulum:
15 L = h + kG^2/h;        //(m)
16 printf("Length of equivalent simple pendulum = %.2f
     m",L);
```

Scilab code Exa 26.17 To find the radius of gyration of the pendulum and the numbe

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
```

```
5  no = 40;      //(No. of oscillations/min)
6  l = 0.45;     //(m)(Distance of centre of gravity
      from the centre of oscillation)
7  AB = 1.5;     //(m)(Length of AB)
8  g = 9.81;     //(m/s^2)(Accn due to Gravity)
9
10 //Radius of gyration:
11 //Let kG = Radius of gyration about an axis through
      centre of gravity and parallel to the knife edge,
12 //L = Length of the equivalent simple pendulum.
13
14 //Frequency of the pendulum:
15 n = no/60;      //(Hz)
16
17 //Time period:
18 T = 1/n;       //(s)
19
20 //Also;
21 //T = (2*%pi)*sqrt(L/g);
22 L = (T^2 * g)/(2*%pi)^2;       //(m)
23
24 //And;
25 //L = l + kG^2/l;
26 kG = sqrt((L - l)*l) * 1000;     //(mm)
27 printf("Radius of gyration about an axis through
      centre of gravity and parallel to the knife edge
      = %.2f mm\n",kG);     //The answers vary due to
      round off error
28
29 //No. of oscillations per minute:
30 h = AB - l;      //(m)
31
32 //Length of the equivalent simple pendulum:
33 L = h + (kG/1000)^2/h;        //(m)
34
35 //No. of oscillations per second of the pendulum
      when supported at B:
36 n1 = (1/(2*%pi))*sqrt(g/L);
```

```
37
38 //No. of oscillations per minute:
39 N = n1 * 60;
40 printf("No. of oscillations per minute = %.2f",N);
```

Scilab code Exa 26.18 To calculate the time taken by the bob for one revolution an

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m = 2;      //(kg)(Mass of the sphere)
6 l = 1.3;    //(m)(Length of the string)
7 r = 0.5;    //(m)(Radius of the horizontal circle)
8 g = 9.8;    //(m/s^2)(Accn due to Gravity)
9
10 //(i) Time taken by the bob for one revolution:
11
12 //Vertical distance between the bob and O(i.e. AO):
13 h = sqrt(l^2 - r^2);          //(m)
14
15 //Time taken by the bob for one revolution:
16 t = (2*%pi) * sqrt(h/g);     //(s)
17 printf("Time taken by the bob for one revolution = %
   .2f s\n",t);
18
19 //(ii) Tension in the string:
20 T = (m*g*l)/h;          //(N)
21 printf("Tension in the string = %.2f N",T);
```

Scilab code Exa 26.19 To find the angle which the string will make with the vertic

```
1 //OS-version - Windows 10
```

```
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 l = 1.5;       //(m)(Length of pendulum)
6 N = 30;        //(r.p.m)(Angular speed of the pendulum
    )
7 r = 0.5;       //(m)(Radius of the circle)
8 g = 9.8;       //(m/s^2)(Accn due to Gravity)
9
10 //Let theta = Angle which the string will make with
    the verticle.
11
12 //Angular velocity of the bob:
13 omega = (2*%pi*N)/60;      //(rad/s)
14
15 //tand(theta) = (W^2 * r)/g;
16 theta = atand((omega^2 * r)/g);
17 printf("Angle which the string will make with the
    verticle = %.2f degrees",theta);
```

Chapter 27

Collision of Elastic Bodies

Scilab code Exa 27.1 To find the velocity of the second ball

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m1 = 1;     //(kg)(Mass of first ball)
6 u1 = 2;     //(m/s)(Initial velocity of first ball)
7 m2 = 2;     //(kg)(Mass of second ball)
8 u2 = 0;     //(Initial velocity of second ball)
9 v1 = 0;     //(Final velocity of first ball after
     impact)
10
11 //Velocity of second ball after impact:
12 //Let v2 = Velocity of the second ball after impact.
13
14 //From law of conservation of momentum:
15 //m1*u1 + m2*u2 = m1*v1 + m2*v2;
16 v2 = (m1*u1 + m2*u2 - m1*v1)/m2;     //(m/s)
17 printf("Velocity of the second ball after impact = %
     .2f m/s\n",v2);
18
19 //Coefficient of restitution:
```

```
20 //Let e = Coefficient of restitution;
21 //From the law of collision of elastic bodies.
22 e = (v2 - v1)/(u1 - u2);
23 printf("Coefficient of restitution = %.2f",e);
```

Scilab code Exa 27.6 To find the loss of kinetic energy

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  m1 = 1;     //(kg)(Mass of first sphere)
6  u1 = 3;     //(m/s)(Initial velocity of first sphere)
7  m2 = 5;     //(kg)(Mass of second sphere)
8  u2 = 0.6;   //(m/s)(Initial velocity of second sphere
       )
9  e = 0.75;   //(Coefficient of restitution)
10
11 //Loss of kinetic energy during impact:
12 EL = [(m1*m2)*(u1-u2)^2*(1-e^2)]/(2*(m1 + m2));
13 printf("Loss of kinetic energy during impact = %.2f
       J\n",EL);
14
15 //Final velocity of the first sphere:
16 //Let v1 = Final velocity of the first sphere,
17 //v2 = Final velocity of the second sphere.
18
19 //From law of conservation of momentum:
20 //m1*u1 + m2*u2 = m1*v1 + m2*v2
21
22 //v1 + 5v2 = 6              -(1)
23
24 //From law of collision of elastic bodies:
25 //(v2-v1) = e*(u1-u2)
26 //v2 - v1 = 1.8             -(2)
```

```
27
28 //From (1) and (2):
29 v2 = 7.8/6;              //(m/s)
30
31 //Substituting this in eqn (1):
32 v1 = 6 - (5 * v2);       //(m/s)
33 printf("Final velocity of the first sphere = %.2f m/
      s",v1);
```

Scilab code Exa 27.7 To find the direction and velocity of the 2 kg and 4 kg ball

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  m1 = 2;      //(kg)(Mass of first ball)
6  u1 = 3;      //(m/s)(Initial velocity of first ball)
7  m2 = 4;      //(kg)(Mass of second ball)
8  u2 = 1;      //(m/s)(Initial velocity of second ball)
9  alpha1 = 30;      //(Degrees)(Angle, which initial
      velocity of first ball makes with the line of
      impact)
10 alpha2 = 30;      //(Degrees)(Angle, which the
      initial velocity of second ball makes with the
      line of impact)
11 e = 0.5;          //(Coefficient of restitution)
12
13 //(a) Direction, in which the 4 kg ball will move
      after the impact:
14 //Let theta1 = Angle, which the 2 kg ball makes with
       the line of impact,
15 //theta2 = Angle, which the 4 kg ball makes with the
       line of impact,
16 //v1 = Velocity of the 2 kg ball after impact,
17 //v2 = Velocity of the 4 kg ball after impact.
```

```
18
19 //Component of velocities perpendicular to the line
      of impact remain unchanged:
20 //v1*sind(theta1) = 1.5          -(1)
21 //v2*sind(theta2) = 0.5          -(2)
22
23 //From law of conservation of momentum:
24 //v1*cosd(theta1) + 2*v2*cosd(theta2) = 4.33
      -(3)
25
26 //From law of collision of lastic bodies:
27 //v2*cosd(theta2) - v1*cosd(theta1) = .866       -(4)
28
29 //(3) + (4):
30 //v2*cosd(theta2) = 1.732             -(5)
31
32 //(2)/(5):
33 theta2 = atand(0.5/1.732);        //(Degrees)
34 printf("Angle, which the 4 kg ball makes with the
      line of impact = %.2f degrees\n",theta2);
35
36 //(b) Velocity of the 4 kg ball after impact:
37 //Substituting theta2 in equation (2):
38 v2 = (0.5)/sind(theta2);          //(m/s)
39 printf("Velocity of the 4 kg ball after impact = %.2
      f m/s\n",v2);
40
41 //(c) Direction, in which the 2 kg ball will move
      after impact:
42 //Substituting the value of theta2 and v2 in
      equation (4):
43 //v1*cosd(theta1) = 0.866;
44
45 //(1)/(6):
46 theta1 = atand(1.5/0.866);        //(Degrees)
47 printf("Angle, which the 2 kg ball makes with the
      line of impact = %.2f degrees\n",theta1);
48
```

```
49 //(d) Velocity of 2 kg ball after impact:
50 //Substituting the value of theta1 in equation (1):
51 v1 = 1.5/sind(theta1);       //(m/s)
52 printf("Velocity of the 2 kg ball after impact = %.2
      f m/s",v1);
```

Scilab code Exa 27.9 To find the coefficient of restitution and the expected heigh

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 h0 = 1;      //(m)(Height from which the ball is
      dropped)
6 h1 = 0.81;   //(m)(Height to which the ball rose
      after first bounce)
7 g = 9.8;     //(m/s^2)(Accn due to Gravity)
8
9 //(a) Coefficient of restitution:
10 //Velocity with which the ball impringes on the
      floor:
11 u = sqrt(2*g*h0);        //(m/s)        -(1)
12
13 //Velocity with which the ball rebounds:
14 v = sqrt(2*g*h1);        //(m/s)        -(2)
15
16 //v = e*u;
17 e = v/u;
18 printf("Coefficient of restituion = %.2f\n",e);
19
20 //(b) Expected height after the second bounce:
21 //Let h2 = Expected height after the second bounce.
22
23 //Velocity with which the ball impringes second time
      :
```

```
u = v;                          //(m/s)

//Velocity, with which the ball rebounds:
//v = sqrt(2*g*h2);             //(m/s)

//Again, v = e*u
h2 = (e*u)^2/(2*g);             //(m)
printf("Expected height after the second bounce = %
   .2f m",h2);     //The answers vary due to round
   off error
```

Scilab code Exa 27.10 To find the height from which the ball must be dropped

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
e = (0.5)^(1/3);        //(Coefficient of restitution)
g = 9.8;                //(m/s^2)(Accn due to Gravity)

//Let h0 = Height from which the ball is dropped,
//h1 = Height after first rebound,
//h2 = Height after second rebound,

//Velocity with which the ball impringes on the
   floor:
//u = sqrt(2*g*h0)              -(1)

//Velocity with which the ball rebounds first time:
//v = sqrt(2*g*h1)              -(2)

//Velocity with which the ball impringes after first
    rebound:
//u1 = v;                       -(3)

```

```
21 //Velocity with which the ball rebound second time:
22 //v1 = sqrt(2*g*h2)
23
24 //Velocity with which the ball impringes after
      second rebound,
25 //u2 = v1                          -(4)
26
27 //Velocity with which the ball rebounds third time:
28 u3 = sqrt(2*g*16);                 //(m/s)     -(5)
29
30 //During first impact:
31 //v = e*u;
32 //sqrt(2*g*h1) = e * sqrt(2*g*h0)          -(6)
33
34 //During second impact:
35 //v1 = e*u1;
36 //sqrt(2*g*h1) = sqrt(2*g*h2)/e            -(7)
37
38 //During third impact:
39 //u3 = e*v1
40 //sqrt(2*g*16) = e * sqrt(2*h*h2)
41 //sqrt(2*g*h2) = sqrt(2*g*16)/e;           -(8)
42
43 //Substituting (8) in (7):
44 //sqrt(2*g*h1) = 4*sqrt(2*g)/e            -(9)
45 //Substituting (7) in (6)
46
47 h0 = ((4*sqrt(2*g)/e^3)^2)/(2*g);      //(m)
48 printf("Height from which the ball is dropped = %.2f
      m",h0);
```

Scilab code Exa 27.11 To find the direction and velocity of the body after impact

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
```

```
clc
clear all
u = 4;          //(m/s)(Initial velocity of the body)
alpha = 90 - 30;     //(Degrees)(Angle, which the
    initial velocity of the body makes with the line
    of impact)
e = 0.5;        //(Coefficient of restitution)

//(a) Direction of the body after impact:
//Let theta = Angle, which the final velocity makes
    with the line of impact, and
//v = Final velocity of the body after impact.

//From law of conservation of momentum:

//v*sind(theta) = u*sind(alpha) = 3.464      -(1)

//From the law of collision of elastic bodies:
//v*cosd(theta) = 1                          -(2)

//(1)/(2):
theta = atand(3.464);      //(Degrees)
printf("Angle, which the final velocity makes with
    the line of impact = %.2f degrees\n",theta);

//(b) Velocity of the body after impact:
//Substituting the value of theta in equation (2):
v = 1/cosd(theta);      //(m/s)
printf("Final velocity of the body after impact = %
    .2f m/s",v);
```

Chapter 28

Motion Along a Circular Path

Scilab code Exa 28.1 To find the centrifugal force acting on the body

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m = 5;      //(kg)(Mass of body)
6 r = 1.5;    //(m)(Radius of circle)
7 omega = 2;      //(rad/s)(Angular velocity of the
     body)
8
9 //Centrifugal force = F = m*W^2*r;
10 F = m * omega^2 * r;      //(N)
11 printf("Centrifugal force = %.2f N",F);
```

Scilab code Exa 28.2 To find the value of centrifugal force acting on the stone

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
```

```
clear all
m = 1;      //(kg)(Mass of stone)
r = 1;      //(m)(Radius of circle)
v = 10;     //(m/s)(Linear velocity of the stone)

//Centrifugal force = F = (m*v^2)/r
F = (m * v^2)/r;        //(N)
printf("Centrifugal force = %.2f N",F);
```

Scilab code Exa 28.3 To find the maximum angular velocity

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
m = 0.25;       //(kg)(Mass of ball)
r = 2;          //(m)(Length of string or radius of
    circle)
F = 25;         //(N)(Maximum tension in the string)

//Let omega = Maximum angular velocity at which the
    ball can be rotated.

//F = m*W^2*r;
omega = sqrt(F/(m*r));      //(rad/s)
printf("Maximum angular velocity at which the ball
    can be rotated = %.2f rad/s",omega);
```

Scilab code Exa 28.4 To find the tensions in the string

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
```

```
clear all
m = 0.5;      //(kg)(Mass of body)
N = 2;        //(rev/s)(Angular rotation of the body)
r = 1.2;      //(m)(Radius of circle)
g = 9.8;      //(m/s^2)(Accn due to Gravity)

//(i) Tension in the string when the body is at the
   top of the circle:

//Angular velocity of the body:
omega = (2 * %pi * N);     //(rad/s)

T1 = m*omega^2*r - m*g;    //(N)
printf("Tension in the string when the body is at
   the top of the circle = %.2f N\n",T1);

//(ii) Tension in the string when the body is at the
    bottom of the circle:

T2 = m*omega^2*r + m*g;      //(N)
printf("Tension in the string when the body is at
   the bottom of the circle = %.2f N",T2);
```

Scilab code Exa 28.5 To find the least velocity

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
r = 3;           //(m)(Radius of spherical cage)
m = 750;         //(kg)(Mass of motor cycle and rider
   )
g = 9.8;         //(m/s^2)(Accn due to gravity)

//Let v = Least velocity of the motor cyclist.
```

```
10
11 //Centrifugal force = F = (m*v^2)/r = 250*v^2
12
13 //In order to maintain the contact with the highest
      point of the cage, the centrifugal force must be
      equal to the weight of the motor cycle and the
      rider.
14 //250*v^2 = m*g;
15 v = sqrt((m*g)/250);      //(m/s)
16 printf("Least velocity of the motor cyclist = %.2f m
      /s",v);
```

Scilab code Exa 28.6 To find the force exerted on the rails

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m = 60;     //(tonnes)(Mass of railway engine)
6 r = 200;    //(m)(Radius of circular path)
7 v = 10;     //(m/s)(Velocity of engine)
8
9 //Force exerted on the rails:
10 //Pc = (m*v^2)/r
11
12 Pc = (m * v^2)/r;     //(kN)
13 printf("Force exerted on the rails = %.2f kN",Pc);
```

Scilab code Exa 28.7 To find the reaction between between the automobile and the r

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
```

```
clear all
m = 1.5;        //(tonnes)(Mass of automobile)
v = 15;         //(m/s)(Velocity of automobile)
r = 25;         //(m)(Radius of the sag)
g = 9.8;        //(m/s^2)(Accn due to Gravity)

//Reaction between the automobile and the load while
    travelling at the lowest part of the sag:
R = (m*v^2)/r + m*g;        //(kN)
printf("Reaction between the automobile and the load
    while travelling at the lowest part of the sag =
    %.2f kN",R);
```

Scilab code Exa 28.8 To find the angle of the track

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
r = 200;        //(m)(Radius of the tack)
v = 25;         //(m/s)(Speed of the car)
g = 9.8;        //(m/s^2)(Accn due to Gravity)

