

Scilab Manual for
Advanced Mathematical Physics-I
by Dr Triranjita Srivastava
Physics
Kalindi College, University Of Delhi¹

Solutions provided by
Dr Triranjita Srivastava
Physics
Kalindi College, University Of Delhi

July 2, 2026

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

Contents

List of Scilab Solutions	3
1 Linear algebra: Power and Inverse Power methods for finding largest and smallest Eigenvalue and eigenvectors of matrices	5
2 Orthogonal Polynomials as Eigenfunctions of Hermitian differential operators	10
3 Determination of the principal axes of moment of inertia through diagonalization	18
4 Study of geodesics in Euclidean and other spaces(surface of a sphere, etc):Physics problem: problem of refraction.	22
5 Application to solve differential equations for a bound system – Eigen value problem	27
6 Application to computer graphics: Write operators for shear, strain, 2D rotational problems, Reflection, Translation	32
7 Lagrangian formulation in classical mechanics with constraints.	46
8 Vector-space of wave functions in Quantum-Mech: Position and Momentum differential operators and their commutator, wave function	50

List of Experiments

Solution 1.01	Power and Inverse Power Method	5
Solution 2.0	Finite Difference Method	10
Solution 3.0	Diagonalization of matrix	18
Solution 4.0	Geodesic	22
Solution 5.0	Finite Difference Method	27
Solution 6.0	Computer Graphics	32
Solution 7.0	Lagrangian Formulation	46
Solution 8.0	Hermitian Differential Op	50

List of Figures

1.1	Power and Inverse Power Method	6
2.1	Finite Difference Method	11
2.2	Finite Difference Method	16
2.3	Finite Difference Method	17
3.1	Diagonalization of matrix	21
4.1	Geodesic	23
4.2	Geodesic	23
4.3	Geodesic	24
5.1	Finite Difference Method	31
5.2	Finite Difference Method	31
6.1	Computer Graphics	33
6.2	Computer Graphics	41
6.3	Computer Graphics	42
6.4	Computer Graphics	43
6.5	Computer Graphics	44
6.6	Computer Graphics	45
7.1	Lagrangian Formulation	49
7.2	Lagrangian Formulation	49
8.1	Hermitian Differential Op	52
8.2	Hermitian Differential Op	53

Experiment: 1

Linear algebra: Power and Inverse Power methods for finding largest and smallest Eigenvalue and eigenvectors of matrices

Scilab code Solution 1.01 Power and Inverse Power Method

```
1 //Operating system: Windows 8
2 //SCILAB Ver: 5.5.2
3 //Expriment No. 1
4 //Objective: Determination of largest and smallest (
    in magnitude) Eigen value &
5 //Eigen Vectors Using Power Method and Inverse Power
    Method respectively.
6
7
8 //Enter the no dimension of a sqaure of matrix A: 3
9 //Enter the element no (1,1):2
```

```
Scilab 5.5.2 Console
Enter the dimension of row of square matrix A:  3
Enter the element no (1,1):
2
Enter the element no (1,2):
1
Enter the element no (1,3):
1
Enter the element no (2,1):
1
Enter the element no (2,2):
2
Enter the element no (2,3):
1
Enter the element no (3,1):
1
Enter the element no (3,2):
1
Enter the element no (3,3):
5

Lowest eigen value

    0.9999987

Corresponding eigen vector

- 0.7066074
  0.7076057
- 0.0003643

Largest Eigen Value

    5.7320494

Corresponding Eigen Vector

    0.3252492
    0.3252493
    0.8879335

-->|
```

Figure 1.1: Power and Inverse Power Method

```

10 //Enter the element no (1,2):1
11 //Enter the element no (1,3):1
12 //Enter the element no (2,1):1
13 //Enter the element no (2,2):2
14 //Enter the element no (2,3):1
15 //Enter the element no (3,1):1
16 //Enter the element no (3,2):1
17 //Enter the element no (3,3):5
18 //Let Matrix A is A=[2,1,1;1,2,1;1,1,5];
19
20 clc
21 clear
22 //
    *****

23 // Creating an input square matrix
24 //
    *****

25 m = input("Enter the dimension of row of square
    matrix A: ")
26
27 for i=1:m
28     for j=1:m
29         mprintf("Enter the element no (%d,%d): ",i,
    j)
30         A(i,j)=input("")
31     end
32 end
33
34 //
    *****

35 // Creating initial approximation x0
36 //
    *****

37 x=rand(m,1)

```

```

38
39 //
    *****

40 //Finding smallest Eigen Value using Inverse Power
    Method
41 //
    *****

42 z=1
43 f=1
44 y0=rand(m,1)
45 while(f>0.00001)
46     y1=inv(A)*y0
47     lowest=norm(y1,2)
48     y0=y1/lowest
49     f=abs(z-lowest)
50     z=lowest
51 end
52
53 disp(lowest,'Lowest eigen value')
54 disp(y0,'Corresponding eigen vector')
55
56 //
    *****

57 //Finding largest Eigen Value using Power Method
58 //
    *****

59 x0=rand(m,1)
60 y=1
61 d=1
62 while (d>0.00001)
63     x1=A*x0
64     highest=norm(x1,2)
65     x0=x1/highest
66     d=abs(y-highest)

```

```
67     y=highest
68 end
69 disp(highest,'Largest Eigen Value')
70 disp(x0,'Corresponding Eigen Vector')
```

Experiment: 2

Orthogonal Polynomials as Eigenfunctions of Hermitian differential operators

Scilab code Solution 2.0 Finite Difference Method

```
1 // Submitted by Dr. Triranjita Srivastava. Assistant  
   Professor , Physics Dept., Kalindi College ,  
   University of Delhi  
2  
3 // Aim: To prove the orthogonality of Hermitian  
   differential Operator  
4  
5 // Two Hermitian Differential Operators  $(-id/dx)$  and  
    $(-d^2/dx^2)$  are taken as an example  
6  
7 // Finite Difference Method is used to formulate the  
   matrices corresponding to the considered  
   Differential Operator  
8  
9 // This method takes the value of eigenfunction
```

