Table of Contents

Lab 02

Simulating Dynamic Models using MATLAB

2.1 Objective

To simulate the mathematical model of dynamic systems using MATLAB

2.2 Pre-Lab Task

Task 1: Answer the following

a) What is meant by the order of a differential equation?
b) What are linear and non-linear differential equations? Write a few examples of each.
c) What is meant by solving a differential equation?

2.3 Introduction

The first step in any control design process is to build a mathematical model for the system. Once the mathematical model is obtained, it can be simulated to understand the system's behavior and performance. When we say, we want to simulate our system, we're actually looking to solve Ordinary Differential Equations (ODEs) with initial conditions. This is because continuous-time systems are mostly modeled using ODEs or PDEs (i.e. Partial Differential Equations).

Solving a differential equation means finding an equation with no derivatives that satisfies the given differential equation. Often the solutions of such equations require a substantial amount of time and effort by the student. The mathematical solution of these equations does not readily provide the student with a "graphical picture" of the result. Therefore, students are uncomfortable with the entire process of solving these types of systems. On the other hand, using software like MATLAB and Simulink for solution of differential equations is very quick and easy. Both linear and nonlinear differential equations can be solved numerically with high precision and speed, allowing system responses to be calculated and graphically represented for different input functions.

There is a set of popular iterative methods used by different software to accomplish this task. These methods are called Runge-Kutta methods. You are encouraged to look into these methods.

2.4 Solving Differential Equations Using MATLAB

MATLAB's standard solver for ordinary differential equations (ODEs) is the function **ode45**. This function implements a Runge-Kutta method with a variable time step for efficient computation. ode45 is designed to handle the following general problem:

$$\frac{d}{dt}y(t) = f(t,y) \qquad y(t_0) = y_0 \tag{1}$$

Where, t is the independent variable, y is a vector of dependent variables to be found and $f(t,y)$ is a function of t and y.

Syntax for ode45 function: $[t,y] = \text{ode45}(\text{odefun}, t_{span}, y_o)$

Where, $t_{span} = [t_o \quad t_f]$ shows time span to integrate the system of differential equations $y' = f(t,y)$ from t_o to t_f with initial conditiony_o. Each row in the solution array y corresponds to a value returned in column vector t. odefun is the name of the function file which defines the functions to be integrated as a function handle.

Example: To solve the equation $y' = 5t - 3$, consider the following MATLAB script. A function file is first created for the differential equation, and then ode45 command is used to obtain solution. The corresponding plot of obtained solution shown in Figure 1 represents the function y that satisfies the given differential equation. Try this example.

MATLAB Script:

```
function dydt=odefun(t,y)
dydt=(5*t)-3;
end
%save the above function file naming it odefun.m

%Try the following commands
[t,y]= ode45('odefun',[0 5], 0); %time span [0 5], initial condition y(0)=0
```

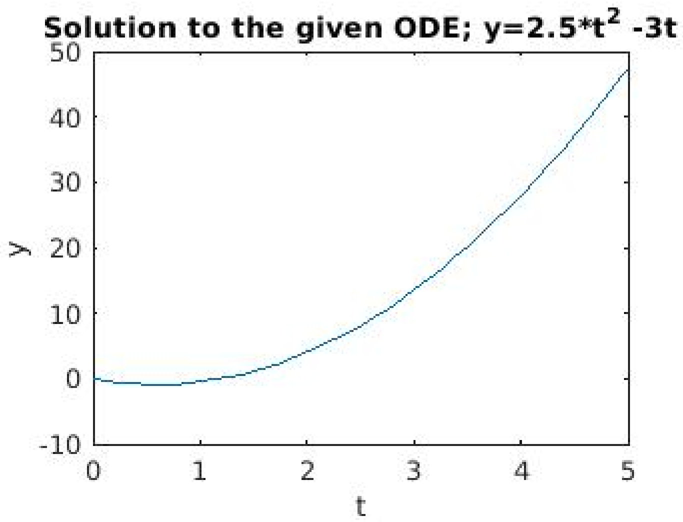


Figure 1: Plot representing solution to the given ODE solved by ode45

You can verify that the analytical solution to this ODE is $y = 2.5t^2 - 3t$ for the initial condition $y(0) = 0$ and the obtained plot represents the same function.

To solve the higher order differential equations, the differential equation is decomposed in a set of first order differential equation and then solved using ode45. Such an example will be practiced later in this lab.

2.5 Mathematical Model of Mass-Spring-Damper System

Mechanical systems obey Newton's law that the sum of the applied forces must be equal to the sum of the reactive forces. The three qualities characterizing elements in a mechanical translation system are mass, elastance, and damping. Consider the mass-spring-damper system shown in Figure 2.

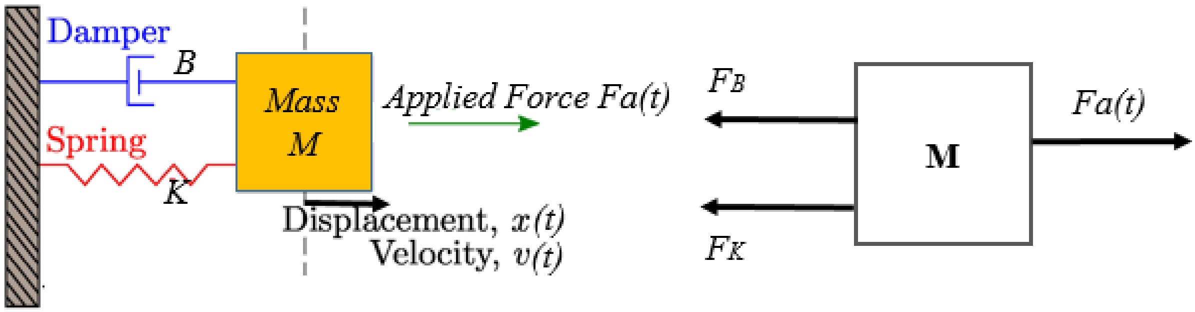


Figure 2: Mass-Spring-Damper system with free body diagram

Here, a force $F_a(t)$is applied on mass M which results in its translational motion. The mass M is the inertial element. A force applied to a mass produces an acceleration of the mass. The force F_M is equal to the product of mass and acceleration. Let, $x(t)$ denotes the displacement of mass, $v(t)$ be its speed and $a(t)$ be its acceleration. Then, this force producing acceleration can be expressed as

$$F_M = Ma(t) = M\frac{d}{dt}v(t) = M\frac{d^2}{dt^2}x(t) = M\ddot{x}(t) \quad (1)$$

As the mass is connected with support through a spring and a damper, therefore, there are two more forces in action.

