Scilab Manual for
Advanced Mathematical Physics-I
by Dr Triranjita Srivastava
Physics
Kalindi College, University Of Delhi[1]

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

November 21, 2024

[1]Funded 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 Experiments

List of Figures

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

```
//Operating system: Windows 8
//SCILAB Ver: 5.5.2
//Expriment No. 1
//Objective: Determination of largest and smallest (in magnitude) Eigen value &
//Eigen Vectors Using Power Method and Inverse Power Method respectively.


//Enter the no dimension of a sqaure of matrix A: 3
//Enter the element no (1,1):2
```

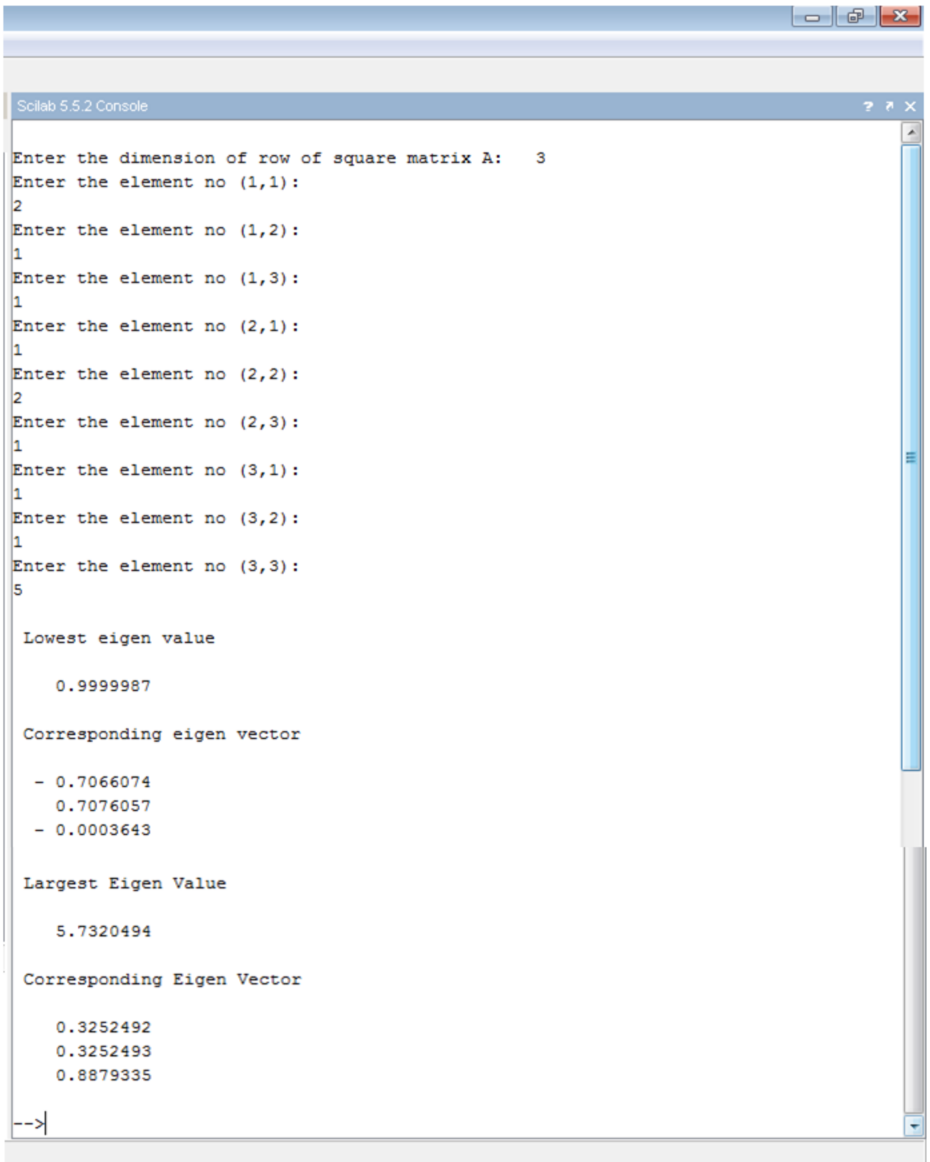

Figure 1.1: Power and Inverse Power Method

```
//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
//Let Matrix A is A=[2,1,1;1,2,1;1,1,5];

clc
clear
//***************************************************************
// Creating an input square matrix
//***************************************************************
m = input("Enter the dimension of row of square matrix A:   ")

for i=1:m
    for j=1:m
        mprintf("Enter the element no (%d,%d):   ",i,j)
        A(i,j)=input("")
    end
end

//***************************************************************
// Creating initial approximation x0
//***************************************************************
x=rand(m,1)
```

```

//***************************************************************
//Finding smallest Eigen Value using Inverse Power Method
//***************************************************************
z=1
f=1
y0=rand(m,1)
while(f>0.00001)
  y1=inv(A)*y0
  lowest=norm(y1,2)
  y0=y1/lowest
  f=abs(z-lowest)
  z=lowest
end

disp(lowest,'Lowest eigen value')
disp(y0,'Corresponding eigen vector')

//***************************************************************
//Finding largest Eigen Value using Power Method
//***************************************************************
x0=rand(m,1)
y=1
d=1
while (d>0.00001)
  x1=A*x0
  highest=norm(x1,2)
  x0=x1/highest
  d=abs(y-highest)
```

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

Experiment: 2

Orthogonal Polynomials as Eigenfunctions of Hermitian differential operators

Scilab code Solution 2.0 Finite Difference Method

```
// Submitted by Dr. Triranjita Srivastava. Assistant
    Professor, Physics Dept., Kalindi College,
   University of Delhi

// Aim: To prove the orthogonality of Hermitian
   differential Operator

// Two Hermitian Differential Operators (-id/dx) and
    (-d^2/dx^2) are taken as an example

// Finite Difference Method is used to formulate the
    matrices coresponding to the considered
   Differential Operator

// This method takes the value of eigenfunction
```

