EXPERIMENT 1:

MODELLING OF FIRST-ORDER SYSTEMS

USING MATLAB/SIMULINK

(4.1 - 4.3)

INDIVIDUAL REPORT

PROGRAMME: MECHATRONICS ENGINEERING

NUR AYU NAZIERA BINTI ROSLI	2119202
DATE OF EXPERIMENT	14 OCTOBER 2024
DATE OF SUBMISSION	21 OCTOBER 2024

Contents

INTRODUCTION 3
OBJECTIVES 3
METHOD AND RESULT 3
4.1 FIRST ORDER SYSTEM 3
4.2 RC TRANSIENT ANALYSIS 4
4.3 DC SERVOMOTOR SYSTEM 10
4.4 TORSIONAL MASS-SPRING-DAMPER SYSTEM 16
4.5 SHOCK ABSORBER 21

INTRODUCTION

In this lab experiment, we explore first-order systems within linear time-invariant dynamic systems. Our primary goal is to understand how these systems react to various inputs, both immediately and over time. We'll use straightforward linear equations with constant coefficients to represent these systems, keeping it practical and easy to grasp. By leveraging tools like MATLAB, we'll visualize the outcomes, making it simpler to comprehend how these systems function in real-world situations. Through analyzing the solutions of these equations and experimenting with different inputs, we'll uncover key insights into the principles that drive system behaviour. This hands-on method will help us build essential skills in system modelling and analysis while equipping us to handle more complex engineering challenges in the future. So, let us explore the fascinating domain of first-order systems in greater depth.

OBJECTIVES

1. To model a first-order system and investigate the effect of system parameters on its response to a step/impulse input.
2. To identify specific system parameters that affect system response.

METHOD AND RESULT

4.1 FIRST ORDER SYSTEM

a. A first-order system is composed of an amplifier and a plant (Fig. 4.1). Find the system

transfer function in standard first-order form, K/(Ts+1).

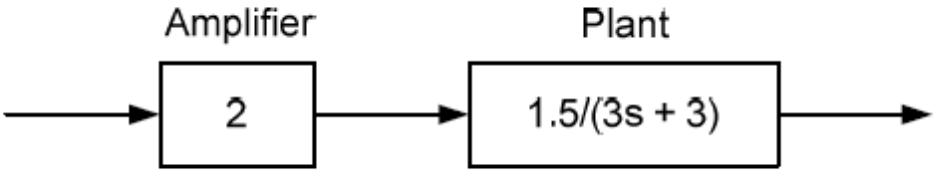


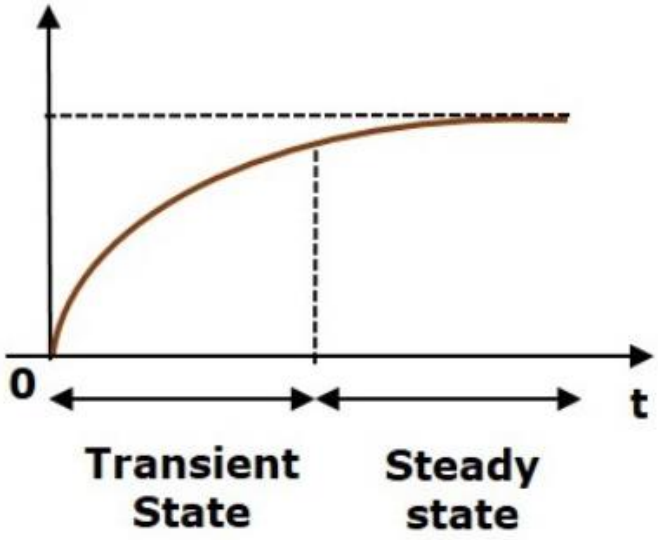
Figure 4.1: First-order system consisting of an amplifier and a plant.

$$G(s) = \frac{K}{Ts + 1}$$

$$G(s) = \frac{(2)(1.5)}{3s + 3}$$

$$G(s) = \frac{3}{3(s + 1)} = \frac{1}{s+1}$$

b. Sketch, as detailed as possible, the output of the above system (Fig. 4.1) for a unit input.



