Scilab Manual for Numerical and Statistical Methods Laboratory by Dr Keval Chandrakant Nikam Mechanical Engineering Savitribai Phule Pune University[1]

Solutions provided by
Dr Keval Chandrakant Nikam
Mechanical Engineering
Savitribai Phule Pune University

May 7, 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

Experiment: 1

Program for roots of Equation using Bisection Method accuracy criteria

Scilab code Solution 1.1 Bisection Method

```
//Scilab code Solution 1 Program for roots of
    Equation using Bisection Method accuracy criteria
// Operating System  Windows 7
// SCILAB version 6.1.1
// Experiment No 1
// Program for roots of Equation using Bisection
    Method accuracy criteria
// Example - Write a computer program in SCILAB
    to find root of equation as x^2-8x+2..
//Take Accuracy as 0.01 using Bisection Method.Take
    x1=0 and x2=1
// Input x1=0,x2=1,acc=0.01,f(x)=x^2-8*x+2
clc;
clear;
close;
```

```
deff('y=f(x)','y=x^2-8*x+2')
x1=input('Enter First Initial Guess = ');
x2=input('Enter Second Initial Guess = ');
acc=input('Enter the value of Accuracy =');
i=0;
printf('Iteration \t x1 \t \t x2 \t \t z \t \t f(z)\
   n ')
while abs(x1-x2)>acc // Condition of Accuracy
      z=(x1+x2)/2
printf ('%i\t \t%f \t%f \t%f \t%f \n ',i,x1,x2,z,f(z
   ))// Print in form of Table
if f(z)*f(x1)>0  // Substitution of initial guess
   for next iteration
      x1=z
else
      x2=z
end
      i=i+1 // Increment in Iteration by 1 for each
         step
end
printf(' \n\n The solution of this equation is %g
   after %i Iterations ',z,i-1)// Display final
   anaswe to User
```

Experiment: 2

Program for roots of Equation using Newton Raphson Method accuracy criteria

Scilab code Solution 2.2 Newton Raphson Method

```
//Scilab code Solution 2 Program for roots of
    Equation using Newton Raphson Method accuracy
    criteria
// Operating System  Windows 7
// SCILAB version 6.1.1
// Experiment No 2
// Program for roots of Equation using Newton
    Raphson Method accuracy criteria
// Example - Solve using Newton Raphson Method x-exp
    (-x)=0
//Take accuracy as 0.001 .Take x0=1
//Input x0=1,acc=0.001,f(x)=x-exp(-x)
clc;
clear;
close;
```

```
deff('y=f(x)','y=x-exp(-x)')
deff('y=f1(x)','y=1+exp(-x)')
x0 =input('Enter Initial Guess = ');
acc =input('Enter the value of Accuracy =');
i=0;
printf('i \t\t x0 \t\t x1 \n')
x1=x0-(f(x0)/f1(x0))
printf('%i\t\t%0.5f\t\t%0.5f \n' ,i,x0 ,x1 )
while abs(x1-x0)>acc // Condition of Accuracy
x0=x1;
x1=x0-(f(x0)/f1(x0)) // Formula of finding root of
    Equation
i=i+1
printf ('%i\t \t%f \t%f \n ',i,x0,x1) // Print in
    form of Table
end
printf('\n\n The root of equation is %0.5f' ,x1) //
    Display final answer to User
```