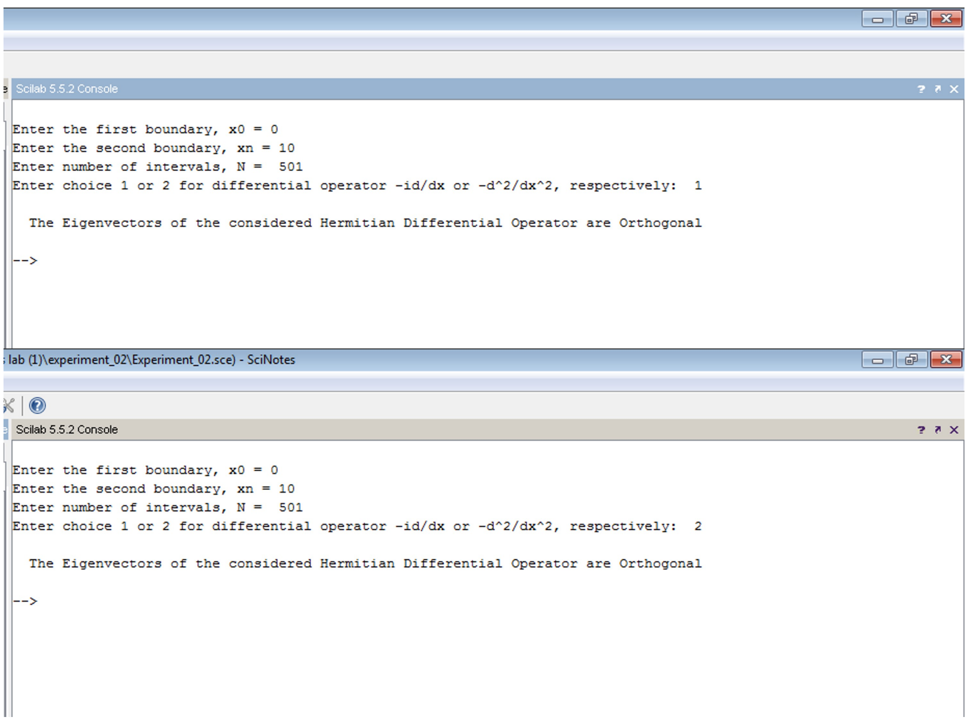

Figure 2.1: Finite Difference Method

```
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)
// N is the number of interval, N should be taken as odd and such that step size is small enough for high accuracy

// 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;

//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



clear
clc
//*******************************************************************
// Boundary over which the function is to be solved
//*******************************************************************
x0=input("Enter the first boundary, x0 = ");
xn=input("Enter the second boundary, xn = ");
N = input("Enter number of intervals, N =  ");
h = (xn-x0)/N;     //step size
s=input("Enter choice 1 or 2 for differential operator -id/dx or -d^2/dx^2, respectively:  ")

select s
case 1
    //
```

```
    ************************************************************
    // Defining D1 Matrix corresponding to differential operator -id/dx
    //************************************************************
    D1=zeros(N-1,N-1);

    for i=1:(N-1)
        x1(1,i)=x0+i*h;
        D1(i,i)=0;
        if i<(N-1)
            D1(i,i+1)=-%i;
            D1(i+1,i)=%i;
        end
    end
    Final_D1=D1/2*h;

    //************************************************************
    // Finding eigenvalue and eigenvector of differential operator -id/dx
    //************************************************************
    [eigenvector,eigenvalue] = spec(Final_D1);

case 2
    //************************************************************
    // Defining D2 Matrix corresponding to differential operator -d^2/dx^2
    //************************************************************
```

```

    D2=zeros(N-1,N-1);

    for i=1:(N-1)
        x1(1,i)=x0+i*h;
        D2(i,i)=2;
        if i<(N-1)
            D2(i,i+1)=-1;
            D2(i+1,i)=-1;
        end
    end
    Final_D2=D2/h^2;

    //*************************************************************
    // Finding eigenvalue and eigenvector of differential operator -d^2/dx^2
    //*************************************************************
    [eigenvector,eigenvalue] = spec(Final_D2);
end

    //*************************************************************
    // Ploting of first three Eigenvector of differential operator -d^2/dx^2
    //*************************************************************

    x=[x0,x1,xn];

    if s==1 then
        title('3 Lowest Order Eigenvectors of -id/dx','fontsize',4);
```

```
else
    title('3 Lowest Order Eigenvectors of -d^2/dx^2','fontsize',4);
end

for k = 1:3
    subplot(3,1,k)
    ylabel('A (m)','fontsize',4)
    a=get("current_axes");//get the handle of the newly created axes
    a.font_size=2
    t=get("hdl")    //get the handle of the newly created object
    t.font_size=2;
    E_vector = [0;eigenvector(:,k);0];
    plot(x,E_vector','linewidth',2);
end
        xlabel('x-coordinate (m)','fontsize',4)
//***************************************************
// Orthogonality Check of eigenvector of differential operator
//***************************************************
for i=1:3
    for j=1:3
        P(i,j) = clean(sum((eigenvector(:,i).*conj(eigenvector(:,j)))));
        if i~=j & P(i,j) ~=0
            disp(" The Eigenvectors of the considered Hermitian Differential Operator are Not Orthogonal")
            abort;
        end
    end
end
```

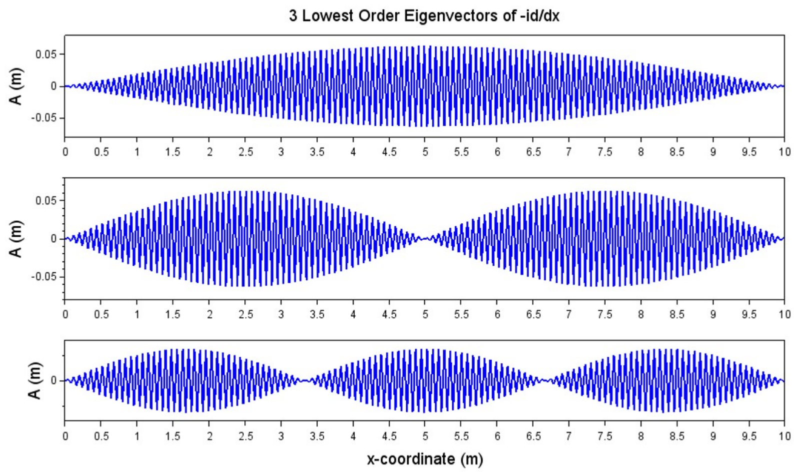

Figure 2.2: Finite Difference Method

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

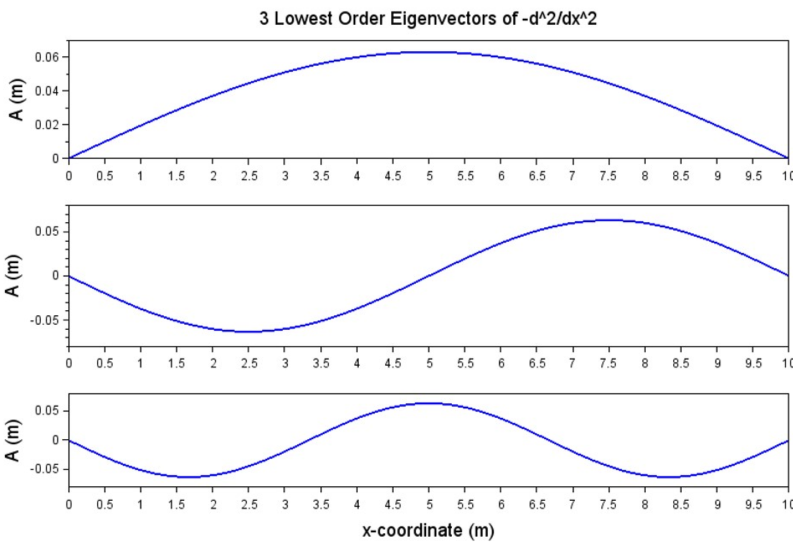

Figure 2.3: Finite Difference Method

Experiment: 3

Determination of the principal axes of moment of inertia through diagonalization

Scilab code Solution 3.0 Diagonalization of matrix

```
// Submitted by Dr. Triranjita Srivastava. Assistant
    Professor, Physics Dept., Kalindi College,
   University of Delhi

// Aim: Determination of the principal axes of
   moment of inertia through diagonalization
// Example is a Dumbell with masses 'm1' and 'm2'
   situated at points, say coordinates are (1,1,0)
   and (-1,-1,0)

clear;
clc;
//
   ************************************************************

//Function for Kronecker Delta
//
```

```
************************************************************

function d=delta(i,j)
    if i==j then
              d=1;
        else d=0;
    end
endfunction

//************************************************************

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

n=input('enter no. of particles  ')

r=zeros(3,3)
for i=1:n
    mprintf("Enter the mass (in kg) at point (%d): ",i)
    M(i)=input("")
    mprintf("Enter the position (x,y,z) coordinate at point (%d):  ",i)
    for j =1:3
        r(i,j)=input("")
end
end

I=zeros(3,3)
for i=1:1:3
    for j=1:1:3
        for k=1:1:n
            I(i,j)=I(i,j)+(M(k)*(sum(r(k,:).^2)*delta(i,j)-(r(k,i).*r(k,j))))
        end
    end
```

