KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University Established Under Section 3 of UGC Act 1956)

Coimbatore – 641 021.

SYLLABUS

	MATHEMATICS -II PRACTICAL	**L T P C**
18PHU414		**- - 4 2**

Objectives

This course enables the students

- To develop skills for quantitative estimation using computer language.
- To solve ordinary differential equations using appropriate coding.

List of Practical

1. Plotting of second order solution family of differential equation.
2. Growth model (exponential case only).
3. Decay model (exponential case only).
4. Solving first order ordinary differential equations.
5. Solution of second order ordinary differential equations with initial conditions.
6. Solving system of linear differential Equations.
7. Computing Lagrange's interpolating polynomial.
8. Computing interpolating polynomial using Newton's formula.

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University Established Under Section 3 of UGC Act 1956)
Pollachi Main Road, Eachanari (Po),
Coimbatore –641 021

CLASS: II B.Sc. Physics **COURSENAME: Mathematics Practical-II**

COURSE CODE: 18PHU414 **BATCH-2018-2021**

LAB MANUAL

CONTENTS

EX. NO : 1

Plotting of second order solution family of differential equation

Question:

Plotting of second order solution family of differential equation using Scilab.

Aim:

To plot second order solution family of differential equation using Scilab.

Algorithm:

Step 1: Start the programme.

Step 2: Define the domain of the function f(t,x).

Step 3: Give the values of dx(1) and dx(2) by real numbers..

Step 4: give the linespace value of t.

Step 5: Plot the function f.

Step 6: Save and execute the programme.

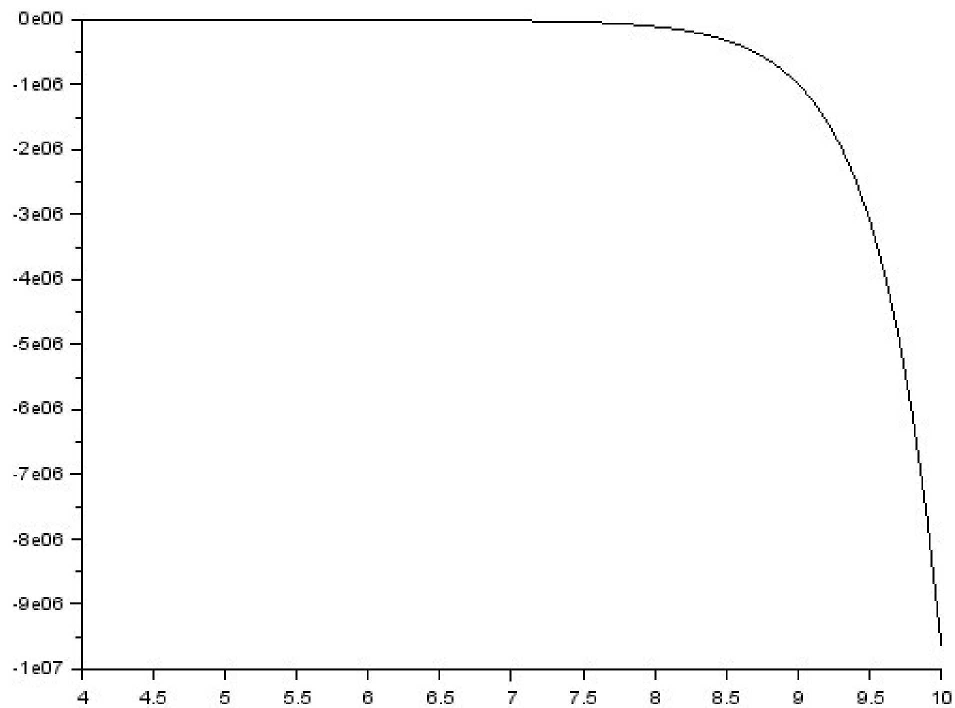
Step 7: Run the programme and view the output in graphic window.

Step 8: Stop the programme.

Coding:

```
function dx=f(t, x)
dx(1)=x(2);
dx(2)=-x(1)/2+5/2*x(2);
endfunction
t=4:0.1:10
sol=ode([6;-1],3,t,f);
disp(sol(1))
plot2d(t,sol(1,:))
```

Output:



Result:

Thus the programme has been executed successfully and the output has been verified.