Experiment: 3

Program for Simultaneous equations using Gauss Elimination Method

Scilab code Solution 3.3 Gaussian Elimination method

```
//Scilab code Solution 3 Program for Simultaneous
    equations using Guass Elimination Method
// Operating System  Windows 7
// SCILAB version 6.1.1
// Experiment No 3
// Program for Simultaneous equations using Guass
    Elimination Method
// Example - Write a computer program in SCILAB
      to solve following set of simultaneous
    equations using Gauss Elimination method.
//3 X + 2Y + Z = 10
//2 X + 3 Y +  2Z = 14
//X + 2Y + 3Z = 14
// Input coeffienct matrix a and solution matrix b
clc;
```

```
clear all;
disp('OUTPUT:');
a=input('Enter coefficient matrix a:=');
b=input('Enter matrix b:=');
[m,n]=size(a);
if m~=n // Check condition of square matrix
   error('Matrix A must be square');
end
//Perform Partial Pivoting
for i=1:1:n-1
    for u=i+1:1:n
         if (abs(a(u,i))>abs(a(i,i))) // Comparison of Pivot Element
            for v=1:1:n
               temp=a(i,v); // Replacement of Pivot Element
               a(i,v)=a(u,v);
               a(u,v)=temp;
            end
            temp=b(i);
            b(i)=b(u);
            b(u)=temp;
         end
    end
//Gauss Elimination - operation of Rows
   for k=i+1:1:n
         factor=a(k,i)/a(i,i);
      for j=1:1:n
         a(k,j)=a(k,j)-factor*a(i,j); // Formula to make Coefficent Matrix in Upper Triangular Matrix
      end
      b(k)=b(k)-factor*b(i); // Formula also applicable to solution matrix
   end
end
disp('Final augmented matrix is:');
disp([a,b]); //Display formed Upper Triagular Matrix
```

```
// Back Substitution
for i=n:-1:1
    temp=b(i);
    for j=i+1:1:n
        temp=temp-a(i,j)*x(j);
    end
    x(i)=temp/a(i,i); // Calculating the value of x
        (3),x(2) and x(1) resp.
end
disp('Answer is:');
disp(x);
```

Experiment: 4

Program for Ordinary differential equation using Euler Method

Scilab code Solution 4.4 Euler method

```
//Scilab code Solution 4 Program for Ordinary
    differential equation using Euler Method
// Operating System  Windows 7
// SCILAB version 6.1.1
// Experiment No 4
// Program for Ordinary differential equation using
    Euler Method
// Example - Write a computer program in SCILAB to
    solve the ODE
//dy/dx=-x*y^2 using Eulers method under the
    condition x=0,y=2.Find y at x=1 with h=0.1.
//Input function -xy^2,x0=0,y0=2,xn=1,h=0.1
clc;
close;
clear;
```

```
deff('y=f(x,y)','y=-x*y^2');// Enter the Function
x0=input('Enter the value of x0=');
y0=input('Enter the value of y0=');
xn=input('Enter the value of xn=');
h=input('Enter the value of h=');
n=(xn-x0)/h; // Formuale for finding number of Step Size
disp(n)
for i=1:1:n
   yn=y0+h*f(x0,y0);// Formulae use in Euler Method
   x0=x0+h; // Increment in Step size
   y0=yn; // Replacement of y0 as yn for next iteration
   printf('Value of y(%f)=%f\n',x0,y0);
end
```

Experiment: 5

Program for Ordinary differential equation using Runge Kutta 4th order

Scilab code Solution 5.5 RK4ORDER

```
//Scilab code Solution 5 Program for Ordinary
    differential equation using Runge-Kutta 4th order
// Operating System  Windows 7
// SCILAB version 6.1.1
// Experiment No 5
// Program for Ordinary differential equation using
    Runge-Kutta 4th order
// Example - Write a computer program in SCILAB to
    obtain the numerical solution of
//dy/dx=x^2+y^2, y(0)=0,h=0.2.Estimate  y(0.4) using
     Runge Kutta 4 order method
//Input function x^2+y^2,x0=0,y0=0,h=0.2,xn=0.4
clc;
close;
clear;
```

```
deff('y=f(x,y,z)','y=x*x+y*y');// Enter the Function
x0=input('Enter the value of x0=');
y0=input('Enter the value of y0=');
xn=input('Enter the value of xn=');
h=input('Enter the value of h=');
n=(xn-x0)/h;// Formuale for finding number of Step Size
disp(n)
for i=1:1:n
   k1=h*f(x0,y0);// Calculate value of k1
   k2=h*f(x0+h/2,y0+k1/2);// Calculate value of k2
   k3=h*f(x0+h/2,y0+k2/2);// Calculate value of k3
   k4=h*f(x0+h,y0+k3);// Calculate value of k4
   k=(k1+2*k2+2*k3+k4)/6.0;// Calculate value of k
   yn=y0+k;// Increment in Step size
   x0=x0+h;// Increment in Step size
   y0=yn;// Replacement of z0 as z1 for next iteration
   printf('Value of y(%f)=%f\n',x0,y0);// Display y(n)
end
```

Experiment: 6

Program for Ordinary differential equation using Simultaneous equations using Runge Kutta 2nd order method

