**Experiment- 1(A)**

**Title of the Experiment:** To study about ODEs(ordinary differential equations) in Controls System and their solving techniques in Matlab.

**Software:** Matlab

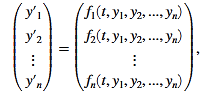
**Theory:**

A control system commands direct or regulates the behavior of other devices or systems using control loops. It can range from a single home heating controller using a thermostat controlling a domestic boiler to large Industrial control systems which are used for controlling processes or machines. Control Systems of the single input single output linear time-invariant systems are often associated with ordinary differential equations, the valued entity to be controlled is often varied with time as a summation of its derivatives and itself. There are two common classes of control action: open loop and closed loop. In an open-loop control system, the control action from the controller is independent of the process variable. In a closed-loop control system, the control action from the controller is dependent on the desired and actual process variable. In the case of the boiler analogy, this would utilize a thermostat to monitor the building temperature, and feedback a signal to ensure the controller output maintains the building temperature close to that set on the thermostat. A closed-loop controller has a feedback loop which ensures the controller exerts a control action to control a process variable at the same value as the set point. For this reason, closed-loop controllers are also called feedback controllers.

**Ordinary Differential Equations:** An ordinary differential equations (ODE) contains one or more derivatives of a dependent variable, y, w.r.t a single independent variable t, usually refered to as a time. Commonly used ODE Solving Functions are:

1. Ode45
2. Ode23
3. Ode113
4. Ode15s
5. Ode23s
6. Ode23t
7. Ode15i

**Systems of ODEs:** You can specify any number of coupled ODE equations to solve, and in principle the number of equations is only limited by available computer memory. If the system of equations has n equations,



**Using Ode45:** Solve non-stiff differential equations — medium order method.

**Syntax:**

**Description**: where  integrates the system of differential equations from with initial conditions . Each row in the solution array y corresponds to a value returned in column vector t.

**Example**: Solve the ODE use a time interval of [0,5] and initial conditions

**Code:**

clc;

clear all;

close all;

tspan = [0 5];

y0 = 0;

[t,y] = ode45(@(t,y) 2\*t,tspan,y0);

plot(t,y);

title('Analysis of ODE using @ode45');

xlabel('time(s)'),ylabel('y(t)');

**Output:**



**Using Ode23:** ode23 can be more efficient than ode45 at problems with crude tolerances, or in the presence of moderate stiffness.

**Syntax:**

**Description**: where  integrates the system of differential equations from with initial conditions . Each row in the solution array y corresponds to a value returned in column vector t.

**Example**: Solve the ODE use a time interval of [0,5] and initial conditions

**Code:**

clc;

clear all;

close all;

tspan = [0 5];

y0 = 0;

[t,y] = ode23(@(t,y) 2\*t,tspan,y0);

plot(t,y);

title('Analysis of ODE using @ode23');

xlabel('time(s)'),ylabel('y(t)');

**Output:**



**Using Ode113:** ode113 can be more efficient than ode45 at problems with stringent error tolerances, or when the ODE function is expensive to evaluate.

**Syntax:**

**Description**: where  integrates the system of differential equations from with initial conditions . Each row in the solution array y corresponds to a value returned in column vector t.

**Example**: Solve the ODE . use a time interval of [0,5] and initial conditions

**Code:**

clc;

clear all;

close all;

tspan = [0 5];

y0 = 0;

[t,y] = ode113(@(t,y) 2\*t,tspan,y0);

plot(t,y);

title('Analysis of ODE using @ode113');

xlabel('time(s)'),ylabel('y(t)');

**Output:**



**Using Ode15s:** Try ode15 and ode45 fails or is inefficient and you suspect that the problem is stiff. Also use ode15s when solving differential algebraic equations(DAEs).

**Syntax:**

**Description**: where  integrates the system of differential equations from with initial conditions . Each row in the solution array y corresponds to a value returned in column vector t.

**Example**: Solve the ODE . use a time interval of [0,5] and initial conditions

**Code:**

clc;

clear all;

close all;

tspan = [0 5];

y0 = 1;

[t,y] = ode15s(@(t,y) -10\*t,tspan,y0);

plot(t,y);

title('Analysis of ODE using @ode15s');

xlabel('time(s)'),ylabel('y(t)');

**Output:**

****

**Using Ode23s:** ode23s can be more efficient than ode15s at problems with crude error tolerances. It can solve some stiff problems for which od15s is not effective. Ode23s computes the Jacobian so it is beneficial to provide the jacobian via odeset to maximize efficiency and accuracy.

**Syntax:**

**Description**: where  integrates the system of differential equations from with initial conditions . Each row in the solution array y corresponds to a value returned in column vector t.