//Let theta = Angle of the bank.
//tand(theta) = v^2/(g*r);
theta = atand(v^2/(g*r));     //(Degrees)
printf("Angle of the bank = %.2f degrees",theta);
      //The answers vary due to round off error
```

Scilab code Exa 28.9 To find the amount by which the outer rail must be elevated

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
```

```
clc
clear all
G = 1.67;       //(m)(Gauge of the track)
r = 500;        //(m)(Radius of the curve)
v = 12.5;       //(m/s)(Radius of the curve)
g = 9.8;        //(m/s^2)(Accn due to Gravity)

//Superelevation:
S = (G * v^2)/(g * r) * 1000;          //(mm)
printf("Elevation of the outer rail = %.2f mm",S);
      //The answers vary due to round off error
```

Scilab code Exa 28.10 To find the superelevation

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
G = 1.67;      //(m)(Gauge of the track)
r = 1000;      //(m)(Radius of curved track)
g = 9.8;       //(m/s^2)(Accn due to Gravity)
n1 = 15;       //(trains)
n2 = 10;       //(trains)
n3 = 5;        //(trains)
n4 = 2;        //(trains)

v1 = 50;       //(kmph)(Speed of n1 trains)
v2 = 60;       //(kmph)(Speed of n2 trains)
v3 = 70;       //(kmph)(Speed of n3 trains)
v4 = 80;       //(kmph)(Speed of n4 trains)

//Equilibrium speed:
v = [(n1 * v1) + (n2 * v2) + (n3 * v3) + (n4 * v4)
   ]/(n1 + n2 + n3 + n4) * 1000/3600;          //(km.
   p.h.)
```

```
20
21 //Superelevation:
22 S = [(G * v^2)/(g * r)] * 1000;          //(mm)
23 printf("Superelevation = %.2f mm",S);     //(Answer
   varies due to round off error)
```

Scilab code Exa 28.11 To find the reactions of the wheels

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m = 1.2;      //(tonnes)(Mass of the vehicle)
6 r = 100;      //(m)(Radius of the curve)
7 v = 8.33;     //(m/s)(Velocity of the vehicle)
8 h = 1;        //(m)(Height of the c.g. of the vehicle
    from the road level)
9 a = 0.75;     //(m)(Half of the distance between the
   centre lines of the wheel)
10 g = 9.8;     //(m/s^2)(Accn due to Gravity)
11
12 //Reaction at the inner wheel:
13 RA = (m*g)/2 * (1 - (v^2*h)/(g*r*a));     //(kN)
14 printf("Reaction at the inner wheel = %.2f kN\n",RA)
   ;
15
16 //Reaction at the outer wheel:
17 RB = (m*g)/2 * (1 + (v^2*h)/(g*r*a));     //(kN)
18 printf("Reaction at the outer wheel = %.2f kN",RB);
```

Scilab code Exa 28.12 To find the speed of the vehicle

```
1 //OS-version - Windows 10
```

```
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 w = 1;          //(tonne)(Weight of the vehicle)
6 r = 40;         //(m)(Radius of the curve)
7 h = 0.75;       //(m)(Height of the centre of gravity
     of the vehicle from road level)
8 a = 0.6;        //(m)(Half of the distance between
     centre lines of the wheels)
9 g = 9.8;        //(m/s^2)(Accn due to gravity)
10
11 //Maximum speed at which the vehicle should run, in
     order to avoid overturning:
12 vmax = sqrt((g*r*a)/h) * 3600/1000;       //(km.p.h.)
13 printf("Maximum speed at which the vehicle should
     run, in order to avoid overturning = %.2f km.p.h.
     ",vmax);     //The answers vary due to round off
     error
```

Scilab code Exa 28.13 To find the maximum speed of the car

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 r = 50;       //(m)(Radius of level track)
6 mu = 0.45;    //(Coefficient of friction)
7 g = 9.8;      //(m/s^2)(Accn due to Gravity)
8
9 //Maximum speed at which the car can travel:
10 vmax = sqrt(mu*g*r) * 3600/1000;        //(km.p.h.)
11 printf("Maximum speed at which the car can travel =
     %.2f km.p.h.",vmax);     //The answers vary due to
      round off error
```

Scilab code Exa 28.14 To find the least radius of the curve

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 v = 5;      //(m/s)(Velocity of the cycle)
6 mu = 0.25;  //(Coefficient of friction)
7 g = 9.8;    //(m/s^2)(Accn due to Gravity)
8
9 //Let r = Radius of the curve in metres.
10
11 //v = sqrt(mu*g*r);
12 r = v^2/(mu*g);      //(m)
13 printf("Radius of the curve in metres = %.2f m",r);
```

Chapter 29

Balancing of Rotating Masses

Scilab code Exa 29.1 To find the radius

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m1 = 10;      //(kg)(Mass of body A)
6 r1 = 250;     //(mm)(Radius of rotating body)
7 m2 = 4;       //(kg)(Mass of body B)
8
9 //Let r2 = Radius at which c.g. of mass B should be
     placed.
10
11 //m1*r1 = m2*r2;
12 r2 = (m1 * r1)/m2;       //(mm)
13 printf("Radius at which c.g. of mass B should be
     placed = %.2f mm",r2);
```

Scilab code Exa 29.2 To find the magnitudes of the balancing masses

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  m1 = 40;    //(kg)(Mass of the body A)
6  r1 = 1;     //(m)(Distance between the centre of mass
       A and the axis of rotation)
7  l2 = 1;     //(m)(Distance between the lines of
       action of the bodies A and B)
8  l3 = 2;     //(m)(Distance between the lines of
       action of the bodies A and C)
9  r2 = 1;     //(m)(Radius of the rotating body B)
10 r3 = 2;     //(m)(Radius of the rotating body C)
11
12 //Let m2 = Magnitude of the mass B in kg and
13 //m3 = Magnitude of the mass C in kg.
14
15 //Using:
16 //m1*r1 = m2*r2 + m3*r3
17
18 //m2 + 2*m3 = 40        -(1)
19
20 //Using:
21 //m2*r2*l2 = m3*r3*l3
22 //m2 = 4*m3             -(2)
23
24 //Substituting value of m2 in eqn (1):
25 m3 = 40/6;       //(kg)
26 m2 = 4*m3;       //(kg)
27 printf("Magnitude of the mass B in kg = %.2f kg\n",
       m2);
28 printf("Magnitude of the mass C in kg = %.2f kg",m3)
       ;
```

Scilab code Exa 29.3 To find the magnitude and position of the balancing masses

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
m1 = 2;              //(kg)
m2 = 4;              //(kg)
m3 = 6;              //(kg)
m4 = 8;              //(kg)
theta1 = 0;          //(degrees)
theta2 = 60;         //(degrees)
theta3 = 120;        //(degrees)
theta4 = 135;        //(degrees)
r1 = 1.2;            //(m)
r2 = 0.8;            //(m)
r3 = 0.4;            //(m)
r4 = 0.2;            //(m)
omega = 500;          //(r.p.m.)(Angular velocity of the
    shaft)
r = 0.8;         //(m)(Radius of blancing mass)

//Let m = Magnitude of the balancing mass, and
//theta = Angle, which the balancing mass makes with
    A.

//Resolving all the assumed forces horizontally:
H = m1*r1*cosd(theta1) + m2*r2*cosd(theta2) + m3*r3*
   cosd(theta3) + m4*r4*cosd(theta4);           //
   -(1)

//Resolving all the assumed forces vertically:
V = m1*r1*sind(theta1) + m2*r2*sind(theta2) + m3*r3*
   sind(theta3) + m4*r4*sind(theta4);          //-(2)

//Resultant assumed force:
R = sqrt(H^2 + V^2);

//m * r = R;
m = R/r;          //(kg)
```

```
34 printf("Magnitude of the balancing mass = %.2f kg\n"
      ,m);
35
36 theta = atand(V/H);     //(Degrees)
37
38 //Since, E and V are both positive, therefore the
      resultant of these assumed forces lies in the
      first quadrant. Thus the balancing force must act
       in its opposite direction i.e., in the third
      quadrant.
39 A = 180 + theta;      //(Degrees)
40 printf("Angle, which the balancing mass makes with A
       = %.2f degrees",A);
```

Scilab code Exa 29.4 To find the position and magnitude of the body D

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m1 = 10;        //(kg)(Mass of body A)
6 r1 = 100;       //(mm)(Radius of the rotating body A)
7 theta1 = 0;     //(Angle which the body A makes with
      the horizontal)
8 m2 = 9;         //(kg)(Mass of body B)
9 r2 = 125;       //(mm)(Radius of the rotating body B)
10 theta2 = 60;   //(Degrees)(Angle which the body B
      makes with the horizontal)
11 m3 = 16;       //(kg)(Mass of body C)
12 r3 = 50;       //(mm)(Radius of rotating body C)
13 theta3 = 135;  //(Degrees)(Angle which the body C
      makes with the horizontal)
14 omega = 100;       //(r.p.m.)(Angular velocity)
15 r = 150;       //(mm)(Radius of balancing mass D)
16
```

```
17 //Let m = Mass of the balancing body D, and
18 //theta = Angle, which the balancing body makes with
       A.
19
20 //Resolving all the assumed forces horizontally:
21 H = m1*r1*cosd(theta1) + m2*r2*cosd(theta2) + m3*r3*
       cosd(theta3);     //-(1)
22
23 //Resolving the assumed forces vertically:
24 V = m1*r1*sind(theta1) + m2*r2*sind(theta2) + m3*r3*
       sind(theta3);     //-(2)
25
26 //Resultant assumed force:
27 R = sqrt(H^2 + V^2);
28
29 //m * r = R
30 m = R/r;          //(kg)
31 printf("Mass of the balancing body D = %.2f kg\n",m)
       ;
32
33 theta = atand(V/H);        //(Degrees)
34
35 //Since, H and E are both positive, therefore the
       resultant of these forces lies in the first
       quadrant. It is thus obvious, that the balancing
       force must act in its opposite direction.
       Therefore actual angle of the balancing body:
36 A = 180 + theta;             //(Degrees)
37 printf("Angle, which the balancing body makes with A
       = %.2f degres",A);     //The answers vary due to
       round off error
```

Scilab code Exa 29.5 To determine the unbalanced force on the spindle

```
1 //OS-version - Windows 10
```

```
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 N = 240;          //(r.p.m.)(No. of revolution)
6 r = 0.2625;       //(m)(Radius of balancing mass)
7
8 ad = 1575;
9 m = ad/(r*1000);      //(kg)
10 printf("Mass = %.2f kg\n",m);
11
12 theta = 263;     //(Degrees)(By measuring)
13 printf("theta = %.2f degrees\n",theta);
14
15 //Angular velocity of the spindle:
16 omega = (2 * %pi * N)/60;     //(rad/s)
17
18 //Unbalanced force on the spindle:
19 P = m * omega^2 * r;          //(N)
20 printf("Unbalanced force on the spindle = %.2f N",P)
   ;    //The answers vary due to round off error
```

Scilab code Exa 29.6 To find the height and speed of the governor

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m = 2.5;      //(kg)(Mass of flyballs)
6 N = 75;       //(r.p.m.)(Angular frequency)
7 g = 9.8;      //(m/s^2)(Accn due to Gravity)
8
9 //Height of the governor:
10
11 //Angular velocity of the governor:
12 omega = (2*%pi*N)/60;      //(rad/s)
```

```
13
14 //Height of governor:
15 h = g/omega^2 * 1000;              //(mm)
16 printf("Height of governor = %.2f mm\n",h);     //The
       answers vary due to round off error
17
18 //(i) Speed of the governor when the balls rise by
      20 mm:
19 //In this case height of governor,
20 h1 = (h - 20)/1000;      //(m)
21 omega = sqrt(g/h1);           //(rad/s)
22
23 //Speed of the governor:
24 N1 = (60 * omega)/(2*%pi);      //(r.p.m.)
25 printf("Speed of the governor when the balls rise by
       20 mm = %.2f r.p.m.\n",N1);
26
27 //(ii) Speed of the governor when the balls fell by
      20 mm:
28 //In this case height of governor,
29 h2 = (h + 20)/1000;      //(m)
30 omega = sqrt(g/h2);           //(rad/s)
31
32 //Speed of the governor:
33 N2 = (60 * omega)/(2*%pi);      //(r.p.m.)
34 printf("Speed of the governor when the balls fell by
       20 mm = %.2f r.p.m.\n",N2);     //The answers
      vary due to round off error
```

Chapter 30

Work Power and Energy

Scilab code Exa 30.1 To find the work done

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 F = 300;      //(N)(Pull)
6 v = 75;       //(m/min)(Velocity)
7 t = 5;        //(min)(Time)
8
9 //Distance travelled in 5 minutes:
10 s = v*t;     //(m)
11
12 //Work done by the horse:
13 W = (F * s)/1000;       //(kJ)
14
15 printf("Work done by the horse = %.2f kJ",W);
```

Scilab code Exa 30.2 To find the work done

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
s = 50;    //(mm)(Spring stretched by the application
    of force)
d = 1;     //(mm)(Stretching of spring)
F = 10;    //(N)(Force)

//Force required to stretch the spring by 50 mm:
FS = F * s;       //(N)

//Average force:
FA = FS/2;        //(N)

//Work done:
W = (FA * s)/1000;      //(J)
printf("Work done = %.2f J",W);
```

Scilab code Exa 30.3 To calculate the brake power of the engine

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
D = 1.2;          //(m)(Diameter of flywheel)
d = 0.0125;       //(m)(Diameter of rope)
N = 200;          //(r.p.m.)(Engine speed)
w = 600;          //(N)(Dead load on brake)
S = 150;          //(N)(Spring balance reading)

//Brake power of the engine:
BP = [[(w - S) * %pi * (D + d) * N]/60]/1000;      //
    (kW)
printf("Brake power of the engine = %.2f kW",BP);
```

Scilab code Exa 30.4 To find the brake power of the engine

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 W = 100;          //(N)(Weight hung from the lever)
6 L = 1.2;          //(m)(Distance between weight and
     pulley)
7 N = 150;          //(r.p.m.)(Shaft speed)
8
9 //Brake power of the engine:
10 BP = [[(2*%pi*N) * (W * L)]/60]/1000;     //(kW)
11 printf("Brake power of the engine = %.2f kW",BP);
```

Scilab code Exa 30.5 To find the power of an engine

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 W = 1200;         //(J)(Work)
6 t = 8;            //(s)(Time)
7
8 //Work done by the engine in one second:
9 W1 = W/t;     //(J/s)
10
11 //Power:
12 P = W1;       //(W)
13 printf("Power of the engine = %.2f W",P);
```

Scilab code Exa 30.6 To determine the power of the boat engine

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 v = 10;       //(m/s)(Speed of the motor boat)
6 R = 600;      //(N)(Resistance)
7
8 //Work done by the boat in one second:
9 W1 = (R * v)/1000;          //(kJ/s)
10
11 //Power:
12 P = W1;                    //(W)
13
14 printf("Power of the boat engine = %.2f kW",P);
```

Scilab code Exa 30.7 To find the power of the engine

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m = 20;           //(tonnes)(Mass of the railway
     engine)
6 v = 12.5;         //(m/s)(Velocity)
7 Fr = 1.6;         //(kN)(Frictional Resistance)
8 eta = 0.8;          //(Efficiency of the engine)
9
10 //Work done by the railway engine in one second:
11 P = Fr * v;      //(kW)
12
```

```
13 //Since, efficiency = 0.8;
14 //Actual power:
15 AP = P/eta;      //(kW)
16 printf("Power of the engine = %.2f kW",AP);
```

Scilab code Exa 30.8 To find the power of the engine

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 W = 1000;     //(kN)(Weight of the train)
6 v = 12.5;     //(m/s)(Speed of the train)
7
8 //Resistance due to friction = 1% of the weight of
     the train
9
10 //Frictional resistance:
11 Fr = (1/100) * W;     //(kN)(1% of weight of train)
12
13 //Work done in one second:
14 P = Fr * v;           //(kW)
15 printf("Power of the engine = %.2f kW",P);
```

Scilab code Exa 30.9 To find the power of the engine

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m = 400;    //(tonnes)(Mass of the locomotive)
6 v = 54;     //(kmph)(Velocity acquired)
7 v1 = (v*1000)/3600;     //(m/s)
```

```
t = 300;    //(s)(Time)
Fr = 16;    //(kN)(Frictional resistance)
u = 0;      //(Initial velocity)
A = 500;   //(N)(Air resistance)

//Let a = Acceleration of the locomotive train.

//Using: v = u + a*t;
a = (v1 - u)/t;       //(m/s^2)

//Force acquired for this acceleration:
F = m * a;       //(kN)                -(i)
//Air resistance at 54 km.p.h.:
AR = [A * (v/18)^2]/1000;        //(kN)(Converting to
    kN)     -(ii)

//Total Resistance:
TR = Fr + F + AR;        //(kN)


//Work done in one second:
P = TR * v1;      //(kJ/s)
printf("Power of the engine = %.2f kW",P);
```