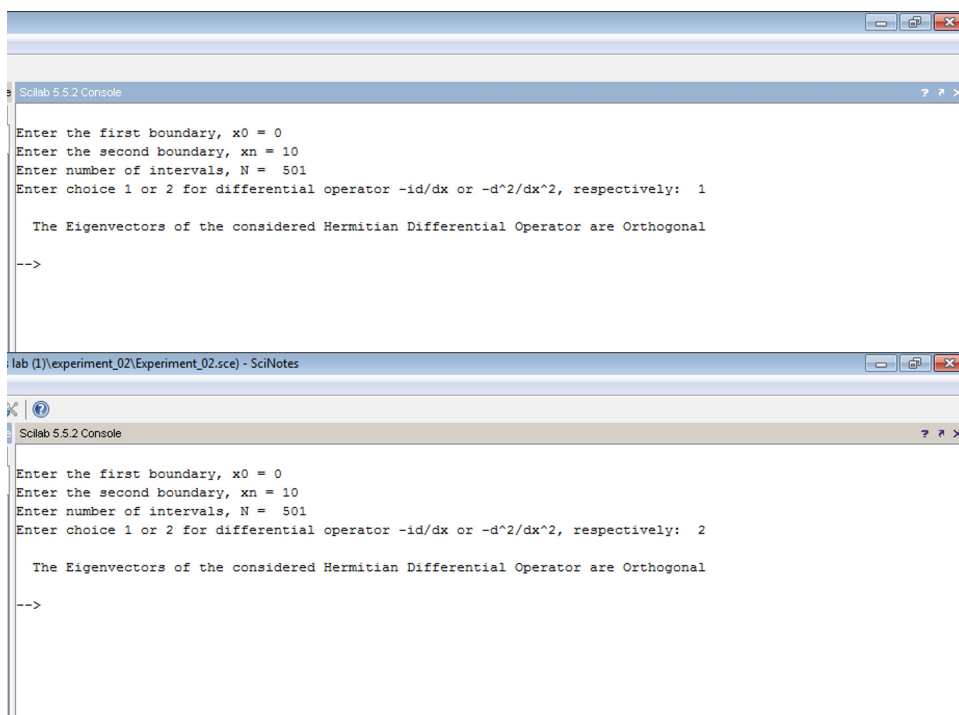


Figure 2.1: Finite Difference Method

```

    equal to 0 at the initial (x0) and final boundary
    (xn), and x is defined as x=x0+i*h (h is step
    size and i is integer)
10 // N is the number of interval, N should be taken as
    odd and such that step size is small enough for
    high accuracy
11
12 // Using Central Differences, tridiagonal matrix is
    obtained for  $(-d/dx)$  which has 0 as diagonal
    element and  $-1$  as upper adjacent diagonal and 1
    as lower adjacent diagonal elements;
13
14 // Similarly, using Central Differences, tridiagonal
    matrix is obtained for  $(-d^2/dx^2)$  which has 2
    and  $-1$  as upper and lower adjacent diagonal
    elements
15
16
17
18 clear
19 clc
20 //
    *****

21 // Boundary over which the function is to be solved
22 //
    *****

23 x0=input("Enter the first boundary, x0 = ");
24 xn=input("Enter the second boundary, xn = ");
25 N = input("Enter number of intervals, N = ");
26 h = (xn-x0)/N; //step size
27 s=input("Enter choice 1 or 2 for differential
    operator  $-id/dx$  or  $-d^2/dx^2$ , respectively: ")
28
29 select s
30 case 1
31 //

```

```

*****

32 // Defining D1 Matrix corresponding to
    differential operator  $-id/dx$ 
33 //
    *****

34 D1=zeros(N-1,N-1);
35
36 for i=1:(N-1)
37     x1(1,i)=x0+i*h;
38     D1(i,i)=0;
39     if i<(N-1)
40         D1(i,i+1)=-%i;
41         D1(i+1,i)=%i;
42     end
43 end
44 Final_D1=D1/2*h;
45
46 //
    *****

47 // Finding eigenvalue and eigenvector of
    differential operator  $-id/dx$ 
48 //
    *****

49 [eigenvector,eigenvalue] = spec(Final_D1);
50
51 case 2
52 //
    *****

53 // Defining D2 Matrix corresponding to
    differential operator  $-d^2/dx^2$ 
54 //
    *****

```

```

55
56     D2=zeros(N-1,N-1);
57
58     for i=1:(N-1)
59         x1(1,i)=x0+i*h;
60         D2(i,i)=2;
61         if i<(N-1)
62             D2(i,i+1)=-1;
63             D2(i+1,i)=-1;
64         end
65     end
66     Final_D2=D2/h^2;
67
68     //
69     // Finding eigenvalue and eigenvector of
70     // differential operator  $-d^2/dx^2$ 
71     //
72     [eigenvector,eigenvalue] = spec(Final_D2);
73     end
74     //
75     // Plotting of first three Eigenvector of
76     // differential operator  $-d^2/dx^2$ 
77
78     x=[x0,x1,xn];
79
80     if s==1 then
81         title('3 Lowest Order Eigenvectors of  $-id/dx$ 
82             ', 'fontsize',4);

```

```

82     else
83         title('3 Lowest Order Eigenvectors of  $-d^2/dx^2$ ', 'fontsize',4);
84     end
85
86     for k = 1:3
87         subplot(3,1,k)
88         ylabel('A (m)', 'fontsize',4)
89         a=get("current_axes");//get the handle of
          the newly created axes
90         a.font_size=2
91         t=get("hdl") //get the handle of the newly
          created object
92         t.font_size=2;
93         E_vector = [0;eigenvector(:,k);0];
94         plot(x,E_vector', 'linewidth',2);
95     end
96         xlabel('x-coordinate (m)', 'fontsize',4)
97     //
          *****
98     // Orthogonality Check of eigenvector of
          differential operator
99     //
          *****
100    for i=1:3
101        for j=1:3
102            P(i,j) = clean(sum((eigenvector(:,i).*
              conj(eigenvector(:,j))))));
103            if i~=j & P(i,j) ~=0
104                disp(" The Eigenvectors of the
              considered Hermitian Differential
              Operator are Not Orthogonal")
105                abort;
106            end
107        end
108    end

```

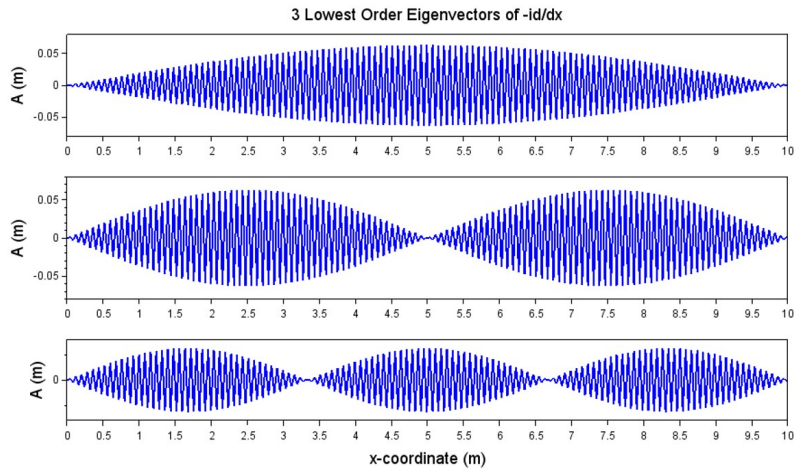


Figure 2.2: Finite Difference Method

109 `disp(" The Eigenvectors of the considered
Hermitian Differential Operator are
Orthogonal")`