4.2 RC TRANSIENT ANALYSIS

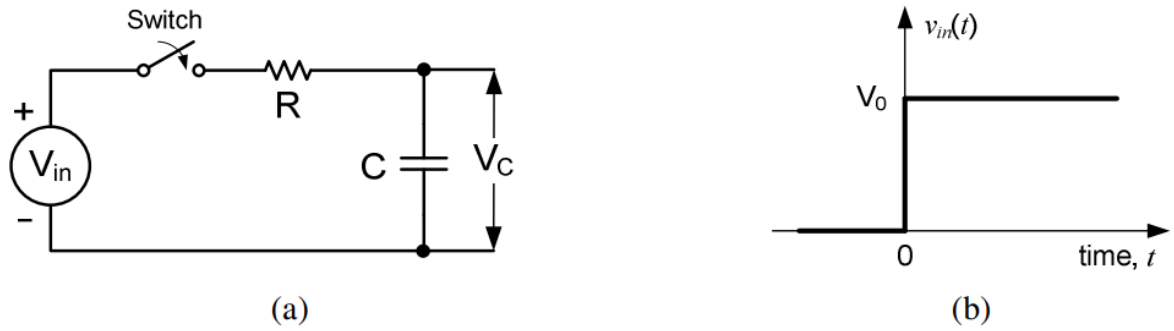


Figure 4.2: A simple RC filter.

a. Derive the differential equations for the transient state shown in Fig. 4.2.

$-V_{in} + V_R + V_C = 0$

$-V_{in} + R\left(C\frac{dv_c}{dt}\right) + \frac{1}{C}\int i\,dt = 0$

$-V_{in} + R\left(C\frac{dv_c}{dt}\right) + \frac{1}{C}\int\left(C\frac{dv_c}{dt}\right)dt = 0$

$-V_{in} + R\left(C\frac{dv_c}{dt}\right) + v_c(t) = 0$

b. Find v_c (t) immediately after the switch is closed (on).

$Vin = R\left(C\frac{dv_c}{dt}\right) + v_c(t)$ ——————— (eq 4.2a)

The input signal is unit step, so, Vin is $\frac{A}{s} = \frac{1}{s}$ in laplace form.

The laplace transform of equation 4.2a is,

$\frac{1}{s} = (RCs + 1) * V_C(s)$

This equation is rearranged to obtain $V_C(s)$;

$V_C(s) = \frac{1}{s}\frac{1}{(RCs+1)}$

Partial fraction of the above equation is;

$V_C(s) = \frac{1}{s} - \frac{1}{(s+\frac{1}{RC})}$

Transforming the equation to time-domain;

$V_C(t) = 1 - e^{\frac{1}{RC}t}$

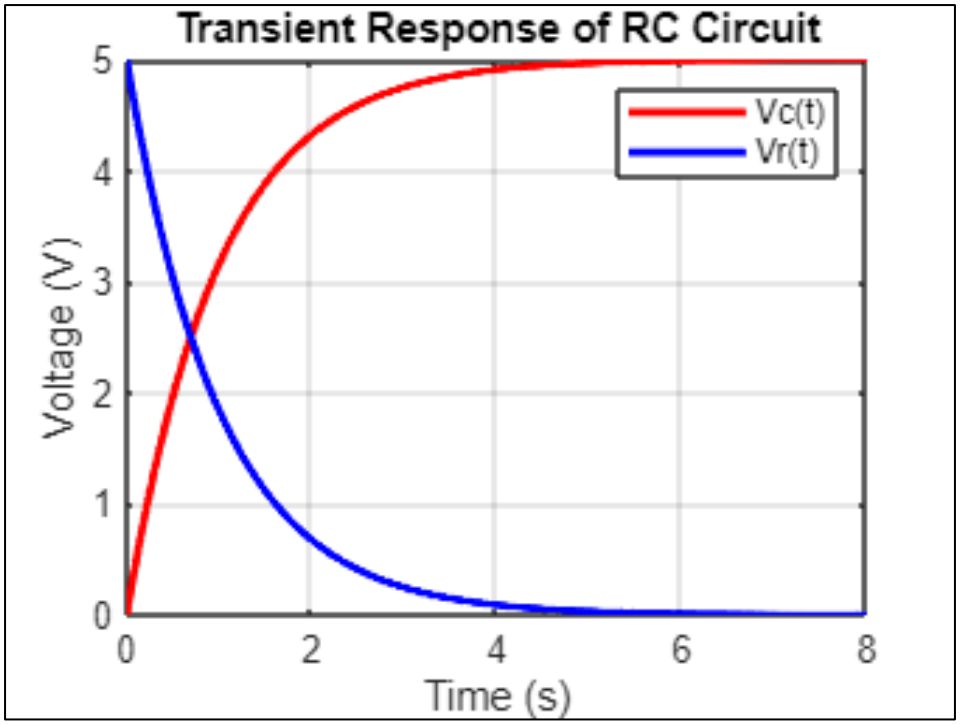
c. If V0 = 5V, R = 10 kΩ, and C = 100 µF, plot the transient curve of v_c(t) and v_R(t) in

MATLAB for t ≤ 8τ. Compute the time constant, τ, of the system. Show your work.