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

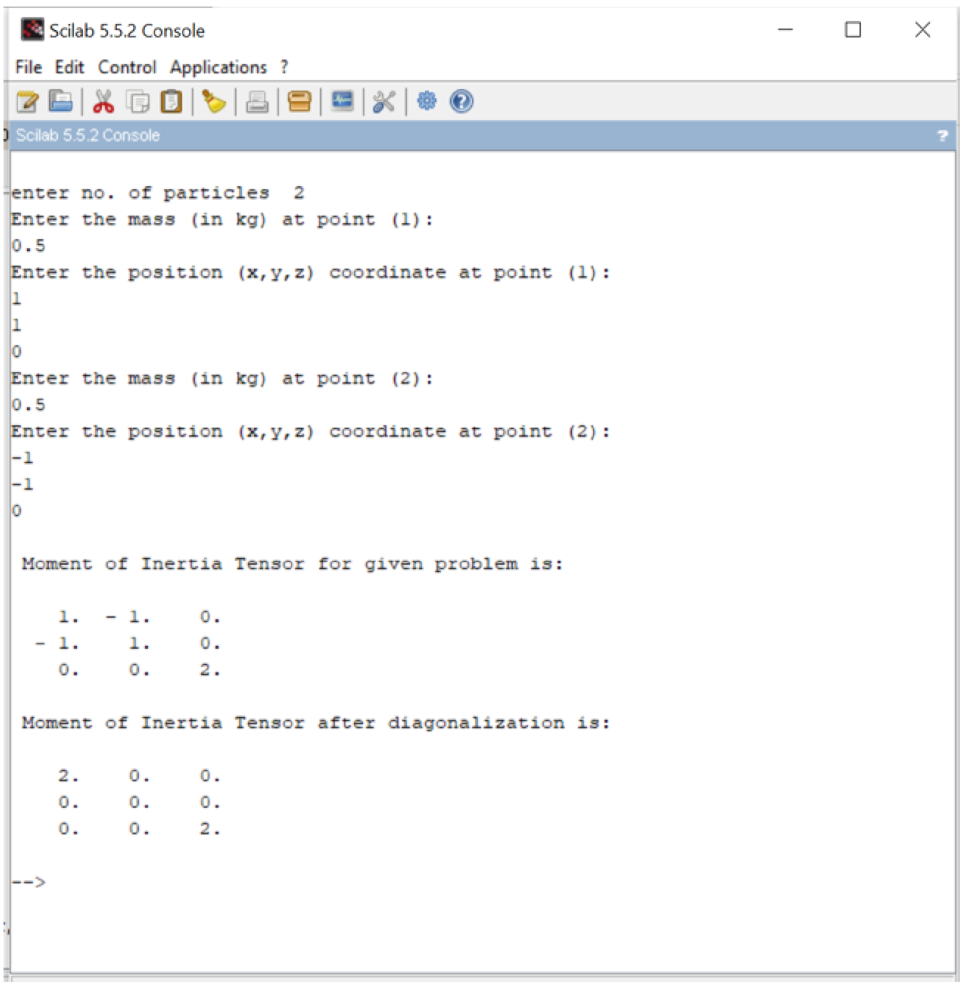

Figure 3.1: Diagonalization of matrix

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

```
// Submitted by Dr. Triranjita Srivastava. Assistant
    Professor, Physics Dept., Kalindi College,
    University of Delhi

// Aim: To study geodesics in Euclidean and
    Cylindrical Polar coordinate System

clc;
clear;
```

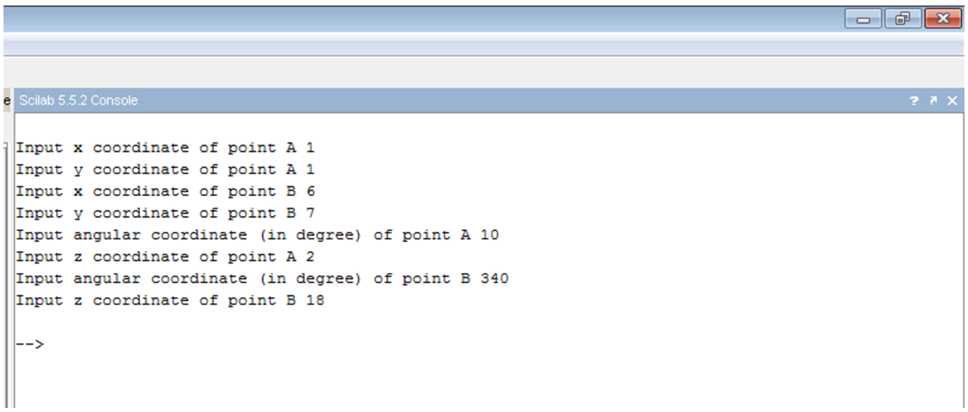

Figure 4.1: Geodesic

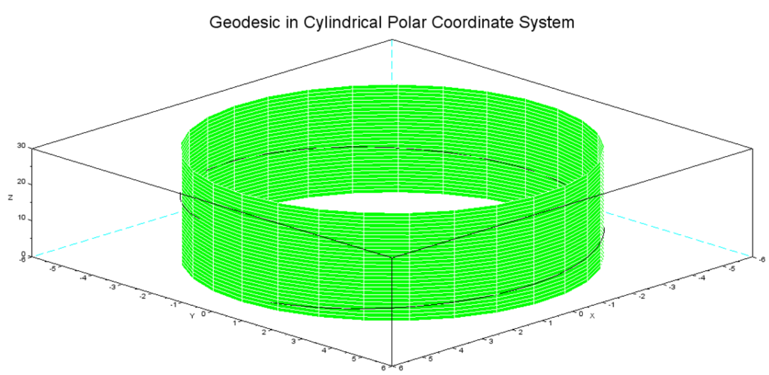

Figure 4.2: Geodesic

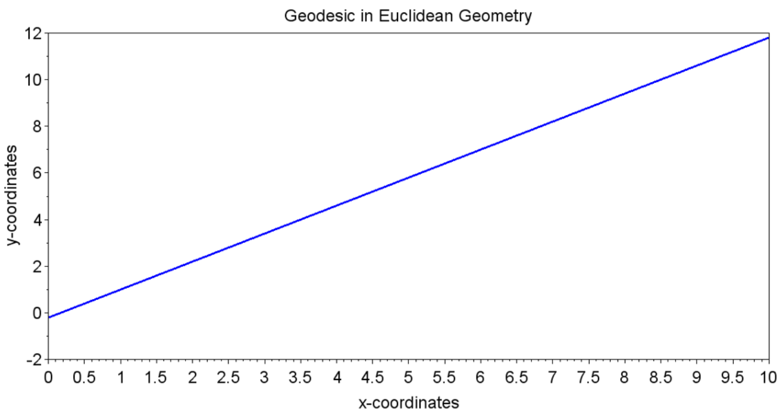

Figure 4.3: Geodesic

```
////***************************************************************
////Equation of Geodesic (straight line) passing through two points in Euclidean Geometry
////***************************************************************
x1=input ("Input x coordinate of point A ")
y1=input ("Input y coordinate of point A ")
x2=input ("Input x coordinate of point B ")
y2=input ("Input y coordinate of point B ")
x=[0,0.1,10]
m=(y2-y1)/(x2-x1);
y=y1+m*(x-x1);
scf()
xlabel('x-coordinates','fontsize',5)
ylabel('y-coordinates','fontsize',5)
title('Geodesic in Euclidean Geometry','fontsize',5)
a=get("current_axes")        //get the handle of the newly created axes
a.font_size=4
t=get("hdl")                 //get the handle of the
```

```
newly created object
t.font_size=5
plot(x,y,'linewidth',3)
////************************************************************
////Plotting of cylinder
////************************************************************
a=5;
theta=linspace(0,2*%pi,30)
z=linspace(0,30,30)
[theta,z]=meshgrid(theta,z)
x=a*cos(theta);
y=a*sin(theta);
scf()
surf(x,y,z,'facecolor','green','edge','white')

//************************************************************
//Equation of Geodesic (helix) in cylindrical Coordinate System
//************************************************************
theta1=input("Input angular coordinate (in degree) of point A ")
z1=input("Input z coordinate of point A ")
theta2=input("Input angular coordinate (in degree) of point B ")
z2=input("Input z coordinate of point B ")
t1=theta1*%pi/180;
t2=theta2*%pi/180;
t=linspace(t1,t2,100)
z=z1+(z2-z1)*(t-t1)/(t2-t1);
title('Geodesic in Cylindrical Polar Coordinate
```

```
    System','fontsize',5)
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