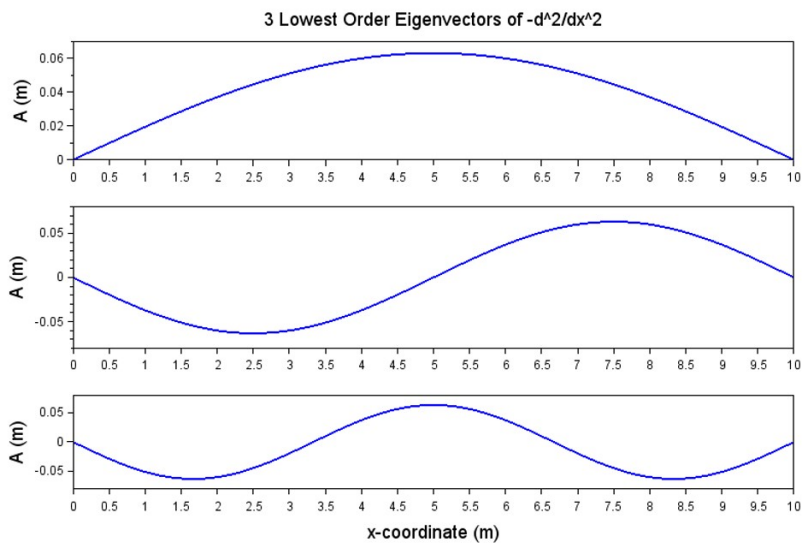


Figure 2.3: Finite Difference Method

Experiment: 3

Determination of the principal axes of moment of inertia through diagonalization

Scilab code Solution 3.0 Diagonalization of matrix

```
1 // Submitted by Dr. Triranjita Srivastava. Assistant
  Professor , Physics Dept., Kalindi College ,
  University of Delhi
2
3 // Aim: Determination of the principal axes of
  moment of inertia through diagonalization
4 // Example is a Dumbell with masses 'm1' and 'm2'
  situated at points , say coordinates are (1,1,0)
  and (-1,-1,0)
5
6 clear;
7 clc;
8 //
  *****

9 //Function for Kronecker Delta
10 //
```

```

*****

11 function d=delta(i,j)
12     if i==j then
13         d=1;
14     else d=0;
15     end
16 endfunction
17
18 //
*****

19 // Input of number of particles at discrete points
20 //
*****

21 n=input('enter no. of particles ')
22
23 r=zeros(3,3)
24 for i=1:n
25     mprintf("Enter the mass (in kg) at point (%d):
26         ",i)
27     M(i)=input("")
28     mprintf("Enter the position (x,y,z) coordinate
29         at point (%d): ",i)
30     for j =1:3
31         r(i,j)=input("")
32     end
33 end
34 I=zeros(3,3)
35 for i=1:1:3
36     for j=1:1:3
37         for k=1:1:n
38             I(i,j)=I(i,j)+(M(k)*(sum(r(k,:).^2)*
39                 delta(i,j)-(r(k,i).*r(k,j))))
38         end
39     end

```

```
40 end
41 disp("Moment of Inertia Tensor for given problem is:
      ")
42 disp(I)
43 [ab,x,bs]=bdiag(I);
44 disp("Moment of Inertia Tensor after diagonalization
      is:  ")
45 disp(ab)
```

```
Scilab 5.5.2 Console
File Edit Control Applications ?
enter no. of particles 2
Enter the mass (in kg) at point (1):
0.5
Enter the position (x,y,z) coordinate at point (1):
1
1
0
Enter the mass (in kg) at point (2):
0.5
Enter the position (x,y,z) coordinate at point (2):
-1
-1
0

Moment of Inertia Tensor for given problem is:

  1.  - 1.  0.
- 1.   1.  0.
  0.   0.  2.

Moment of Inertia Tensor after diagonalization is:

  2.  0.  0.
  0.  0.  0.
  0.  0.  2.

-->
```

Figure 3.1: Diagonalization of matrix

Experiment: 4

Study of geodesics in Euclidean and other spaces(surface of a sphere, etc):Physics problem: problem of refraction.

Scilab code Solution 4.0 Geodesic

```
1 // Submitted by Dr. Triranjita Srivastava. Assistant  
   Professor , Physics Dept. , Kalindi College ,  
   University of Delhi  
2  
3 // Aim: To study geodesics in Euclidean and  
   Cylindrical Polar coordinate System  
4  
5 clc ;  
6 clear ;
```

```
Scilab 5.5.2 Console
Input x coordinate of point A 1
Input y coordinate of point A 1
Input x coordinate of point B 6
Input y coordinate of point B 7
Input angular coordinate (in degree) of point A 10
Input z coordinate of point A 2
Input angular coordinate (in degree) of point B 340
Input z coordinate of point B 18
-->
```

Figure 4.1: Geodesic

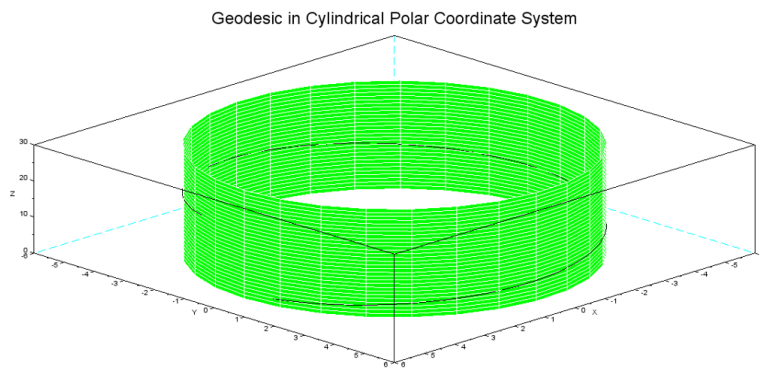


Figure 4.2: Geodesic

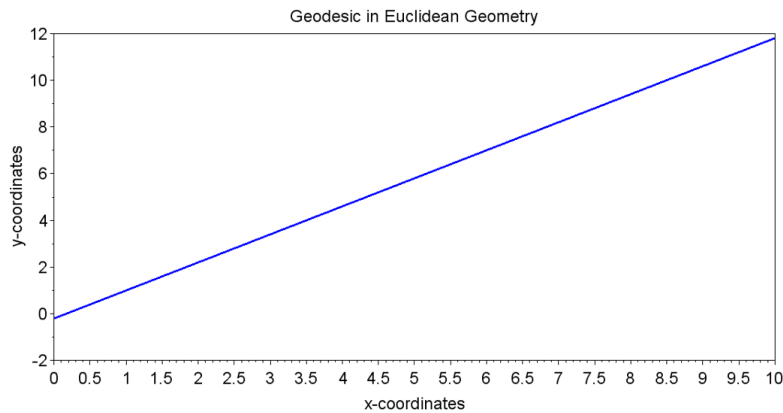


Figure 4.3: Geodesic

```

7 //
  // *****

8 ///Equation of Geodesic (straight line) passing
  through two points in Euclidean Geometry
9 //
  // *****

10 x1=input ("Input x coordinate of point A ")
11 y1=input ("Input y coordinate of point A ")
12 x2=input ("Input x coordinate of point B ")
13 y2=input ("Input y coordinate of point B ")
14 x=[0,0.1,10]
15 m=(y2-y1)/(x2-x1);
16 y=y1+m*(x-x1);
17 scf()
18 xlabel('x-coordinates','fontsize',5)
19 ylabel('y-coordinates','fontsize',5)
20 title('Geodesic in Euclidean Geometry','fontsize',5)
21 a=get("current_axes") //get the handle of the
  newly created axes
22 a.font_size=4
23 t=get("hdl") //get the handle of the