Scilab code Solution 6.6 RK2Order Simultaneous

```
//Scilab code Solution 6 Program for Ordinary differential equation using Simultaneous equations using Runge-Kutta 2nd  order method
// Operating System  Windows 7
// SCILAB version 6.1.1
// Experiment No 6
// Program for Ordinary differential equation using Simultaneous equations using Runge-Kutta 2nd order method
// Example - Write a computer program in SCILAB to to solve the equation
//dy/dx=-0.5*y,dz/dx=4-0.3*z-0.1*y Using runge kutta second order simultaneous method where at x = 0, y =4, z =6.
//Find y & z at x = 0.5 (take h=0.5)
```

```
//Input function f(x,y,z)=-0.5*y, g(x,y,z)= 4-0.3*z-0.1*y,x0=0,y0=4,z0=6,h=2,xn=0.5
clc;
clear;
deff('y=f(x,y,z)','y=-0.5*y');// Enter the Function
deff('z=g(x,y,z)','z=4-0.3*z-0.1*y');// Enter the Function
x0=input('Enter the value of x0=');
y0=input('Enter the value of y0=');
z0=input('Enter the value of z0=');
xn=input('Enter the value of xn=');
h=input('Enter the value of h=');
n=(xn-x0)/h;// Formuale for finding number of Step Size
for i=1:1:n
   k1=h*f(x0,y0,z0);// Calculate value of k1
   L1=h*g(x0,y0,z0);// Calculate value of L1
   k2=h*f(x0+h,y0+k1,z0+L1);// Calculate value of k2
   L2=h*g(x0+h,y0+k1,z0+L1);// Calculate value of L2
   k=(k1+k2)/2.0; // Finding out increment in y direction
   y1=y0+k;// Increment in Step size
   L=(L1+L2)/2.0;// Finding out increment in z direction
   z1=z0+L;// Increment in Step size
   x0=x0+h;// Increment in Step size
   y0=y1;// Replacement of y0 as y1 for next iteration
   z0=z1;// Replacement of z0 as z1 for next iteration
  printf('value of y(%f)=%f\n',x0,y0);// Display y(n)
  printf('value of z(%f)=%f\n',x0,z0);// Display z(n)
end
```

Experiment: 7

Program for Partial differential equation using Simple Laplace method

Scilab code Solution 7.7 Laplace Method

```
//Scilab code Solution 7    Program for Partial
    differential equation using Simple Laplace method
// Operating System  Windows 7
// SCILAB version 6.1.1
// Experiment No 7
// Program for Partial differential equation using
    Simple Laplace method
// Example - A steel Plate of 750x750mm has its two
    adjancent sides maintained at
//100 C .While the two other sides are maintained at
     0 C .What will be the
//steady state temperature at interior assuming a
    grid size of 250mm.Solve upto 11 iteration
```

```
//Input function n1=11,n=4,m=4,u(1,1)=0,u(2,1)=0,u(3,1)=0,u(4,1)=0
//u(4,2)=0,u(4,3)=0,u(4,4)=0,u(3,4)=100,u(2,4)=100,u(1,4)=100
//u(1,3)=100,u(1,2)=100
clc;
clear;
n1=input('Enter the no. of iteration to solve simultaneous eqn:');
n=input(' Enter the no.of mesh point(No.of B.V. values) in x-direction:');
m=input(' Enter the no.of mesh point(No.of B.V. values) in y-direction:');
printf(' Enter boundary value(B.V.) in anticlockwise direction Starting from bottom left corner\n');
u=zeros(m,n); //to create matrix of total size and to take initial guess as 0,0,0...
for i=1:n// to take input as a boundary value at bottom side
   printf('Enter u(%d,1)=',i);
   u(i,1)=input('');
end
for j=2:m //to take I/P at Right hand side B.V. bottom to top
   printf('Enter u(%d,%d)=',n,j);
   u(n,j)=input('');
end
for i=n-1:-1:1 // to take I/P at top side B.V. right to left
   printf('Enter u(%d,%d)=',i,m);
   u(i,m)=input('');
end
for j=m-1:-1:2 //to take I/P at left hand side B.V. top to bottom
   printf('Enter u(1,%d)=',j);
   u(1,j)=input(' ');
end
for k=1:n1 // To repeat n1 iterations
```

```
    for j=2:m-1 //To calculate value at intermediate
       point by Gauss Seidal method
        for i=2:n-1
            u(i,j)=1/4*(u(i-1,j)+u(i,j+1)+u(i+1,j)+u(i,
               j-1));// Formula for finding Internal
               Elements
        end
    end
    printf('Value after iteration no.:%d\n',k);
    for j=m:-1:1 //To print value after each
       Iteration in Tabulated form
        for i=1:n
            printf('\t %0.4f \t',u(i,j));
        end
      printf('\n');
   end
end
```