```
// Submitted by Dr. Triranjita Srivastava. Assistant Professor, Physics Dept., Kalindi College, University of Delhi
//Operating system: Windows
//SCILAB Ver: 5.5.2

//Objective: Application to solve differential equations for a bound system − Eigenvalue Problem

// 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'
//We implement Finite Difference Method (FDM) to obtain the eigenvalues
// By using FDM the second order differential operator  is replaced by a trigonal matrix and
```

```
the problem reduces to a simple eigenvlaue problem


clc
clear
h_cut=1.05457*10^-34                    //(Plancks constant/2pi) J-s
L=input("Enter the width of the potential well L (in m) = ")
M=input("Enter mass of particle M (in kg) = ")
n=250                                   // Number of divisions for FDM
N=(2*n)+1
x1=0                                    // Initial value of x-coordinate
s=(L-x1)/N                              // Step size for implementing FDM
EV=6.242*10^18                          // joule to eV conversion
//**********************************************************************
// Hamiltonion Matrix H=T+V;  T=Kinetic energy operator (-d^2/dx^2)*h_cut^2/2M);   V= 0 (for infinite potential well)
//**********************************************************************
T=zeros(N-1,N-1)
for i=1:(N-1)
    x1=x1+s
    T(i,i)=2
    if (i<(N-1))
        T(i,i+1)=-1
        T(i+1,i)=-1
end
end
```

```

H=(T*h_cut^2*EV/(2*M*s^2))                        // Hamiltonion Matrix

//
***********************************************************

// Finding eigenvalues and corresponding wavefunctions
//
***********************************************************

eigenvalues=spec(H)
disp("The eigenvalues (eV) of three lowest states obtained by FDM are " )
disp(eigenvalues(1:3))
[U,z]=spec(H)

//
***********************************************************

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

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

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

//***********************************************************************************
// Comparison of obtained eigenvalues with analytical solution
//***********************************************************************************
disp("The eigenvalues (eV) of three lowest states obtained by analytical results are " )
for j=1:3
    E(j)=j^2*%pi^2*h_cut^2*EV/(2*M*L^2)
    disp (E(j))
end
```

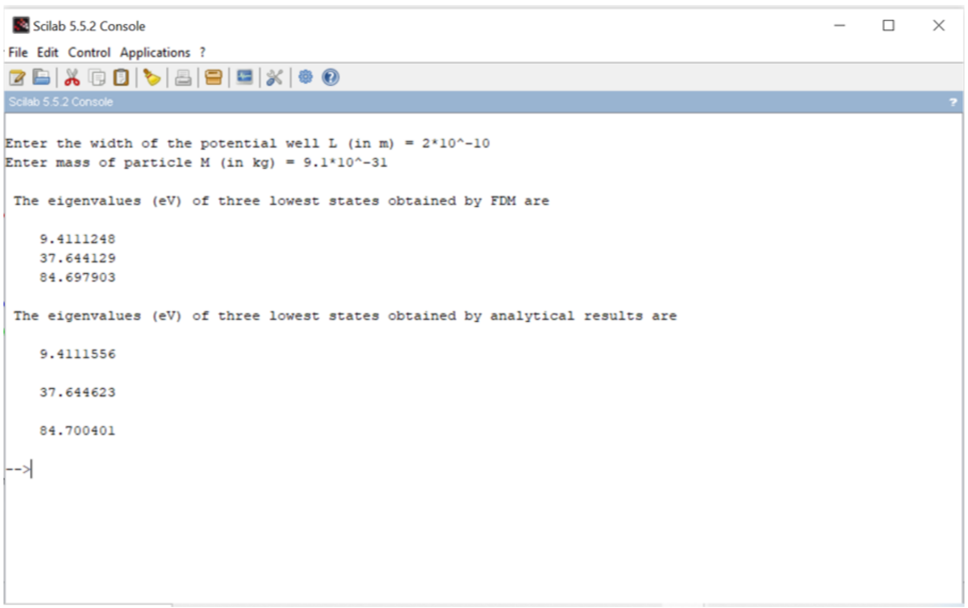

Figure 5.1: Finite Difference Method

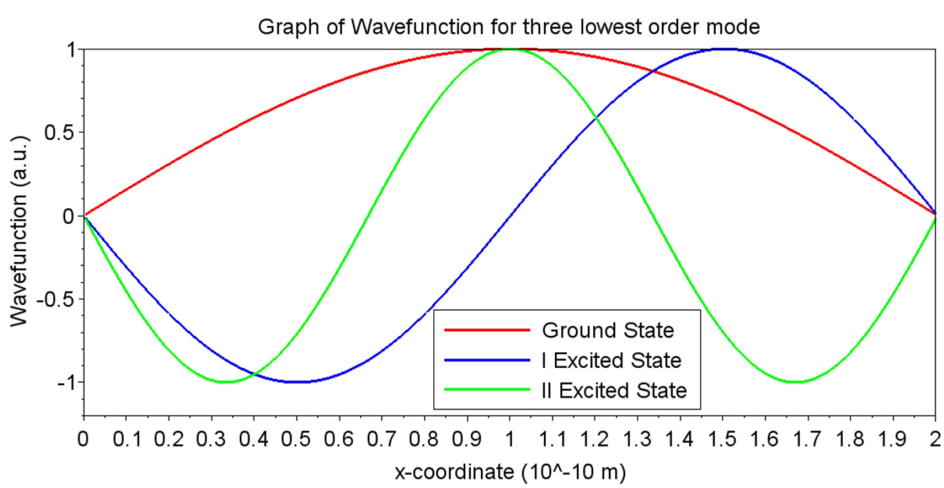

Figure 5.2: Finite Difference Method

Experiment: 6

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

Scilab code Solution 6.0 Computer Graphics

```
// Submitted by Dr. Triranjita Srivastava. Assistant
    Professor, Physics Dept., Kalindi College,
   University of Delhi

//Operating system: Windows 8
//SCILAB Ver: 5.5.2

// Objective: To study computer graphics.
// One can create any object of choice and implement
    various tranformations, like, Shear, Strain, 2D
   rotation, Reflection, Translation,
```

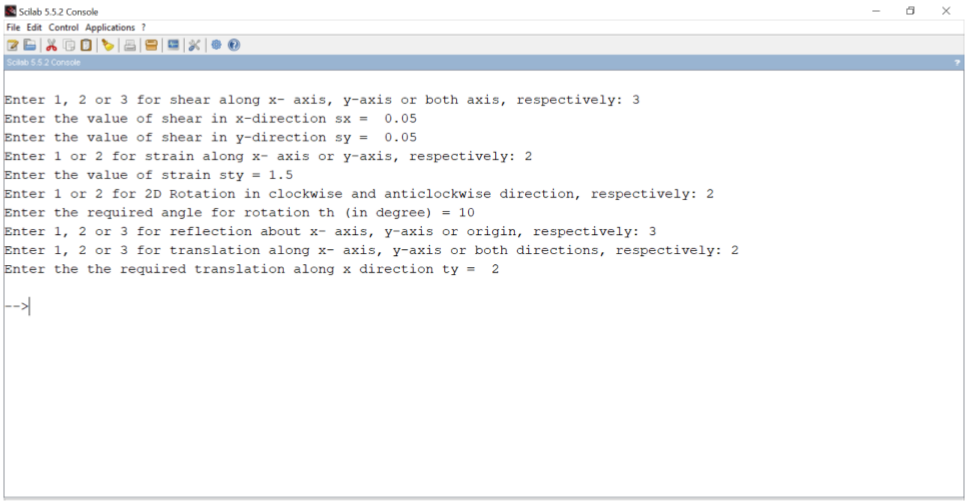

Figure 6.1: Computer Graphics