Scilab code Exa 30.10 To calculate the work done

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
m = 200;          //(kg)(Mass of the block)
s = 10;           //(m)(Distance)
alpha = 15;       //(Degrees)(Inclination of plane)
g = 9.8;          //(m/s^2)(Accn due to Gravity)

```

```
10 //Resistance due to inclination:
11 RI = m*g*sind(alpha);     //(N)
12
13 //Work done:
14 W = [RI * s]/1000;       //(kJ)
15 printf("Work done = %.2f kJ",W);
```

Scilab code Exa 30.11 To find the power of the locomotive and also speed

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m = 200;                  //(tonnes)(Mass of the body)
6 Tr = 20;                  //(kN)(Tractive resistance)
7 alpha = asind(1/125);
8 v = 8;                    //(m/s)(Speed of the train)
9 g = 9.8;                  //(m/s^2)(Accn due to
     Gravity)
10
11 //Power of the locomotive:
12
13 //Resistance due to inclination:
14 RI = m*g*sind(alpha);    //(kN)
15
16 //Total resisting force:
17 TR = Tr + RI;            //(kN)
18
19 //Work done in one second:
20 P = TR * v;              //(kW)
21 printf("Power = %.2f kW\n",P);     //The answers vary
     due to round off error
22
23 //Speed which the train can attain on a level track:
24
```

```
25 //P = Tractive resistance * Speed of the train;
26
27 //Speed of the train:
28 V = P/Tr * 3600/1000;      //(km.p.h.)
29 printf("Speed of the train = %.2f km.p.h.",V);     //
   The answers vary due to round off error
```

Scilab code Exa 30.12 To find the power transmitted by the engine

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m = 300;    //(tonnes)(Mass of the body)
6 alpha = asind(1/125);     //(Degrees)
7 v = 10;     //(m/s)(Speed of the train)
8 Tr = 18;    //(kN)(Tractive resistance)
9 a = 0.2;    //(m/s^2)(Acceleration of the engine)
10 g = 9.8;   //(m/s^2)(Accn due to Gravity)
11
12 //Power transmitted by the engine when the train
   moves with a uniform speed;
13
14 //Resistance due to Inclination:
15 RI = m*g*sind(alpha);        //(kN)
16
17 //Total resisting force:
18 TR = RI + Tr;       //(kN)
19
20 //Work done in one second:
21 P = TR * v;         //(kW)
22 printf("Power transmitted by the engine when the
   train moves with a uniform speed = %.2f kW\n",P);
23
24 //Power tansmitted by the engine when the train is
```

```
    moving up with a uniform acceleration:
25
26 //Force required to accelerate the the engine and
    train:
27 F = m*a;                    //(kN)
28
29 //Total resisting force:
30 TR = RI + Tr + F;           //(kN)
31
32 //Work done in one second:
33 P = TR * v;                 //(kW)
34 printf("Power tansmitted by the engine when the
    train is moving up with a uniform acceleration =
    %.2f kW",P);
```

Scilab code Exa 30.13 To determine the tack resistance

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m = 5000;      //(kg)(Mass of the truck)
6 v = 5;         //(m/s)(Velocity of the truck)
7 alpha = asind(1/50);      //(Degrees)
8 g = 9.8;       //(m/s^2)(Accn due to Gravity)
9
10 //Track resistance per tonne mass of truck:
11
12 //Resistance due to inclination:
13 RI = m*g*sind(alpha);      //(N)
14
15 //Track resistance = Resistance due to inclination:
16 Tr = RI;                   //(N)
17 //Track resistance per tonne mass of the truck:
18 Trp = Tr/m * 1000;          //(N)
```

```
19 printf("Track resistance per tonne mass of the truck
       = %.2f N\n",Trp);
20
21 //Power to be exerted by the engine for moving the
     truck upwards:
22
23 //Total resisting force:
24 TR = (Tr + RI)/1000;        //(kN)
25
26 //Work done in one second:
27 P = TR * (2 * v);           //(kW)
28 printf("Power to be exerted by the engine for moving
       the truck upwards = %.2f kW",P);
```

Scilab code Exa 30.14 To find the power

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m = 5000;       //(kg)(Mass of the truck)
6 Tr = 750;       //(N)
7 u = 10;         //(m/s)(Speed of the truck)
8 alpha = asind(1/100);      //(Degrees)
9 g = 9.8;        //(m/s^2)(Accn due to Gravity)
10
11 //(a) Power required to propel the truck up the
     incline,
12 //Resistance due to inclination:
13 RI = m*g*sind(alpha);          //(N)
14
15 //Total resisting force:
16 TR = Tr + RI;                  //(N)
17
18 //Work done in one second:
```

```
19 P = [TR * u]/1000;                    //(kW)
20
21 printf("Power required to propel the truck up the
      incline = %.2f kW\n",P);
22
23 //(b) Power required to propel the truck on a level
      track:
24 //Work done in one second:
25 P = [Tr * u]/1000;                     //(kW)
26 printf("Power required to propel the truck on a
      level track = %.2f kW\n",P);
27
28 //(c) Power required to propel the truck down the
      incline:
29 //Net resisting force:
30 NR = Tr - RI;                          //(N)
31
32 //Work done in one second:
33 P = [NR * u]/1000;                     //(kW)
34 printf("Power required to propel the truck down the
      incline = %.2f kW",P);
```

Scilab code Exa 30.15 To find the power of the engine and tension in the coupling

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m1 = 50;          //(tonnes)(Mass of the engine)
6 m2 = 300;         //(tonnes)(Mass of the train)
7 m = m1 + m2;      //(tonnes)(Total mass)
8 alpha = asind(1/100);       //(Degrees)
9 u = 0;            //(Initial velocity)
10 Tr = 17.5;       //(kN)(Tractive resistance)
11 v = 10;          //(m/s)(Final velocity)
```

```
12 s = 1000;          //(m)(Distance)
13 g = 9.8;           //(m/s^2)(Accn due to Gravity)
14
15 //Power of the engine:
16 //Let a = Acceleration of the train.
17
18 //Resistance due to inclination:
19 RI = m*g*sind(alpha);          //(kN)
20
21 //Using: v^2 = u^2 + 2*a*s;
22 a = [v^2 - u^2]/(2*s);         //(m/s^2)
23
24 //Force required to accelerate the engine and the
      train:
25 F = m*a;                       //(kN)
26
27 //Total resisting force:
28 TR = Tr + RI + F;              //(kN)
29
30 //Work done in one second:
31 P = TR*v;                      //(kW)
32 printf("Power of the engine = %.2f kW\n",P);
33
34 //Tension in the coupling:
35 //Tractive resistance:
36 Tr = [m1 * m2]/s;              //(kN)
37
38 //Resistance due to inclination:
39 RI = m2*g*sind(alpha);         //(kN)
40
41 //Force required for acceleration(for train only):
42 F = m2 * a;                    //(kN)
43
44 //Tension in the coupling:
45 T = F + RI + Tr;               //(kN)
46 printf("Tension in the coupling = %.2f kN",T);
```

Scilab code Exa 30.16 To find the average resistance of water

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m = 60;     //(kg)(Mass of the mass)
6 h = 20;     //(m)(Height of the tower)
7 g = 9.8;    //(m/s^2)(Accn due to Gravity)
8
9 //Let P = Average resistance of the water.
10
11 //Potential energy of the man before jumping:
12 PE = m*g*h;          //(N-m)          -(1)
13
14 //Work done by the average resistance of water =
     Average resistance of water * Depth of water;
15 //W = P*2            //(N-m)          -(2)
16
17 //Since the total potential energy of the man is
     used in the work done by the water:
18
19 P = PE/2;            //(N)
20 printf("Average resistance of the water = %.2f N",P)
     ;
```

Scilab code Exa 30.17 To find the maximum compression of the spring

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
```

```
m = 15;        //(tonne)(Mass of the truck)
v = 1.6;       //(m/s)(Velocity of the truck)
k = 1.25;      //(mm/kN)(Buffer spring constant)

//Let x = Maximum compression of the spring in mm

//Kinetic energy of the truck:
KE = (1/2)*m*v^2 * 1000;          //(kN/mm)       -(1)

//Compressive load:
//CL = 0.8*x                    (kN)

//Work done in compressing the spring = Average
   compressive load * Displacement;

//W = (1/2)*(0.8*x)*x;          (kN-mm)       -(2)

//Since the entire kinetic energy of the truck is
   used to compress the spring therefore equating
   (1) and (2):
x = sqrt(KE/0.4);               //(mm)
printf("Maximum compression of the spring in mm = %
   .2f mm",x);
```

Scilab code Exa 30.18 To find the velocity with which the wagon strikes the post

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
m = 50;       //(tonnes)(Mass of the wagon)
u = 0;        //(Initial velocity)
s = 30;       //(m)(Distance)
theta = asind(0.01);     //(Degrees)
Tr = 2.5;     //(kN)(Track resistance)
```

```
10 k = 0.05;    //(mm/kN)(Bumper spring constant)
11 g = 9.8;     //(m/s^2)(Accn due to Gravity)
12
13 //Velocity with which the wagon strikes the post:
14 //Let v = Velocity with which the wagon strikes the
      post
15
16 //Gravitational pull:
17 GP = m*g*sind(theta);       //(kN)
18
19 //Net force responsible for moving the wagon:
20 NF = GP - Tr;               //(kN)
21
22 //F = m*a;
23 a = NF/m;                    //(m/s^2)
24
25 //Using: v^2 = u^2 + 2*a*s;
26 v = sqrt(u^2 + 2*a*s);     //(m/s)
27 printf("Velocity with which the wagon strikes the
      post = %.2f m/s\n",v);
28
29 //Amount by which the spring will be compressed:
30 //Let x = Amount by which the spring will be
      compressed in mm.
31
32 //Kinetic energy:
33 KE = (1/2)*(m*v^2) * 1000;        //(kN-mm)     -(i)
34
35 //Compressive load:
36 //CL = 20*x
37
38 //Work done in compressing the spring:
39 //W = Average load * Displacement
40 //W = 10*x^2                                  -(ii)
41
42 //Since the entire kinetic energy of wagonis used to
      compress the spring, therefore equating (i) and
      (ii)
```

```
43 x = sqrt(KE/10);                    //(mm)
44 printf("Amount by which the spring will be
      compressed in mm = %.2f mm",x);     //The answers
      vary due to round off error
```

Scilab code Exa 30.19 To find the velocity of the bullet

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  m = 0.03;     //(kg)(Mass of bullet)
6  M = 10;       //(kg)(Mass of body)
7  g = 9.8;      //(m/s^2)(Acn due to Gravity)
8  H = 0.8;      //(m)(Height)
9  theta = 30;   //(degrees)(Angle made at the roof)
10
11 //Let u = Initial velocity of the bullet,
12 //v = Velocity of the body after impact.
13
14 //From geometry:
15 h = H - H*cosd(theta);       //(m)
16
17 //Kinetic energy of the body and bullet after impact
      :
18 //KE = 5.015*v^2              (N-m)          -(i)
19
20 //Potential energy of the body B:
21 PEB = (m + M)*g*h;             //(N-m)      -(ii)
22
23 //Equating (i) and (ii):
24 v = sqrt(PEB/((m + M)/2));          //(m/s)
25
26 //Momentum of the body and bullet just after impact:
27 MM = (M + m)*v;                 //(kg-m/s)    -(iii)
```

```
28
29 //Momentum of the bullet just before impact:
30 //MB = 0.03*u                    //(kg-m/s)    -(iv)
31
32 //Equating (iii) and (iv):
33 u = MM/m;                         //(m/s)
34 printf("Initial velocity of the bullet = %.2f m/s",u
      );
35 //The answers vary due to round off error
```

Scilab code Exa 30.20 To find the average force of resistance of the ground

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 m = 200;          //(kg)(Mass of hammer)
6 x = 0.5;          //(m)(Distance through which the
      pile is driven into ground)
7 h = 4;            //(m)(Height through which the
      hammer falls)
8 g = 9.8;          //(m/s^2)(Acn due to Gravity)
9
10 //Average force of resistance of the ground:
11 R = [(m*g)*(h/x + 1)]/1000;              //(kN)
12 printf("Average force of resistance of the ground =
      %.2f kN",R);
```

Scilab code Exa 30.21 To find the resistance offered by the wooden block

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
```

```
4  clear all
5  m = 0.5;          //(kg)(Mass of hammer)
6  M = 0.025;        //(kg)(Mass of nail)
7  v = 5;            //(m/s)(Velocity of hammer)
8  x = 0.025;        //(m)(Distance through which the nail
       is driven into wooden block)
9  g = 9.8;          //(m/s^2)(Accn due to Gravity)
10
11 //Let h = Height through which the pile hammer fell
       before striking the pile.
12
13 //v = sqrt(2*g*h);
14 h = v^2/(2*g);        //(m)
15
16 //Resistance offered by the wooden block:
17 R = [m^2 * g * h]/[x * (m + M)] + (m + M)*g;     //(N
       )
18 printf("Resistance offered by the wooden block = %.2
       f N",R);
19 //The answers vary due to round off error
```

Chapter 31

Kinetics of Motion of Rotation

Scilab code Exa 31.1 To find the kinetic energy of the circular wheel

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 M = 50;          //(kg)(Mass of the wheel)
6 r = 0.2;         //(m)(Radius of the wheel)
7 omega = 300;       //(r.p.m.)(Angular velocity)
8
9 //Moment of inertia of the circular wheel:
10 I = (1/2)*M*r^2;       //(kg-m^2)
11
12 //Angular velocity of the wheel:
13 omega = 300/60 * 2*%pi;     //(rad/s)(Converting
      velocity from r.p.m to rad/s)
14
15 //Kinetic energy of the rotating wheel:
16 E = (1/2)*I*omega^2;         //(J)
17 printf("Kinetic energy of the rotating wheel = %.2f
      J",E);     //The answers vary due to round off
      error
```

Scilab code Exa 31.2 To find the fluctuation of energy

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 M = 6500;      //(kg)(Mass of flywheel)
6 k = 1.8;       //(m)(Radius of gyration)
7 N1 = 120;      //(r.p.m.)(Maximum speed of flywheel)
8 N2 = 118;      //(r.p.m.)(Minimum speed of flywheel)
9
10 //Average speed of the flywheel:
11 N = (1/2)*(N1 + N2);       //(r.p.m.)
12
13 //Mass moment of inertia:
14 I = M*k^2;      //(kg-m^2)
15
16 //Fluctuation of energy:
17 E = [(%pi^2)/900 * I * N * (N1 - N2)]/1000;      //(
   kJ)
18 printf("Fluctuation of energy = %.2f kJ",E);
```

Scilab code Exa 31.3 To find the moment of inertia of the wheel and mass of the fl

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 k = 0.9;     //(m)(Radius of gyration of flywheel)
6 N = 360;     //(r.p.m.)(Mean speed of the flywheel)
7 E = 30000;   //(N-m)(Fluctuation of energy)
8
```

```
//(i) Moment of inertia of the wheel:
//Let I = Moment of inertia of the wheel.

//Maximum speed of the wheel:
N1 = N + (0.02 * N);     //(r.p.m.)

//Minimum speed of the wheel:
N2 = N - (0.02 * N);     //(r.p.m.)

//Fluctuation of energy:
//E = %pi^2/900 * I * N * (N1 -N2);
I = (E * 900)/(%pi^2 * N * (N1 - N2));    //(kg-m^2)
printf("Moment of inertia of the wheel = %.2f kg-m
    ^2\n",I);

//(ii) Mass of the wheel:
M = I/k^2;      //(kg)
printf("Mass of the wheel = %.2f kg",M);
```

Scilab code Exa 31.4 To find the average torque exerted on the flywheel

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
M = 8000;         //(kg)(Mass of the flywheel)
omega0 = 0;           //(Initial angular speed)
omega = 6*%pi;        //(rad/s)(Final angular speed)
t = 180;          //(s)(Time)
k = 0.6;          //(m)(Radius of gyration of the
    flywheel)

//Let alpha = Constant angular acceleration of the
    flywheel.