```

```

        newly created object
24 t.font_size=5
25 plot(x,y, 'linewidth',3)
26 //
    //*****

27 ///Plotting of cylinder
28 //
    //*****

29 a=5;
30 theta=linspace(0,2*pi,30)
31 z=linspace(0,30,30)
32 [theta,z]=meshgrid(theta,z)
33 x=a*cos(theta);
34 y=a*sin(theta);
35 scf()
36 surf(x,y,z, 'facecolor', 'green', 'edge', 'white')
37
38 //
    *****

39 //Equation of Geodesic (helix) in cylindrical
    Coordinate System
40 //
    *****

41 theta1=input ("Input angular coordinate (in degree)
    of point A ")
42 z1=input ("Input z coordinate of point A ")
43 theta2=input ("Input angular coordinate (in degree)
    of point B ")
44 z2=input ("Input z coordinate of point B ")
45 t1=theta1*pi/180;
46 t2=theta2*pi/180;
47 t=linspace(t1,t2,100)
48 z=z1+(z2-z1)*(t-t1)/(t2-t1);
49 title('Geodesic in Cylindrical Polar Coordinate

```

```
System', 'fontsize', 5)  
50 param3d(a*cos(t), a*sin(t), z)
```

Experiment: 5

Application to solve differential equations for a bound system – Eigen value problem

Scilab code Solution 5.0 Finite Difference Method

```
1 // Submitted by Dr. Triranjita Srivastava. Assistant
   Professor , Physics Dept., Kalindi College ,
   University of Delhi
2 //Operating system: Windows
3 //SCILAB Ver: 5.5.2
4
5 //Objective: Application to solve differential
   equations for a bound system – Eigenvalue Problem
6
7 // Example:Let us find out the energy eigenvalues
   and corresponding wavefunction of a particle of
   mass 'M' trapped in infinite potential Well (
   potential V=0) of width 'L'
8 //We implement Finite Difference Method (FDM) to
   obtain the eigenvalues
9 // By using FDM the second order differential
   operator is replaced by a trigonal matrix and
```

```

    the problem reduces to a simple eigenvalue
    problem
10
11
12 clc
13 clear
14 h_cut=1.05457*10^-34           //(Plancks
    constant/2pi) J-s
15 L=input("Enter the width of the potential well L (in
    m) = ")
16 M=input("Enter mass of particle M (in kg) = ")
17 n=250                          // Number of
    divisions for FDM
18 N=(2*n)+1
19 x1=0                            // Initial value
    of x-coordinate
20 s=(L-x1)/N                      // Step size for
    implementing FDM
21 EV=6.242*10^18                 // joule to eV
    conversion
22 //
    *****

23 // Hamiltonian Matrix H=T+V; T=Kinetic energy
    operator (-d^2/dx^2)*h_cut^2/2M); V= 0 (for
    infinite potential well)
24 //
    *****

25 T=zeros(N-1,N-1)
26 for i=1:(N-1)
27     x1=x1+s
28     T(i,i)=2
29     if (i<(N-1))
30         T(i,i+1)=-1
31         T(i+1,i)=-1
32 end
33 end

```

```

34
35 H=(T*h_cut^2*EV/(2*M*s^2)) //
    Hamiltonion Matrix
36
37 //
    *****

38 // Finding eigenvalues and corresponding
    wavefunctions
39 //
    *****

40 eigenvalues=spec(H)
41 disp("The eigenvalues (eV) of three lowest states
    obtained by FDM are ")
42 disp(eigenvalues(1:3))
43 [U,z]=spec(H)
44
45 //
    *****

46 // Plotiing of three lowest order wavefunctions
47 //
    *****

48 x=linspace(s,L,N-1) // creating
    x-coordinates for potential well
49 xlabel('x-coordinate (10^-10 m)', 'fontsize',5)
50 ylabel('Wavefunction (a.u.)', 'fontsize',5)
51 title('Graph of Wavefunction for three lowest order
    mode', 'fontsize',5)
52 a=get("current_axes") //get the handle of the
    newly created axes
53 a.font_size=2
54 t=get("hdl") //get the handle of the
    newly created object
55 t.font_size=5
56 plot(x*10^10,U(:,1)'/max(U(:,1)),'r','linewidth',3)

```

```

57 plot(x*10^10,U(:,2)'./max(U(:,2)),'b','linewidth',3)
58 plot(x*10^10,U(:,3)'./max(U(:,3)),'g','linewidth',3)
59 h1=legend(['Ground State';'I Excited State';'II
           Excited State'],5)
60 h1.font_size=2
61
62 //
           *****

63 // Comparison of obtained eigenvalues with
           analytical solution
64 //
           *****

65 disp("The eigenvalues (eV) of three lowest states
           obtained by analytical results are ")
66 for j=1:3
67     E(j)=j^2*%pi^2*h_cut^2*EV/(2*M*L^2)
68     disp (E(j))
69 end

```

```

Scilab 5.5.2 Console
File Edit Control Applications ?
Enter the width of the potential well L (in m) = 2*10^-10
Enter mass of particle M (in kg) = 9.1*10^-31

The eigenvalues (eV) of three lowest states obtained by FDM are

9.4111248
37.644129
84.697903

The eigenvalues (eV) of three lowest states obtained by analytical results are

9.4111556
37.644623
84.700401

-->|

```

Figure 5.1: Finite Difference Method

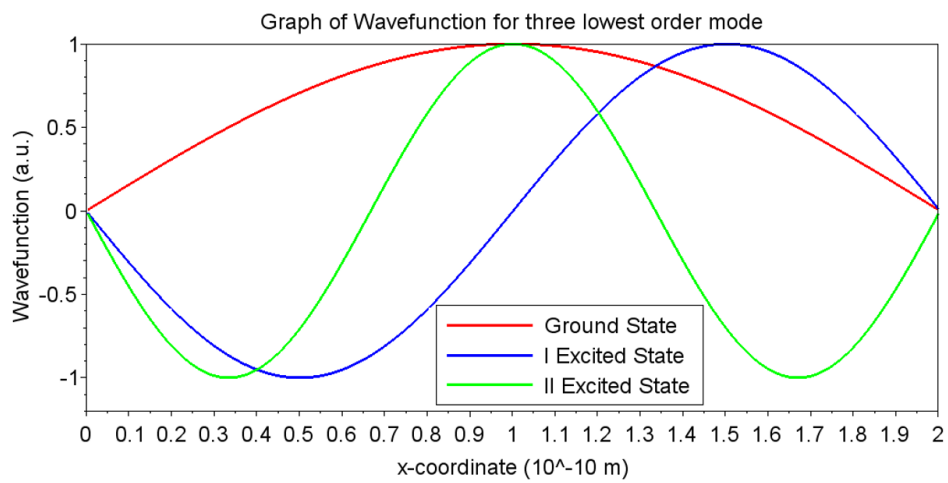


Figure 5.2: Finite Difference Method

Experiment: 6

Application to computer graphics: Write operators for shear, strain, 2D rotational problems, Reflection, Translation

Scilab code Solution 6.0 Computer Graphics

```
1 // Submitted by Dr. Triranjita Srivastava. Assistant
   // Professor , Physics Dept., Kalindi College ,
   // University of Delhi
2
3 //Operating system: Windows 8
4 //SCILAB Ver: 5.5.2
5
6 // Objective: To study computer graphics.
7 // One can create any object of choice and implement
   // various tranformations , like , Shear , Strain , 2D
   // rotation , Reflection , Translation ,
```