```

clc
clear
//
    *************************************************************
//Creation of an object (say, rectangle)
//
    *************************************************************
x=[0,5,5,0,0]
y=[0,0,3,3,0]
N=[x;y]

//
    *************************************************************
//To Study Shear
//
    *************************************************************
```

```
l=input("Enter 1, 2 or 3 for shear along x- axis, y-axis or both axis, respectively: ")

figure(1)
xlabel('x-coordinates (cm)','fontsize',5)
ylabel('y-coordinates (cm)','fontsize',5)
a=get("current_axes")       //get the handle of the newly created axes
a.font_size=4
t=get("hdl")                //get the handle of the newly created object
t.font_size=5

select l
    case 1
        // Transformation Matrix for Shear parallel to x-axis
        s=input("Enter the value of shear s = ")
        Sx=[1 s; 0 1]
        S=Sx*N
        title('Shear parallel to x-axis','fontsize',5)
        a.data_bounds=[0,0;8,5]
    case 2
        // Transformation Matrix for Shear parallel to y-axis
        s=input("Enter the value of shear s = ")
        Sy=[1 0; s 1]
        S=Sy*N
        title('Shear parallel to y-axis','fontsize',5)
        a.data_bounds=[0,0;6,8]
    case 3
        // Transformation Matrix for Shear in x and y-direction
        sx=input("Enter the value of shear in x-direction sx = ")
        sy=input("Enter the value of shear in y-
```

```
direction sy = ")
        Sxy=[1 sx; sy 1]
        S=Sxy*N
        title('Shear in x and y direction','fontsize',5)
        a.data_bounds=[0,0;6,8]
end
plot(x,y,'linewidth',3)
plot(S(1,:),S(2,:),'--r','linewidth',3)
hl=legend(['old coordinates';'new coordinates'])
h1.font_size=3


//*************************************************************
//To Study Strain
//*************************************************************

p=input("Enter 1 or 2 for strain along x- axis or y-axis, respectively: ")

figure(2)
xlabel('x-coordinates (cm)','fontsize',5)
ylabel('y-coordinates (cm)','fontsize',5)
a=get("current_axes");  //get the handle of the newly created axes
a.font_size=4
t=get("hdl")   //get the handle of the newly created object
t.font_size=5;

select p
    case 1
//          Transformation Matrix for Strain along x-axis
```

```
        stx=input("Enter the value of strain stx = ")
        Str_x=[stx 0; 0 1]
        ST=Str_x*N;
        title('Strain along x-axis','fontsize',5);
        a.data_bounds=[0,0;8,5];
    case 2
//          Transformation Matrix for strain along y-axis
        sty=input("Enter the value of strain sty = ")
        Str_y=[1 0; 0 sty]
        ST=Str_y*N;
        title('Strain along y-axis','fontsize',5);
        a.data_bounds=[0,0;6,8];
end
plot(x,y,'linewidth',3);
plot(ST(1,:),ST(2,:),'--r','linewidth',3)
hl=legend(['old coordinates';'new coordinates']);
h1.font_size=3



//************************************************************
//To Study 2D Rotation
//************************************************************

k=input("Enter 1 or 2 for 2D Rotation in clockwise and anticlockwise direction, respectively: ")
th=input("Enter the required angle for rotation th (in degree) = ")
figure(3)
xlabel('x-coordinates (cm)','fontsize',5)
ylabel('y-coordinates (cm)','fontsize',5)
```

```
a=get("current_axes");//get the handle of the newly created axes
a.font_size=4
t=get("hdl")    //get the handle of the newly created object
t.font_size=5;

select k
    case 1
        // Transformation Matrix for Rotation in clockwise direction
        Cl=[cosd(th),sind(th);-sind(th),cosd(th)]
        Rot=Cl*N;
        title('Rotation in clockwise direction','fontsize',5);
        a.data_bounds=[0,0;8,5];
    case 2
        // Transformation Matrix for Rotation in anticlockwise direction
        Anti=[cosd(th),-sind(th);sind(th),cosd(th)]
        Rot=Anti*N;
        title('Rotation in anticlockwise direction','fontsize',5);
        a.data_bounds=[0,0;6,8];
end
plot(x,y,'linewidth',3);
plot(Rot(1,:),Rot(2,:),'--r','linewidth',3)
hl=legend(['old coordinates';'new coordinates']);
h1.font_size=3

//*************************************************************
//To Study the reflection
//*************************************************************
j=input("Enter 1, 2 or 3 for reflection about x-
```

```
axis, y-axis or origin, respectively: ")

figure(4)
xlabel('x-coordinates (cm)','fontsize',5)
ylabel('y-coordinates (cm)','fontsize',5)

a=get("current_axes");//get the handle of the newly created axes
a.font_size=4
t=get("hdl")   //get the handle of the newly created object
t.font_size=5;
select j
    case 1
    // Transformation Matrix for Reflection about x-axis
        Rx=[1 0; 0 -1]
        R=Rx*N;
        title('Reflection about x-axis','fontsize',5);
        a.data_bounds=[0,-4;6,4];
    case 2
    // Transformation Matrix for Reflection about y-axis
        Ry=[-1 0; 0 1]
        R=Ry*N;
        title('Reflection about y-axis','fontsize',5);
        a.data_bounds=[0,0;8,4];
    case 3
        //Transformation Matrix for Reflection about origin
        Rxy=[-1 0; 0 -1]
        R=Rxy*N;
        title('Reflection about origin','fontsize',5);
        a.data_bounds=[-8,-5;8,5];
end
```

```
plot(x,y,'linewidth',3);
plot(R(1,:),R(2,:),'--r','linewidth',3)
h1=legend(['old coordinates';'new coordinates']);
h1.font_size=5

//*************************************************************
//To Study translation
//*************************************************************
i=input("Enter 1, 2 or 3 for translation along x-axis, y-axis or both directions, respectively: ")

figure(5)
xlabel('x-coordinates (cm)','fontsize',5)
ylabel('y-coordinates (cm)','fontsize',5)
a=get("current_axes");    //get the handle of the newly created axes
a.font_size=4
t=get("hdl")    //get the handle of the newly created object
t.font_size=5;

select i
    case 1
        // Transformation Matrix for translation along to x-axis
        tx=input("Enter the required translation along x direction tx =  ")
        T1=[ones(1,length(x));zeros(1,length(x))];
        X=N+tx*T1;
        title('Translation along to x-axis','fontsize',5);
        a.data_bounds=[0,0;8,5];
    case 2
```