The matlab code to plot the transient curve v_c(t) and v_R(t):

```
% Given values
V0 = 5;          % Initial voltage (V)
R = 10e3;        % Resistance (Ohms)
C = 100e-6;      % Capacitance (Farads)

% Compute time constant T
T = R * C;
% Time vector
t = linspace(0, 8*T, 1000); % From 0 to 8T
% Voltage across the capacitor (vC) and resistor (vR)
vC = V0 * (1 - exp(-t/tau));  % Capacitor voltage
vR = V0 * exp(-t/tau);        % Resistor voltage
% Plotting
figure;
plot(t, vC, 'r', 'LineWidth', 2); % Plot Vc(t)
hold on;
plot(t, vR, 'b', 'LineWidth', 2); % Plot Vr(t)
xlabel('Time (s)');
ylabel('Voltage (V)');
title('Transient Response of RC Circuit');
legend('Vc(t)', 'Vr(t)');
grid on;
```

v_C(t) and v_R(t) plot:



The time constant, τ, of the system is the RC value.

$\tau = RC$

$\tau = (10k\Omega)(100\ \mu F)$

$\tau = 1s$

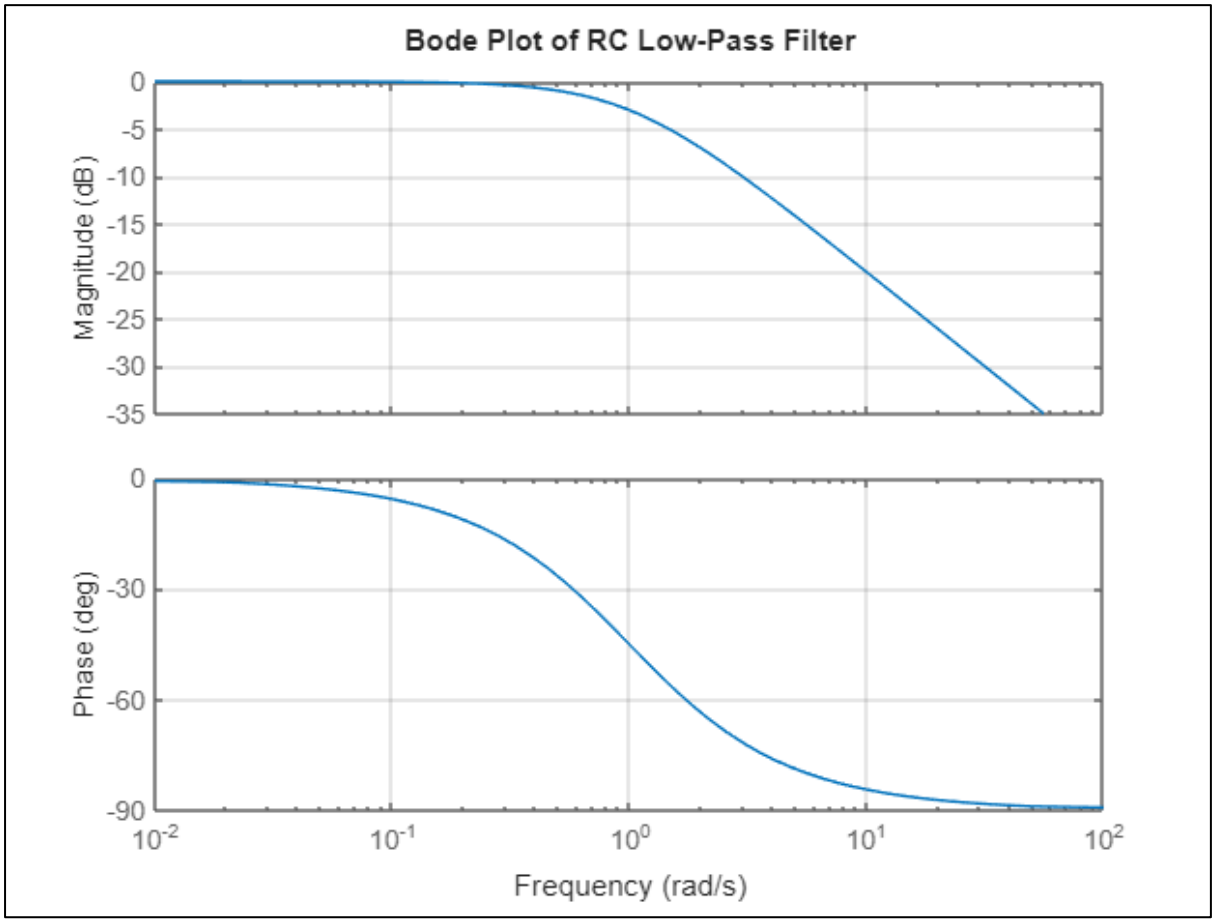
d. What kind of filter does the above circuit represent? Using MATLAB, draw a Bode plot