**Example**: Solve the ODE use a time interval of [0,5] and initial conditions

**Code:**

clc;

clear all;

close all;

tspan = [0 5];

y0 = 1;

[t,y] = ode23s(@(t,y) -10\*t,tspan,y0);

plot(t,y);

title('Analysis of ODE using @ode23s');

xlabel('time(s)'),ylabel('y(t)');

**Output:**



**Using Ode23t:** use ode23t if the problem is only moderately stiff and you need a solution without numerical damping.

**Syntax:**

**Description**: where  integrates the system of differential equations from with initial conditions . Each row in the solution array y corresponds to a value returned in column vector t.

**Example**: Solve the ODE. use a time interval of [0,5] and initial conditions

**Code:**

clc;

clear all;

close all;

tspan = [0 5];

y0 = 1;

[t,y] = ode23t(@(t,y) -10\*t,tspan,y0);

plot(t,y);

title('Analysis of ODE using @ode23t');

xlabel('time(s)'),ylabel('y(t)');

**Output:**



**Using Ode15i:**

**Syntax:**

**Description**: where  integrates the system of differential equations from with initial conditions . Each row in the solution array y corresponds to a value returned in column vector t.

**Example:** Solve the ODE use a time interval of [0,5] and initial conditions

**Code:**

clc;

clear all;

close all;

tspan = [0 5];

y0 = 1;

[t,y] = ode15i(@(t,y) -10\*t,tspan,y0);

plot(t,y);

title('Analysis of ODE using @ode15i');

xlabel('time(s)'),ylabel('y(t)');

**Output:**

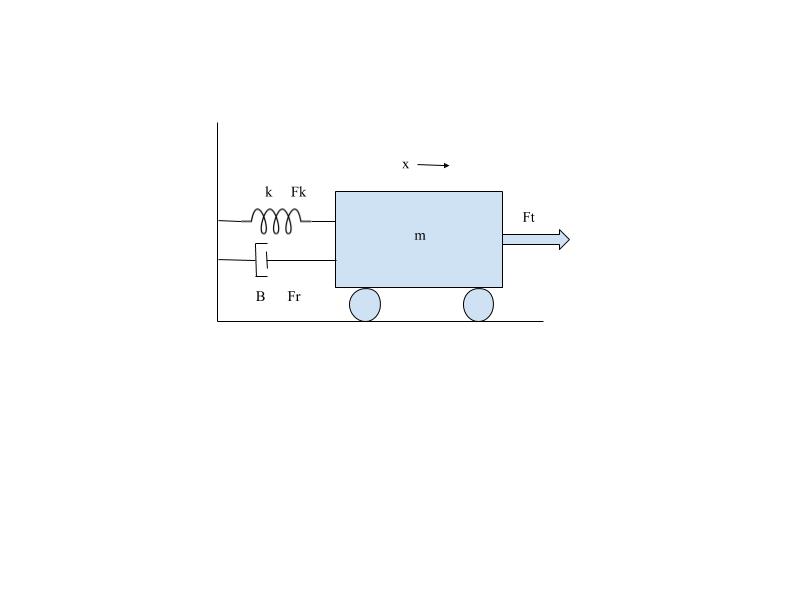


**Conclusions:** Various ODE solving functions were studied and solved using Matlab. It was observed that for different conditions such as stiffness and damping of different system equations there are various ODE’s functions from which we can solve them.

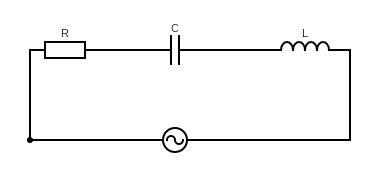
**Experiment 1(B)**

**Title of the experiment:**

1. To solve the ode equation: .
2. To model the physical system using 2nd order ODE.



1. Assume: form a 1st order ODE.
2. Using Mupad solve the 1st order ODE with initial condition
3. Plot ‘v’ vs ‘t’ in MATLAB for
4. Plot using @ode45 Function.



.

1. To model the given RLC circuit in series model in Matlab to a 2nd order ODE.
2. Solve using Mupad for given values.
3. Plot for t: 0 to 200 sec.

**Software Used:** Matlab 2018

**Theory:**

1. **Mupad**: Mupad is a GUI driven MATLAB package that helps you do algebra, calculus, as well as to graph and visualize functions. As you know, MATLAB is good for writing simple programs and working with numbers, but is cumbersome for doing symbolic calculations. Mupad is useful here.
2. **ode@45:** Syntax »

Where tspan = [t0 tf], integrates the system of differential equations y′=f (t,y) from  with initial conditions y0. Each row in the solution array y corresponds to a value returned in column vector t.

 additionally finds where functions of , called event functions, are zero. In the output,  is the time of the event, is the solution at the time of the event, and  is the index of the triggered event.