A spring is governed by the Hooke's law which states that within certain limits, the force required to stretch an elastic object such as a metal spring is directly proportional to the extension of the spring. Therefore, the restoring force F_K for the above spring with spring constant or stiffness $\boldsymbol{K}$ is given as

$$F_K = Kx(t) \quad (2)$$

The damper is a dissipative device which consumes energy of the system through viscous friction. This frictional or damping force F_B is proportional to velocity. For a damper with damping coefficient $\boldsymbol{B}$, the force is given as

$$F_B = Bv = B\frac{d}{dt}x(t) \quad (3)$$

This is also known as the *drag* force as per the Stokes' law.

Applying Newton's law on the given system using free-body diagram, we get

$$F_M + F_B + F_K = F_a(t) \quad (4)$$

Substituting F_K, F_M and F_B in the above equation, we get the following differential equation for the Mass-Spring-Damper System.

$$Ma + Bv + Kx = F_a(t) \tag{5}$$

$$M\frac{d^2}{dt^2}x(t) + B\frac{d}{dt}x(t) + Kx(t) = F_a(t) \tag{6}$$

This is a second order differential equation which describes the relationship between the displacement and the applied force. The differential equation can then be used to study the time behavior of $x(t)$ under various changes of the applied force. Note that (6) is a second-order system because there are two energy storage elements – the mass and the spring.

2.6 Modelling Speed Cruise Control

Automatic cruise control is an excellent example of a feedback control system found in many modern vehicles. The purpose of the cruise control system is to maintain a constant vehicle speed despite external disturbances, such as changes in wind or road grade. This is accomplished by measuring the vehicle speed, comparing it to the desired or reference speed, and automatically adjusting the throttle according to a control law. Figure 3 shows different forces acting on a vehicle.

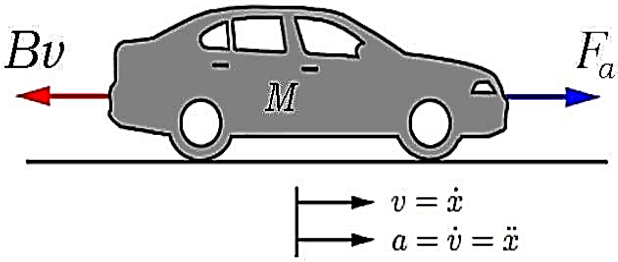


Figure 3: Representation of forces acting on a vehicle

As compared to the mass-spring system, the cruise system has no spring force i.e., $F_s(x) = 0$ which means that $K = 0$. Equation (6) becomes

$$M\frac{d^2}{dt^2}x(t) + B\frac{d}{dt}x(t) = F_a \tag{7}$$

Or,

$$M\frac{d}{dt}v(t) + Bv(t) = F_a \tag{8}$$

The equation (8) reveals that a cruise speed control system is first-order in nature and the reason is that there is only one energy storage element which is the mass of the vehicle. From (7), it appears that the system is second-order; however, this is due to the fact that one of a state, the position $x(t)$, is just redundant.

Task 2: To simulate the mathematical model of speed cruise control

To solve the differential equation (8), use ode45 as explained in the example.

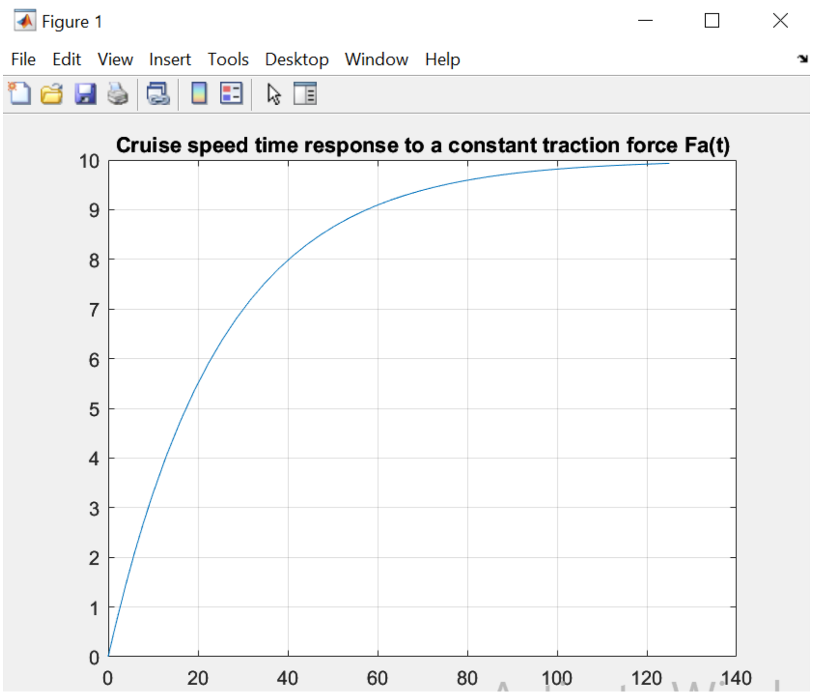
1. Create a MATLAB-function cruise_speed.m and represent the differential equation in terms of parameters; mass M, friction coefficient B and applied force Fa.

```
function dvdt=cruise_speed (t,v)
M=750;           % (Kg)
B=30;            % (Nsec/m)
Fa=300;          % (N)
```

2. Create a new MATLAB m-file and write the following piece of code:

```
v0=0;   % initial speed
[t,v]= ode45('cruise_speed', [0 125], v0);
plot(t,v); grid on;
```

After running the code, the following plot was obtained



3. Run the code and observe the speed represented by the obtained plot. By changing different parameters, you can observe their impact on the steady-state values of speed, and the time required to reach the steady-state.

4. Here, the parameters M, B and Fa are initialized in cruise_speed.m function. We can use them as input arguments of the function. In that case, we use ode45 command multiple times with different values of parameters in the same script through which the resulting plots can be obtained in one figure. To do this, modify the cruise_speed.m function as follows:

```
function dvdt=cruise_speed (t, v, Fa, M, B)
dvdt = Fa/M - (B/M)*v;
```

With this function, use the given variant of ode45.

Syntax:

[t,v]=ode45(@(t,v) odefun(t,v,a1,a2,a3), [time span], initial value);

Note that a1,a2,a3… are the input arguments of function.