justifying your answer.

-The circuit represents a low-pass filter.

Here is the MATLAB code to draw the Bode plot for this circuit:

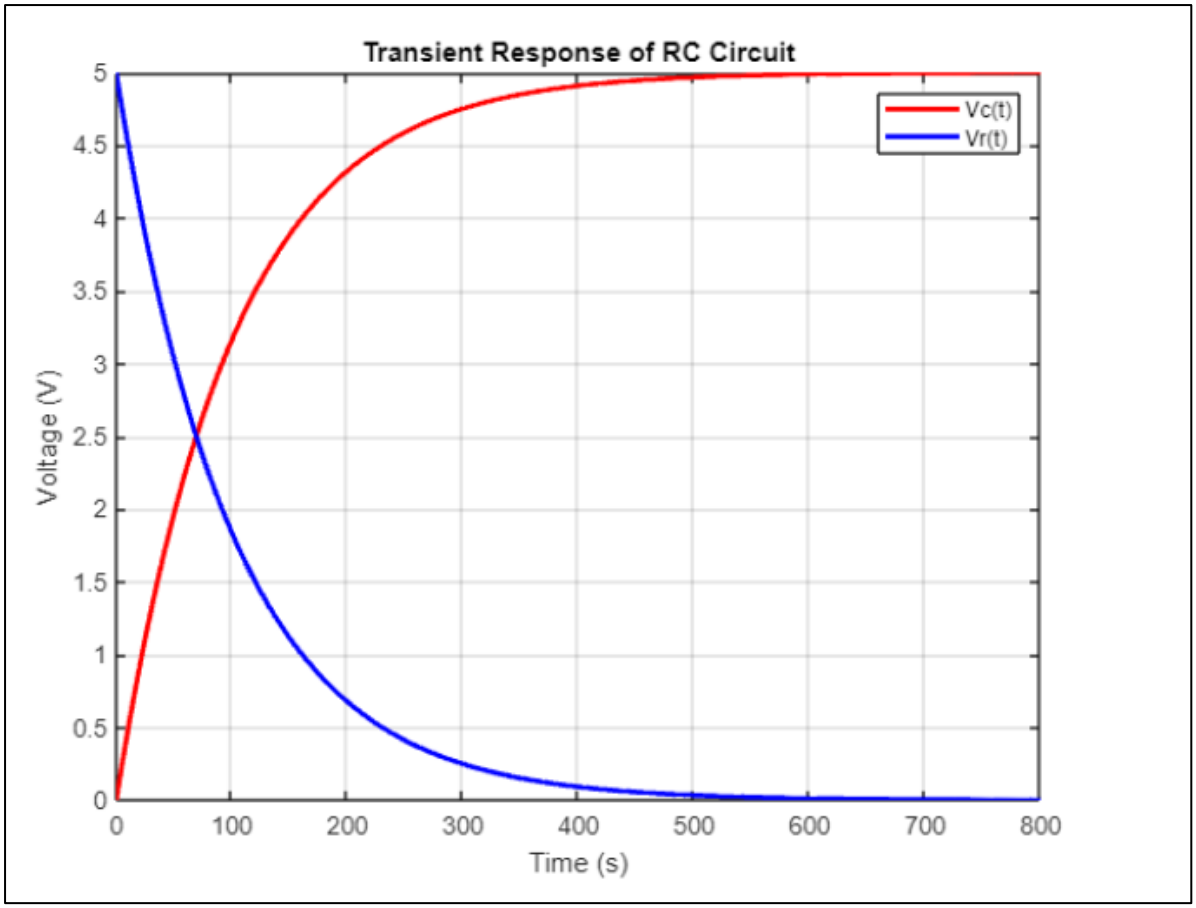
```
% Given values
R = 10e3;        % Resistance (Ohms)
C = 100e-6;      % Capacitance (Farads)
tau = R * C;     % Time constant (seconds)

% Transfer function H(s) = 1 / (1 + sRC)
s = tf('s');
H = 1 / (1 + s*R*C);

% Bode plot
figure;
bode(H);
grid on;
title('Bode Plot of RC Low-Pass Filter');
```