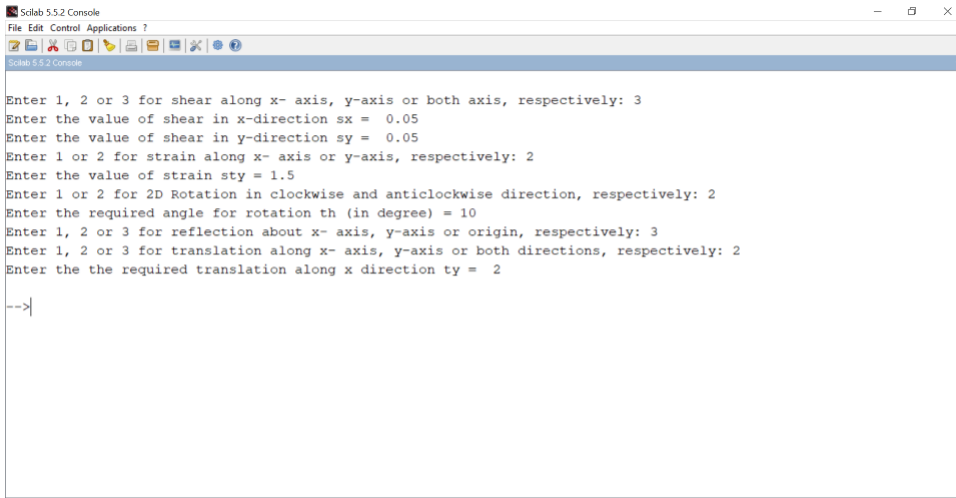


Figure 6.1: Computer Graphics

```

8
9  clc
10 clear
11 //
    *****

12 //Creation of an object (say, rectangle)
13 //
    *****

14 x=[0,5,5,0,0]
15 y=[0,0,3,3,0]
16 N=[x;y]
17
18 //
    *****

19 //To Study Shear
20 //
    *****

```

```

21 l=input("Enter 1, 2 or 3 for shear along x- axis , y-
    axis or both axis , respectively: ")
22
23 figure(1)
24 xlabel('x-coordinates (cm)', 'fontsize',5)
25 ylabel('y-coordinates (cm)', 'fontsize',5)
26 a=get("current_axes")           //get the handle of the
    newly created axes
27 a.font_size=4
28 t=get("hdl")                   //get the handle of the
    newly created object
29 t.font_size=5
30
31 select l
32     case 1
33         // Transformation Matrix for Shear parallel
            to x-axis
34         s=input("Enter the value of shear s = ")
35         Sx=[1 s; 0 1]
36         S=Sx*N
37         title('Shear parallel to x-axis', 'fontsize'
            ,5)
38         a.data_bounds=[0,0;8,5]
39     case 2
40         // Transformation Matrix for Shear parallel
            to y-axis
41         s=input("Enter the value of shear s = ")
42         Sy=[1 0; s 1]
43         S=Sy*N
44         title('Shear parallel to y-axis', 'fontsize'
            ,5)
45         a.data_bounds=[0,0;6,8]
46     case 3
47         // Transformation Matrix for Shear in x and
            y-direction
48         sx=input("Enter the value of shear in x-
            direction sx = ")
49         sy=input("Enter the value of shear in y-

```

```

        direction sy = ")
50     Sxy=[1 sx; sy 1]
51     S=Sxy*N
52     title('Shear in x and y direction','fontsize
        ',5)
53     a.data_bounds=[0,0;6,8]
54 end
55 plot(x,y,'linewidth',3)
56 plot(S(1,:),S(2:,:),'—r','linewidth',3)
57 hl=legend(['old coordinates';'new coordinates'])
58 hl.font_size=3
59
60
61 //
        *****

62 //To Study Strain
63 //
        *****

64
65 p=input("Enter 1 or 2 for strain along x- axis or y-
        axis , respectively: ")
66
67 figure(2)
68 xlabel('x-coordinates (cm)','fontsize',5)
69 ylabel('y-coordinates (cm)','fontsize',5)
70 a=get("current_axes"); //get the handle of the
        newly created axes
71 a.font_size=4
72 t=get("hdl") //get the handle of the newly created
        object
73 t.font_size=5;
74
75 select p
76     case 1
77 // Transformation Matrix for Strain along x-
        axis

```

```

78         stx=input("Enter the value of strain stx = "
79             )
80         Str_x=[stx 0; 0 1]
81         ST=Str_x*N;
82         title('Strain along x-axis','fontsize',5);
83         a.data_bounds=[0,0;8,5];
84     case 2
85 //         Transformation Matrix for strain along y-
86 axis
87         sty=input("Enter the value of strain sty = "
88             )
89         Str_y=[1 0; 0 sty]
90         ST=Str_y*N;
91         title('Strain along y-axis','fontsize',5);
92         a.data_bounds=[0,0;6,8];
93     end
94     plot(x,y,'linewidth',3);
95     plot(ST(1,:),ST(2,:),'—r','linewidth',3)
96     hl=legend(['old coordinates';'new coordinates']);
97     hl.font_size=3
98 //
99 //*****
100 //To Study 2D Rotation
101 //*****
102 k=input("Enter 1 or 2 for 2D Rotation in clockwise
103 and anticlockwise direction , respectively: ")
104 th=input("Enter the required angle for rotation th (
105 in degree) = ")
106 figure(3)
107 xlabel('x-coordinates (cm)','fontsize',5)
108 ylabel('y-coordinates (cm)','fontsize',5)

```

```

107 a=get("current_axes");//get the handle of the newly
    created axes
108 a.font_size=4
109 t=get("hdl") //get the handle of the newly created
    object
110 t.font_size=5;
111
112 select k
113     case 1
114         // Transformation Matrix for Rotation in
            clockwise direction
115         Cl=[cosd(th),sind(th);-sind(th),cosd(th)]
116         Rot=Cl*N;
117         title('Rotation in clockwise direction ','
            fontsize ',5);
118         a.data_bounds=[0,0;8,5];
119     case 2
120         // Transformation Matrix for Rotation in
            anticlockwise direction
121         Anti=[cosd(th),-sind(th);sind(th),cosd(th)]
122         Rot=Anti*N;
123         title('Rotation in anticlockwise direction ',
            'fontsize ',5);
124         a.data_bounds=[0,0;6,8];
125 end
126 plot(x,y,'linewidth ',3);
127 plot(Rot(1,:),Rot(2,:),'—r','linewidth ',3)
128 hl=legend(['old coordinates';'new coordinates']);
129 hl.font_size=3
130
131 //
            *****

132 //To Study the reflection
133 //
            *****

134 j=input("Enter 1, 2 or 3 for reflection about x-