```
        // Transformation Matrix for translation along to y-axis
        ty=input("Enter the the required translation along x direction ty = ")
        T1=[zeros(1,length(x));ones(1,length(x))];
        X=N+ty*T1;
        title('Translation along to y-axis','fontsize',5);
        a.data_bounds=[0,0;6,8];
    case 3
        // Transformation Matrix for translation along to y-axis
        tx=input("Enter the required translation along x direction tx = ")
        ty=input("Enter the required translation along y direction ty = ")
        T1=[ones(1,length(x));zeros(1,length(x))];
        T2=[zeros(1,length(x));ones(1,length(x))];
        X=N+tx*T1+ty*T2;
        title('Translation along to y-axis','fontsize',5);
        a.data_bounds=[0,0;6,8];
end
plot(x,y,'linewidth',3);
plot(X(1,:),X(2,:),'--r','linewidth',3)
hl=legend(['old coordinates';'new coordinates']);
h1.font_size=3
```

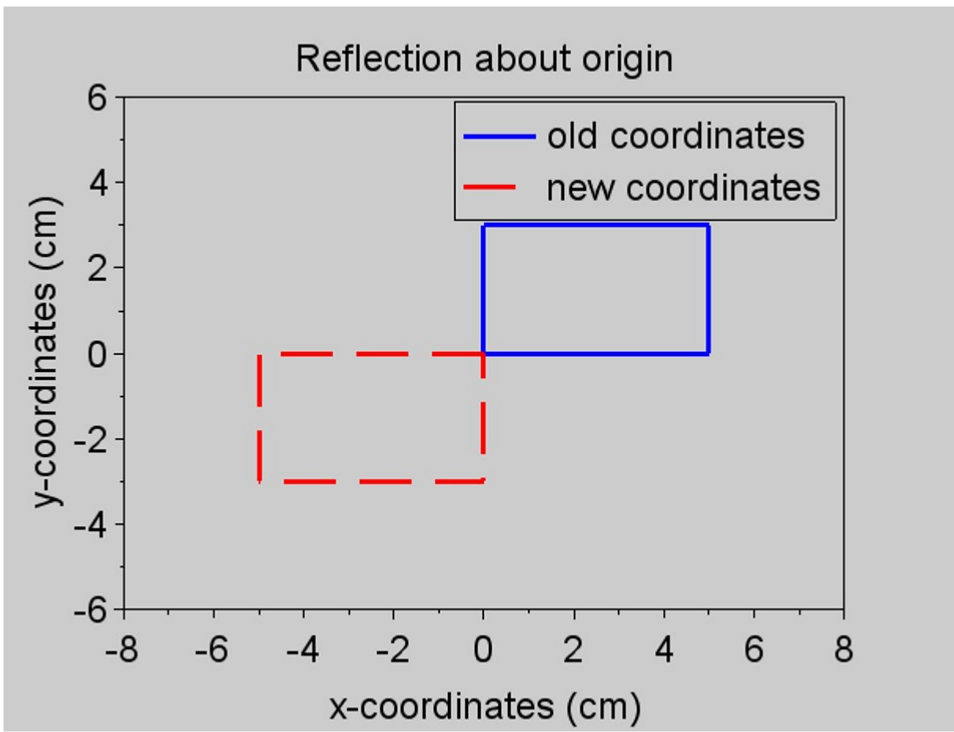

Figure 6.2: Computer Graphics

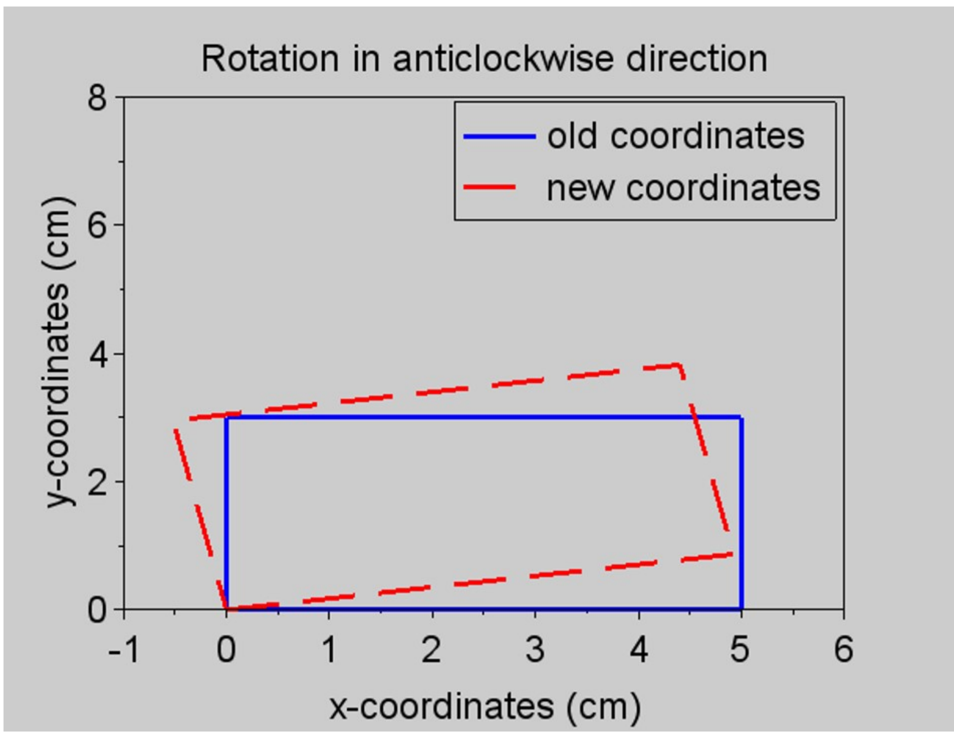

Figure 6.3: Computer Graphics

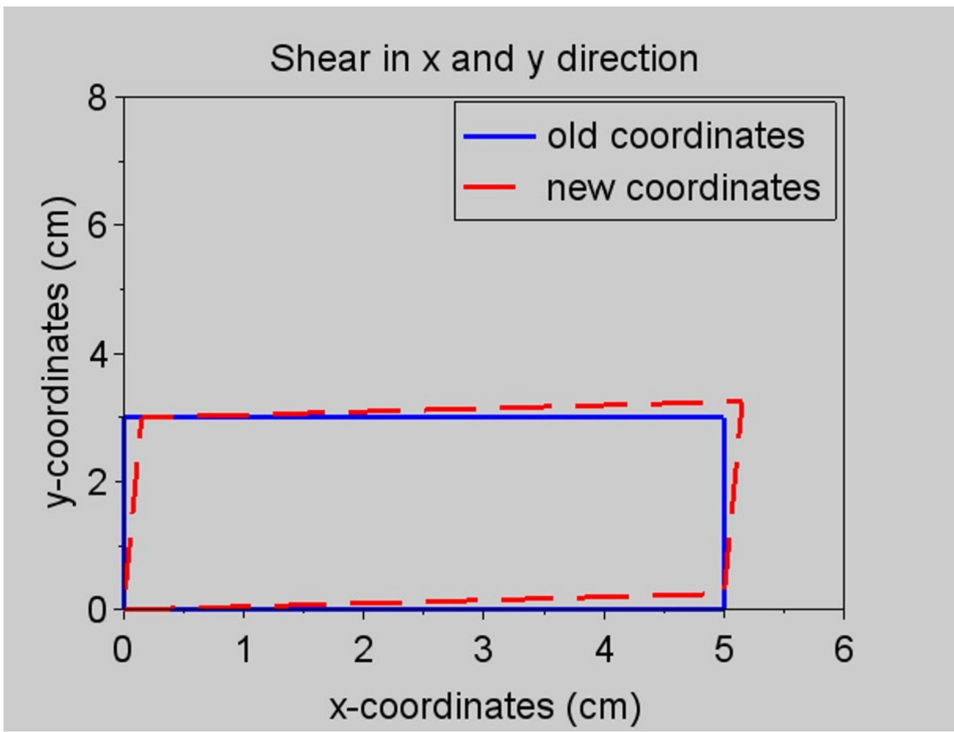

Figure 6.4: Computer Graphics

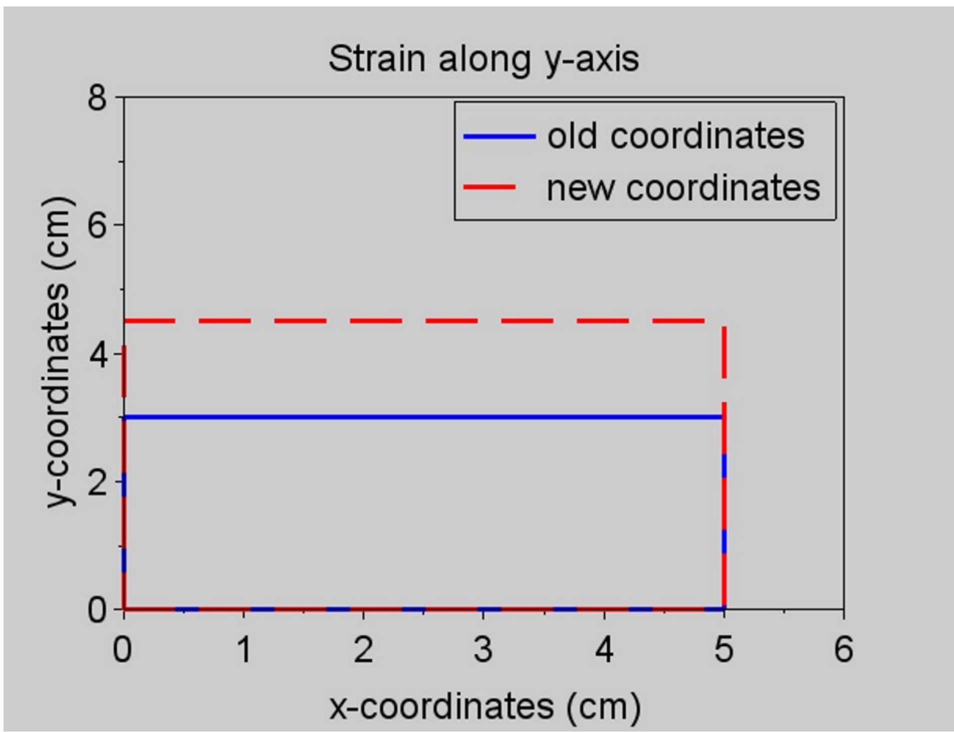

Figure 6.5: Computer Graphics

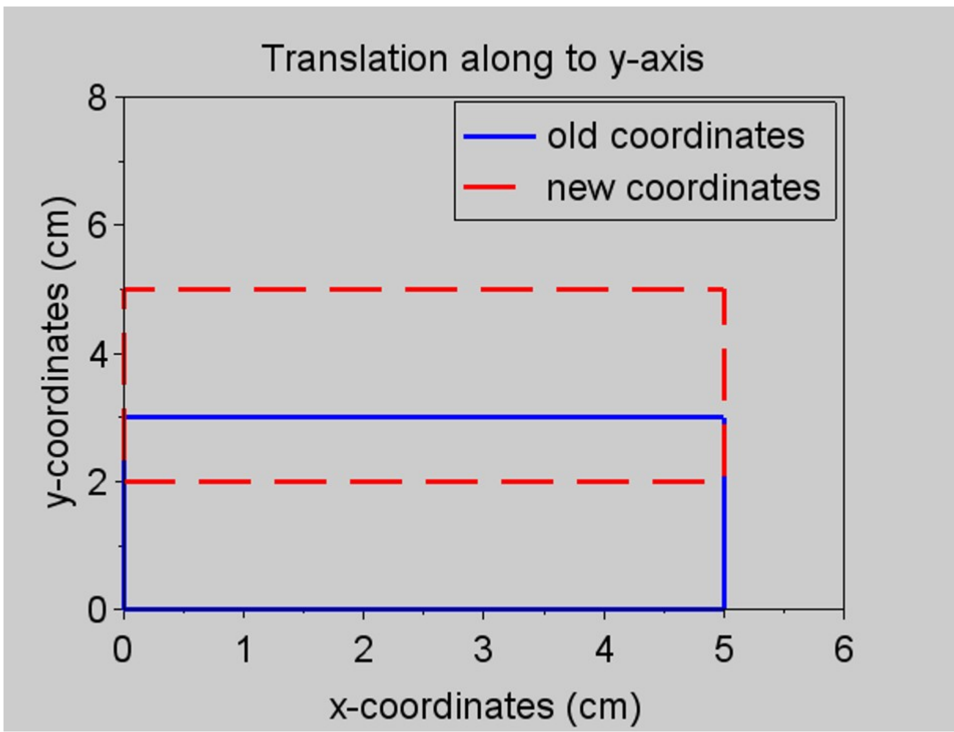

Figure 6.6: Computer Graphics

Experiment: 7

Lagrangian formulation in classical mechanics with constraints.

Scilab code Solution 7.0 Lagrangian Formulation