```

```
13 //Mass moment of inertia of the flywheel:
14 I = M*k^2;       //(kg-m^2)
15
16 //Using: omega = omega0 + alpha*t;
17 alpha = (omega - omega0)/t;       //(rad/s^2)
18
19 //Average torque exerted by the flywheel:
20 T = I*alpha;                 //(N-m)
21 printf("Average torque exerted by the flywheel = %.2
     f N-m",T);     //The answers vary due to round off
     error
```

Scilab code Exa 31.5 To find the time taken by the wheel in coming to rest

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 h = 0.04;       //(m)(Thickness of the flywheel)
6 d = 0.2;        //(m)(Width of the flywheel)
7 r = 1;          //(m)(Mean radius)
8 omega0 = 300/60 * 2*%pi;    //(rad/s)(Initial angular
     speed)
9 Fc = 100;       //(N-m)(Frictional couple)
10 rho = 7900;    //(kg/m^3)(Density of steel)
11
12 //Let alpha = Constant angular acceleration of
     flywheel,
13 //t = Time taken by the flywheel in coming to rest.
14
15 //Volume of the flywheel:
16 V = %pi * 2 * d *h;     //(m^3)
17
18 //Mass of the flywheel:
19 M = V * rho;          //(kg)
```

```
20
21 //Mass moment of inertia:
22 I = M*r^2;          //(kg-m^2)
23
24 //Frictional couple:
25 //Fc = I*alpha;
26 alpha = Fc/I;       //(rad/s^2)
27
28 //Using: W = W0 - alpha*t (retardation);
29 omega = 0;
30 t = (omega0 - omega)/alpha;    //(s)
31 printf("Time taken by the flywheel in coming to rest
       = %.2f s",t);     //The answers vary due to round
       off error
```

Scilab code Exa 31.6 To find the angular velocity of the cylinder

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 M = 100;     //(kg)(Mass of cylinder)
6 r = 0.5;     //(m)(Radius of the cylinder)
7 m = 10;      //(kg)(Mass of the block)
8 t = 2;       //(s)(Time)
9 g = 9.8;     //(m/s^2)(Accn due to Gravity)
10 omega0 = 0;      //(Initial angular velocity)
11
12 //Linear acceleration of the solid cylinder:
13 a = (2*m*g)/(2*m + M);      //(m/s^2)
14
15 //Angular acceleration:
16 alpha = a/r;                //(rad/s^2)
17
18 //Angular velocity of the cylinder 2 seconds after
```

```
    the motion:
omega = omega0 + alpha*t;          //(rad/s)
printf("Angular velocity of the cylinder 2 seconds
    after the motion = %.2f rad/s",omega);     //(The
    answers vary due to round off error)
```

Scilab code Exa 31.7 To find the acceleration of the descending mass pull in the r

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
m = 6;     //(kg)(Mass of the body)
M = 60;    //(kg)(Mass of the solid disc)
D = 50;    //(cm)(Diameter of the disc)
s = 15;    //(m)(Distance)
g = 9.8;   //(m/s^2)(Accn due to Gravity)
u = 0;     //(Initial velocity)

//(i) Acceleration of the descending mass:
a = (2*m*g)/(2*m + M);      //(m/s^2)
printf("Acceleration of the descending mass = %.2f m
    /s^2\n",a);

//(ii) Pull in the rope:
P = (M*m*g)/(2*m + M);     //(N)
printf("Pull in the rope = %.2f N\n",P);

//(iii) Velocity after the mass has descended 15 m:
//Let v = Velocity after the mass has descended 15 m
    .
//Using: v^2 = u^2 + 2*a*s;
v = sqrt(u^2 + 2*a*s);
printf("Velocity after the mass has descended 15 m =
    %.2f m/s",v);
```

Scilab code Exa 31.8 To find the moment of inertia of the wheel about its axis

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 l = 5.4;      //(m)(Length of the string)
6 P = 98;       //(N)(Force)
7 omega = 3 * 2*%pi;    //(rad/s)(Angular velocity)
8
9 //Let I = Moment of inertia of the wheel about its
     axis.
10
11 //Work done in pulling the string:
12 omega = P * l;      //(N-m)                          -(
     i)
13
14 //Kinetic energy of the wheel:
15 //E = (1/2)*I*W^2 = 177.7 * I;        (N-m)     -(ii)
16
17 //Equating (i) and (ii):
18 I = omega/177.7;              //(kg-m^2)
19 printf("Moment of inertia of the wheel about its
     axis = %.2f kg-m^2",I);
```

Scilab code Exa 31.9 To find the acceleration of the body and tension in the rope

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
```

```
M = 800;      //(kg)(Mass of pulley)
k = 0.8;      //(m)(Radius of gyration)
r = 1;        //(m)(Radius of pulley)
T = 60000;    //(N-m)(Torque)
m = 3000;     //(kg)(Mass of the body to be lifted)

//(i) Acceleration of the body:

//Let a = Acceleration of the body,
//alpha = Angular acceleration of the body,
//P = Tension in the rope.

//Considering the motion of the hanging body:
//Resultant force = P - 29400                -(i)
//Force acting on the body = 3000*a          -(ii)

//Equating (i) and (ii):
//P - 29400 = 3000*a                         -(iii)

//Moment of inertia of the pulley:
I = M*k^2;      //(kg-m^2)

//Accelerating torque:
//T1 = T*alpha = 512 * alpha;

//Torque due to tension in the rope:
//T2 = P*r = P;

//Total torque:
//T = T1 + T2;

//60000 = T;
//P = 60000 - 512*a

//Substituting the value of P in equation (iii):
a = 30600/3512;        //(m/s^2)
printf("Acceleration of the body = %.2f m/s^2\n",a);

```

```
43 //(ii) Tension of the rope:
44 //Substituting the value of a in equation (iii):
45 P = [3000*a + 29400]/1000;        //(kN)
46 printf("Tension of the rope = %.2f kN",P);
```

Scilab code Exa 31.10 To find the acceleration of the masses and pulls on either s

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  m1 = 15;        //(kg)(Mass of the first body)
6  m2 = 5;         //(kg)(Mass of the second body)
7  r = 0.2;        //(m)(Radius of the pulley)
8  M = 10;         //(kg)(Mass of the pulley)
9  k = 0.15;       //(m)(Radius of gyration)
10 g = 9.8;        //(m/s^2)(Accn due to Gravity)
11
12 //Acceleration of the masses:
13 //Mass moment of inertia of the pulley:
14 I = M*k^2;       //(kg-m^2)
15
16 //Acceleration of the masses:
17 a = (g*(m1-m2))/[I/r^2 + (m1 + m2)];     //(m/s^2)
18 printf("Acceleration of the masses = %.2f m/s^2\n",a
   );
19
20 //Pulls on either side of the rope:
21 //!5 kg mass:
22 P1 = m1*(g - a);      //(N)
23 printf("Pull in the rope with 15 kg of mass = %.2f N
   \n",P1);     //(The answers vary due to round off
   error)
24
25 //5 kg mass:
```

```
26 P2 = m2*(a + g);     //(N)
27 printf("Pull in the rope with 5 kg of mass = %.2f N"
      ,P2);
```

Scilab code Exa 31.11 To find the torque

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  m1 = 800;      //(kg)(Mass of the body A)
6  m2 = 600;      //(kg)(Mass of the body B)
7  r = 0.4;       //(m)(Radius of pulley)
8  M = 100;       //(kg)(Mass of pulley)
9  k = 0.4;       //(m)(Radius of gyration)
10 a = -1;        //(m/s^2)(Acceleration)
11 g = 9.8;       //(m/s^2)(Accn due to Gravity)
12
13 //Pull in the rope carrying 800 kg mass:
14 P1 = m1*(g - a);      //(N)
15
16 //Pull in the rope carrying 600 kg mass:
17 P2 = m2*(g + a);      //(N)
18
19 //Moment of inertia of the body:
20 I = M*k^2;            //(kg-m^2)
21 //Angular acceleration of the body:
22 alpha = a/r;          //(rad/s^2)
23
24 //Torque due to rotation of the pulley:
25 T1 = I*alpha;         //(N-m)
26
27 //Torque acting on the pulley due to difference of
      pulls in the two ropes:
28 T2 = (P1 -P2)*r;      //(N-m)
```

```
29
30 //Torque which must be applied to the pulley:
31 T = T2 - T1;            //(N-m)
32
33 printf("Torque which must be applied to the pulley =
       %.2f N-m",T);
```

Scilab code Exa 31.12 To determine the tension in the rope

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  M = 6;        //(tonnes)(Mass of trains and cars)
6  theta = asind(1/30);        //(Degrees)
7  r = 0.5;      //(m)(Radius of the drum)
8  m = 1;        //(tonne)(Mass of drawn)
9  k = 0.45;     //(m)(Radius of gyration)
10 T = 3;        //(kN-m)(Torque applied)
11 g = 9.8;      //(m/s^2)(Accn due to Gravity)
12
13 //Let P = Tension in the rope,
14 //a = Linear acceleration of the rope,
15 //alpha = Angular acceleration of the drum.
16
17 //Considering the motion of the cage:
18
19 //Component of the weight of the cage along the
      plane:
20 CP = M*g*sind(theta);      //(kN)
21
22 //Resultant force along the plane:
23 //R = P - CP;            -(i)
24
25 //Force acting on it:
```

```
26 //F = 6*a;                -(ii)
27
28 //Equating (i) and (ii):
29 //P - 1.96 = 6*a          -(iii)
30
31 //Considering the motion of the drum:
32 I = m*k^2;                //(kg-m^2)
33
34 //Linear acceleration of the train is equal to the
      angular acceleration of the drum:
35 //alpha = 2*a;
36
37 //Accelerating torque:
38 //T1 = 0.4*a;           (kN-m)
39
40 //Torque due to tension in the rope:
41 //T2 = 0.5*P;
42
43 //Total torque:
44 //T = T1 + T2;
45
46 //This torque is equal to the torque applied to the
      drum:
47 //P = 6 - 0.8*a
48 //Substituting this value of P in equation (iii):
49 a = 4.04/6.8;             //(m/s^2)
50
51 //Substituting this value of a in equation (iii):
52 P = 6*a + CP;             //(kN)
53 printf("Tension in the rope = %.2f kN",P);
```

Scilab code Exa 31.13 To find the pulls in the two parts of the string and the ang

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
```

```
clc
clear all
m1 = 30;     //(kg)(Mass of the body A)
m2 = 10;     //(kg)(Mass of the body B)
k = 0.07;    //(m)(Radius of gyration of the pulley)
M = 4;       //(kg)(Mass of the pulley)
r1 = 0.05;   //(m)(Internal radius of the pulley)
r2 = 0.1;    //(m)(External radius of the pulley)

//Angular acceleration of the pulley:
//Let P1 = Pull in the string carrying 30 kg mass,
//P2 = Pull in the string 10 kg mass,
//alpha = Angular acceleration of the body.

//Let a1 = Acceleration of the mass 30 kg,
//a2 = Acceleration of the mass 10 kg.

//Mass moment of inertia of the pulley:
I = M*k^2;          //(kg-m^2)

//Considering the motion of mass 30 kg;
//Resultant force = 294 - P1                    -(i)

//Force acting on the body = 30*a1              -(ii)
//Equating (i) and (ii):
//294 - P1 = 30*a1                              -(iii)

//Considering mass 10 kg:
//Resultant force = P2 - 98                     -(iv)

//Force acting on it = 10*a2                    -(v)

//Equating (iv) and (v):
//P2 - 98 = 10*a2                               -(vi)

//Considering the motion of the pulley:
//a1 = 0.05*alpha
//a2 = 0.1*alpha
```

```
41
42 //Torque:
43 //T = P1*r1 - P2*r2 = 0.05*P1 - 0.1*P1          -(vii)
44
45 //Torque on the pulley:
46 //T = 0.02*alpha                                -(viii
      )
47
48 //Equating (vii) and (viii):
49 //P1 = 0.4*alpha + 2*P2                         -(ix)
50
51 //Substituting this value of P1 in equation (iii):
52 //147 - P2 = 0.95*alpha                         -(x)
53
54 //from equation (vi):
55 //P2 - 98 = alpha;                              -(xi)
56
57 //Adding (x) and (xi):
58 alpha = 49/1.95;                  //(rad/s^2)
59
60 //Pulls in the two parts of the pulley:
61 P2 = 147 - 0.95*alpha;                    //(N)
62
63 P1 = (0.4 * alpha) + 2*P2;                //(N)
64
65 printf("Angular acceleration of the body = %.2f rad/
      s^2\n",alpha);
66 printf("Pull in the string carrying 30 kg mass = %.2
      f N\n",P1);
67 printf("Pull in the string 10 kg mass = %.2f N",P2);
         //The answers vary due to round off error
```

Scilab code Exa 31.14 To find the accelerations of the masses A and B

```
1 //OS-version - Windows 10
```

```
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  m1 = 150;       //(kg)(Mass of the body A)
6  m2 = 75;        //(kg)(Mass of the body B)
7  M = 75;         //(kg)(Mass of the pulley)
8  k = 0.1;        //(m)(Radius of gyration)
9  r1 = 0.25;      //(m)(External radius of the pulley)
10 r2 = 0.125;     //(m)(Internal radius of the pulley)
11
12 //Pulls in the strings:
13 //Let P1 = Pull in the string carying 150 kg mass,
14 //P2 = Pull in the string carrying 75 kg mass.
15
16 //Let a1 = Acceleration of the 150 kg mass,
17 //a2 = Acceleration of the 75 kg mass,
18 //alpha = Angular acceleration of the pulley.
19
20 //Mass moment of inertia of the pulley:
21 I = M*k^2;          //(kg-m^2)
22
23 //Considering the motion of 150 kg mass:
24 //Resultant force = 1039 - P1;                -(i)
25
26 //Force acting on the body = 150*a1;          -(ii)
27
28 //Equating (i) and (ii):
29 //1039 - P1 = 150*a1                          -(iii)
30
31 //Considering the motion of the 75 kg mass:
32 //Resultant force = P2 - 367.5                -(iv)
33
34 //Force acting on the body = 75*a2            -(v)
35
36 //Equating (iv) and (v):
37 //P2 = 367.5 = 75*a2                          -(vi)
38
39 //Considering the motion of the pulley:
```

```
40 //a1 = 0.25*alpha
41 //a2 = 0.125*alpha
42
43 //Torque:
44 //T = 0.25*P1 - 0.125*P2                          -(vii)
45
46 //Torque on the pulley:
47 //T = 0.75*alpha                                  -(viii)
48
49 //Equating (vii) and (viii):
50 //P1 = 0.5*P2 + 3*alpha                          -(ix)
51
52 //Substituting the value of P1 in equation (iii):
53 //2078 - P2 = 81*alpha                           -(x)
54
55 //From equation (vi):
56 //P2 - 367.5 = 9.4*alpha                         -(xi)
57
58 //(x) + (xi):
59 alpha = 1710.5/90.4;        //(rad/s^2)
60
61 //Substituting the value of alpha in (x):
62 P2 = 2078 - 81*alpha;                  //(N)
63
64 //Substituting the value of alpha in (ix):
65 P1 = (0.5 * P2) + (3 * alpha);         //(N)
66
67 //Acceleration of the masses:
68 a1 = r1*alpha;          //(m/s^2)
69 a2 = r2*alpha;          //(m/s^2)
70
71 printf("Pull in the string carying 150 kg mass = %.2
      f N\n",P1);
72 printf("Pull in the string carrying 75 kg mass = %.2
      f N\n",P2);
73 printf("Acceleration of the 150 kg mass = %.2f m/s
      ^2\n",a1);
74 printf("Acceleration of the 75 kg mass = %.2f m/s^2"
```

```
    ,a2);
75 //The answers vary due to round off error
```

Scilab code Exa 31.15 To find acceleration

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  M = 20;       //(kg)(Mass of roller)
6  m = 10;       //(kg)(Mass of hanging body B)
7  g = 9.8;      //(m/s^2)(Accn due to Gravity)
8
9  //3*k^2 = 0.5*r^2
10
11 //Acceleration on the horizontal plane:
12 //a = (m*g)/[M + 2*m + M*k^2/r^2];
13 a = [m*g]/[M + 2*m + M*0.5];     //(m/s^2)
14 printf("Acceleration on the horizontal plane = %.2f
     m/s^2",a);
```

Scilab code Exa 31.16 To find the minimum value of coefficient of friction between

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  theta = 30;    //(Degrees)(Inclination of the plane)
6
7  //k^2 = 0.4*r^2
8  //Minimum value of coefficient of friction:
9  //u = (tand(theta))/[(k^2 + r^2)/k^2];
10 mu = tand(theta) * (0.4/1.4);
```

```
11 printf("Minimum value of coefficient of friction = %
      .2f",mu);
```

Scilab code Exa 31.17 To determine the minimum value of coefficient of friction an

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  r = 1;              //(m)(Radius of the wheel)
6  m = 40;             //(kg)(Mass of the wheel)
7  theta = 30;         //(Degrees)(Inclination of the plane
       )
8  g = 9.8;            //(m/s^2)(Accn due to Gravity)
9  u = 0;              //(Initial velocity)
10 s = 4;              //(m)(Distance)
11
12 //(i) Minimum value of coefficient of friction:
13 //k^2 = 0.5*r^2
14
15 //Minimum coefficient of friction:
16 //mu = [tand(theta)]/[(k^2 + r^2)/k^2];
17 mu = [tand(theta)] * [0.5/1.5];
18 printf("Minimum coefficient of friction = %.2f\n",mu
       );
19
20 //(ii) Velocity of the centre of the wheel after it
       has travelled a distance of 4 m:
21 //Let v = Velocity of the centre of the wheel.
22
23 //Acceleration of the whel when it rolls down the
       plane:
24 a = (g*sind(theta)) * (1/1.5);        //(m/s^2)
25
26 //Using: v^2 = u^2 + 2*a*s;
```

```
27 v = sqrt(u^2 + 2*a*s);                    //(m/s)
28 printf("Velocity of the centre of the wheel after it
       has travelled a distance of 4 m = %.2f m/s",v);
```