Example:

[t,v] = ode45(@(t,v) cruise_speed(t,v,150,750,30), [0 300], v0);

5. Now to observe the effect of each parameter, you will vary the mass, applied force, and damping coefficient, one by one (keeping the other parameters constant) and compare the response with the one obtained with actual parameters. This can be done by using ode45 each time with different value of a parameter and plotting the response with command: **hold on**

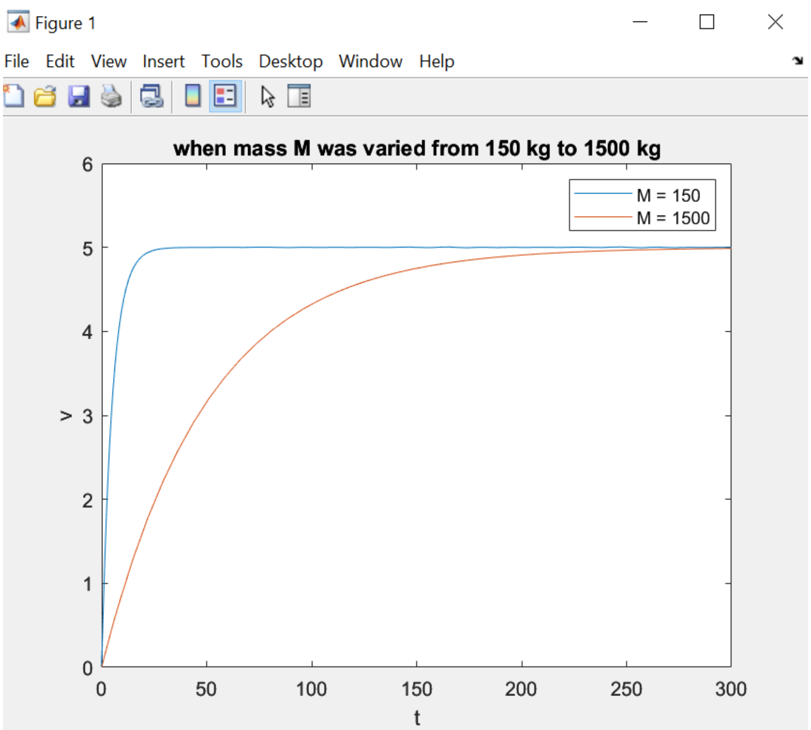
   Make sure that the plots have suitable title and legends.
   Remember: Without suitable labels and title, plots are just random lines.

6. Conclude the effect of each parameter on cruise speed (steady-state speed and time taken to reach the steady-state). You may use stepinfo(v,t) to obtain the information regarding each response.

7. Hints for analysis:
   - Relate mass with inertia and observe its impact on the magnitude of cruise speed in steady-state and the time taken to reach the steady-state.
   - To find the parameters that affect the final value of cruise-speed, consider that the acceleration will be zero at steady-state (constant velocity). Apply this on equation (8).

   For each of the following cases, add capture of plots with your analysis.

   a. M= 150 Kg and M= 1500 Kg. Provide commentary on your findings.

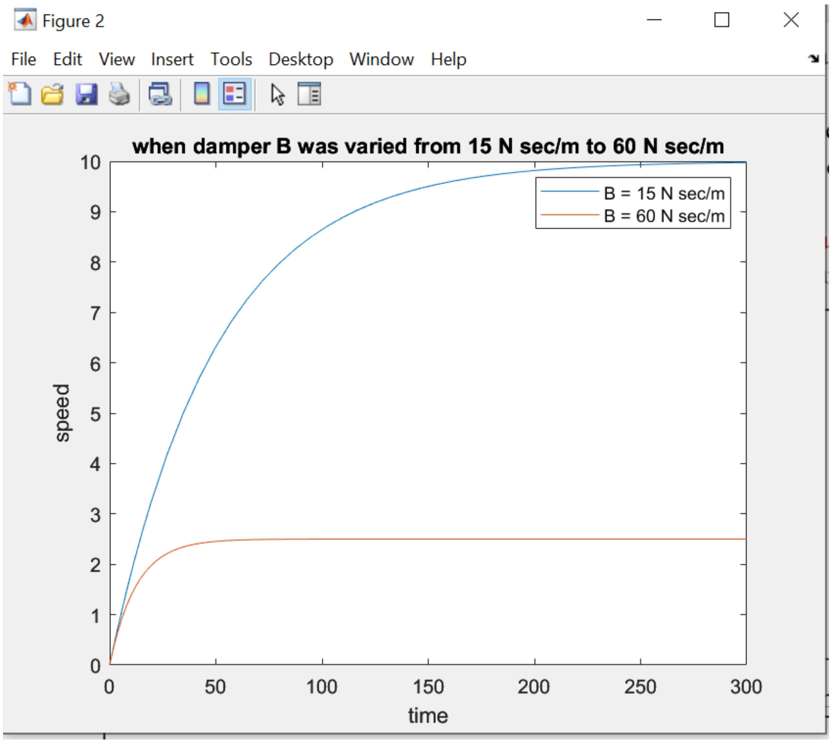


Commentary:

As can be seen from the graphs, the steady state with both masses is the same. This shows that the mass is not affecting the steady state. This is because the steady state velocity is reached when the drag force (B.v) equals the driving force. Therefore, the steady state velocity is independent of the mass and is equal to $(\frac{F_a}{B})$.

However, the mass does affect the transient response, and for a greater mass the transient response's duration is longer that is the settling time of the velocity is greater. The reason for this is that the greater mass has a greater inertia and therefore it takes greater time to accelerate it to the steady state velocity. So, m=150kg achieves steady state velocity faster.

b. B=15 N sec/m and B= 60 N sec/m. Provide commentary on your findings.

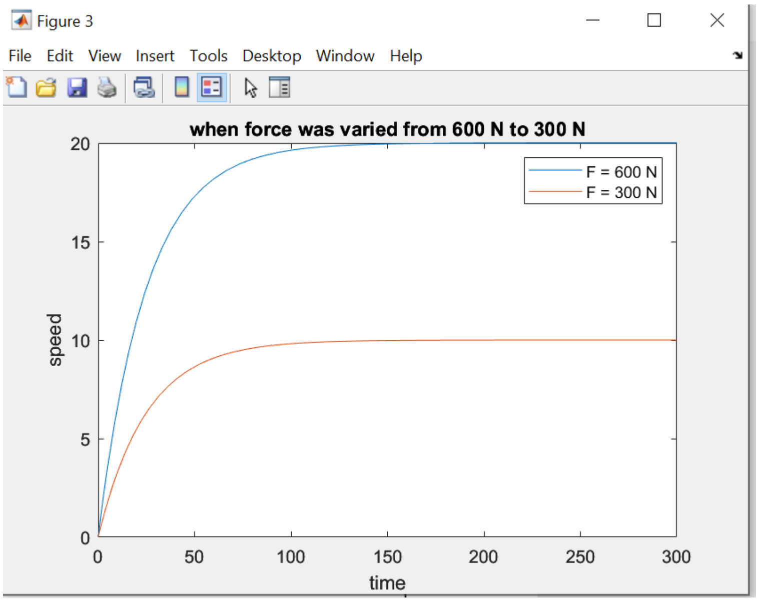


Commentary:

The graphs demonstrate that both the steady state and transient responses are unique. As was previously mentioned, the steady state velocity is first attained when the net force acting on the vehicle is zero, or when the drag force and driving force are equal. When that happens, v=F_a/b. The graph above illustrates how the steady state velocity would be lower for the higher drag coefficient.