Experiment: 8

Program for Numerical Integration using Trapezoidal rule

Scilab code Solution 8.8 TrapezoidalRule

```
//Scilab code Solution> 8 Program for Numerical
    Integration using Trapezoidal rule
// Operating System  Windows 7
// SCILAB version 6.1.1
// Experiment No 8
// Program for Numerical Integration using
    Trapezoidal rule
// Example - Write a computer program in SCILAB to
    solve integration 4*x+2
//limits x0=0,xn=1 by using Trapezoidal Method.
//Take h=0.5.
//Program on Trapezoidal Rule
clc;
close;
clear;
```

```
deff('y=f(x)','y=4*x+2');// Enter the Function
x0=input('Enter lower limit:');// Enter Lower Limit
xn=input('Enter upper limit:');// Enter Upper Limit
h=input('Enter step Size h:');// Enter Step Size
x=x0;
n=(xn-x0)/h;// Enter number of Step size
s=0;
for i=1:n-1
    x=x+h;
    s=s+2*f(x);
end
s=f(x0)+s+f(xn);
I=h/2*s;// Formula for finding Area by using Trapezoidal Rule
printf('Integration of given function is=%f\n',I);
```

Experiment: 9

Program for Numerical Integration using Simpsons 1/3rd Rule

Scilab code Solution 9.9 Simpson 1 3rd Rule

```
//Scilab code Solution 9    Program for Numerical
   Integration using Simpsons 1/3 rd Rule
// Operating System  Windows 7
// SCILAB version 6.1.1
// Experiment No 9
// Program for Numerical Integration using
   Simpsons 1/3 rd Rule
// Example - Write a Program in Scilab for finding
   area of fucntion (sin(x))/(2+3*sin(x)) for
//upper limit of 1 and lower limit of 0.Take n=6 by
   using Simpson's 1/3 Rule
//Program on Simpson's 1/3rd Rule
clc;
clear;
deff('y=f(x)','y=(sin(x))/(2+3*sin(x))');// Enter
```

```
the Function
x0=input('Enter lower limit:');// Enter the lower limit of x
xn=input('Enter upper limit:');// Enter the upper limit of x
n=input('Enter number of steps:');// Enter the number of steps
x=x0;
h=(xn-x0)/n;// Calculate step size
s=0;
for i=1:n-1
   x=x+h;
   if modulo(i,2)==0 // Calculating Even Term of Simpson 1/3rd Formula
      s=s+2*f(x);
   else
      s=s+4*f(x);
   end
   end
s=f(x0)+s+f(xn);
I=(h/3)*s;// Finding Integrating value
printf('\nIntegration of given function is=%f\n',I); // Display Value
```

Experiment: 10

Program for Numerical Integration using Simpsons 3/8th Rule.

Scilab code Solution 10.10 Simpson 3 8th Rule

```
//Scilab code Solution 10   Program for Numerical
    Integration using Simpsons 3/8 Rule
// Operating System  Windows 7
// SCILAB version 6.1.1
// Experiment No 10
// Program for Numerical Integration using
    Simpsons 3/8 Rule
// Example - Write a Program in Scilab for finding
    area of fucntion exp(x)/x for
//upper limit of 2 and lower limit of 1.Take n=6 by
    using Simpson's 3/8 Rule
//Program on Simpson's 3/8th Rule
clc;
close;
clear;
```

```
deff('y=f(x)','y=exp(x)/x');// Enter function
x0=input('Enter lower limit:');// Enter lower limit of x
xn=input('Enter upper limit:');// Enter upper limit of x
n=input('Enter number of steps:');// Enter number of step
x=x0;
h=(xn-x0)/n;// Finding out of step size
s=0;
for i=1:n-1
   x=x+h;
   if modulo(i,3)==0 // Condition for adding the odd value together
     s=s+2*f(x);
   else
     s=s+3*f(x);
   end
   end
s=f(x0)+f(xn)+s;
I=((3*h)/8)*s; // Calculating Area
printf('\nIntegration of given function is=%f\n',I);
```

Experiment: 11

Program for Numerical Integration using Gauss Quadrature 2-point and 3-point method