```

```

        axis , y-axis or origin , respectively: ")
135
136 figure(4)
137 xlabel('x-coordinates (cm)', 'fontsize',5)
138 ylabel('y-coordinates (cm)', 'fontsize',5)
139
140 a=get("current_axes");//get the handle of the newly
    created axes
141 a.font_size=4
142 t=get("hdl") //get the handle of the newly created
    object
143 t.font_size=5;
144 select j
145     case 1
146         // Transformation Matrix for Reflection about x-
            axis
147             Rx=[1 0; 0 -1]
148             R=Rx*N;
149             title('Reflection about x-axis', 'fontsize'
                ,5);
150             a.data_bounds=[0,-4;6,4];
151     case 2
152         // Transformation Matrix for Reflection about y-
            axis
153             Ry=[-1 0; 0 1]
154             R=Ry*N;
155             title('Reflection about y-axis', 'fontsize'
                ,5);
156             a.data_bounds=[0,0;8,4];
157     case 3
158         //Transformation Matrix for Reflection about
            origin
159             Rxy=[-1 0; 0 -1]
160             R=Rxy*N;
161             title('Reflection about origin', 'fontsize'
                ,5);
162             a.data_bounds=[-8,-5;8,5];
163 end

```

```

164
165 plot(x,y,'linewidth',3);
166 plot(R(1,:),R(2,:),'—r','linewidth',3)
167 hl=legend(['old coordinates';'new coordinates']);
168 hl.font_size=5
169
170 //
        *****

171 //To Study translation
172 //
        *****

173 i=input("Enter 1, 2 or 3 for translation along x-
        axis, y-axis or both directions, respectively: ")
174
175 figure(5)
176 xlabel('x-coordinates (cm)','fontsize',5)
177 ylabel('y-coordinates (cm)','fontsize',5)
178 a=get("current_axes"); //get the handle of the
        newly created axes
179 a.font_size=4
180 t=get("hdl") //get the handle of the newly created
        object
181 t.font_size=5;
182
183 select i
184     case 1
185         // Transformation Matrix for translation
            along to x-axis
186         tx=input("Enter the required translation
            along x direction tx = ")
187         T1=[ones(1,length(x));zeros(1,length(x))];
188         X=N+tx*T1;
189         title('Translation along to x-axis',
            'fontsize',5);
190         a.data_bounds=[0,0;8,5];
191     case 2

```

```

192         // Transformation Matrix for translation
           along to y-axis
193     ty=input("Enter the the required translation
              along x direction ty = ")
194     T1=[zeros(1,length(x));ones(1,length(x))];
195     X=N+ty*T1;
196     title('Translation along to y-axis',
           fontsize',5);
197     a.data_bounds=[0,0;6,8];
198     case 3
199         // Transformation Matrix for translation
           along to y-axis
200     tx=input("Enter the required translation
              along x direction tx = ")
201     ty=input("Enter the required translation
              along y direction ty = ")
202     T1=[ones(1,length(x));zeros(1,length(x))];
203     T2=[zeros(1,length(x));ones(1,length(x))];
204     X=N+tx*T1+ty*T2;
205     title('Translation along to y-axis',
           fontsize',5);
206     a.data_bounds=[0,0;6,8];
207     end
208     plot(x,y,'linewidth',3);
209     plot(X(1,:),X(2:,:),'—r','linewidth',3)
210     h1=legend(['old coordinates';'new coordinates']);
211     h1.font_size=3