Additionally, the higher drag force has a lower settling time or transient response. It's because the drag force multiplies more quickly as velocity rises due to the higher drag coefficient. As a result, for higher drag coefficients than for lower drag coefficients, the drag force equals the driving force more quickly, and as a result, has smaller transient response.

c. Fa= 600 N and Fa=300 N. Provide commentary on your findings.



The graph shows that varying the applied force or the driving force affects the steady state response. As discussed earlier, the steady state velocity is directly proportional to the driving force. Therefore, for the higher force, the steady state velocity is greater. This is shown in the graph above, by doubling the driving force, the steady state velocity was also doubled.

Task 3: To simulate the model of mass-spring system using matrix approach

To solve the second-order differential equation (6), it has to be decomposed in a set of first order differential equations as illustrated in Table 1.

Table 1: Relation between state variables and their derivatives

Variables	New Variable	Differential Equation
$x(t)$	X_1	$\frac{dX_1}{dt} = X_2$
$\frac{dx(t)}{dt}$	X_2	$\frac{dX_2}{dt} = -\frac{B}{M}X_2 - \frac{K}{M}X_1 + \frac{F_a}{M}$

In vector form, let

$$X = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$$

Taking derivative of both sides

$$\frac{d}{dt}X = \begin{bmatrix} \frac{d}{dt}X_1 \\ \frac{d}{dt}X_2 \end{bmatrix}$$

Then, the system can be written as

$$\frac{d}{dt}X = \begin{bmatrix} X_2 \\ -\frac{B}{M}X_2 - \frac{K}{M}X_1 + \frac{F_a}{M} \end{bmatrix}$$

ode45 solver can be now be used but instead of representing one differential equation in function, set of the above first order differential equations will be expressed in the function as two elements of a matrix *X*.

8. Create a MATLAB-function mass_spring.m

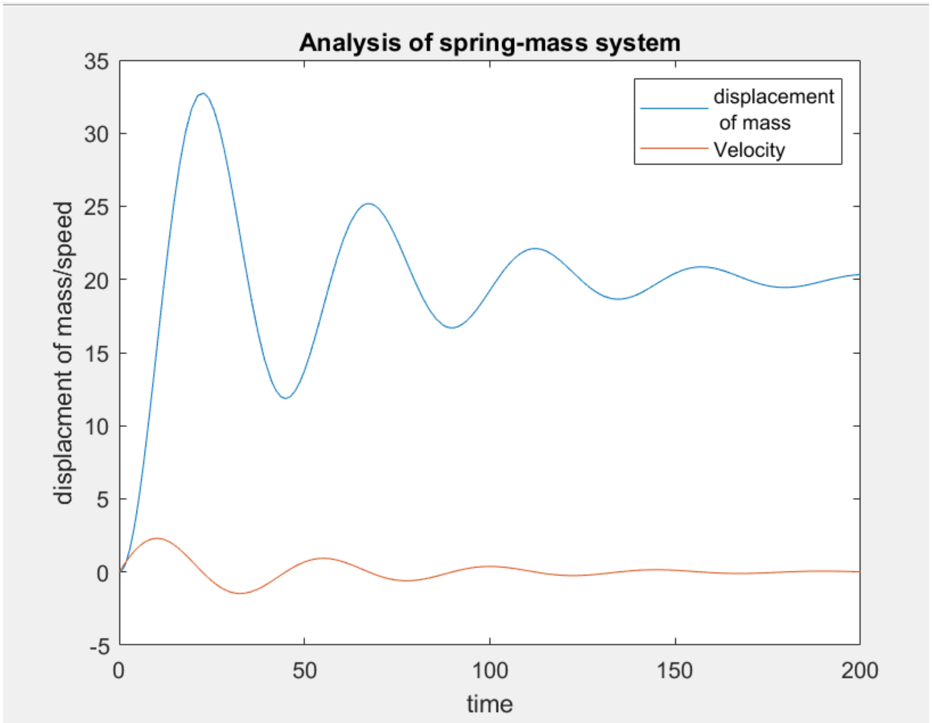
```
function dXdt=mass_spring(t,X)
M=750;          % (Kg)
B=30;           % (Nsec/m)
Fa=300;         % (N)
K=15;           % (N/m)
dXdt(1,1)= X(2);    %(differential equation 1)
```

9. Now write the following piece of code to solve the equations above using ode45.

```
X0= [0; 0]; % initial position and speed
Options = odeset('RelTol', [1e-4 1e-4], 'AbsTol', [1e-5 1e-5],'Stats', 'on');
```

10. Plot the solution of two differential equations returned in matrix X. Identify which one represents speed and which one represents the displacement of mass.

11. Analyze the results. Add captures of the obtained plots. Don’t forget to add labels and titles in plots.



As spring is displaced from the reference point, it oscillates around the reference. For increased displacement, velocity is also high and it gradually decreases to 0 as the spring loses energy and displacement from the reference point becomes 0. The displacement curve dies down till 20 since that is the relaxed length of spring.

Task 4: To simulate the response of a non-linear pendulum

Figure 4**Error! Reference source not found.** shows the forces acting on a non-linear pendulum where the tension in the string 'F_T' is acting along the string of length 'L' and the weight of mass 'm' is acting downward. 'θ' is the angle of rotation.

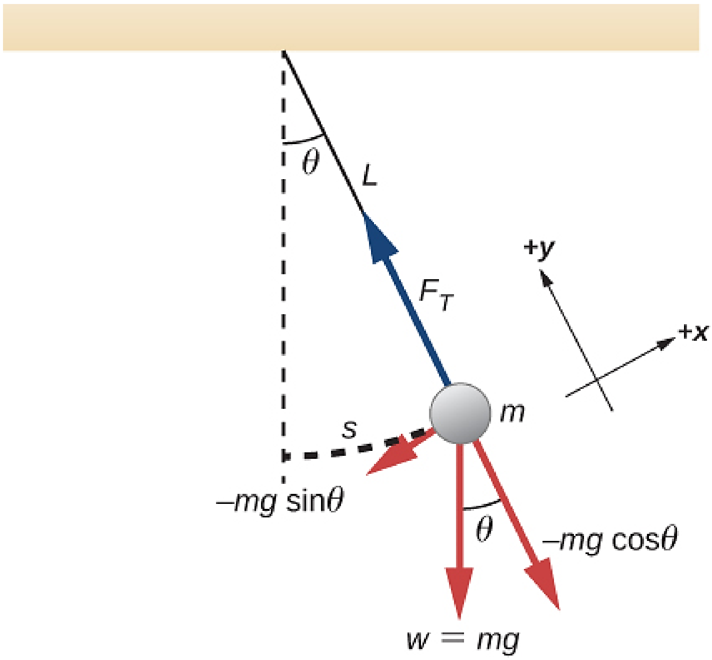


Figure 4: Forces acting on a pendulum

The system can be expressed by the second-order differential equations (9) and (10).