Scilab code Solution 11.11 Gauss 2 and 3 Point Method

```
//Scilab code Solution 11  Program for Numerical
   Integration using Gauss Quadrature 2-point and 3-
   point method
// Operating System  Windows 7
// SCILAB version 6.1.1
// Experiment No 11
// Program for Numerical Integration using Gauss
   Quadrature 2-point and 3-point method
// Example - Write a Program in Scilab to solve
   using two-point or three point Gauss quadrature
   rule to
//approximate the distance covered by a rocket from
   t = 8 to t = 30 as given by
//x=(2000*log(140000/(140000-2100*t))-9.8*t)
// Enter a=lower limit=8,b=upper limit=30,n=Enter 2
```

```
or 3 depend upon Guass 2 point or 3 point formula
//Program on Gauss Quadrature 2-point and 3-point method
clc;
clear;
deff('x=f(t)','x=(2000*log(140000/(140000-2100*t))-9.8*t)');// Enter the function
a=input('Enter lower limit:');// Enter the lower limit of Integration
b=input('Enter upper limit:');// Enter the upper limit of Integration
n=input('Enter 2 point or 3 point method:');// Enter which method you are suppose to use
if n==2  // For executing 2 Point Method
    c=(b-a)/2;
    d=(b+a)/2;
    z1=-1/sqrt(3);
    z2=1/sqrt(3);
    x1=c*z1+d;
    x2=c*z2+d;
    I=c*(f(x1)+f(x2));// Formula for finding Integration value
else // For executing 3 Point Method
    c=(b-a)/2;
    d=(b+a)/2;
    z1=sqrt(3/5);
    z2=-sqrt(3/5);
    x1=c*z1+d;
    x2=c*z2+d;
    x3=d;
    I=c*(5/9*f(x1)+5/9*f(x2)+8/9*f(x3));// Formula for finding Integration value
end
printf('\n Integration of given function is=%f\n',I);// Display the Integration
```

Experiment: 12

Program for Numerical Double Integration using Trapezoidal rule

Scilab code Solution 12.12 Trapezoidal Double Rule

```
//Scilab code Solution   12 Program for Numerical Double Integration using Trapezoidal rule
// Operating System  Windows 7
// SCILAB version 6.1.1
// Experiment No 12
//  Program for Numerical Double Integration using Trapezoidal rule
// Example - Write a Program in Scilab for finding area of fucntion x+y for
//upper limit of 0 and lower limit of 1 for x,y.Take n=m=6 by using Numerical Double Integration using Trapezoidal rule
//Take f(x,y)=x+y,x0=0,xn=1,y0=0,yn=1,n=m=6
//  Program for Numerical Double Integration using Trapezoidal rule
```

```
clc;
close;
clear;
deff('y=f(x,y)','y=x+y');// Enter function
x0=input('Enter x0 lower limit of x:');// Enter lower limit of x
xn=input('Enter xn upper limit of x:');// Enter upper limit of x
n=input('Enter no. of steps in x-direction:');// Enter number of step size in x axis
y0=input('Enter y0 lower limit of y:');// Enter lower limit of y
ym=input('Enter ym upper limit of y:');// Enter upper limit of y
m=input('Enter no. of steps in y-direction:');// Enter number of step size in y axis
h=(xn-x0)/n;// Enter step size in x axis
k=(ym-y0)/m;// Enter step size in y axis
s=0;
x=x0;// Replacement of x by x0
y=y0;// Replacement of y by y0
for i=1:1:m+1
    for j=1:1:n+1
        a(i,j)=f(x,y);// Alloting pivot point by putting value in function
        x=x+h;//Increament in x axis
    end
     y=y+k;//Increament in y axis
     x=x0;
 end
 disp([a]);
 for i=1:1:m
    for j=1:1:n
        s=s+a(i,j)+a(i,j+1)+a(i+1,j)+a(i+1,j+1);
    end
 end
 I=h*k/4*s;// Calculating the Area
 printf('Integration of given function is=%f\n',I);
```

```
// Display the Area
```