EX. NO : 2

Growth model (exponential case only)

Question:

Plot the graphs of growth model in (exponential case) using Scilab.

Aim:

To plot the graphs of growth model in (exponential case) using Scilab.

Algorithm:

Step 1: Start the programme.

Step 2: Use linspace command to fix the values of t.

Step 3: Define the exponential function xt.

Step 4: Display the values of xt

Step 5: Use plot2d command to plot the graphs of functions.

Step 6: Save and execute the programme.

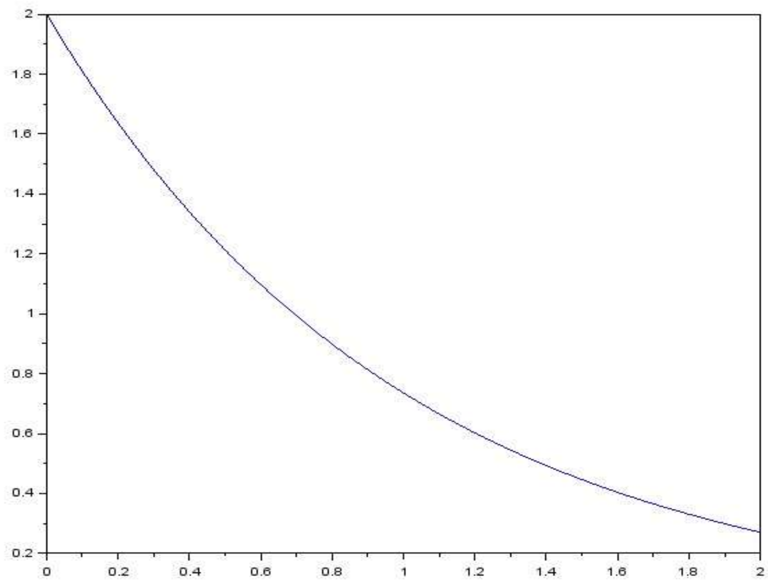
Step 7: Run the programme and view the output in graphic window.

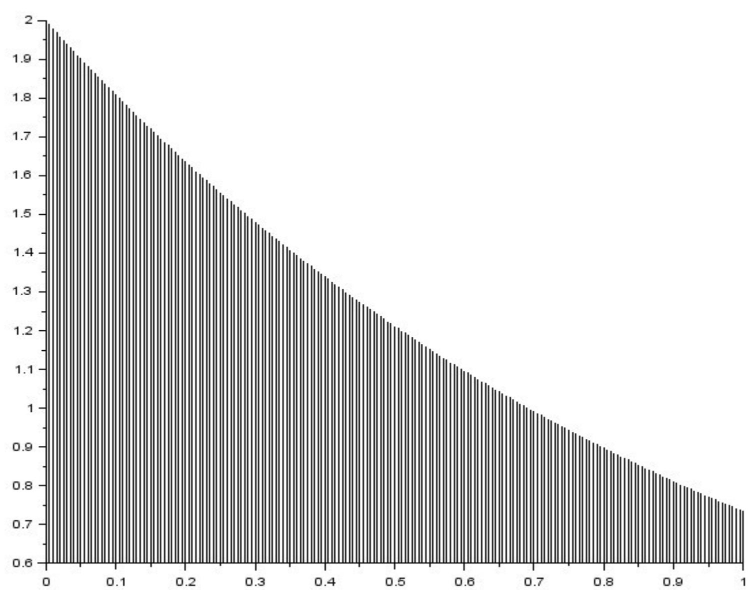
Step 8: Stop the programme.

Coding:

```
clc;
t=0:0.01:2;
xt=2*exp(-1*t);
subplot(1,2,1);
plot(t,xt);
T=2;
n=t/T;
xn=2*exp(-1*n);
subplot(1,2,2);
plot2d3(n,xn);
```

Output:





Result:

Thus the programme has been executed successfully and the output has been verified.

EX. NO : 3

Decay model (exponential case only)

Question:

Decay model (exponential case only) using Scilab.

Aim:

To solve the Decay model problem (exponential case only) using Scilab.

Algorithm:

Step 1: Start the programme.