$$mL\frac{d^2}{dt^2}\theta = -mg\sin(\theta) \tag{9}$$

$$\frac{d^2}{dt^2}\theta = -\frac{g}{L}\sin(\theta) \tag{10}$$

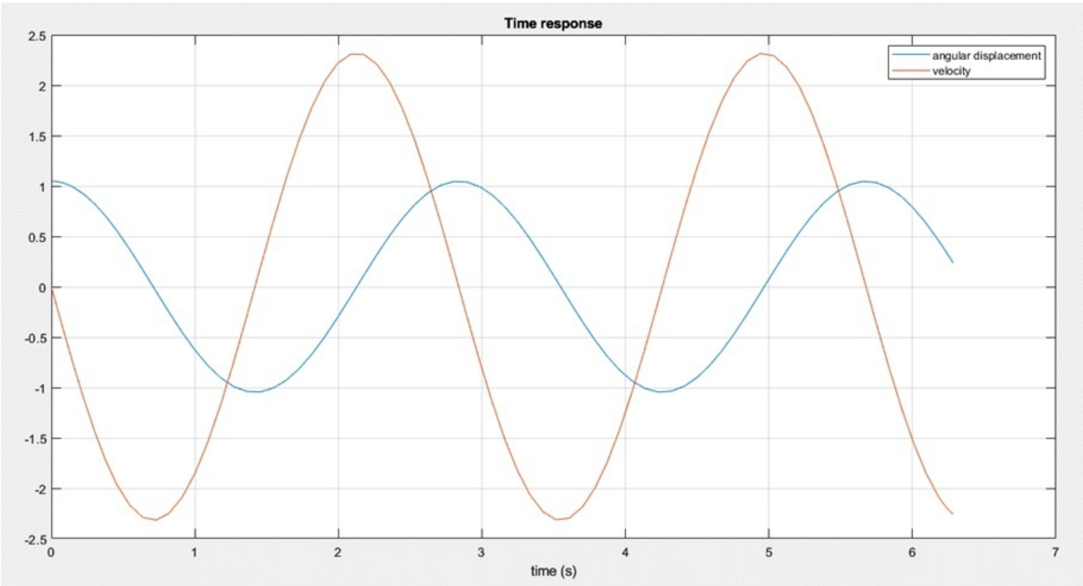
Where, θ is the angular displacement and $\frac{d^2}{dt^2}\theta$ is the angular acceleration. These equations are non-linear since there is a sine function with variable θ involved.

Decompose this second-order differential equation in a set of two first order differential equations as done in Task 3.

1. Write the equations in matrix form.

$$\frac{d}{dt}X = \begin{bmatrix} X_2 \\ -\frac{g}{L}\sin(X1) \end{bmatrix}$$

2. Create a MATLAB function to represent the set of differential equations with the given parameters: L= 2m, g=9.8m/s
3. Solve the equation using ode45 for time span of [0 2π] with initial value of angular displacement 'θ' to be $\frac{\pi}{3}$.
4. Plot angular displacement and velocity with respect to time. Explain the plots obtained.



CODE

```
function dXdt=mass_spring(t,X)
L=2;
g=9.8;
dXdt(1,1)= X(2);    %(differential equation 1)
dXdt(2,1)= -(g/L)*X(1); %(differential equation 2)
end
```

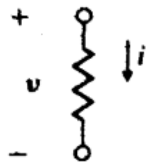
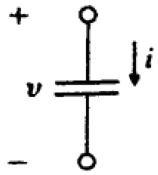
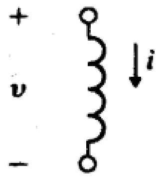
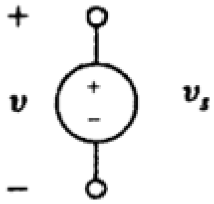
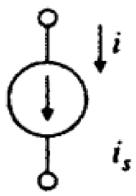
Explanation: The graph shows that the angular displacement and angular velocity keeps oscillating and the response is sinusoidal. We observe no damping in the displacement and velocity no frictional force against it was modelled, so there are no energy losses. Thus, the pendulum keeps oscillating about its mean position.

2.7 Modelling of Electrical Circuits

Electric circuits consist of interconnections of sources of electric voltage and current, and other electronic elements such as resistors, capacitors, inductors, and transistors. Passive circuits consist of interconnections of resistors, capacitors, and inductors. Symbols for some linear circuit elements and their current-voltage relations are given in Table 2.

Table 2: Linear circuit elements with current-voltage relationship

Element	Symbol	Equation

Resistor: *R*		$v = iR$
Capacitor: *C*		$i = C\frac{dv}{dt}$
Inductor: *L*		$v = L\frac{di}{dt}$
Voltage Source		$v = v_s$
Current Source		$i = i_s$

For mathematical modelling of electrical circuits, Kirchhoff's laws are applied which are stated below.

- **Kirchhoff's Current Law (KCL):** The algebraic sum of currents leaving a junction or node equals the algebraic sum of currents entering that node.
- **Kirchhoff's Voltage Law (KVL):** The algebraic sum of all voltages taken around a closed path in a circuit is zero.

From Table 2, we can see that the equation for inductor involves derivative since its voltage is proportional to the rate of change of current. Similarly, for capacitor the rate of change of voltage across it affects the current through it. Therefore, such circuits which involve energy storing elements; inductors and capacitors, are represented through a set of differential equations. The highest order of differential equations in this set is equal to the number of energy storing elements in a circuit.

Task 5: To estimate the response of an RC circuit

Consider the RC circuit shown in Figure 5.

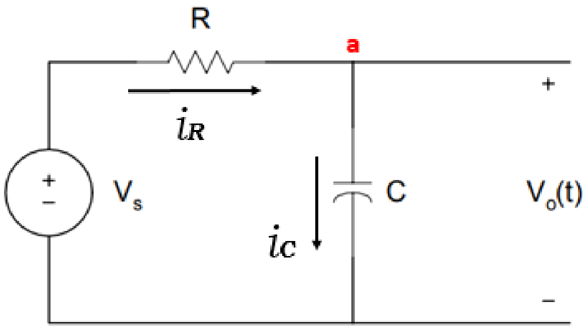


Figure 5: RC circuit with a constant voltage source

The voltage across capacitor is $v_o(t)$ and the source voltage is v_s. Let, the current through resistor is i_R and the current through capacitor is i_C. Applying KCL at node **a**, we get