**Code 1:**

eq:= ode({y(x) = x^2 + x + 4},y(x))

solve(eq)

**Output:**

**Code 2:**

1. Balancing forces on the free body diagram we get: and Substituting the given and , we get: .
2. **Using Mupad:**

eq:= ode({25\*v'(t) + v(t) = 10,v(0)=0},v(t))

solve(eq)

1. **Using Editor:**

clc;

close all;

clear all;

syms t;

v(t) = 10 - 10\*exp(-t/25);

t = 0:200;

plot(t,v(t));

title('Analysis of Physical System');

xlabel('time(s)'),ylabel('v(t)');

d) **Using @ode45:**

function dv = asdf(t,v)

dv = (10-v)/25;

end

clc;

close all;

clear all;

[t,v] = ode45(@asdf,0:200,0);

plot(t,v);

title('Analysis of Physical System using @ode45');

xlabel('time(s)'),ylabel('v(t)');

**Output:**



**Conclusions:** 2nd order derivative equation were derived for given Physical System, 1st ODE solved using mupad editor and graphs were plot for the same using ode45 function and matlab text editor.

**Code 3:**

1. Adding the voltages to with same current we get:

Using initial conditions we get:

1. **Using Mupad:**

eq:= ode({500\*sin(50\*t) + i’(t) + 2\*i’’(t) + i(t) = 0,i(0)=0,i’(0) = 0},i(t))

solve(eq)

1. **Using Editor:**

clc

clear all

close all

syms t;

i(t) = cos((7^(1/2)\*t)/4)\*((12500\*cos(50\*t - (7^(1/2)\*t)/4))/24992501 … - (499925000\*7^(1/2)\*exp(-t/4)\*sin((7^(1/2)\*t)/4))/174947507;

t = 0:1:200;

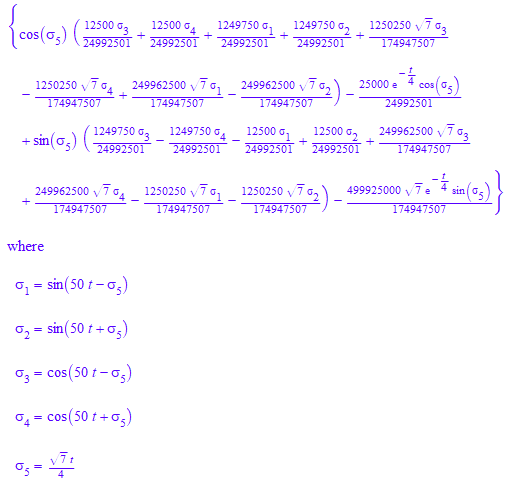
plot(t,i(t));

title('Analysis of Electrical System');

xlabel('time(s)'),ylabel('i(t)');

**Output:**





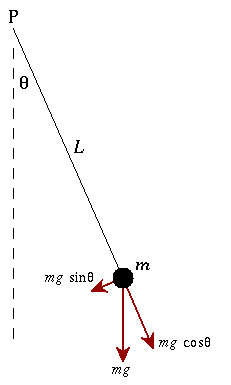


**Conclusions:** 2nd order derivative equation were derived for given System, solved using mupad editor and graph was plot for the same. The characteristics of simple RLC circuit were analyze using Mupad Editor and Matlab.

**Assignment 1**

**Aim:**

1. Assume .



1. To model the given physical system to a 2nd order ODE.
2. Solve the 2nd Order Differential Equation.
3. Plot for t: 0-200

**Code:**

1. Balancing forces on the free body diagram we get: and assuming the values of respectively and

, we get: . Also assuming tends to small value.

1. **Using Mupad:**

eq:= ode({q''(t) = -4.8\*q(t),q(0) = PI/6,q'(0) = 0},q(t))

x(t) :=solve(eq)

1. **Using Editor:**

Clc;

clear all;

close all;

syms t;

theta(t) = 0.5235987756\*cos(2.19089023\*t);

t = 0:0.01:2\*pi;

plot(t,theta(t));

title('Analysis of Physical System');

xlabel('time(s)'),ylabel('theta(t)');

**Output:**





****

**Conclusions:** 2nd order derivative equation were derived for given free pendulum System, solved using mupad editor and graph was plot for the same. It was observed that the pendulum slow down its oscillation over time due damping force acting on it. Hence, after sometime the pendulum starts to come in its steady state.