Step 2 : Define the Value of Number of atoms in 10e-10kg.

Step 4: Apply the Half life t_h value.

Step 5: Apply the activity value A=N*D and power produced by one dps.

Step 6: Save and execute the programme.

Step 7: Run the programme and view the output for print value in the console page .

Step 8: Stop the programme.

Coding:

```
N=2.87e+019;
t_h=138*24*3600;
D=0.693/t_h;
A=N*D;
E=5.3*1.6E-013;
P=A*E;
printf("\n the power produced by 1.667e+012 dps : %3.1f W",P)
```

Output:

-->the power produced by 1.667e+012 dps : 1.4 W

Result:

Thus the programme has been executed successfully and the output has been verified.

EX. NO : 4

Solving first order ordinary differential equations

Question:

Evaluate the Solving first order ordinary differential equations using Scilab.

Aim:

To evaluate the Solving first order ordinary differential equations using Scilab.

Algorithm:

Step 1: Start the programme.

Step 2: Define the function f(x,y).

Step 3: Apply dx value.

Step 4: Given initial conditions of the problem.

Step 5: Save and execute the programme.

Step 6: Run the programme and view the output in console.

Step 7: Stop the programme.

Coding:

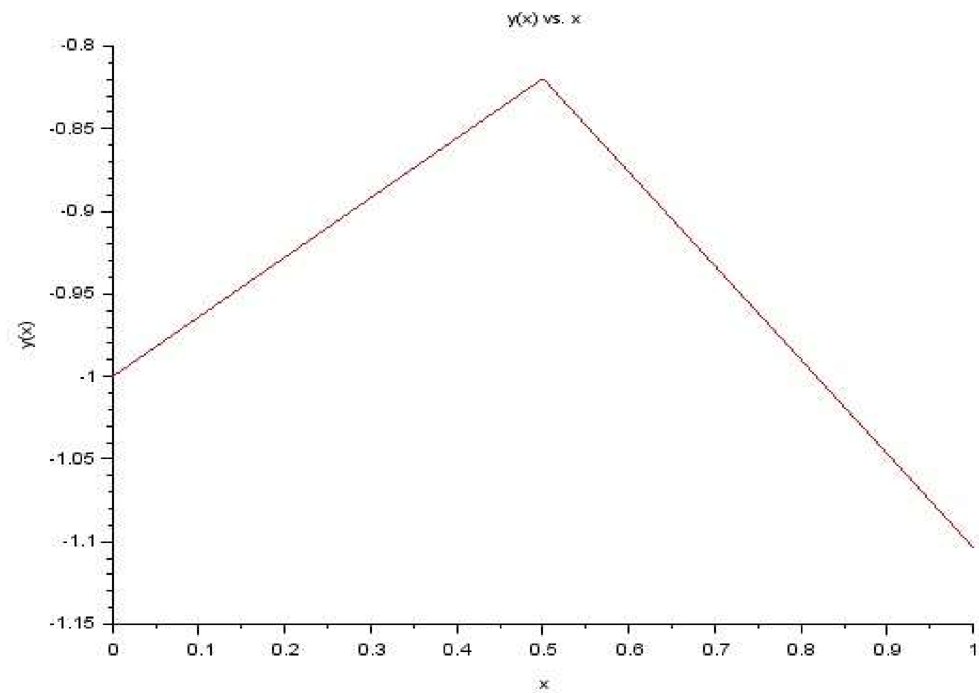
```
funcprot(0)
function dx=f(x, y)
dx=-2*x-y;
endfunction
y0=-1;
x0=0;
x=[0:0.5:1];
sol=ode(y0,x0,x,f);
disp(sol,"answer");
plot2d(x,sol,5)
```

```
xlabel('x');
ylabel('y(x)');
xtitle('y(x) vs. x');
```

Output:

→ **Answer**

-1.   -0.8195921   -1.1036384



Result:

Thus the programme has been executed successfully and the output has been verified.

EX. NO : 5

Solution of second order ordinary differential equations with initial conditions

Question:

Solution of second order ordinary differential equations with initial conditions using Scilab.

Aim:

To evaluate Solution of second order ordinary differential equations with initial conditions using Scilab.

Algorithm:

Step 1: Start the programme.