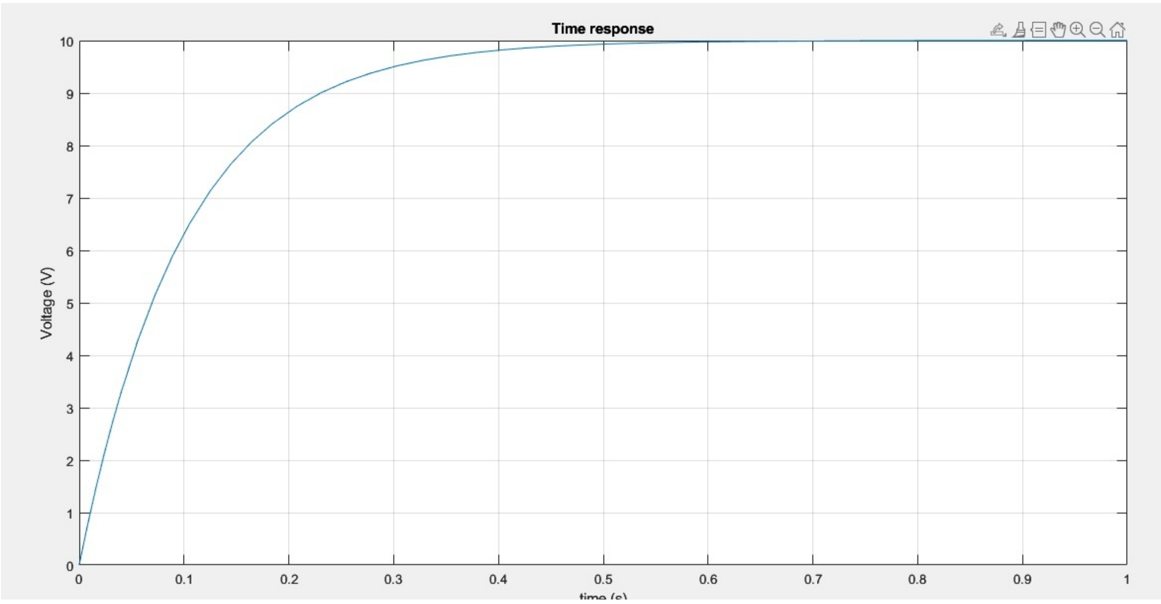
$$i_R = i_C \tag{11}$$

$$\frac{1}{R}\left(v_s - v_o(t)\right) = C\frac{d}{dt}v_o(t) \tag{12}$$

If the capacitor is initially discharged i.e., $v_o(0) = 0$ then, the supply will charge capacitor equal to the source voltage when it is switched on. This doesn't happen instantly due to the capacitor's nature of opposing change in its voltage. The initial response is called transient response. This $v_o(t)$ can be estimated by solving the above differential equation.

1. For the given RC circuit, $v_s = 10V, R = 10k\Omega$ and $C = 10\mu F$. Find the output voltage $v_o(t)$ for the time span of 0 to 1 sec. Represent the obtained solution through properly labelled graph. From the estimated response, find out time the capacitor takes to charge up to source voltage.

Theoretically it takes 5 tau for a capacitor to fully charge. Here, tau= Rc= 0.1. 5 tau is 0.5. The graph also shows that the capacitor voltage reaches Vs i.e becomes fully charged at 0.5s



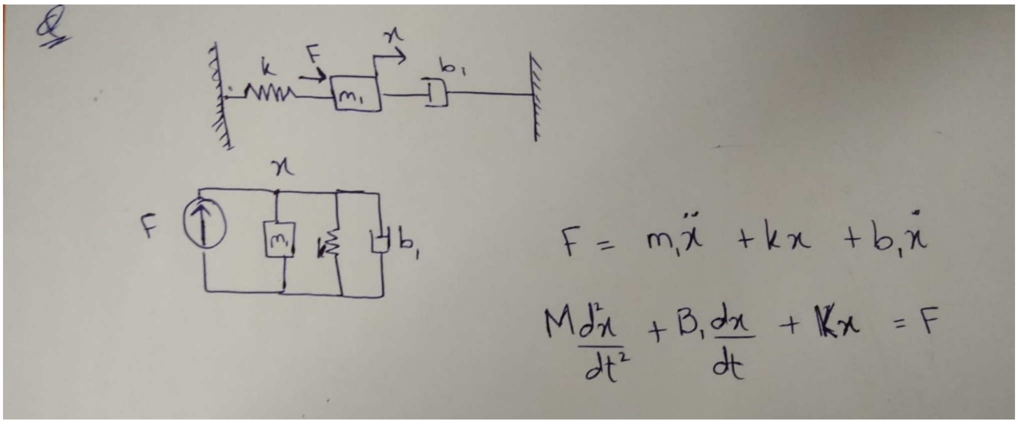
2.8 Post Lab Tasks

Task 6: To simulate the response of a vertical mass-spring system

Consider the mechanical system depicted in Figure 6 below.

(a) Show that the differential equation governing the system is:

$$M\frac{d^2}{dt^2}x(t) + B\frac{d}{dt}x(t) + Kx(t) = F(t)$$



(b) Write an m-file and plot the system response such that forcing function *F(t)* = 1N. Let, M=10 Kg, K= 1 N sec/m and B= 1 N/m. Choose non-zero initial conditions for *x(t)* and its derivative. Discuss your choices.