```
// Submitted by Dr. Triranjita Srivastava. Assistant
    Professor, Physics Dept., Kalindi College,
    University of Delhi

//Operating system: Windows 8
//SCILAB Ver: 5.5.2
//Objectiv: Lagrangian formulation in classical
   mechanics with constraints
//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/dt^2)+(g/L)sin(theta)]=0. Here theta is
   angular displacement.
// We implemented ordinary differential equation (
   ODE) Solver to solve the second order
   differential equation
```

```
//We present plot of solution of angular displacement for t=0 to t=10 seconds

clear
clc
L=input ('Enter the length of pendulum (m) L = ')
g = 9.8                          //acceleration due to gravity (m/s^2)
k=g/L
theta=input('Enter the initial angular displacement (radian) at (t = 0) = ') ;          // Initial angular displacement at t = 0
dt=input('Enter initial d_theta/dt (radian) at (t = 0) = ') ;          // Initial boundary condition d_theta/dt at t = 0
////***************************************************************
//// Function declaration for ODE
////***************************************************************
t=linspace(0,10,200)
function dx=f(t,x,k)
    dx(1)=x(2)
    dx(2)=-k*sin(x(1))
endfunction
////***************************************************************
//// Solving second order differential equation by ODE  solver
////***************************************************************
    y=ode([theta;dt],0,t,f)
    ysol=y(1,:)
    ydotsol = y(2,:)
```

```
////**********************************************************
//// Plotting the solution (angular displacement (theta)and d_theta/dt)
////**********************************************************
scf()
title('Solution of Simple Pendulum', 'fontsize',5)
ylabel('Solution -->','fontsize',5)
xlabel('t (sec) --> ','fontsize',5)
a=get("current_axes")        //get the handle of the newly created axes
a.font_size=4
t=get("hdl")                 //get the handle of the newly created object
t.font_size=5
plot(t,ysol,'r','linewidth',3)
plot(t,ydotsol,'k','linewidth',3)
h1 = legend(['$\theta$'; '$d\theta/dt$'])
h1.font_size=3
```

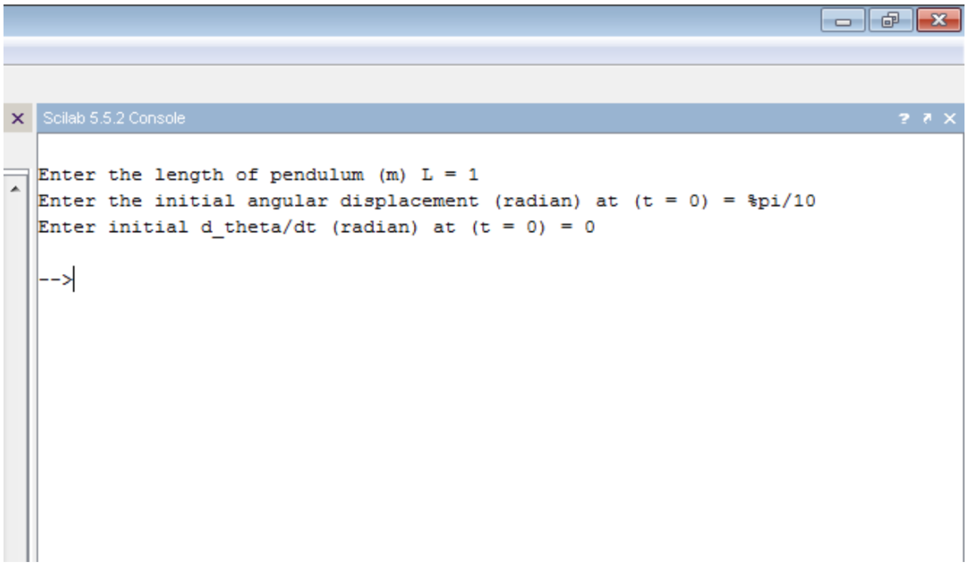

Figure 7.1: Lagrangian Formulation

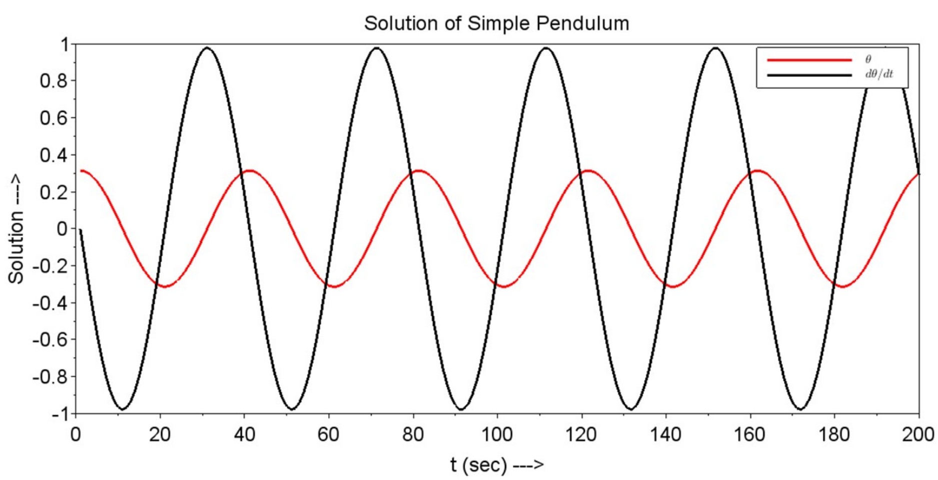

Figure 7.2: Lagrangian Formulation

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