Chapter 32

Motion of Vehicles

Scilab code Exa 32.1 To find the acceleration of the vehicle frictional resistance

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 M = 8;     //(tonnes)(Mass of the truck)
6 r = 0.4;   //(m)(Radius of each wheel)
7 m = 1;     //(tonne)(Mass of each pair of wheels with
      axles)
8 k = 0.3;   //(m)(Radius of gyration of each wheel)
9 x = 1.2;   //(m)(Half of the horizontal distance
     between the centre of axles)
10 h = 1.5;  //(m)(Height of the centre of gravity of
     the truck above road level)
11 P = 5.4;  //(kN)(Tractive force)
12 g = 9.8;  //(m/s^2)(Accn due to Gravity)
13
14 //(i) Acceleration of the vehicle:
15 //Mass moment of inertia of each pair of wheels:
16 I = m*k^2;          //(t-m^2)
17
18 //Acceleration of the vehicle:
```

```
19 a = [P/(I/r^2 + M)];      //(m/s^2)
20 printf("Acceleration of the vehicle = %.2f m/s^2\n",
      a);
21
22 //(ii) Frictional resistance:
23 F = (I*a)/(2*r^2);       //(kN)
24 printf("Frictional resistance = %.2f kN\n",F);
25
26 //(iii) Reactions on the wheels:
27 //Reaction of the front pair of wheels:
28 RF = (1/2)*[M*g + (h/x)*[(I*P)/(I + M*r^2)]];     //(
      kN)
29 printf("Reaction of the front pair of wheels = %.2f
      kN\n",RF);
30
31 //Reaction on the rear pair of wheels:
32 RR = (1/2)*[M*g - (h/x)*[(I*P)/(I + M*r^2)]];     //(
      kN)
33 printf("Reaction on the rear pair of wheels = %.2f
      kN",RR);
```

Scilab code Exa 32.2 To find the reactions of the front and rear wheels

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 M = 5;          //(tonnes)(Mass of the vehicle)
6 r = 0.5;        //(m)(Radius of each wheel)
7 m = 0.75;       //(tonne)(Mass of each pair of wheel
      with axle)
8 k = 0.4;        //(m)(Radius of gyration of each
      wheel)
9 x = 1;          //(m)(Half of the horizontal distance
       between the centre of the axle)
```

```
10 h = 1.2;          //(m)(Height of the centre of gravity
       of the vehicle above road surface)
11 P = 6;            //(kN)(Tractive force)
12 y = 0.05;         //(m)(Distance between the centre of
     gravity of the vehicle and the point through
     which the tractive force acts)
13 g = 9.8;          //(m/s^2)(Accn due to Gravity)
14
15 //Mass moment of inertia of a pair of wheels:
16 I = m*k^2;        //(t-m^2)
17
18 //Reaction at the front wheels:
19 RF = (1/2)*[M*g + (P/x)*[(h*I)/(M*r^2 + I) - y]];
     //(kN)
20 printf("Reaction at the front wheels = %.2f kN\n",RF
     );
21
22 //Reaction at the rear wheels:
23 RR = (1/2)*[M*g - (P/x)*[(h*I)/(M*r^2 + I) - y]];
     //(kN)
24 printf("Reaction at the rear wheels = %.2f kN",RR);
```

Scilab code Exa 32.3 To find the accelerations of the vehicle and normal reaction

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 M = 1200;      //(kg)(Mass of the vehicle)
6 Tr = 1800;     //(N)(Tractive force)
7 h = 1;         //(m)(Height of the c.g. above road
     surface)
8 y = 1 - 0.75;      //(m)(Distance between the centre
     of gravity of the vehicle and the point through
     which the tractive force act)
```

```
x = 1.2;          //(m)(Half of the horizontal
    distance between centre of wheels)
g = 9.8;          //(m/s^2)(Accn due to Gravity)

//(i) Acceleration of the vehicle:
//RF + RR = M*g = 11760 N                -(i)

//Let a = Acceleration of the vehicle.

//Total force of resistance:
//FF + FR =(1/12)*(RF + RR) = 980 N

//Net tractive force:
P = Tr - 980;       //(N)

//Net accelerating force:
//P = M*a;
a = P/M;            //(m/s^2)
printf("Acceleration of the vehicle = %.2f m/s^2\n",
    a);

//(ii) Normal reactions at the front and rear pair
    of wheels:
//Taking moments about the centre of gravity(G) of
    the vehicle and equating the same:
//RF - RR = 442 N                  -(ii)

//(i) + (ii):
RF = (11760 + 442)/2;       //(N)

//Substituting the value of RF in equation (i):
RR = 11760 - RF;            //(N)

printf("Normal reaction at the front wheel = %.2f N\
    n",RF);
printf("Normal reaction at the rear wheel = %.2f N",
    RR);     //The answer provided in the textbook is
    wrong
```

Scilab code Exa 32.4 To calculate the maximum possible acceleration of the car

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 M = 800;      //(kg)(Mass of the motor car)
6 d = 2.75;     //(m)(Distance between the centre of the
      axles)
7 h = 0.85;     //(m)(Height of the c.g. of the car
     above the ground)
8 x1 = 1.15;   //(m)(Distance of c.g. from front wheel)
9 x2 = 2.75-1.15;    //(m)(Distance of c.g. from rer
     wheel)
10 mu = 0.6;      //(Coefficient of friction)
11
12 //(i) Maximum possible acceleration when the car has
      rear wheel drive:
13 //Let RF = Reaction on the front pair of wheels,
14 //RR = Reaction on the rear pair of wheels,
15 //a = Maximum possible acceleration of the car.
16
17 //RF + RR = M*g = 7840 N          (i)
18
19 //Since the car is moving with an acceleration (a),
     therefore accelerating force acting on it:
20 //F = M*a                        (ii)
21
22 //Force of friction on the rear pair of wheels,
23 //FR = u*RR = 0.6*RR
24
25 //Taking moments about the centre of gravity (G) of
     the car and equating the same:
26 //RF = 1.835 * RR
```

```
27
28 //Substituting this value of RF in equation (i):
29 RR = 7840/2.835;      //(N)
30
31 FR = mu * RR;        //(N)      -(iii)
32
33 //Force of friction = Accelerating force;
34 a = FR/M;             //(m/s^2)
35 printf("Maximum possible acceleration when the car
      has rear wheel drive = %.2f m/s^2\n",a);
36
37 //(ii) Maximum possible acceleration when the car
      has front wheel drive:
38 //Force of friction on the front pair of wheels:
39 //FF = 0.6*RF;
40
41 //Taking moments about the centre of gravity (G) of
      the car and equating the same:
42 //RR = 0.4*RF
43 //Substituting this value of RR in equation (i):
44 RF = 7840/1.4;        //(N)
45 FF = mu * RF;        //(N)     (iv)
46
47 //Force of friction = Accelerating force;
48 a = FF/M;       //(m/s^2)
49 printf("Maximum possible acceleration when the car
      has front wheel drive = %.2f m/s^2\n",a);
50
51 //(iii) Maximum possible acceleration when the car
      has four wheel drive:
52 //FF + FR = 0.6*(RF + RR) = 4704 N      (v)
53
54 //Total force of friction = Accelerating force;
55 a = 4704/M;       //(m/s^2)
56 printf("Maximum possible acceleration when the car
      has four wheel drive = %.2f m/s^2",a);
```

Scilab code Exa 32.5 To find the distance covered by the automobile in coming to s

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 M = 4000;      //(kg)(Mass of the automobile)
6 u = 12.5;     //(m/s)(Speed of the vehicle)
7 h = 1.2;      //(m)(Height of the centre of gravity
     of the vehicle above road surface)
8 d = 2.8;      //(m)(Distance between the two axles)
9 x1 = 1.6;     //(m)(Distance of c.g from the front
     axle)
10 x2 = 2.8 - 1.6;     //(m)(Distance of c.g. from the
     rear axle)
11
12 //(i) Distance covered by the automobile in coming
     to stop, if the brakes are applied on the rear
     pair of wheels only:
13
14 //Let s1 = Distance covered by the automobile in
     coming to stop.
15 //RF = Reaction at front pair of wheels,
16 //RR = Reation at rear pair of wheels.
17
18 //(RF + RR) = M*g = 39200 N          -(i)
19
20 //Kinetic energy of the automobile before the brakes
      are applied:
21 KE = (1/2)*M*u^2;        //(N-m)      -(ii)
22
23 //Force of friction in the rear pair of wheels:
24 //FR = 0.2*RR          (Given)
25
```

```
26 //Taking moments about the centre of gravity(G) of
     the automobile and equating the same:
27 //RF = 0.9*RR
28
29 //Substituting this value of RF in equation (i):
30 RR = 39200/1.9;      //(N)
31 FR = 0.2*RR;;        //(N)
32
33 //This resisting force stops the automobile in a
     distance of s1 metres. Therefore work done by the
      force:
34 //W = 4126*s1          -(iii)
35
36 //Equating (ii) and (iii):
37 s1 = KE/4126;       //(m)
38 printf("Distance covered by the automobile in coming
      to stop, if the brakes are applied on the rear
     pair of wheels only = %.2f m\n",s1);
39
40 //(ii) Distance covered by the automobile in coming
     to stop, if the brakes are applied on the front
     pair of wheels only:
41 //Let s2 = Distance covered by the automobile in
     coming to stop.
42
43 //Force of friction in the front pair of wheels:
44 //FF = 0.2*RF         (Given)
45
46 //Taking moments about the centre of gravity(G) of
     the automobile and equating the same:
47 //RR = 1.13*RF
48 //Substituting this value of RR in equation (i):
49 RF = 39200/2.13;     //(N)
50 FF = 0.2*RF;         //(N)
51
52 //This resisting force stops the automobile in a
     distance of s2 metres. Work done by the force:
53 //W = 3680*s2          (N-m)      (iv)
```

```
54
55 //Equating equations (ii) and (iv):
56 s2 = KE/3680;      //(m)
57 printf("Distance covered by the automobile in coming
       to stop, if the brakes are applied on the front
   pair of wheels only = %.2f m\n",s2);
58
59 //(iii) Distance covered by the automobile in coming
       to stop, if the brakes are applied on both pair
   of wheels:
60 //Let s3 = Distance covered by the automobile in
   coming to stop.
61
62 //Total force of friction:
63 //FF + FR = 0.2*(RF + RR) = 7840 N
64
65 //This total force of friction stops the automobile
   in a distance of s3 metres. Work done by the
   force:
66 //W = 7840*s3            (vi)
67
68 //Equating (ii) and (vi)
69 s3 = KE/7840;      //(m)
70 printf("Distance covered by the automobile in coming
       to stop, if the brakes are applied on both pair
   of wheels = %.2f m",s3);     //The answers vary
   due to round off error
```

Scilab code Exa 32.6 To find the distance covered by the automobile in coming to s

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 M = 2250;     //(kg)(Mass of vehicle)
```

```
6  b = 1.8;      //(m)(Wheel base)
7  u = 10;       //(m/s)(Speed of the vehicle)
8  alpha = 10;   //(Degrees)(Inclination of plane)
9  x1 = 1;       //(m)(Distance of the c.g. from the
       front wheel)
10 h = 0.9;      //(m)(Height of the c.g. above the
       ground level)
11 mu = 0.5;      //(Coefficient of friction)
12 g = 9.8;      //(m/s^2)(Accn due to Gravity)
13
14 //(i) Distance covered and time taken by the vehicle
       in coming to stop when it is going up the plane:
15 //Let RF = Reaction at the front pair of wheels,
16 //RR = Reaction at the rear pair of wheels,
17 //s1 = Distance covered by the vehicle in coming to
       stop.
18
19 //Kinetic energy of the vehicle before the brakes
       are applied:
20 KE = (1/2)*M*u^2;        //(N-m)        -(i)
21
22 //RF + RR  = 21710 N
23 //Braking resistance in both the pair of wheels:
24 //BR = mu*(RF + RR) = 10855 N
25 BR = 10855;              //(N)
26
27 //Resistance due to inclination:
28 RI = M*g*sind(alpha);    //(N)
29
30 //Total resistance:
31 TR = BR + RI;      //(N)
32
33 //This total resistance stops the vehicle in a
       distance of s1 metres. Therefore the work done by
       the force:
34 //W = TR*s1;           -(ii)
35
36 //Equating (i) and (ii):
```

```
37 s1 = KE/TR;          //(m)
38 printf("Distance covered by the vehicle in coming to
       stop when it is going up the plane = %.2f m\n",
     s1);
39
40 //Let a1 = Retardation of the vehicle,
41 //t1 = Time taken by the vehicle in coming to stop.
42 v = 0;      //(Final velocity)
43
44 //Using: v^2 = u^2 + 2*a*s;
45 a1 = (v^2 - u^2)/(2*s1);      //(m/s^2)
46
47 //Using: v = u + a*t;
48 t1 = (v - u)/a1;                //(s)
49 printf("Time taken by the vehicle in coming to stop
     when it is going up the plane = %.2f s\n",t1);
50
51 //(ii) Distance covered and time taken by the
     vehicle in coming to stop when it is coming down
     the plane:
52 //Let s2 = Distance covered by the vehicle in coming
       to stop.
53
54 //Gravitational pull due to inclination:
55 GPI = M*g*sind(alpha);      //(N)
56
57 //Net resistance:
58 NR = BR - GPI;     //(N)
59
60 //This net resistance stops the vehicle in a
     distance of s2 metres. Work done:
61 //W = 7027*s2;            -(iii)
62
63 //Equating (i) and (iii):
64 s2 = KE/NR;          //(m)
65 printf("Distance covered by the vehicle in coming to
       stop when it is coming down the plane = %.2f m\n
     ",s2);
```

```
66
67 //Let a2 = Retardation of the vehicle,
68 //t2 = Time taken by the vehicle in coming to stop.
69
70 //Using: v^2 = u^2 + 2*a*s;
71 a2 = (v^2 - u^2)/(2*s2);        //(m/s^2)
72
73 //Using: v = u + a*t;
74 t2 = (v - u)/a2;                //(s)
75 printf("Time taken by the vehicle in coming to stop
   when coming down the plane = %.2f s",t2);
```

Scilab code Exa 32.7 To find the maximum inclination on which the car can climb

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 M = 1000;    //(kg)(Mass of the car)
6 d = 2.25;    //(m)(Distance between the centre of
   wheel)
7 x1 = 1.25;   //(m)(Distance of c.g. of the car from
   front wheel)
8 h = 0.5;     //(m)(Height of c.g. of the vehicle
   above the road)
9 mu = 0.35;    //(Coefficient of friction)
10 g = 9.8;     //(m/s^2)(Accn due to Gravity)
11
12 //(i) Maximum inclination on which the car can climb
    when it is driven by the rear pair of wheels
   only:
13 //Let RF = Reaction at the front wheel,
14 //RR = Reaction at the rear wheel, and
15 //alpha = Maximum angle of inclination which the car
    can climb.
```

```
16
17 //RF + RR = 9800*cosd(alpha)
18
19 //Force of friction acts on the rear pair of wheels:
20 //FR = u*RR = 0.35*RR                    -(i)
21
22 //Resolving the forces along the plane:
23 //FR = M*g*sind(alpha) = 9800*sind(alpha)     -(ii)
24
25 //Equating (i) and (ii):
26 //RR = 28000*sind(alpha)
27 //RF = 9800*cosd(alpha) - 28000*sind(alpha)
28
29 //Taking moments about the centre of gravity(G) of
      the car and equating the same:
30 alpha = atand(0.35/1.94);      //(Degrees)
31 printf("Maximum angle of inclination which the car
      can climb = %.2f degrees\n",alpha);
32
33 //(ii) Maximum inclination on which the car can
      climb when it is driven by the front pair of
      wheels only:
34 //Force of friction at the front pair of wheels:
35 //FF = 0.35*RF                                -(iii)
36
37 //Resolving the forces along the plane:
38 //FF = M*g*sind(alpha) = 9800*sind(alpha)     -(iv)
39
40 //Equating (iii) and (iv):
41 //RF = 28000*sind(alpha)
42 //RR = 9800*cosd(alpha) - 28000*sind(alpha)
43
44 //Taking moments about the centre of gravity (G) of
      the car and equating the same:
45 alpha = atand(0.35/2.425);      //(Degrees)
46 printf("Maximum inclination on which the car can
      climb when it is driven by the front pair of
      wheels only = %.2f degrees",alpha);
```

Chapter 33

Transmission of Power by Belts and Ropes

Scilab code Exa 33.1 To determine the size of the pulley

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 N2 = 620;     //(r.p.m.)(Speed of the driver)
6 d1 = 300;     //(mm)(Diameter of pulley)
7 N1 = 240;     //(r.p.m.)(Speed of the pulley A)
8
9 //Let d2 = Diameter of the follower;
10 //Since, N2/N1 = d1/d2
11 d2 = (N1/N2)*d1;     //(mm)
12 printf("Diameter of the follower = %.2f mm",d2);
```