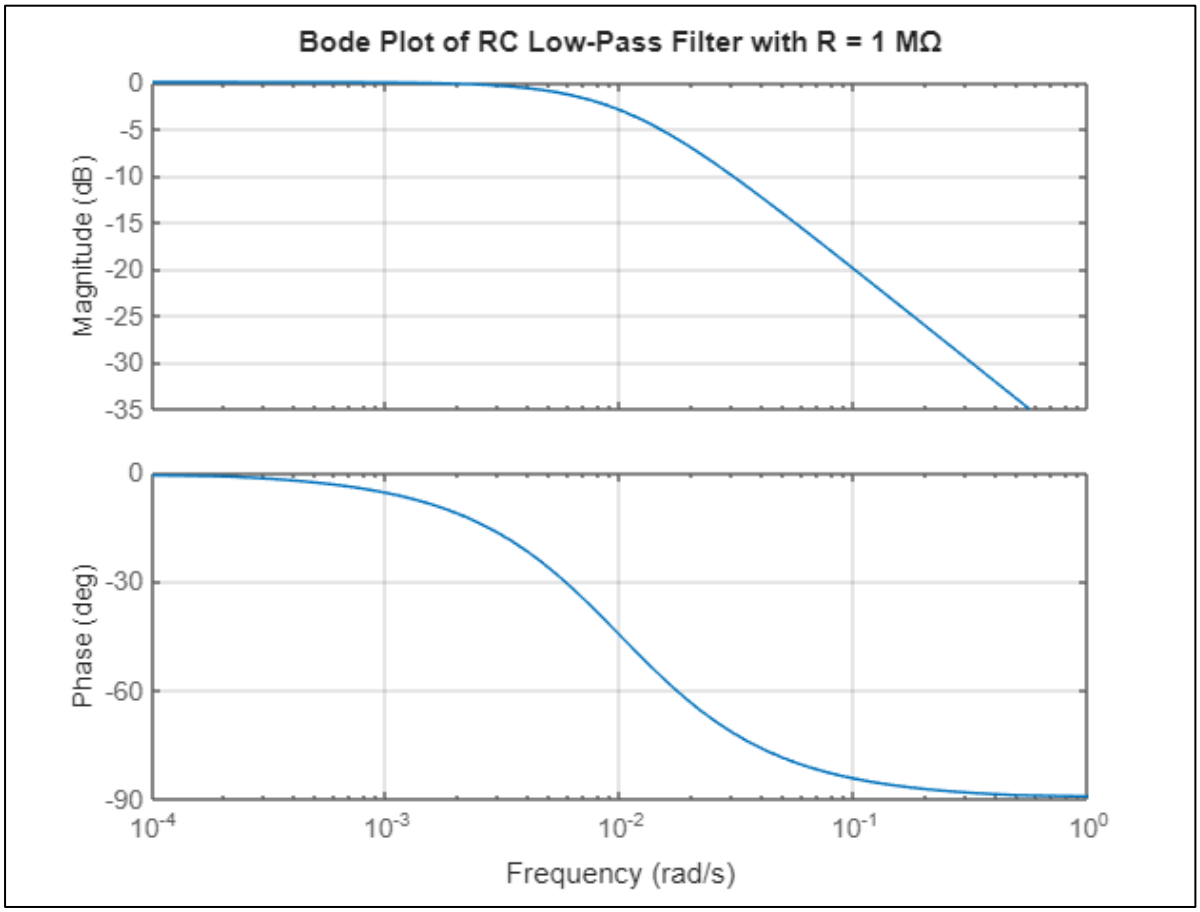


Based on the Bode plot, it is clearly justifying that the circuit is a low-pass filter due to its attenuation of high-frequency signals and its characteristic phase shift behaviour.

e. Now we change the resistor to R = 1 MΩ. What happened to the transient curve and Bode

plot? Write your conclusions.