```
// Submitted by Dr. Triranjita Srivastava. Assistant Professor, Physics Dept., Kalindi College, University of Delhi

// Aim: To show the commutator relation in postion and momentum space [x,p]=ih_cut or n general [x^n,p]=i*h_cut*n*x^(n-1)
// Two examples are shown in this program
//1. Let the first function is fx=x
//2. Let the second function is fx=x^3
// For simplicity let the wavefunction A=x
//[fx,p]=(ih_cut)(dfx/dx)
//h_cut=h/2pi; h is planck constant
```

```
clc
x=poly(0,"x")
h_cut=1.05*(10)^-34                         //h_cut=h/2pi, units is in Joule-sec
A=x                                         //Considered Wavefunction is A=x

s=input("Enter 1 or 2 to choose the function as fx = x or fx = x^3, s= ")
select s
case 1
    fx=x                                    //First wavefunction
case 2
    fx=x^3                  //Second wavefunction
end

fx_p=fx*(-%i*h_cut)*derivat(A)
p_fx=(-%i*h_cut)*derivat(fx*A)
commutator=(fx_p-p_fx)
disp("[fx,p] = ")
disp(commutator)
disp ("The result contains an extra x because the chosen wavefunction is A = x")
```

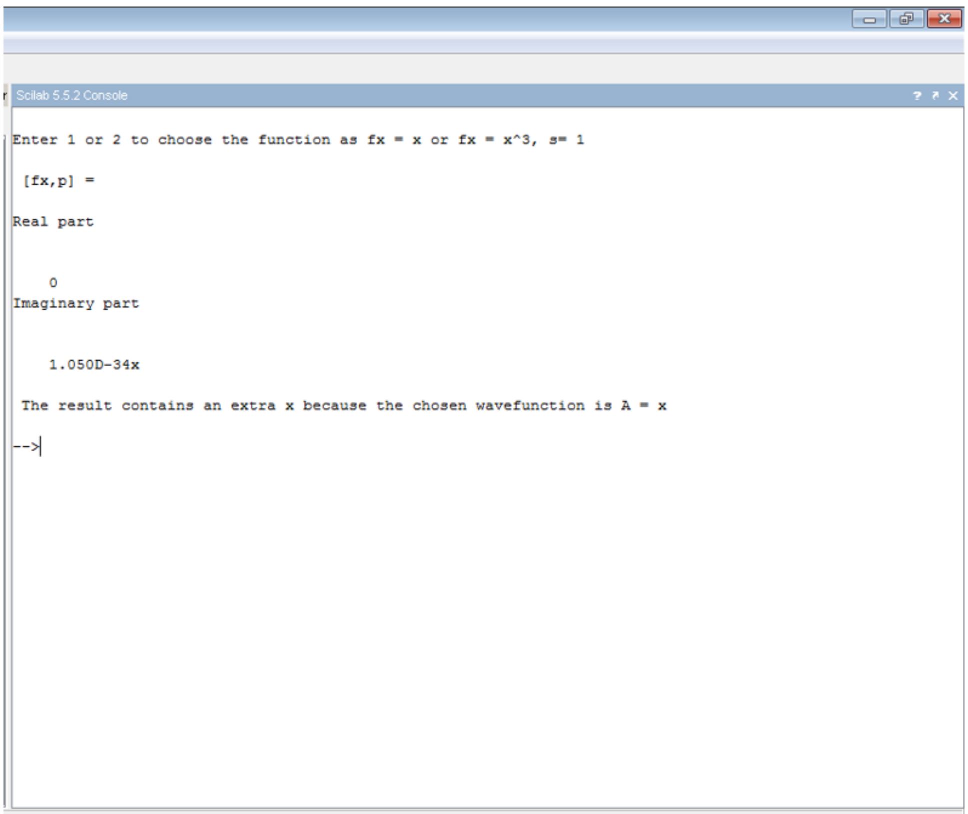

Figure 8.1: Hermitian Differential Op

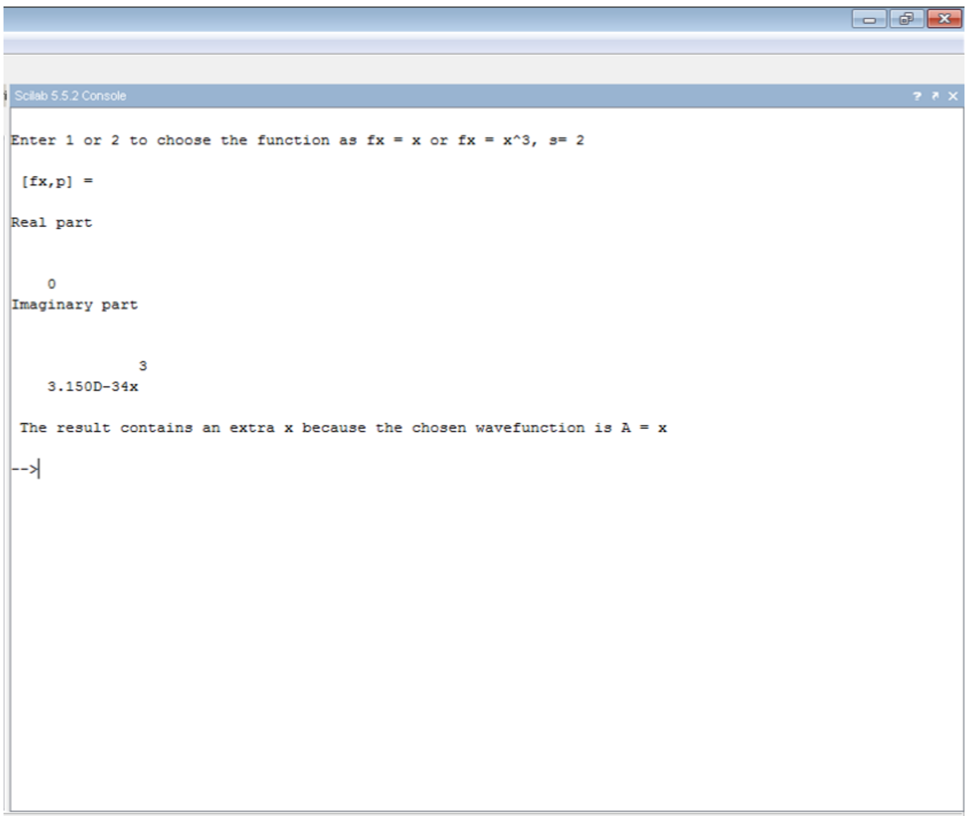

Figure 8.2: Hermitian Differential Op