Step 2: Define the function f(x,y).

Step 3: Apply dx(1) and dx(2) value.

Step 4: Given line space value.

Step 5: Given initial conditions of the problem.

Step 6: Save and execute the programme.

Step 7: Run the programme and view the output in console.

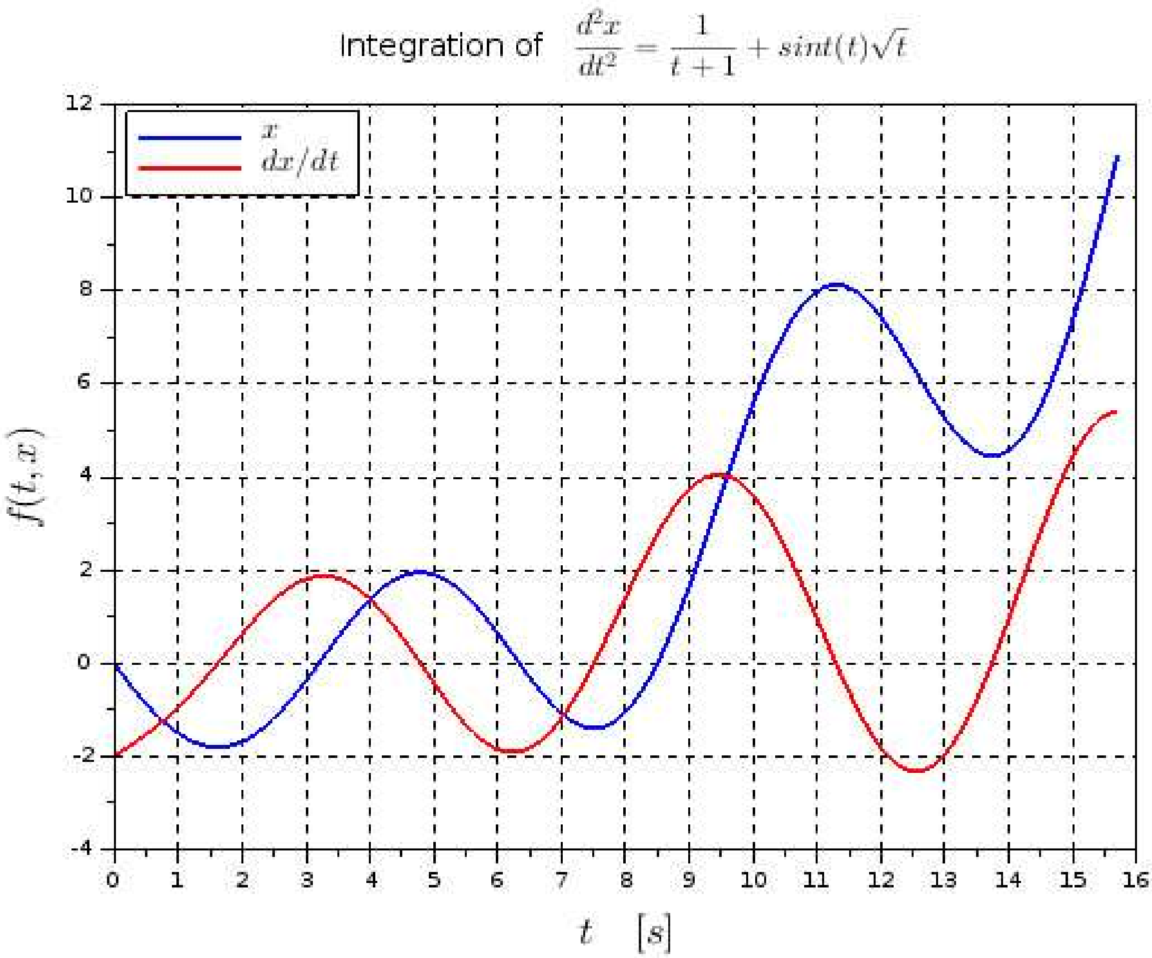
Step 8: Stop the programme.