```

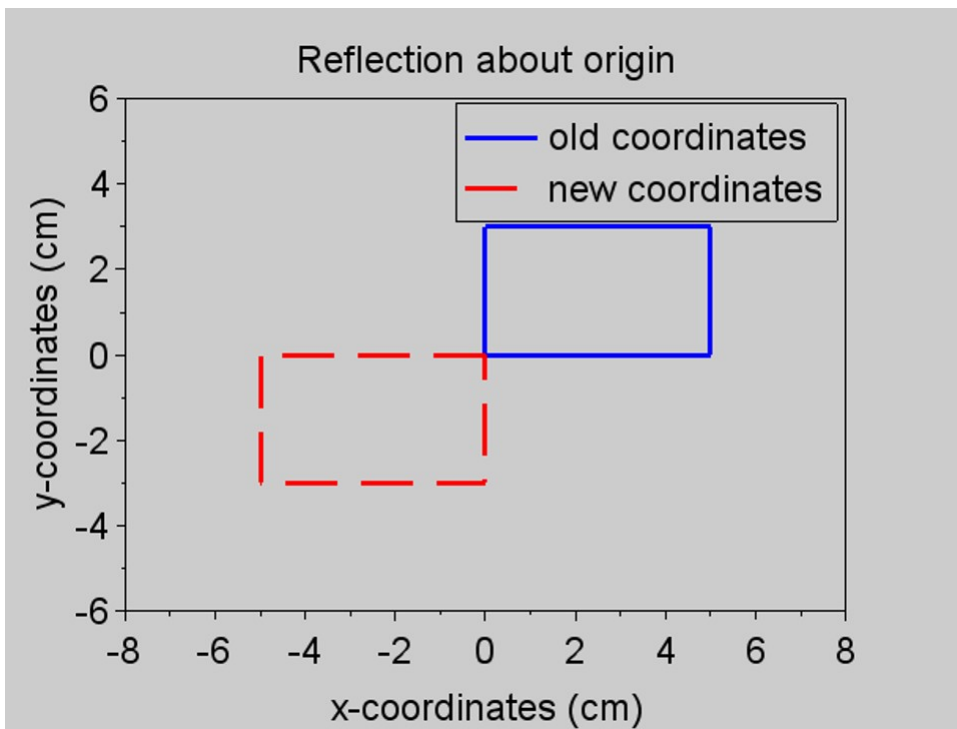


Figure 6.2: Computer Graphics

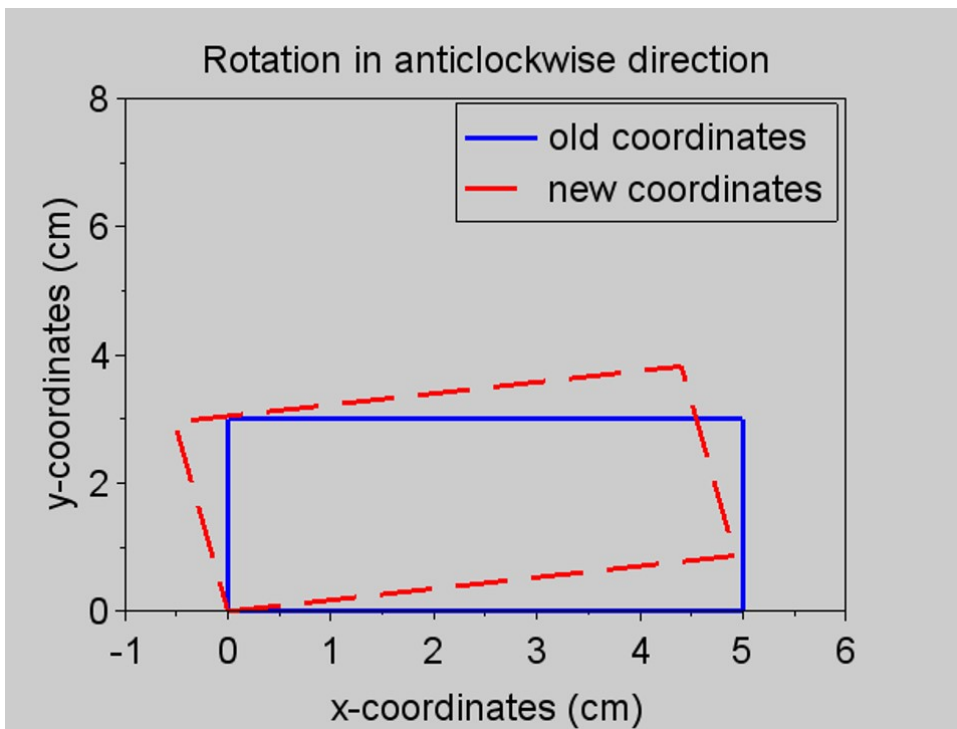


Figure 6.3: Computer Graphics

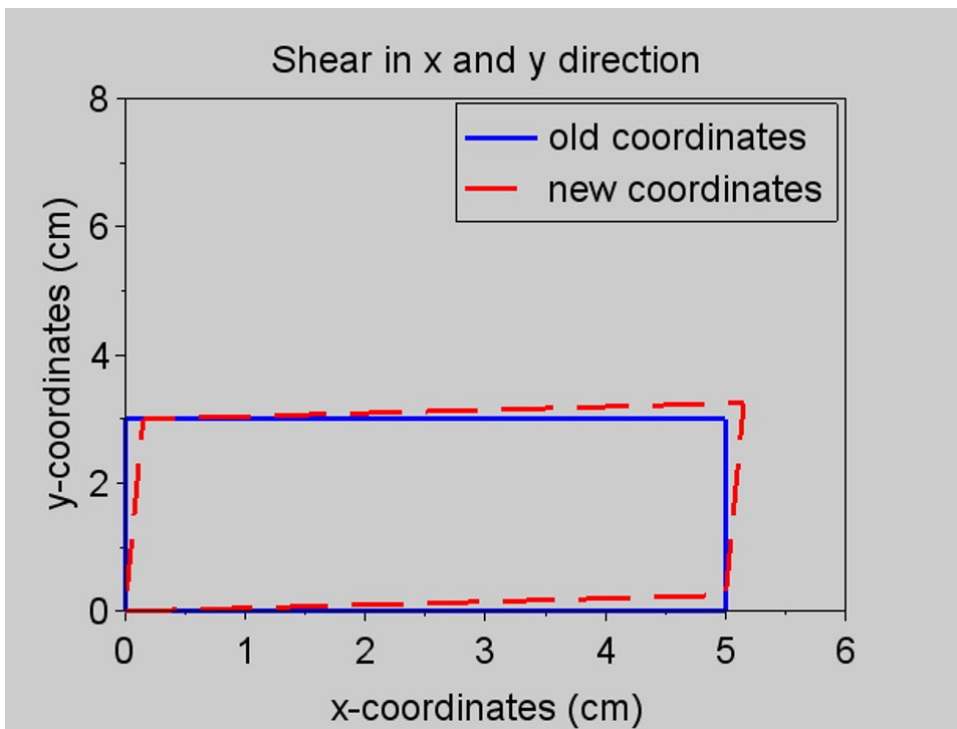


Figure 6.4: Computer Graphics

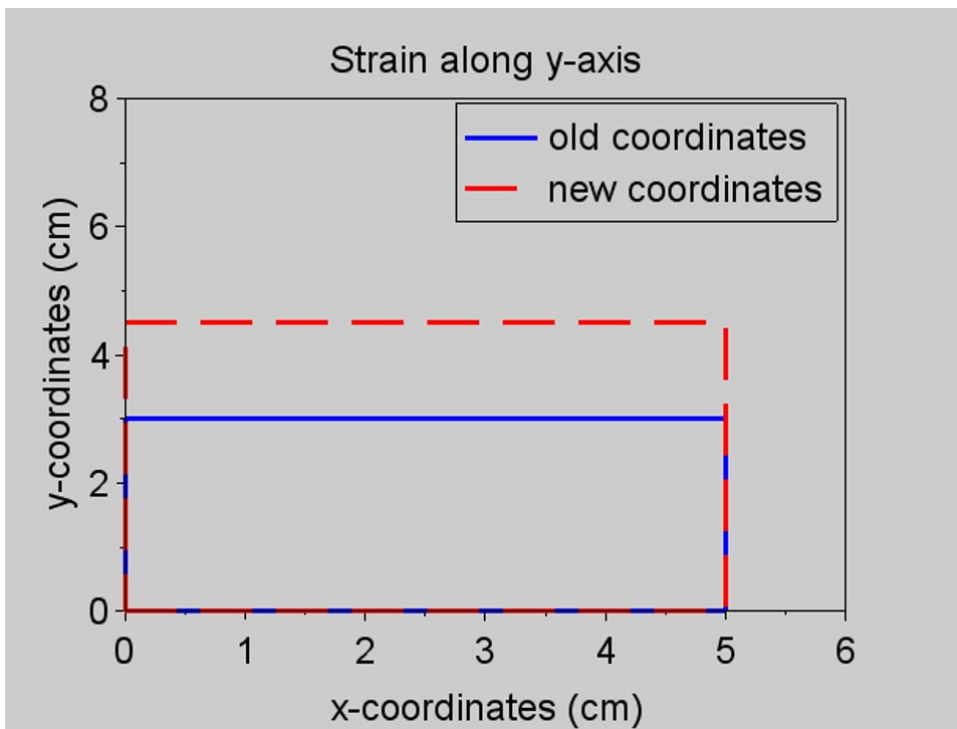


Figure 6.5: Computer Graphics

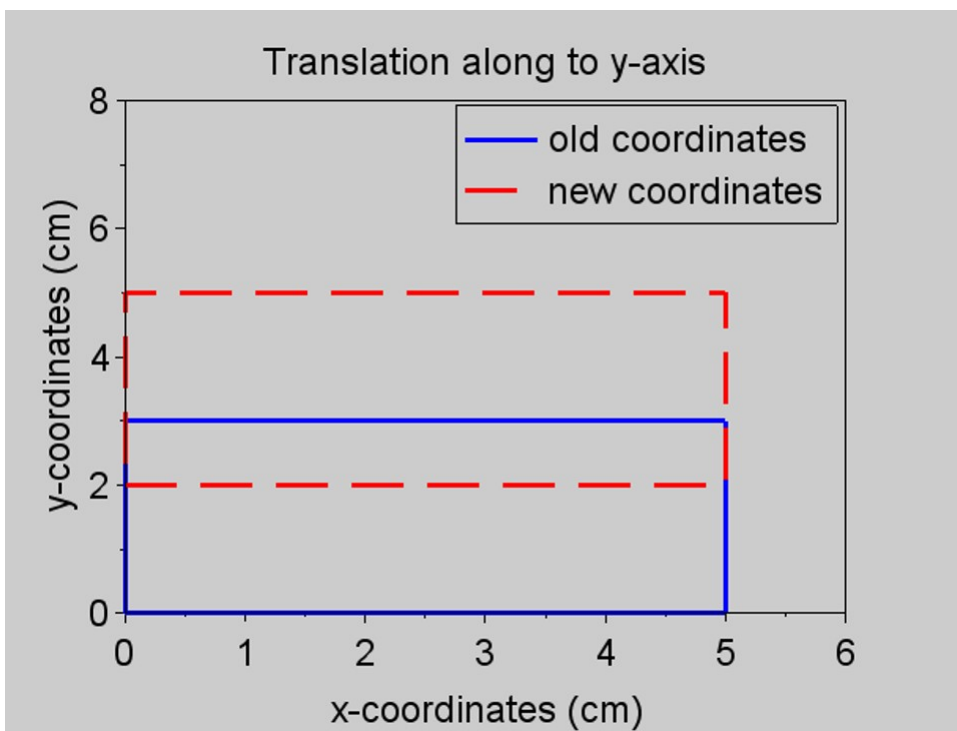


Figure 6.6: Computer Graphics

Experiment: 7

Lagrangian formulation in classical mechanics with constraints.