Scilab code Exa 33.2 To find the speed of the follower

```
1 //OS-version - Windows 10
```

```
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 d1 = 500;    //(mm)(Diameter of the engine pulley)
6 d2 = 250;    //(mm)(Diameter of the shaft pulley)
7 d3 = 700;    //(mm)(Diameter of pulley 2)
8 d4 = 280;    //(mm)(Diameter of the follower)
9 N1 = 180;    //(r.p.m.)(Speed of the engine)
10
11 //Speed of the shaft:
12 N4 = N1 * (d1/d2) * (d3/d4);     //(r.p.m.)
13 printf("Speed of the shaft = %.2f r.p.m.",N4);
```

Scilab code Exa 33.3 To find the speed of the machine shaft

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 N1 = 120;    //(r.p.m.)(Speed of the engine shaft)
6 d1 = 2;      //(m)(Diameter of the pulley on the
     engine shaft)
7 d2 = 1;      //(m)(Diameter of the machine shaft)
8 t = 0.005;   //(m)(Thickness of the belt)
9 s = 3;    //(Slip)
10
11 //(i) Speed of the machine shaft when there is no
     slip:
12 N2 = N1 * [(d1 + t)/(d2 + t)];     //(r.p.m.)
13 printf("Speed of the machine shaft when there is no
     slip = %.2f r.p.m.\n",N2);
14
15 //(ii) Speed of the machine shaft when there is a
     slip of 3% :
16 N2 = N1 * [(d1 + t)/(d2 + t)] * [1 - s/100];     //(r
```

```
    .p.m.)
17 printf("Speed of the machine shaft when there is a
    slip of 3 %% = %.2f r.p.m.",N2);     //(The
    answers vary due to round off error)
```

Scilab code Exa 33.4 To find the length of the belt

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 r2 = 0.25;    //(m)(Radius of the driven pulley)
6 l = 12;       //(m)(Distance between the centres of
    the two pulleys)
7 r1 = 0.8;     //(m)(Radius of the driving pulley)
8
9 //Length of the belt if it is open:
10 L = %pi * (r1 + r2) + 2*l + (r1 - r2)^2/l;
11 printf("Length of the belt if it is open = %.2f m\n"
    ,L);
12
13 //Length of the belt if it is crossed:
14 L = %pi * (r1 + r2) + 2*l + (r1 + r2)^2/l;
15 printf("Length of the belt if it is crossed = %.2f m
    ",L);
```

Scilab code Exa 33.5 To find the length of the belt

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 r1 = 0.3025;    //(m)(Radius of driving pulley)
```

```
6 r2 = 0.1525;    //(m)(Radius of driven pulley)
7 l = 3.5;        //(m)(Distance between the centres of
      the pulleys)
8
9 //Length of the cross belt drive:
10 L = %pi * (r1 + r2) + 2*l + (r1 + r2)^2/l;    //(m)
11 printf("Length of the cross belt drive = %.2f m",L);
        //The answers vary due to round off error
```

Scilab code Exa 33.6 To find the power transmitted by the belt

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 T1 = 1000;    //(N)(Tension in the tight side)
6 T2 = 800;     //(N)(Tension in the slack side)
7 v = 75;       //(m/s)(Speed of the belt)
8
9 //Power transmitted by the belt:
10 P = [(T1 - T2)*v]/1000;      //(kW)
11 printf("Power transmitted by the belt = %.2f kW",P);
```

Scilab code Exa 33.7 To find the difference in tensions

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 p = 120;    //(W)(Transmitting power)
6 v = 30;     //(m/s)(Speed of the belt)
7
```

```
8 //Let T1 - T2 = T12 = Necessary difference in
     tensions in the two sides of the belt.
9
10 //Power transmitted:
11 //P = ((T1 - T2)*v)/(4*N);
12 T12 =p/v;     //(N)
13 printf("Necessary difference in tensions in the two
     sides of the belt = %.2f N",T12);
```

Scilab code Exa 33.8 To find the power transmitted by the belt

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 d = 0.6;     //(m)(Diameter of pulley)
6 N = 200;     //(r.p.m.)(Speed of the pulley)
7 mu = 0.25;    //(Coefficient of friction)
8 theta = 160 * (%pi/180);     //(rad)(Angle of lap)
9 T1 = 2.5;    //(kN)(Maximum tension)
10
11 //Let T2 = Tension in the belt in slack side.
12
13 //Speed of the belt:
14 v = (%pi*d*N)/(60);     //(rad/s)
15
16 //2.3*log(T1/T2) = u*theta;
17 T2 = (T1)/(10^((mu*theta)/2.3));    //(kN)
18
19 //Power transmitted by the belt:
20 P = (T1 - T2)*v;     //(kW)
21 printf("Power transmitted by the belt = %.2f kW",P);
22 //The answers vary due to round off error
```

Scilab code Exa 33.9 To find the power transmitted by the belt

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 r1 = 0.225;      //(m)(Radius of larger pulley)
6 r2 = 0.1;        //(m)(Radius of smaller pulley)
7 l = 1.95;        //(m)(Distance between the centres of
      the pulleys)
8 N = 200;         //(r.p.m.)(Speed of the larger pulley
    )
9 T1 = 1000;       //(N)(Maximum tension in the belt)
10 mu = 0.25;       //(Coefficient of friction)
11
12 //Let T2 = Tension in the belt in slack side.
13 //Length of belt:
14 L = %pi * (r1 + r2) + 2*l + (r1 + r2)^2/l;     //(m)
15 printf("Length of the belt = %.2f m\n",L);     //The
    answers vary due to round off error
16
17 //Angle of contact between the belt and each pulley:
18 //Let theta = Angle of contact between the belt and
    each pulley.
19
20 //From geometry:
21 //Angle(MO1O2) = 180 - theta/2;
22 //Angle(MO1O2) = 80.4;      (Degrees)
23 theta = 2*(180 - 80.4);       //(Degrees)
24 printf(" Angle of contact between the belt and each
    pulley = %.2f degrees\n",theta);
25
26 //Power transmitted by the belt:
27 //Speed of the belt:
```

```
28 v = (%pi*2*r1*N)/60;          //(m/s)
29
30 theta_rad = theta*(%pi/180);      //(rad)
31
32 T2 = (T1)/(10^((mu*theta_rad)/2.3));    //(kN)
33
34 //Power transmitted by the belt:
35 P = ((T1 - T2)*v)/1000;           //(kW)
36 printf("Power transmitted by the belt = %.2f kW",P);
```

Scilab code Exa 33.10 To determine the power transmitted

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  t = 0.008;      //(m)(Thickness of belt)
6  b = 0.15;       //(m)(Width of belt)
7  d = 1.2;        //(m)(Diameter of pulley)
8  N = 180;        //(r.p.m.)(Speed of the pulley)
9  theta = 190 * (%pi/180);       //(rad)(Angle of lap)
10 rho = 1000;     //(kg/m^3)(Mass density of the belt
      material)
11 sigma = 1.5 * 10^6;    //(N/m^2)(Permissible stress
      in the belt)
12 mu = 0.3;       //(Coefficient of friction)
13
14 //(i) Power transmitted when the centrifugal tension
      is considered:
15 //Let T1 = Tension in the tight side of the belt,
16 //T2 = Tension in the slack side of the belt.
17
18 //Speed of the belt:
19 v = (%pi*d*N)/60;          //(m/s)
20
```

```
21 //Maximum tension in the belt:
22 T = sigma*b*t;          //(N)
23
24 //Mass of the belt per metre length:
25 m = (t * b)*1*rho;      //(kg)
26
27 //Centrifugal tension:
28 TC = m*v^2;             //(N)
29
30 //Tension in the tight side:
31 T1 = T - TC;            //(N)
32
33 T2 = (T1)/(10^((mu*theta)/2.3));    //(kN)
34
35 //Power transmitted:
36 P = [(T1 - T2)*v]/1000;          //(N-m/s)
37 printf("Power transmitted when the centrifugal
      tension is considered = %.2f kW\n",P);
38
39 //(ii) Power transmitted when the centrifugal
      tension is neglected:
40 T1 = 1800;      //(N)
41 T2 = (T1)/(10^((mu*theta)/2.3));    //(kN)
42
43 //Power transmitted:
44 P = [(T1 - T2)*v]/1000;         //(kW)
45 printf("Power transmitted when the centrifugal
      tension is neglected = %.2f kW",P);
```

Scilab code Exa 33.11 To determine the width of the belt

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
```

```
P = 35;      //(kW)(Power to be transmitted)
d = 1.5;     //(m)(Effective diameter of pulley)
N = 300;     //(r.p.m.)(Speed of pulley)
theta = 2*%pi*(11/24);     //(rad)
mu = 0.3;     //(Coefficient of friction)
t = 0.0095;     //(m)(Thickness of belt)
rho = 1100;     //(kg/m^3)(Mass density of the belt
    material)
sigma = 2.5 * 10^6;;     //(N/m^2)

//Let b = Width of the belt
//T1 = Tension on the tight side of the belt, and
//T2 = Tension on the slack side of the belt.

//Velocity of the belt:
v = (%pi*d*N)/60;       //(m/s)

//Power transmitted:
//Let T1 - T2 = T12
//P = (T1 - T2)*v;
T12 = P/v;          //(N)          (i)

//Also:
//2.3*log10(T1/T2) = u*theta;
//T1 = 2.375*T2;

//Substituting the value of T1 in equation (i):
T2 = 1486/1.375;      //(N)
T1 = 2.375 * T2;      //(N)

//Maximum tension in the belt:
//T = sigma*b*t = 23750*b N

//Mass of the belt per metre length:
//m = (b * t)*1*rho = 10.45*b N

//Centrifugal tension:
//TC = m*v^2 = 5800*b N
```

```
42
43 //Tension on the tight side of the belt:
44 //T1 = T - TC = 17950*b
45 b = T1/17950 * 1000;      //(mm)
46 printf("Width of the belt = %.2f mm",b);    //The
   answers vary due to round off error
```

Scilab code Exa 33.12 To calculate the power and absolute maximum power

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 b = 0.1;    //(m)(Width of the belt)
6 t = 0.01;   //(m)(Thickness of belt)
7 v = 20;     //(m/s)(Velocity of the belt)
8 l = 1;      //(m)(Length)
9 //Net driving tension(T1 - T2) = 1.8*T2(where T2 is
   tension in slack side)     -(i)
10
11 sigma = 1.8 * 10^6;     //(N/m^2)(Safe stress)
12 rho = 1000;     //(kg/m^3)(Mass density of leather)
13
14 //Power transmitted by the belt:
15 T = sigma*b*t;     //(N)
16
17 //Mass of the belt per metre length:
18 m = (b * t)*l*rho;      //(kg)
19
20 //Centrifugal tension:
21 TC = m*v^2;      //(N)
22
23 //Tension on the tight side:
24 T1 = T - TC;     //(N)
25
```

```
26 //Now calculating T2 from (i):
27 T2 = T1/2.8;     //(N)
28
29 //Power transmitted by the belt:
30 P = [(T1 - T2)*v]/1000;         //(kW)
31 printf("Power transmitted by the belt = %.2f kW\n",P
   );
32
33 //Speed at which absolute maximum power can be
   transmitted:
34 v = sqrt(T/(3*m));          //(m/s)
35 printf("Speed at which absolute maximum power can be
    transmitted = %.2f m/s\n",v);     //The answers
   vary due to round off error
36
37 //Absolute maximum power that can be transmitted by
   the belt:
38
39 //Centrifugal tension:
40 TC = T/3;          //(N)
41
42 //Tension on the tight side of the belt:
43 T1 = T - TC;      //(N)
44
45 //Tension on the slack side of the belt:
46 T2 = T1/2.8;      //(N)
47
48 //Power transmitted by the belt:
49 P = [(T1 - T2)*v]/1000;     //(kW)
50 printf("Absolute maximum power that can be
   transmitted by the belt = %.2f kW",P);
```

Scilab code Exa 33.13 To calculate the power transmitted

```
1 //OS-version - Windows 10
```

```
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 l = 4.8;    //(m)(Distance between the centre of the
     shafts)
6 r1 = 0.75;     //(m)(Radius of the larger pulley)
7 r2 = 0.5;      //(m)(Radius of the smaller pulley)
8 T0 = 3;        //(kN)(Initial tension in the belt)
9 m = 1.5;       //(kg/m)(Mass of the material)
10 mu = 0.3;      //(Coefficient of friction)
11 N2 = 400;      //(r.p.m.)(Speed of the smaller pulley
     )
12
13 //Let T1 = Tension in the tight side of the belt,
14 //T2 = Tension in the slack side of the belt.
15
16 //Velocity of the belt:
17 v = (%pi*2*r2*N2)/60;      //(m/s)(Diameter d2 = 2 *
     Radius r2)
18
19 //Initial tension in the belt when the belt is
     stationary:
20 //3 = (T1 + T2)/2;
21 //T1 + T2 = 6;       (kN)       (i)
22
23 //For an open drive:
24 alpha = asind((r1-r2)/l);       //(Degrees)
25
26 //Angle of lap for the smaller pulley:
27 theta = (180 - 2*alpha)*(%pi/180);    //(rad)
28
29 //Also,
30 //2.3*log10(T1/T2) = u*theta;
31 //T1 = 2.49*T2;
32 //Substituting this value of T1 in equation (i):
33 T2 = 6/3.49;       //(kN)
34 T1 = 2.49*T2;      //(kN)
35
```

```
36 //Power transmitted by the belt:
37 P = (T1 - T2)*v;       //(kW)
38 printf("Power transmitted by the belt = %.2f kW",P);
```

Scilab code Exa 33.14 To find the power transmitted by a rope

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 theta = %pi;      //(rad)
6 alpha = 30;       //(Degrees)(Half of pulley groove
     angle)
7 mu = 0.2;          //(Coefficient of friction)
8 m = 0.4;          //(kg/m)(Mass of rope)
9 T = 1500;         //(N)(Permissible tension)
10 v = 15;          //(m/s)(Velocity of rope)
11
12 //Centrifugal tension:
13 TC = m*v^2;      //(N)
14 T1 = T - TC;     //(N)
15
16 T2 = (T1)/(10^((mu*theta*(1/sind(alpha)))/2.3));
     //(N)
17
18 //Power transmitted by the rope drive:
19 P = [(T1 - T2)*v]/1000;          //(kW)
20 printf("Power transmitted by the rope drive = %.2f
     kW",P);
21 //The answers vary due to round off error
```

Scilab code Exa 33.15 To find the number of ropes required for the drive

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
P = 1000;      //(kW)(Power to be transmit)
d = 1;         //(m)(Diameter of the pulley)
N = 450;       //(r.p.m.)(Speed of the pulley)
T = 2250;      //(N)(Safe pull)
m = 1;         //(kg/m)(Mass of the rope)
theta = 150 * (%pi/180);      //(rad)
alpha = 22.5;       //(Degrees)(Half of the grooe
   angle)
mu = 0.3;      //(Coefficient of friction)

//Velocity of the ropes:
v = (%pi*d*N)/60;      //(m/s)

//Centrifugal tension:
TC = m*v^2;      //(N)
T1 = T - TC;     //(N)

T2 = (T1)/(10^((mu*theta*(1/sind(alpha)))/2.3));
   //(N)

//Power transmitted by one rope:
P1 = [(T1 - T2)*v]/1000;         //(kW)

//No. of ropes:
NOR = P/P1;
printf("No. of ropes = %d",NOR);      //(Answer in
   textbook is rounded off)
```

Chapter 34

Transmission of Power by Gear Trains

Scilab code Exa 34.1 To find the power transmitted

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 T = 100;    //(No. of teeth on driver)
6 p = 40;     //(mm)(Pitch of the two wheels)
7 F = 100;    //(N)(Tangential force exerted by the
     driver)
8 N = 225;  //(r.p.m.)(Speed of the driver)
9
10 //Circumference of the pitch circle:
11 C = [T * p]/1000;          //(m)
12
13 //Velocity of driver at the pitch:
14 v = (C * N)/60;            //(m/s)
15
16 //Power transmitted by the gear:
17 P = [F * v]/1000;          //(kW)
18 printf("Power transmitted by the gear = %.2f kW",P);
```

Scilab code Exa 34.2 To find the speed of F

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 NA = 975;    //(r.p.m.)(Speed of the gear wheel A)
6 TA = 20;     //(No. of teeth on wheel A)
7 TB = 50;     //(No. of teeth on wheel B)
8 TC = 25;     //(No. of teeth on wheel C)
9 TD = 75;     //(No. of teeth on wheel D)
10 TE = 26;    //(No. of teeth on wheel E)
11 TF = 65;    //(No. of teeth on wheel F)
12
13 //Let NF = Speed of the shaft F.
14
15 //NF/NA = (TA/TB) * (TC/TD) * (TE/TF);
16
17 NF = NA * (TA/TB) * (TC/TD) * (TE/TF);
18 printf("Speed of the shaft F = %.2f r.p.m.",NF);
```

Scilab code Exa 34.3 To find the load W

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 l = 0.4;     //(m)(Length of the handle)
6 r = 0.1;     //(m)(Radius of the drum)
7 TA = 20;     //(No. of teeth on wheel A)
8 TB = 80;     //(No. of teeth on wheel B)
```

```
TC = 20;      //(No. of teeth on wheel C)
TD = 100;     //(No. of teeth on wheel D)
P = 10;       //(N)(Effort applied)
eta = 0.6;      //(Efficiency of the system)

//Let W = Load that can be raised by the drum.