Transient Response of RC Circuit
Vc(t)
Vr(t)
Voltage (V)
Time (s)
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
0
100
200
300
400
500
600
700
800


Bode Plot of RC Low-Pass Filter with R = 1 MΩ
Magnitude (dB)
Phase (deg)
Frequency (rad/s)
0
-5
-10
-15
-20
-25
-30
-35
0
-30
-60
-90
10-4
10-3
10-2
10-1
100

When the resistor in the RC circuit is increased to 1MΩ the time constant becomes much larger which is τ =100seconds causing the transient response of both the capacitor voltage and resistor voltage to slow down significantly. It takes much longer for the capacitor to charge and the resistor voltage to decay. In the frequency domain, the cutoff frequency shifts to a much lower value which is fc = 0.00159Hz allowing only very low-frequency signals to pass through while attenuating higher frequencies. The phase shift also transitions over a lower frequency range. In summary, increasing the resistor value in an RC low-pass filter will increases the time constant, making the transient response slower and shifting the cutoff frequency to lower values, resulting in a more selective filter for low frequencies.

4.3 DC SERVOMOTOR SYSTEM

A schematic diagram of an armature-controlled DC servomotor is shown in Fig. 4.3b. The system variables include:

ea: armature drive potential (V).

eb: back EMF or counter-electromotive-force (V)

ia: armature current (A)

T: torque produced by servomotor (Nm)

θ: angular position of rotor (rad)

θ: angular speed of rotor (rad/s)

The parameters of the system include:

Ra: armature electric resistance (Ω)

La: armature electric inductance (H)

J: rotor's moment of inertia (kg m2)

b: damping ratio (Nms)

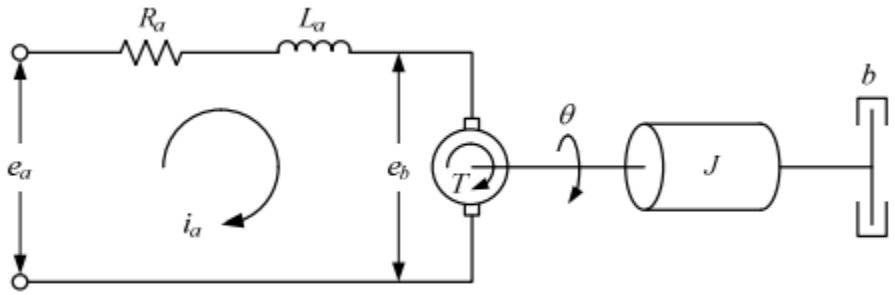


Figure 4.3b: Schematic diagram of DC servomotor.

The system parameters not shown in Fig. 4.3b include:

K_T: torque constant (Nm/A)

K_e: back EMF constant (Nm/A)

The torque constant K_T models the relationship between the electric current input, ia, and servomotor torque output, T:

$$T(s) = K_T i_a(s)$$

The transfer function of the servomotor, with armature drive potential, ea(s), as input and motor speed, sθ(s), as output, can be written as [3].

$$G(s) = \frac{s\theta(s)}{e_a(s)} = \frac{K_T}{JR_a s + (bR_a + K_T K_e)}$$

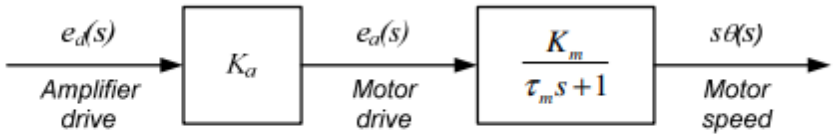


Figure 4.4: Block diagram of the servomotor and amplifier system.

Modelling DC servomotor

For the model of DC servomotor, the following parameters are available:

• Simulation interval, 0 ≤ ≤ 3 s

• Time step (for simulation), Δt = 0.1 ms

• Rotor's moment of inertia, J = 0.01 kg m2

• Damping ratio, b = 0.1 Nms

• Back EMF constant, Ke = 0.01 Nm/A

• Torque constant, K_T= 0.01 Nm/A

• Amplifier gain, Ka = 25

• Armature electric resistance, Ra = 1 Ω

• Armature electric inductance, La = 0.01 H – 0.5 H (varying)