Scilab code Solution 7.0 Lagrangian Formulation

```
1 // Submitted by Dr. Triranjita Srivastava. Assistant
   Professor , Physics Dept., Kalindi College ,
   University of Delhi
2
3 //Operating system: Windows 8
4 //SCILAB Ver: 5.5.2
5 //Objectiv: Lagrangian formulation in classical
   mechanics with constraints
6 //Example: Simple Pendulum of length L (m)operating
   in gravitational field . After applying
   Lagrangian formulation this problem reduces to a
   simple second order differential equation  $[(d^2$ 
   theta/dt2)+(g/L)sin(theta)]=0. Here theta is
   angular displacement.
7 // We implemented ordinary differential equation (
   ODE) Solver to solve the second order
   differential equation
```

```

8 //We present plot of solution of angular
  displacement for t=0 to t=10 seconds
9
10 clear
11 clc
12 L=input ('Enter the length of pendulum (m) L = ')
13 g = 9.8 //acceleration due to
  gravity (m/s^2)
14 k=g/L
15 theta=input('Enter the initial angular displacement
  (radian) at (t = 0) = '); // Initial
  angular displacement at t = 0
16 dt=input('Enter initial d_theta/dt (radian) at (t =
  0) = '); // Initial boundary condition
  d_theta/dt at t = 0
17 //
  //*****
18 /// Function declaration for ODE
19 //
  //*****
20 t=linspace(0,10,200)
21 function dx=f(t,x,k)
22     dx(1)=x(2)
23     dx(2)=-k*sin(x(1))
24 endfunction
25 //
  //*****
26 /// Solving second order differential equation by
  ODE solver
27 //
  //*****
28     y=ode([theta;dt],0,t,f)
29     ysol=y(1,:)
30     ydotsol = y(2,:)

```

```

31
32 //
    //*****
33 //// Plotting the solution (angular displacement (
    theta)and d_theta/dt)
34 //
    //*****

35 scf()
36 title('Solution of Simple Pendulum', 'fontsize',5)
37 ylabel('Solution —>', 'fontsize',5)
38 xlabel('t (sec) —>', 'fontsize',5)
39 a=get("current_axes") //get the handle of the
    newly created axes
40 a.font_size=4
41 t=get("hdl") //get the handle of the
    newly created object
42 t.font_size=5
43 plot(t,ysol,'r','linewidth',3)
44 plot(t,ydotsol,'k','linewidth',3)
45 h1 = legend(['$\theta$'; '$d\theta/dt$'])
46 h1.font_size=3

```

```
Scilab 5.5.2 Console
Enter the length of pendulum (m) L = 1
Enter the initial angular displacement (radian) at (t = 0) = %pi/10
Enter initial d_theta/dt (radian) at (t = 0) = 0
-->
```

Figure 7.1: Lagrangian Formulation

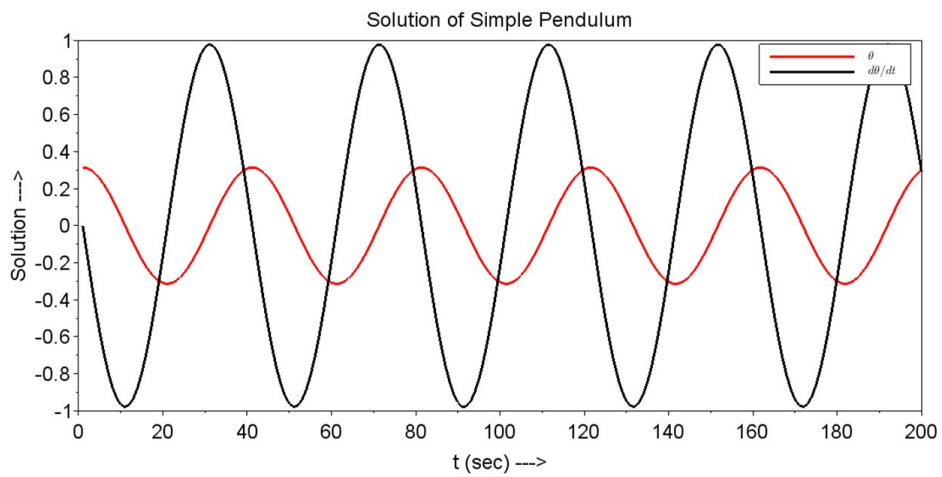


Figure 7.2: Lagrangian Formulation

Experiment: 8

Vector-space of wave functions in Quantum-Mech: Position and Momentum differential operators and their commutator, wave function

Scilab code Solution 8.0 Hermitian Differential Op

```
1 // Submitted by Dr. Triranjita Srivastava. Assistant
  Professor , Physics Dept., Kalindi College ,
  University of Delhi
2
3 // Aim: To show the commutator relation in position
  and momentum space  $[x,p]=ih\_cut$  or n general  $[x^n
  ,p]=i*h\_cut*n*x^{(n-1)}$ 
4 // Two examples are shown in this program
5 //1. Let the first function is  $fx=x$ 
6 //2. Let the second function is  $fx=x^3$ 
7 // For simplicity let the wavefunction  $A=x$ 
8 //  $[fx ,p]=(ih\_cut)(dfx/dx)$ 
9 //  $h\_cut=h/2\pi$ ; h is planck constant
```

```

10
11
12 clc
13 x=poly(0,"x")
14 h_cut=1.05*(10)^-34 //h_cut=h/2pi,
    units is in Joule-sec
15 A=x //Considered
    Wavefunction is A=x
16
17 s=input("Enter 1 or 2 to choose the function as fx =
    x or fx = x^3, s= ")
18 select s
19 case 1
20     fx=x //First
        wavefunction
21 case 2
22     fx=x^3 //Second wavefunction
23 end
24
25 fx_p=fx*(-%i*h_cut)*derivat(A)
26 p_fx=(-%i*h_cut)*derivat(fx*A)
27 commutator=(fx_p-p_fx)
28 disp(" [fx ,p] = ")
29 disp(commutator)
30 disp ("The result contains an extra x because the
    chosen wavefunction is A = x")

```

```
Scilab 5.5.2 Console
Enter 1 or 2 to choose the function as fx = x or fx = x^3, s= 1
[fx,p] =
Real part
0
Imaginary part
1.050D-34x
The result contains an extra x because the chosen wavefunction is A = x
-->|
```

Figure 8.1: Hermitian Differential Op

```
Scilab 5.5.2 Console
Enter 1 or 2 to choose the function as fx = x or fx = x^3, s= 2
[fx,p] =
Real part
0
Imaginary part
3.150D-34x3
The result contains an extra x because the chosen wavefunction is A = x
-->
```

Figure 8.2: Hermitian Differential Op