//Velocity ratio of the system:
VR = (1/r)*(TB/TA)*(TD/TC);

//Mechanical advantage:
//MA = W/P = W/10;

//Efficiency:
//n = MA/VR;
//0.6 = (W/10)/80;
W = eta * VR * P;      //(N)
printf("Load that can be raised by the drum = %.2f N
   ",W);
```

Scilab code Exa 34.4 To design the wheels

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
l = 600;      //(mm)(Distance between the centres of
   the two shafts)
N1 = 120;     //(r.p.m.)(Speed of the first shafts)
p = 25;       //(mm)(Pitch of the wheel)

//Let d1 = Diameter of the first wheel,
//T1 = No. of teeth on the first wheel,
//d2, T2 = Corresponding values for the follower.

```

```
13 //Distance between the shafts:
14 //600 = (1/2)*(d1 + d2)
15 //(d1 + d2) = 1200          -(i)
16
17 //d1/d2 = N2/N1
18 //d1 = 3*d2                 -(ii)
19
20 //From (i) and (ii):
21 d2 = 300;      //(mm)
22 d1 = 3*d2;     //(mm)
23
24 //No. of teeth on the first wheel:
25 T1 = ceil((%pi * d1)/p);
26 T2 = ceil((%pi * d2)/p);
27
28 //Exact diameter of first wheel:
29 d1_dash = (T1 * p)/%pi;      //(mm)
30 d2_dash = (T2 * p)/%pi;      //(mm)
31
32 //Exact distance between the two shafts:
33 l_dash = (1/2)*(d1_dash + d2_dash);     //(mm)
34
35 printf("No. of teeth of the first wheel = %.2f\n",T1
   );
36 printf("Diameter of first wheel = %.2f mm\n\n",
   d1_dash);
37 printf("No. of teeth of the second wheel = %.2f\n",
   T2);
38 printf("Diameter of second wheel = %.2f mm\n\n",
   d2_dash);
39 printf("Exact distance between the two shafts = %.2f
    mm",l_dash);
40 //(The answers vary due to round off error)
```

Scilab code Exa 34.5 To find the velocity ratio

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  T = 4.5;      //(Turns taken by the spring spindle
      for one day)
6
7  //Velocity ratio of main spindle to the hour hand
      spindle:
8  VR = T/2;
9  printf("Velocity ratio of main spindle to the hour
      hand spindle = %.2f\n",VR);
10
11 //Velocity ratio of the minute hand spindle to the
      hour spindle:
12 VRM = 12/1;
13 printf("Velocity ratio of the minute hand spindle to
       the hour spindle = %.2f\n",VRM);
14
15 //Train of wheels:
16 //Let T1 = No. of teeth on the hour hand spindle,
17 //T2 = No. of teeth on the wheel 2,
18 //T3 = No. of teeth on the wheel 3,
19 //T4 = No. of teeth on the minute hand spindle.
20
21 //T1 + T2 = T3 + T4                -(i)
22 //T1 * T3 = 12 * (T2 * T4)         -(ii)
23
24 //Assuming:
25 T4 = 8;
26 T3 = 4*T4;
27
28 //Substituting the values of T3 and T4 in equation (
      i):
29 //T1 = 40 - T2                     -(iii)
30
31 //Substituting the value of T3 and T4 in equation (
      ii):
```

```
32 //T1 = 3*T2                                -(iv)
33
34 //Substituting this value of T1 in equation (iii):
35 T2 = 40/4;
36
37 //Substituting this value of T2 in equation (iii):
38 T1 = 40 - 10;
39
40 printf("No. of teeth on the hour hand spindle = %.2f
      \n",T1);
41 printf("No. of teeth on the wheel 2 = %.2f\n",T2);
42 printf("No. of teeth on the wheel 3 = %.2f\n",T3);
43 printf("No. of teeth on the minute hand spindle = %
      .2f",T4);
```

Scilab code Exa 34.6 To find the speed of B

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 TA = 36;     //(No. of teeth on wheel A)
6 TB = 45;     //(No. of teeth on wheel B)
7 NC = 150;    //(r.p.m.)(Speed of arm C)
8
9 //Let NA = Speed of wheel A,
10 //NB = Speed of wheel B,
11 //NC = Speed of arm C.
12
13 //Speed of the wheel A relative to arm C = NA - NC
14
15 //Speed of wheel B relative to arm C = NB - NC
16
17 //Since the wheels A and B revolve in opposite
      directions:
```

```
18 //(NB - NC)/(NA - NC) = - TA/TB      -(i)
19
20 //Speed of wheel B when wheel A is fixed:
21 NA = 0;
22 NC = 150;     //(r.p.m.)
23
24 //Substituting this value of NC in equation (i):
25 NB = (NC * TA/TB) + NC;          //(r.p.m.)
26 printf("Speed of wheel B when wheel A is fixed = %.2
      f r.p.m.\n",NB);
27
28 //Speed of wheel B when wheel A makes 300 r.p.m.:
29 //Since the wheel A makes 300 r.p.m. anticlockwise,
      therefore:
30 NA = -300;     //(r.p.m.)
31
32 //Substituting this value of NA in equation (i):
33 NB = -(NA - NC)*(TA/TB) + NC;     //(r.p.m.)
34 printf("Speed of wheel B when wheel A makes 300 r.p.
      m. = %.2f r.p.m.",NB);
```

Scilab code Exa 34.7 To determine the speed of wheels B and C

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 TA = 72;     //(No. of teeth on wheel A)
6 TC = 32;     //(No. of teeth on wheel C)
7 ND = 18;     //(r.p.m.)(Speed of arm D)
8
9 //Speed of wheel C:
10 y = 18;     //(r.p.m.)
11
12 //Since wheel A is ficed:
```

```
13 x = (y * TA)/TC;      //(r.p.m.)
14
15 //Speed of wheel C:
16 NC = x + y;          //(r.p.m.)
17 printf("Speed of wheel C = %.2f r.p.m.\n",NC);
18
19 //Speed of wheel B:
20 //Let dA, dB and dC be the pitch circle diameters of
       wheels A, B and C respectively. From geometry:
21 //2*dB + dC = dA
22
23 //Since no. of teeth are proportional to their
     diameters:
24 TB = 20;
25 //Speed of wheel B:
26 NB = y - x*(TC/TB);     //(r.p.m.)
27 printf("Speed of wheel B = %.2f r.p.m.\n",NB);
```

Scilab code Exa 34.8 To determine the speed of the driven shaft

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 TS = 30;     //(No. of teeth on sun wheel)
6 TP = 50;     //(No. of teeth on planet wheels)
7 P = 4000;    //(W)(Power transmitted)
8 v = 300;     //(r.p.m.)(Speed of driving shaft)
9 n = 0.95;    //(Efficiency)
10
11 //Power tansmitted by the driven shaft:
12 PDS = P * n;     //(W)
13
14 //Speed of the driven shaft:
15
```

```
16 //Let dA, dP and dS be the pitch circle diameters of
       wheels A, P and S respectively. From geometry:
17 //dA = dS + 2*dP
18
19 //Since the no. of teeth are proportional to their
       diameters:
20 TA = TS + 2*TP;
21
22 //As the wheel A is fixed:
23 //x = -y              -(i)
24
25 //As the sun wheel rotates at 300 r.p.m.
26 //300 = y - x*(TA/TS)
27 y = (v * 3/16);
28 x = -y;
29
30 //Speed of the driven shaft:
31 N = y;     //(r.p.m.)
32 printf("Speed of the driven shaft = %.2f r.p.m.\n",N
       );
33
34 //Torque tansmitted by the driven shaft:
35 //Let T = Torque transmitted by the shaft.
36
37 //Power tansmitted by the shaft:
38 //3800 = 2*%pi*N*T;
39 T = 3800/353.4;      //(N-m)
40 printf("Power tansmitted by the shaft = %.2f N-m",T)
       ;
```

Scilab code Exa 34.9 To determine the speed of the driven shaft

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
```

```
clear all
TA = 40;    //(No. of teeth on wheel A)
TB = 50;    //(No. of teeth on wheel B)
TC = 20;    //(No. of teeth on wheel C)
v = -50;    //(r.p.m.)(Speed of driving shaft)
u = 100;    //(r.p.m.)(Speed of arm)

//Since the speed of the arm is 100 r.p.m.:
y = 100;

//As the speed of the driving shaft is 50 r.p.m.:
//x + y = -50;
x = -50 - y;

//Speed of the driving shaft:
S = y - x*(TA/TC);      //(r.p.m.)
printf("Speed of the driving shaft = %.2f r.p.m.",S)
   ;
```

Chapter 35

Hydrostatics

Scilab code Exa 35.1 To find the pressure at a point

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 H = 4;     //(m)(Depth of the point below the free
     surface of water)
6 w = 9.8;   //(kN/m^3)
7
8 //Pressure at the point:
9 p = w * H;      //(kN/m^2)
10 printf("Pressure at the point = %.2f kN/m^2",p);
```

Scilab code Exa 35.2 To find the depth of oil

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
```

```
5 w = 9.8 * 0.8;      //(kN/m^3)(Specific weight)
6 p = 50;             //(kN/m^2)(Intensity of pressure)
7
8 //Let H = Depth of oil.
9
10 //p = w * H;
11 H = p/w;       //(m)
12 printf("Depth of oil = %.2f m",H);
```

Scilab code Exa 35.3 To calculate the pressure on the base of the tank

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 l = 5;     //(m)(Length of the tank)
6 b = 2;     //(m)(Width of the tank)
7 d = 2.5;   //(m)(Depth of the tank)
8 w = 9.8 * 1;    //(kN/m^3)(Specific weight)
9
10 //Surface area of the base of the tank:
11 A = l * b;      //(m^2)
12
13 //Total pressure on the base of the tank:
14 P = w * A * d;      //(kN)
15 printf("Total pressure on the base of the tank = %.2
    f kN",P);
```

Scilab code Exa 35.4 To find the total pressure

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2.
3 clc
```

```
clear all
b = 2;    //(m)(Width of plate)
d = 4;    //(m)(Depth of plate)
w = 9.8; //(kN/m^3)(Specific weight of water)

//Area of the rectangular plate:
A = b * d;          //(m^2)

//Depth of centre of gravity of the plate from the
   water surface:
x = 2.5 + d/2;      //(m)

//Total pressure on the plate:
P = w * A * x;      //(kN)
printf("Total pressure on the plate = %.2f kN",P);
```

Scilab code Exa 35.5 To determine the total pressure on the door

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
d = 1;    //(m)(Diameter of the door)
x = 2;    //(m)(Depth of centre of opening from water
    level)
n = 1.03;    //(Specific gravity of water)
w = 9.8 * n;    //(Specific weight of water)

//Area of the circular door:
A = (%pi/4)*(d^2);     //(m^2)

//Total pressure:
P = w * A * x;       //(kN)
printf("Total pressure = %.2f kN",P);
```

Scilab code Exa 35.7 To find the total pressure on the plate

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 b = 1;    //(m)(Base of plate)
6 h = 1.5; //(m)(Altitude of plate)
7 theta = 30;    //(Degrees)(Inclination of the plane
     with the free surface of water)
8 w = 9.8 * 1;  //(kN/m^3)(Specific weight of water)
9 d = 2;    //(m)(Depth of base)
10
11 //Area of the triangular plate:
12 A = (b * h)/2;     //(m^2)
13
14 //Depth of centre of gravity of the plate from the
     water surface:
15 x = d + (h/3)*sind(theta);     //(m)
16
17 //Total pressure on the plate:
18 P = w * A * x;
19 printf("Total pressure on the plate = %.2f kN",P);
```

Scilab code Exa 35.9 To locate the centre of pressure both vertically and laterall

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 b = 3;    //(m)(Base of the triangle)
6 h = 6;    //(m)(Altitude of the triangle)
```

```
x_bar = 9;    //(m)(Head of the water on its axis)

//Vertical location of centre of pressure:
//Area of the triangular plate:
A = (b * h)/2;      //(m^2)

//Moment of inertia of triangle ABD about AD:
IABD = (6 * 1.5^3)/12;      //(m^4)

//Moment of inertia of triangle ADC about AD:
IADC = IABD;       //(m^4)

//Moment of inertia of the triangle ABC about AD:
IG = IABD + IADC;     //(m^4)

//Depth of centre of pressure of the plate from the
   water surface:
h_bar = (IG)/(A*x_bar) + x_bar;     //(m)
printf("Depth of centre of pressure of the plate
   from the water surface = %.2f m\n",h_bar);

//Horizontal location of centre of pressure:
//Will coincide with the centre of the triangle:
vl = 6/3;      //(m)(From BC)
printf("Horizontal location of centre of pressure =
   %.2f m from BC",vl);
```

Scilab code Exa 35.10 To find the total pressure and the centre of pressure

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
d = 3;      //(m)(Diameter of the circular plate)
gd = 2;     //(m)(Greatest depth)
```

```
7 ld = 1;     //(m)(Least depth)
8 w = 9.8 * 1;     //(Specific weight of water)
9
10 //Total pressure on one face of the plate:
11 //Let theta = Inclination of the plate with the
      water surface.
12
13 theta = asind((2 - 1)/3);    //(Degrees)
14
15 //Area of the circular plate:
16 A = (%pi/4)*(d^2);    //(m^2)
17
18 //Depth of centre of gravity from the water surface:
19 x_bar = (1 + 2)/2;      //(m)
20
21 //Total pressure on one face of the plate:
22 P = w * A * x_bar;      //(kN)
23 printf("Total pressure on one face of the plate = %
      .2f kN\n",P);
24
25 //Position of the centre of presure:
26 //Moment of inertia of a circular plate, about its
      centre of gravity:
27 IG = (%pi/64)*(d^4);     //(m^4)
28
29 //Depth of centre of pressure from the water surface
      :
30 h_bar = (IG * sind(theta)^2)/(A * x_bar) + x_bar;
       //(m)
31 printf("Depth of centre of pressure from the water
      surface = %.2f m",h_bar);
```

Scilab code Exa 35.11 To find the total pressure and the centre of pressure

```
1 //OS-version - Windows 10
```

```
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 b = 1;     //(m)(Base of the plate)
6 h = 1.5;   //(m)(Altitude of the plate)
7 theta = 30;     //(Degrees)(Inclination of the plate
     with the water surface)
8 w = 9.8 * 1;    //(Specific weight of water)
9 d = 2;     //(m)(Depth of base)
10
11 //Total pressure on the plate:
12 //Area of the triangular plate:
13 A = (b * h)/2;      //(m^2)
14
15 //Depth of centre of gravity of the plate from the
     water surface:
16 x_bar = d + (h/3)*sind(theta);     //(m)
17
18 //Total pressure on the plate:
19 P = w * A * x_bar;     //(kN)
20 printf("Total pressure on the plate = %.2f kN\n",P);
21
22 //Centre of pressure:
23 //Moment of inertia of the triangular section about
     its centre of gravity and parallel to the base:
24 IG = (b * h^3)/36;      //(m^4)
25
26 //Depth of centre of pressure from the water surface
     :
27 h_bar = (IG * sind(theta)^2)/(A * x_bar) + x_bar;
       //(m)
28 printf("Depth of centre of pressure from the water
     surface = %.2f m",h_bar);
```

Scilab code Exa 35.13 To find the total pressure on the plate and the depth of the

```
1  //OS-version - Windows 10
2  //Scilab-version - 6.0.2
3  clc
4  clear all
5  D = 4;     //(m)(Diameter of the plate)
6  d = 1;     //(m)(Diameter of the hole)
7  theta = 30;     //(Degrees)(Inclination of the plate
       with the horizontal)
8  w = 9.8 * 1;     //(Specific weight of water)
9
10 //Total pressure on the plate:
11
12 //Area of the main plate:
13 A1 = (%pi/4)*(D^2);     //(m^2)
14
15 //Area of hole:
16 A2 = (%pi/4)*(d^2);     //(m^2)
17
18 //Depth of centre of gravity of the main plate from
       the water surface:
19 x1_bar = 2 + 2*sind(theta);    //(m)
20 x2_bar = 2 + 1*sind(theta);    //(m)
21
22 //Pressure on the main plate:
23 P1 = w * A1 * x1_bar;       //(kN)
24 P2 = w * A2 * x2_bar;       //(kN)
25
26 //Total pressure on the plate:
27 P = P1 - P2;      //(kN)
28 printf("Total pressure on the plate = %.2f kN\n",P);
         //The answers vary due to round off error
29
30 //Centre of pressure:
31 //Let h_bar = Depth of the centre of pressure of the
       plate from the water surface.
32
33 //Moment of inertia of the main circular section
       about its centre of gravity:
```