Coding:

```
function dx=f(t, x)
 dx(1) = x(2);
 dx(2) = 1/(t+1) + sin(t)*sqrt(t);
endfunction
t = 0:0.01:5*%pi;
t0 = min(t);
y0 = [0; -2];
y = ode(y0, t0, t, f);
plot(t,y(1,:),'LineWidth',2)
```

```
plot(t,y(2,:),'r','LineWidth',2)
xgrid();
xlabel('$t \quad [s]$','FontSize',3)
ylabel('$f(t,x)$','FontSize',3)
title(['Integration of ' '$\frac{d^2 x}{dt^2} = \frac{1}{t+1} +
sint(t)\sqrt{t}$'],'FontSize',3)
legend(['$\Large{x}$' '$\Large{dx/dt}$'],2)
```

Output:



Result:

Thus the programme has been executed successfully and the output has been verified.

EX. NO : 6

Solving system of linear differential Equations

Question:

Solving system of linear differential Equations using Scilab.

Aim:

To Solving system of linear differential Equations using Scilab.

Algorithm:

Step 1: Start the programme.

Step 2: Apply the system of equation coefficient value.

Step 3: Define the eigen value and eigen vector value.

Step 4: Display the line solving value.

Step 5: Save and execute the programme.

Step 6: Run the programme and view the output in console.

Step 7: Stop the programme.

Coding:

```
clc;
A=[1 0 1;1 1 -1;5 1 1]
[c,d]=spec(A);
disp(spec(A),"the eigenvalues of the matrix A arc:")
disp(c,"the corresponding eigen vector is:")
x=c; y=[1;2;3]
B=linsolve(x,y)
S=x*[B B B]
disp(S,'s=',B,'B=')
```

Output:

the eigenvalues of the matrix A arc:

3.1149075

- 0.8608059

0.7458983

the corresponding eigen vector is:

```
- 0.4170021  - 0.3827458    0.1983289
  0.2198294    0.5884340  - 0.9788391
- 0.8819208    0.7122156  - 0.0503957
```

B=

```
3.5181246
0.3600066
3.049763
```

s=

```
- 1.  - 1.  - 1.
- 2.  - 2.  - 2.
- 3.  - 3.  - 3.
```

Result:

Thus the programme has been executed successfully and the output has been verified.

EX. NO : 7

Computing Lagrange's interpolating polynomial

Question:

Computing Lagrange's interpolating polynomial using Scilab.

Aim:

To Computing Lagrange's interpolating polynomial by using Scilab.

Algorithm:

Step 1: Start the programme.

Step 2: Define the domain of the lagranges function P.

Step 3: Define the domain of the function X nodes,Y values.

Step 4: Use n is the number of nodes. (n-1) is the degree.

Step 5: Save and execute the programme.

Step 6: Run the programme and view the output.

Step 7: Stop the programme.

Coding:

```
function [P]=lagrange(X, Y)
n=length(X);//
x=poly(0,"x");P=0;
for i=1:n, L=1;
 for j=[1:i-1,i+1:n] L=L*(x-X(j))/(X(i)-X(j));end
 P=P+L*Y(i);
end
endfunction
```

Output:

```
-->X=[0;2;4];Y=[1;5;17];P=lagrange(X,Y)
```

$$P = 1 + x^2$$

Result:

Thus the programme has been executed successfully and the output has been verified.

EX. NO : 8

Computing interpolating polynomial using Newton's formula

Question:

Computing interpolating polynomial using Newton's formula.

Aim:

To Computing interpolating polynomial using Newton's formula in Scilab.

Algorithm:

Step 1: Start the programme.

Step 2: Define the domain of the newton function value.

Step 3: Define the domain of the function and store its length i and j value.

Step 4: Use command to print the output value.

Step 5: Save and execute the programme.

Step 6: Run the programme and view the output.

Step 7: Stop the programme.

Coding:

```
function [P]=newton(X, Y)
n=length(X);
for j=2:n,
for i=1:n-j+1,Y(i,j)=(Y(i+1,j-1)-Y(i,j-1))/(X(i+j-1)-X(i));end
end,x=poly(0,"X");
P=Y(1,n);
for i=2:n,P=P*(x-X(i))+Y(i,n-i+1);end
endfunction
```

Output:

```
-->X=[0;2;4];Y=[1;5;17];P=newton(X,Y)
```

$P = 1 + x^2$

Result:

Thus the programme has been executed successfully and the output has been verified.