Experiment: 13

Program for Curve fitting using least square technique for first order equation

Scilab code Solution 13.13 First Order Equation

```
//Scilab code Solution 13   Program for Curve
    fitting using least square technique for first
    order equation
// Operating System  Windows 7
// SCILAB version 6.1.1
// Experiment No 13
// Program for Curve fitting using least square
    technique for first order equation
// Example - Write a computer program in SCILAB to
    fit a straight line to the data given below :
// x=[1 2 3 4 5 6 7];
//y=[0.5 2.5 2.0 4.0 3.5 6.0 5.5];

clc;
close;
```

```
clear;
//x=input('enter value of x matrix')
x=[1 2 3 4 5 6 7];// Enter the x values (Dependent
    Variables)
disp([x]);
//y=input('enter value of y matrix')
y=[0.5 2.5 2.0 4.0 3.5 6.0 5.5];// Enter the y
    values(Independent Variables)
disp([y]);
n=length(x);// Enter the data in x values
Y=y;
X=x;
X2=X.*X;// Calculating X*X Values( .* indicates that
     multiplication between respective value of x)
XY=X.*Y;// Calculating X*y Values( .* indicates that
     multiplication between respective value of x and
     y)
a0=(sum(Y)*sum(X2)-sum(X)*sum(XY))/(n*sum(X2)-(sum(X
    )^2)); // Calculating coefficient a0
a1=((n*sum(XY)-sum(X)*sum(Y))/(n*sum(X2)-(sum(X))^2)
    );// Calculating coefficient a1
a=a1;//Replacement value of a
b=a0;//Replacement value of b
printf('\n y=%f*x+%f',a,b);// Display y=ax+b
```

Experiment: 14

Program for Curve fitting using least square technique for power equation

Scilab code Solution 14.14 Power Equation

```
//Scilab code Solution 14   Program for Curve
   fitting using least square technique for power
   equation
// Operating System  Windows 7
// SCILAB version 6.1.1
// Experiment No 14
// 14   Program for Curve fitting using least square
    technique for power equation
// Example - Write a computer program in SCILAB to
   fit a power equation y=ax^b to the data given
   below :
// x=[1 2 3 4 5];
//y=[0.5 1.7 3.4 5.7 8.4];
//  Program for Curve fitting using least square
   technique for power equation
```

```
clc;
close;
clear;
//x=input('enter value of x matrix')
x=[1 2 3 4 5];// Enter X values( Dependent Variables
    )
disp([x]);
//y=input('enter value of y matrix')
y=[0.5 1.7 3.4 5.7 8.4];// Enter Y Values(
    Independent variables)
disp([y]);
n=length(x);// calculate length of x
Y=log(y);// Calculate valur of Y
X=log(x);// Calculate valur of X
X2=X.*X;// Calculate valur of X*X
XY=X.*Y// Calculate valur of X*Y
a0=(sum(Y)*sum(X2)-sum(X)*sum(XY))/(n*sum(X2)-(sum(X
    )^2));// Calculating coefficienct a0
a1=((n*sum(XY)-sum(X)*sum(Y))/(n*sum(X2)-(sum(X))^2)
    );// Calculating coefficient a1
a=exp(a0); // Replacement of value
b=a1;//Replacement of value
printf('\n y=%f*x^%f',a,b);//Display y= ax^b
```

Experiment: 15

Program for Curve fitting using least square technique for exponential equation

Scilab code Solution 15.15 Exponential Equation

```
//Scilab code Solution 15   Program for Curve
   fitting using least square technique for
   exponential equation
// Operating System  Windows 7
// SCILAB version 6.1.1
// Experiment No 15
// 15   Program for Curve fitting using least square
    technique for exponential equation
// Example - Write a computer program in SCILAB to
   fit a exponential equation y=ae^bx to the data
   given below :
// x=[1 3 5 7 9];
//y=[2.473 6.722 18.274 49.673 135.026];
//  Program for Curve fitting using least square
   technique for exponential equation
clc;
close;
```

```
clear;
//x=input('enter value of x matrix')
x=[1.0 3.0 5.0 7.0 9.0];// Enter x values(Dependent Variables)
disp([x]);
//y=input('enter value of y matrix')
y=[2.473 6.722 18.274 49.673 135.026];// Enter y values(Indepent Variables)
disp([y]);
n=length(x);// Calculate number of data enter in x
Y=log(y);// Calculating Y Value
X=x;
X2=X.*X;// Calculating X*X Value( .* indicates respective value of x is multipled with respective x value)
XY=X.*Y//// Calculating X*Y Value
a0=(sum(Y)*sum(X2)-sum(X)*sum(XY))/(n*sum(X2)-(sum(X)^2));// Calculating coefficient a0
a1=((n*sum(XY)-sum(X)*sum(Y))/(n*sum(X2)-(sum(X))^2));// Calculating coefficient a1
a=exp(a0);// Finding the value of a
b=a1;// Finding the value of a
printf('\n y=%f*e^%f*x',a,b);//Displaying y=ae^b
```