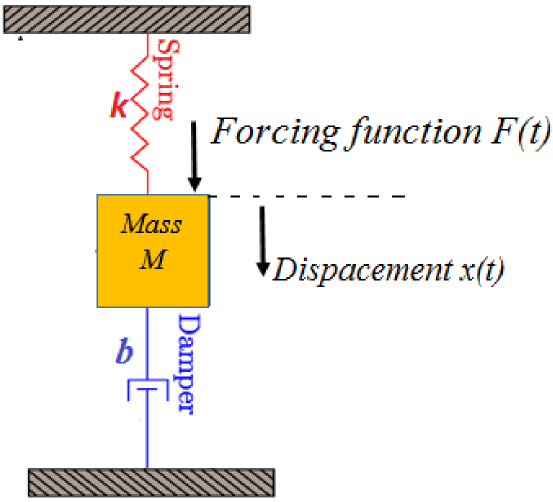


Figure 6: Vertical mass-spring system

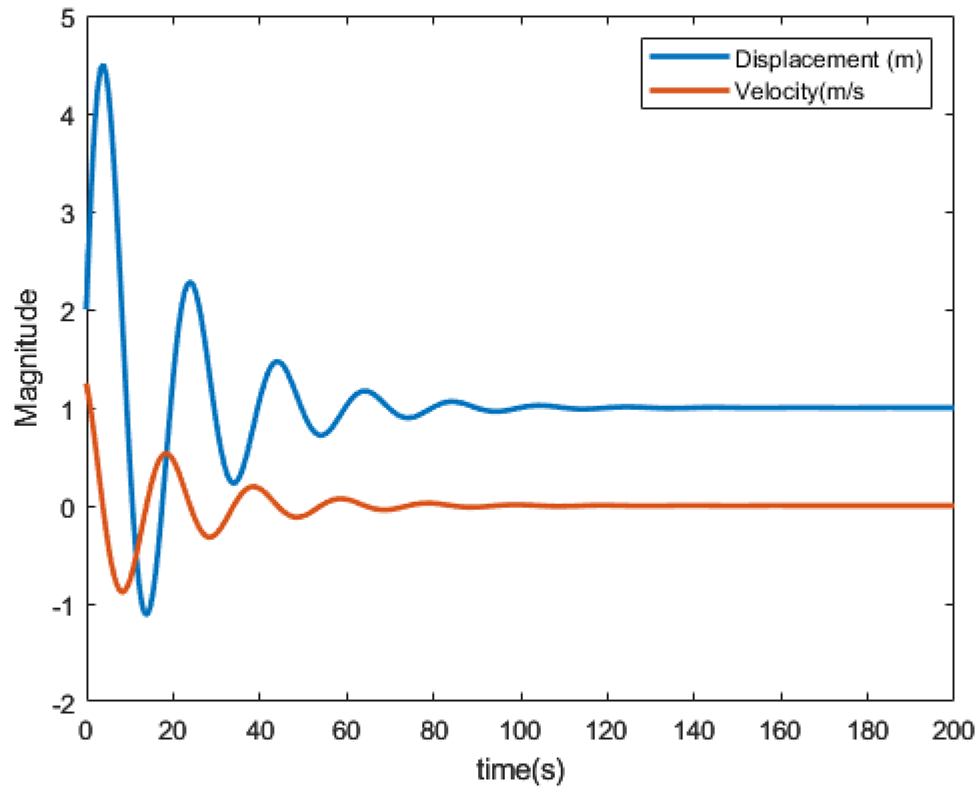
Code:

```
function dXdt=vertical_spring(t,X)
M=10;           % (Kg)
B=1;            % (Nsec/m)
Fa=1;           % (N)
```

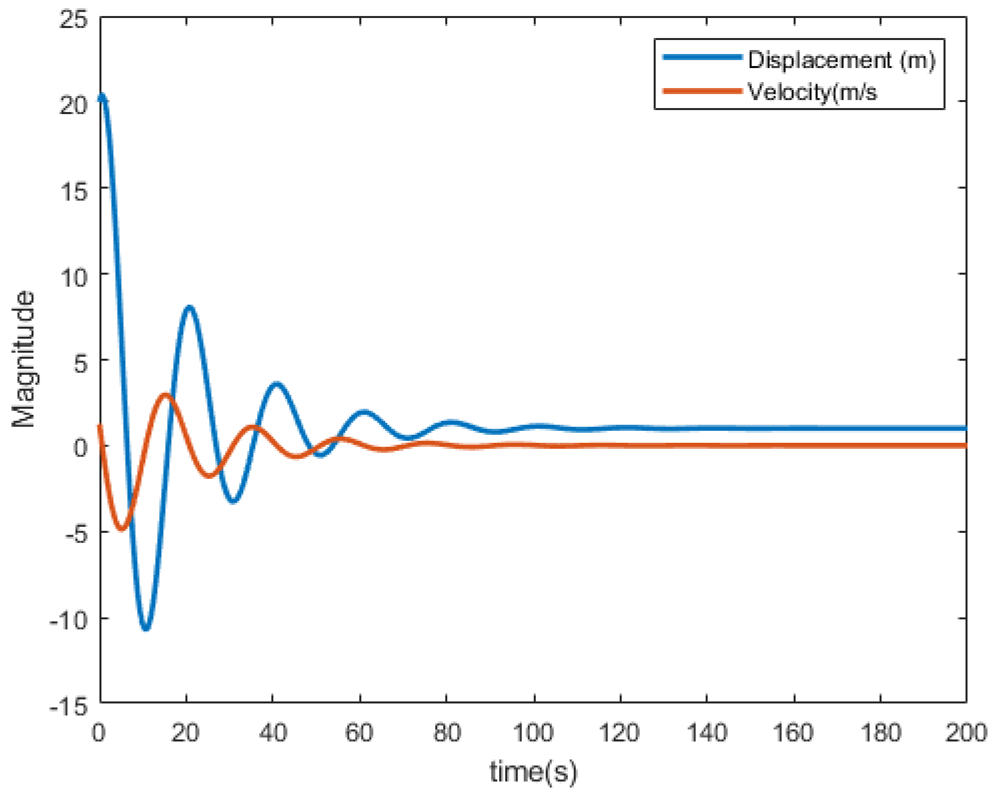
```
K=1;               % (N/m)
dXdt(1,1)= X(2);    %(differential equation 1)
dXdt(2,1)= -(B/M)*X(2) -(K/M)*X(1) + (Fa/M); %(differential equation 2)
end
```

```
X0= [2; 1.25]; % initial position and speed
Options = odeset('RelTol', [1e-4 1e-4], 'AbsTol', [1e-5 1e-5],'Stats', 'on');
[t,X]= ode45('vertical_spring',[0 200], X0);
figure;
plot(t,X(:,1),'LineWidth',2)
xlabel("time(s)")
hold on;
plot(t,X(:,2),"LineWidth",2)
xlabel("time(s)")
ylabel("Magnitude")
legend("Displacement (m)","Velocity(m/s")
```

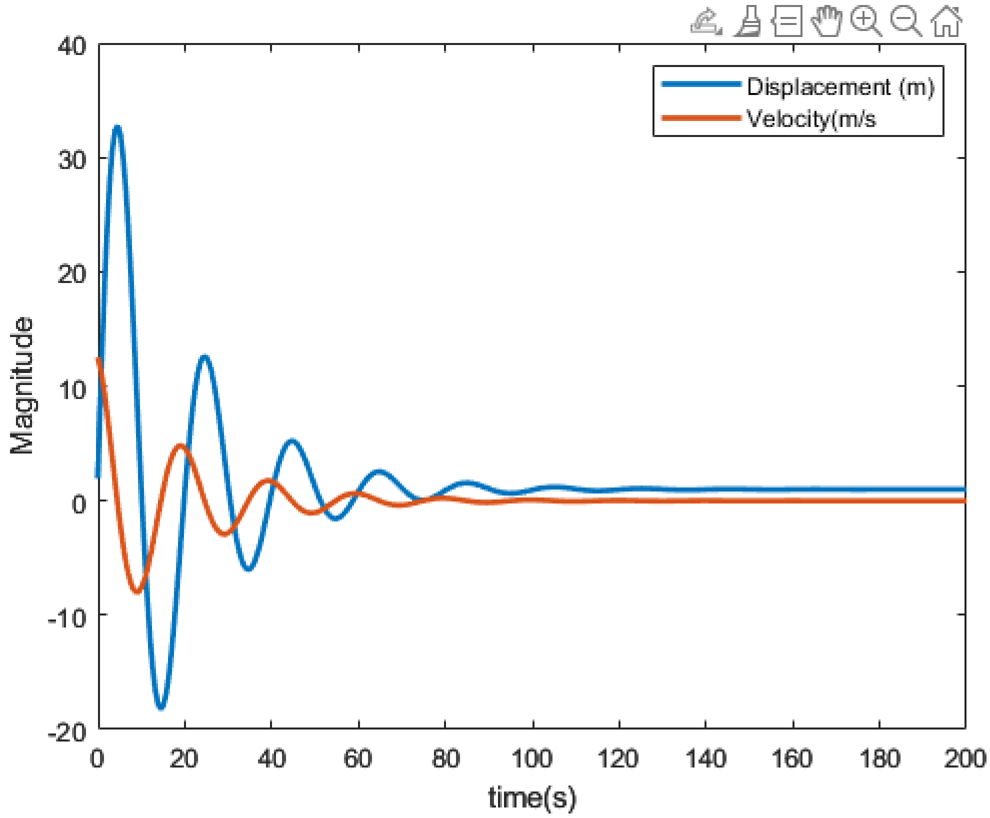
Initial velocity = 1.25 m/s Displacement = 2 m