```
34 IG1 = (%pi/64)*(D^4);     //(m^4)
35 IG2 = (%pi/64)*(d^4);     //(m^4)
36
37 //Depth of centre of pressure of the main plate from
       the water surface:
38 h1_bar = (IG1 * sind(theta)^2)/(A1*x1_bar) + x1_bar;
         //(m)
39 h2_bar = (IG2 * sind(theta)^2)/(A2*x2_bar) + x2_bar;
         //(m)
40
41 //Taking moments about the water surface and
      equating the same:
42 h_bar = [(P1 * h1_bar) - (P2 * h2_bar)]/P;      //(m)
43 printf("Depth of the centre of pressure of the plate
       from the water surface = %.2f m",h_bar);
44 //The answers vary due to round off error
```

Scilab code Exa 35.14 To find the pressure exerted on the wall and the point where

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 l = 10;    //(m)(Length of the vertical side of tank)
6 h_bar = 1.5;  //(m)(Depth of water)
7 w = 9.8 * 1;     //(kN/m^3)(Specific weight of water)
8
9 //Total pressure exerted on wall:
10
11 //Pressure BC:
12 PBC = w * h_bar;      //(kN/m^2)
13
14 //Total pressure per metre length of the storage
      tank:
15 TP = (1/2) * h_bar * PBC;     //(kN)
```

```
16
17 //Total pressure exerted on the 10 m long wall:
18 P = 10 * TP;                    //(kN)
19 printf("Total pressure exerted on the 10 m long wall
       = %.2f kN\n",P);     //The answers vary due to
      round off error
20
21 //Point where the pressure acts:
22 h = (2 * h_bar)/3;     //(m)
23 printf("Point where the pressure acts = %.2f m from
      A",h);
```

Scilab code Exa 35.15 To find the magnitude and the line of action of the resultan

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 s = 10;     //(m)(Side of the square tank)
6 d = 1;      //(m)(Depth of tank)
7 w1 = 9.8 * 1;
8 w2 = 9.8 * 2;
9 H1 = 0.5;     //(m)
10 H2 = 0.5;     //(m)
11 BF = 4.9;     //(m)
12
13 //Magnitude of the resultant force(i.e., pressure):
14
15 //Pressure DE:
16 PDE = w1 * H1;      //(kN/m^2)
17 PFC = w2 * H2;      //(kN/m^2)
18
19 //Pressure per metre length of the tank:
20 P = (1/2)*(BF * H1) + (H2 * BF) + (1/2)*(w1 * H1);
       //(kN/m^2)
```

```
21
22 //Magnitude of the total pressure on 10 m long wall:
23 PP = 10 * P;     //(kN)
24 printf("Magnitude of the total pressure on 10 m long
       wall = %.2f kN\n",PP);
25
26 //Line of action of the resultant force(i.e.,
     pressure):
27
28 //Let hh = Depth of the line of action of the
     resultant pressure from A.
29
30 //Taking moments of all the pressures about A, and
     equating the same:
31 h_bar = [((1/2)*BF*H1)*(1/3) + (H2*BF)*(3/4) + (1/2)
     *(w1 * H1)*(5/6)]/P;     //(m)
32 printf("Depth of the line of action of the resultant
       pressure from A = %.2f m",h_bar);
```

Scilab code Exa 35.16 To determine the resultant pressure on the bulkhead and the

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 l = 3;     //(m)(Length of bulkhead)
6
7 //Resultant pressure on bulkhead:
8 w1 = 9.8 * 0.78;     //(kN)
9 w2 = 9.8 * 0.88;     //(kN)
10 H1 = 1.8;        //(m)
11 H2 = 0.9;        //(m)
12 BO = 13.76;      //(m)
13 OE = 7.76;       //(m)
14
```

```
15 //Pressure BO:
16 PBO = w1 * H1;      //(kN/m^2)
17
18 //Pressure OE:
19 POE = w2 * H2;      //(kN/m^2)
20
21 //Pressure per metre length of the bulkhead:
22 P = (1/2)*(PBO * H1) - (1/2)*(POE * H2);    //(kN)
23
24 //Resultant pressure on 3 m long bulkhead:
25 P3 = 3 * P;      //(kN)
26 printf("Resultant pressure on 3 m long bulkhead = %
      .2f kN\n",P3);
27
28 //Position of the resultant pressure:
29 //Let hh = Height of the point of the resultant
      pressure from O.
30
31 //Taking moments of all the pressures anout O, and
      equating the same:
32 h_bar = [[(1/2)*(BO * H1) * 0.6] - [(1/2)*(OE * H2)
      *0.3]]/P;
33 printf("Height of the point of the resultant
      pressure from O = %.2f m",h_bar);
```

Chapter 36

Equilibrium of Floating Bodies

Scilab code Exa 36.2 To find the volume of water displaced and the position of the

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 l = 4;          //(m)(Length)
6 b = 2;          //(m)(Breadth)
7 h = 1;          //(m)(Height)
8 V = l * b * h;    //(m^3)
9 rho = 7;          //(kN/m^3)
10 g = 9.8;          //(m/s^2)(Acceleration due to
    gravity)
11
12 //Volume of water displaced:
13 //Weight of the block:
14 w = V * rho;     //(kN)
15
16 //Volume of water displaced:
17 VD = w/g;      //(m^3)
18
19 //Position of the centre of buoyancy:
20 //Depth of immersion:
```

```
21 d = VD/(l * b);      //(m)
22
23 //Centre of buoyancy:
24 CB = d/2;             //(m)
25 printf("Centre of buoyancy = %.2f m",CB);      //The
   answers vary due to round off error
```

Scilab code Exa 36.3 To find the fraction of steel that will stay below the surfac

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 ss = 7.8;         //(Specific gravity of steel)
6 sm = 13.6;        //(Specific gravity of mercury)
7 g = 9.8;          //(m/s^2)(Acceleration due to gravity
   )
8
9 //Let x = Part of the steel piece inside the mercury
   .
10 //(1 - x) = Part of the steel piece outside the
   mercury, i.e., inside water.
11
12 //Considering one cubic metre of the steel piece:
13 //(9.8 * 7.8)*1 = (9.8 * 13.6)*x + 9.8*(1 - x)
14 x = (g*ss - g)/(g*sm - g);
15
16 //Fraction of steel inside the mercury:
17 Fr = x/1;
18 printf("Fraction of steel inside the mercury = %.2f"
   ,Fr);
```

Scilab code Exa 36.4 To determine the metacentric height

```
//OS-version - Windows 10
//Scilab-version - 6.0.2
clc
clear all
sw = 0.8;    //(Specific gravity of wood)
l = 1.2;     //(m)(Length of the wooden block)
b = 0.4;     //(m)(Breadth of the block)
d = 0.3;     //(m)(Height or depth of the block)

//Depth of immersion of the block:
di = sw * d;     //(m)

//Distance of centre of buoyancy, from the bottom of
    the block:
OB = di/2;       //(m)

//Distance of c.g. from the bottom of the block:
OG = d/2;        //(m)
BG = OG - OB;    //(m)                -(i)

//Moment of inertia of rectangular section about the
    central axis and parallel to the long side:
I = (l * b^3)/12;      //(m^4)

//Volume of water displaced:
l = 1.2;                    //(m)(Length)
b = 0.4;                    //(m)(Breadth)
h = 0.3;                    //(m)(Height)
V = l * b * di;      //(m^3)

BM = I/V;                   //(m)

//Metacentric height:
GM = (BM - BG)*1000;        //(mm)
printf("Metacentric height = %.2f mm",GM);     //The
   answers vary due to round off error
```

Scilab code Exa 36.5 To find the metacentric height of the buoy

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 d = 2;          //(m)(Diameter of buoy)
6 dt = 1.2;       //(m)(Depth of buoy)
7 V = 0.4;        //(m^3)(Volume of curved portion)
8 OB1 = 1.3;      //(m)(Centre of buoyancy of the
     curved portion below the top of the cylinder)
9 OG = 0.8;       //(m)(Centre of gravity of the whole
     buoy below the top of the cylinder)
10 v = 2.6;       //(m^3)(Total volume of water
     displaced)
11
12 //Let h = Distance between the water surface and top
      of the buoy,
13 //B1 = Centre of buoyancy of the cylindrical buoy.
14
15 //Volume of water displaced by the cylindrical
     portion:
16 VV = v - V;     //(m^3)
17
18 //VV = (%pi/4)*(2^2) * (1.2 - h);
19 h = dt - VV * 4/(%pi * d^2);
20
21 //Distance of the centre of buoyancy of the
     cylindrical buoy from the top of the buoy:
22 OB2 = h + (dt - h)/2;      //(m)
23
24 //Let B = Centre of buoyancy for the whole buoy.
25
26 OB = [(V * OB1) + (VV * OB2)]/(V + VV);  //(m)
```

```
27
28 BG = OB - OG;      //(m)
29
30 //Moment of inertia of the cylindrical portion(top
      portion) about its centre of gravity:
31 I = (%pi/64)*(2^4);     //(m^4)
32 BM = I/v;               //(m)
33
34 //Metacentric height:
35 GM = (BM - BG)*1000;      //(mm)
36 printf("Metacentric height = %.2f mm",GM);
37 //The answers vary due to round off error
```

Scilab code Exa 36.6 To find the metacentric height and state whether its equilibr

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 d = 3;     //(m)(Diameter of cylinder)
6 h = 3;     //(m)(Height of cylinder)
7 s = 0.8;   //(Specific gravity)
8
9 //Depth of immersion of the cylinder:
10 di = s * d;        //(m)
11
12 //Distance of centre of buoyancy, from the bottom of
      the cylinder:
13 OB = di/2;          //(m)
14
15 //Distance of c.g. from the bottom of the cylinder:
16 OG = h/2;           //(m)
17
18 BG = OG - OB;       //(m)
19
```

```
20 //Moment of inertia of the circular section:
21 I = (%pi/64)*(d^4);     //(m^4)
22
23 //Volume of water displaced:
24 V = (%pi/4)*(d^2) * di;      //(m^3)
25
26 BM = I/V;          //(m)
27
28 //Metacentric height:
29 GM = BM - BG;      //(m)
30 printf("Metacentric height = %.2f m\n",GM);     //(
   Answer rounded off in console)
31 printf("The cylinder is in an unstable equilibrium."
   );
```

Scilab code Exa 36.7 To determine if the solid cylinder can float vertically in wa

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 l = 500;      //(mm)(Length of cylinder)
6 d = 100;      //(mm)(Diameter of the cylinder)
7 t = 10;       //(mm)(Base thickness)
8 sb = 7;       //(Specific gravity of base)
9 sr = 0.5;     //(Specific gravity of remaining portion
   )
10 h = l - t;   //(mm)(Height of water)
11
12 A = 2*%pi*(d/2)^2;     //(m^2)
13 //Distance between the combined centre of gravity
   and the bottom of cylinder (O):
14 OG = [[sr*A * h * (t + h/2)] + [sb*A * t * t/2]]/[(
   sr*A * h) + (sb*A * t)];
15
```

```
16 //Combined specific gravity:
17 CSG = [(sr * h) + (sb * t)]/(h + t);
18
19 //Depth of immersion of the cylinder:
20 di = CSG * l;     //(mm)
21
22 //Distance of centre of buoyancy from the bottom of
      the cylinder:
23 OB = di/2;        //(mm)
24 BG = OG - OB;      //(mm)
25
26 //Moment of inertia of the circular section about
      its centre of gravity:
27 I = (%pi/64)*(d^4);      //(mm^4)
28
29 //Volume of water displaced:
30 V = (%pi/4)*(d^2)*di;       //(m^3)
31 BM = I/V;                  //(mm)
32
33 //Metacentric height:
34 GM = BM - BG;     //(mm)
35 printf("Metacentric height = %.2f mm\n",GM);
36 printf("The cylinder is in an unstable equilibrium."
      );
```

Scilab code Exa 36.9 To determine if the solid cylinder can float vertically in wa

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 l = 100;     //(cm)(Length of the cylinder)
6 d1 = 20;      //(cm)(Diameter of the cylinder)
7 b = 2.5;     //(cm)(Thickness of base)
8 sb = 8;      //(Specific gravity of base)
```

```
9  sr = 0.5;    //(Specific gravity of remaining portion
       )
10 d = l - b;  //(cm)(Depth of water)
11
12 //Floating of the cylinder:
13 //Cross-section area of the cylinder:
14 A = (%pi/4)*(d^2);     //(cm^2)
15
16 //Distance between the combined centre of gravity (G
       ) and bottom of the cylinder (O):
17 OG = [[sr*A * d*(b + d/2)] + [sb*A * b * b/2]]/[(sr*
       A * d) + (sb*A * b)];     //(cm)
18
19 //Combined specific gravity of the cylinder:
20 CSG = [(d * sr) + (b * sb)]/(d + b);
21
22 //Depth of immersion of the cylinder:
23 DI = l * CSG;     //(cm)
24
25 //Distance of centre of buoyancy from the bottom of
       the buoy:
26 OB = DI/2;      //(cm)
27 BG = OG - OB;     //(cm)
28
29 //Moment of inertia of the circular section:
30 I = (%pi/64)*(d1^4);     //(cm)
31
32 //Volume of water displaced:
33 V = (%pi/4)*(d1^2)*DI;     //(cm^3)
34 BM = I/V;                  //(cm)
35
36 //Metacentric height:
37 GM = BM - BG;              //(cm)
38 printf("Metacentric height = %.2f cm\n",GM);
39 printf("The cylinder is in unstable equilibrium.\n")
       ;      //The answers vary due to round off error
40
41 //Maximum permissible length of the cylinder:
```

```
42
43 //Let l = Length of cylinder excluding metal portion
       in cm.
44
45 //Distance between the combined centre of gravity (G
       ) and the bottom of the cylinder (O):
46 //OG = (l^2 + 5*l + 100)/(2*l + 80)
47
48 //Combined specific gravity of cylinder:
49 //CSG = (0.5*l + 20)/(l + 2.5)
50
51 //Depth of immersion of the cylinder:
52 //DI = (0.5*l + 20)          (cm)
53
54 //Distance of centre of buoyancy from the bottom of
       the buoy:
55 //OB = 0.25*l + 10           (cm)
56
57 //Volume of water displaced:
58 //V = 100*%pi*(0.5*l + 20)
59 //BM = (50)/(l + 40)
60 //OM = OB + BM;
61
62 //For stable equilibrium, the metacentre (M) should
       be above centre of gravity (G) or may coincide
       with G:
63 //i.e.,       OM <= OG
64
65 //Solving:
66 //l^2 - 70*l - 1600 <= 0
67 l = (70 + sqrt(70^2 + 4*1600))/2;     //(cm)
68
69 //Maximum permissible length of the cylinder
       including the metal portion:
70 L = l + 2.5;     //(cm)
71 printf("Maximum permissible length of the cylinder =
       %.2f cm",L);
```

Scilab code Exa 36.11 To determine the minimum weight of the buoy

```
1 //OS-version - Windows 10
2 //Scilab-version - 6.0.2
3 clc
4 clear all
5 L = 1;        //(m)(Length of the conical buoy)
6 D = 1.2;      //(m)(Diameter of base of the conical
     buoy)
7 g = 9.8;      //(Acceleration due to gravity)
8
9 //Let l = Length of the cone immersed in water.
10
11 //Volume of water displaced:
12 //V = (1/3) * %pi * (0.6 * l)^2 * l       (m^3)
13 //V = 0.377 * l^3                          (m^3)
14
15 //Moment of inertia of circular section:
16 //I = (%pi/64) * (1.2 * l)^4;
17
18 //BM = I/V = 0.27 * l                      (m)
19
20 //Distance of buoyancy from the apex:
21 //OB = 0.75 * l;
22
23 //Distance of c.g. from the apex:
24 OG = 0.75 * L;          //(m)
25
26 //For stable equilibrium, the metacentre (M) should
     be above G or may coincide with G:
27 //i.e.,          BG <= BM
28 //Solving:
29 l = 0.75/1.02;          //(m)
30
```

```
31 //Volume of water displaced:
32 V = (%pi/3)*(D/2)^2 * l^3;          //(m^3)
33
34 //This should be equal to the weight of the buoy,
      weight of the buoy:
35 W = V * g;                  //(kN)
36 printf("Weight of the buoy = %.2f kN",W);
```