Initial velocity = 1.25 m/s Displacement = 20 m



Initial velocity = 12.5 m/s Displacement = 2 m



The above graphs show that as the initial velocity and initial displacement increase, the oscillations take longer to dissipate, increasing the transient response. The mass takes longer to reach stationery. The reason for this is that the system's energy input is determined by the initial velocity and displacement. Greater kinetic and elastic energy are introduced into the system with higher initial velocity and initial displacement. As a result, the damping needs more time to cancel out this input energy.

Task 7: To obtain the response of an op-amp based integrator

The circuit shown in Figure 7 represents an op-amp based integrator.

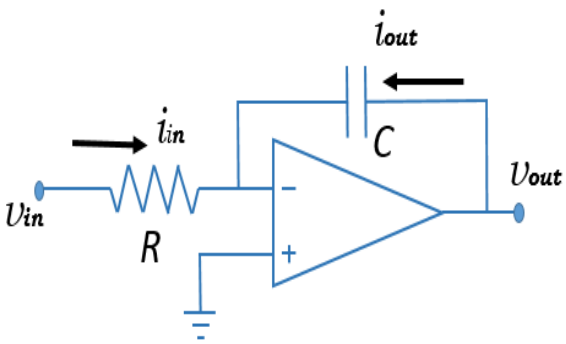
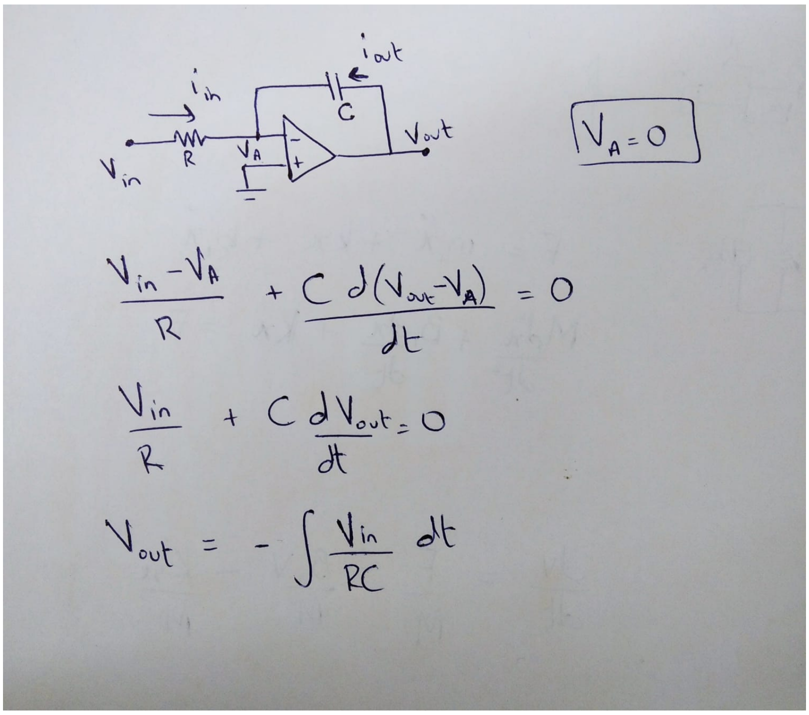


Figure 7: An op-amp based integrator circuit

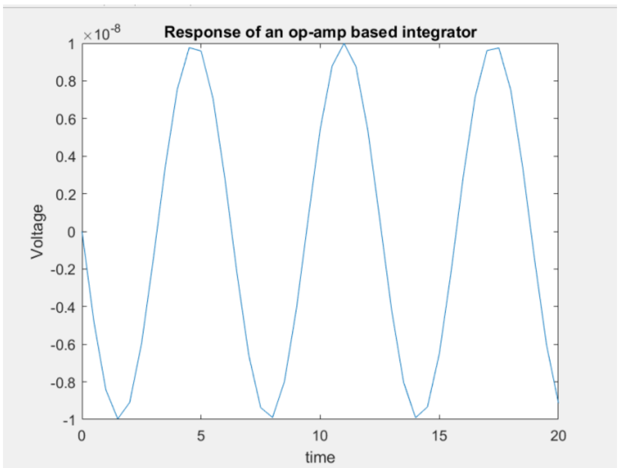
For this circuit:

1. Prove that the output v_{out} is the integral of input signal v_{in} by deriving a mathematical relation between the input and output.



2. Show the system can be solved for v_{out} using ode45 function of MATLAB. Simulate it for $R = 10k\ \Omega$, $C = 100\mu F$ and $v_{in} = \cos(t)$ at initial condition $v_{out}(0) = 0$ for time span of 20 sec. Verify that the response is as per the expression obtained in part 1.

The graph shows that Vout is a -sine wave, which is the integral of Vin, cosine wave. The negative sign is from the formula (working shown above).



```
function dvdt=opamp(t,V)
vin=cos(t);
r=10000;
c=100*10^-6;
dvdt(1,1)=(-vin)/r/c;
end
```

test code:

```
clc;
V0= [0]; % initial position and speed
Options = odeset('RelTol', [1e-4 1e-4], 'AbsTol', [1e-5 1e-5],'Stats', 'on');
[t,V]= ode45('opamp',[0 20], V0);
plot(t,cos(t));
grid on;
hold on;
plot(t,V);
legend ('Vin',"Vout")
title("Time response")
xlabel("time (s)")
ylabel("Voltage (V)")
```

Assessment Rubric

Lab 02
Simulating Dynamic Models using MATLAB

Name: Afsah Hyder	Student ID: 07065

Points Distribution

Task No.	LR2 Simulation/Model/ Code	LR5 Results/Plots	LR 10 Analysis	AR 6 Class Participation
Task 1	-	-	/8	-
Task 2	/4	/4	/12	-
Task 3	/6	/4	/8	-

Task 4	/6	/4	/6	-
Task 5	/6	/4	/4	-
Task 6	/4	/4	/4	-
Task 7	/4	/4	/4	
SEL	-	-	-	/20
Course Learning Outcomes	CLO 1		CLO 4	
Total Points	/100		/20	
	/120			

For details on rubrics, please refer to *Lab Evaluation Assessment Rubrics*.