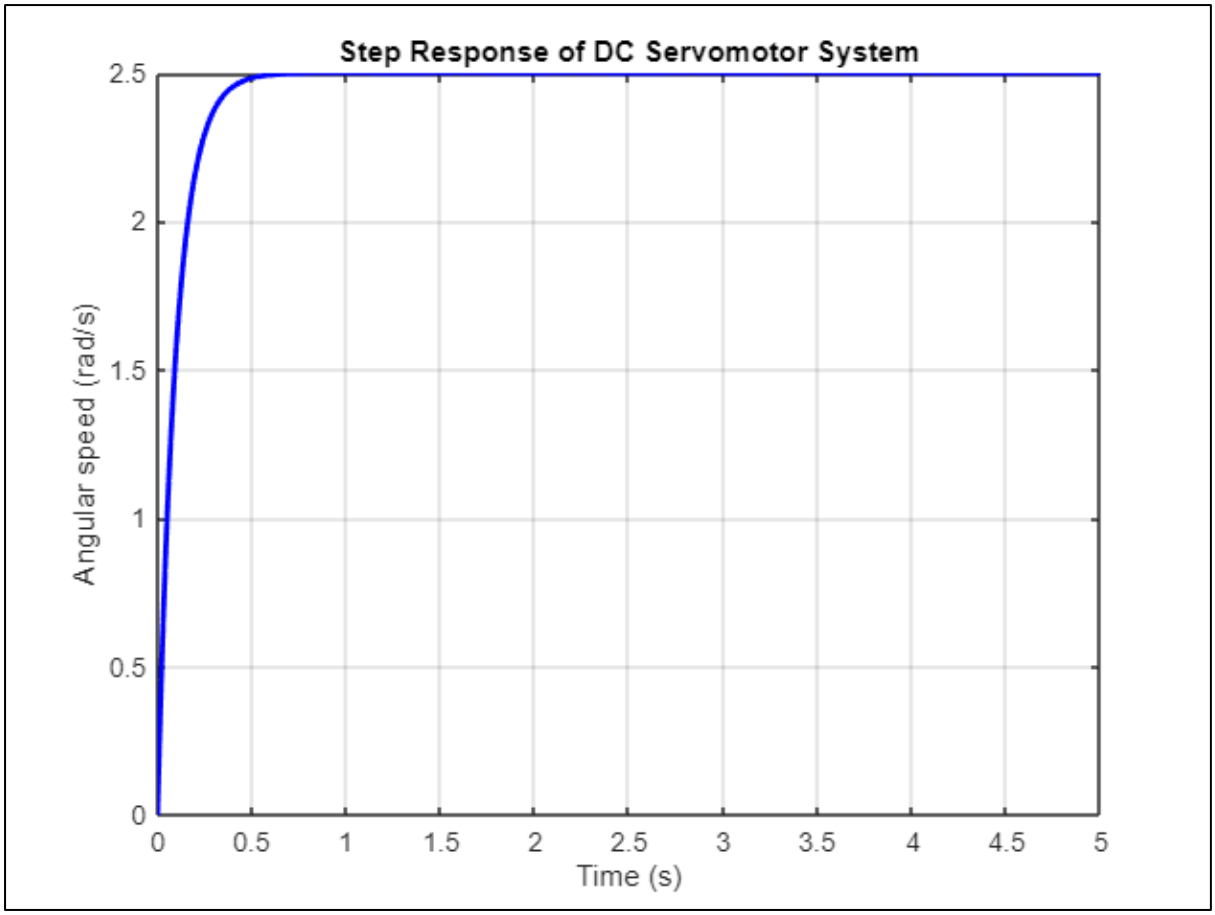
a. For the time being we assume that the armature electric inductance, La → 0 (negligible small). Write a MATLAB routine for simulating the step response of the open-loop DC servomotor system with amplifier driver, ed, as the input and angular speed, θ, as the output.

From equation 2.4 the denominator is;

$(Js + b)(Ls + R) + K_T K_e$
$JLs^2 + JRs + bLs + bR$
s^2 (JL) + s(JR + bL) + bR

The MATLAB code:

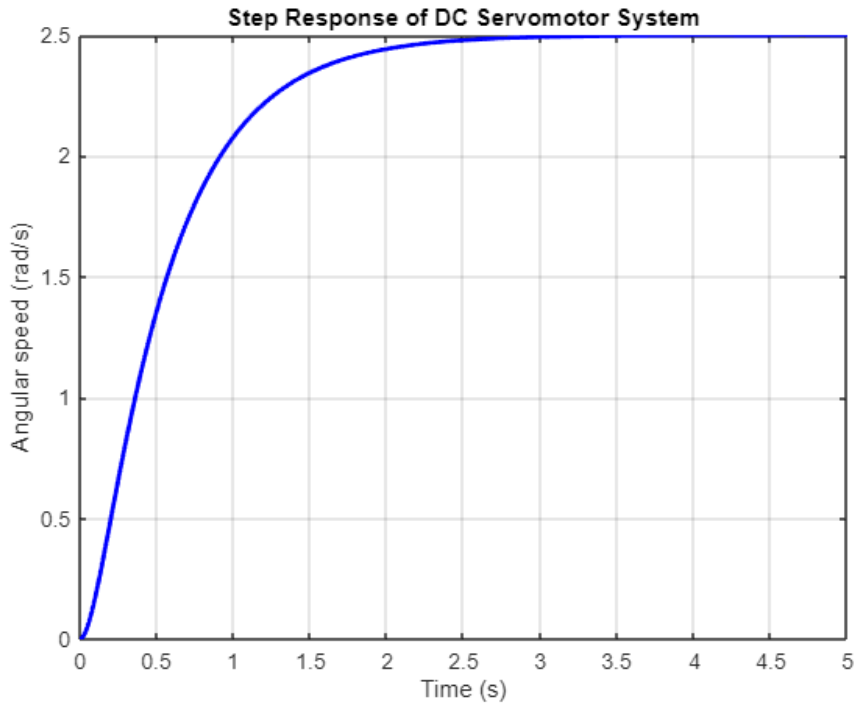
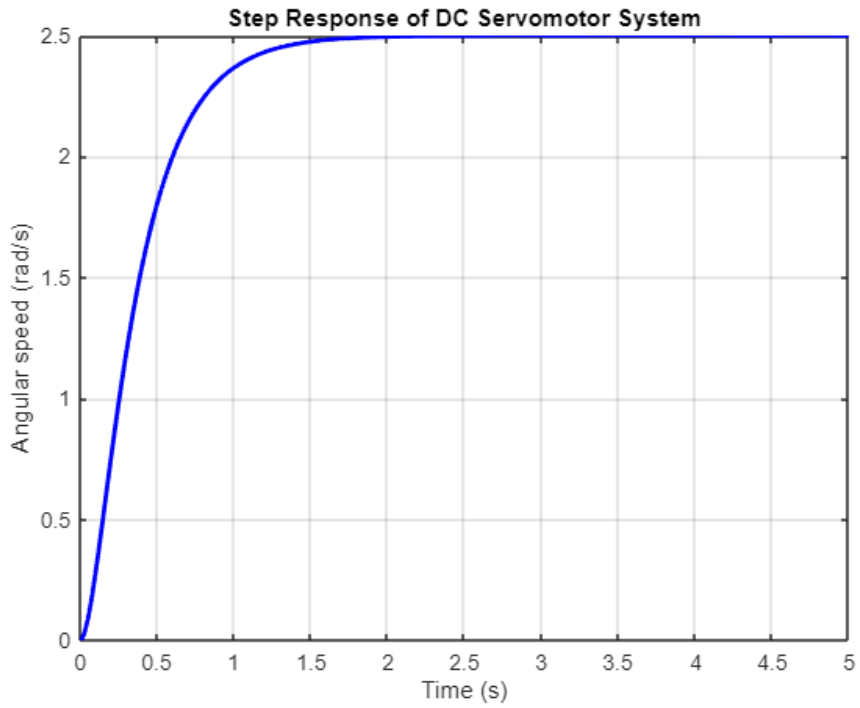
```
% Define parameters
Kt = 0.01; % Motor torque constant (Nm/A)
Kb = 0.01; % Back EMF constant (V/(rad/s))
J = 0.01; % Moment of inertia (kg.m^2)
R = 1; % Armature resistance (ohms)
L = 0; % Armature inductance (negligible)
Ka = 25; % Amplifier gain
b = 0.1; %damping ratio
% Transfer function of the system
num = Kt*Ka;
den = [J*L (J*R+b*L) b*R];
sys = tf(num, den);
% Step response simulation
t = 0:0.01:5; % Time vector
u = ones(size(t)); % Step input
[y, ~] = lsim(sys, u, t); % Simulate step response
% Plot the results
plot(t, y, 'b', 'LineWidth', 2);
xlabel('Time (s)');
ylabel('Angular speed (rad/s)');
title('Step Response of DC Servomotor System');
grid on;
```

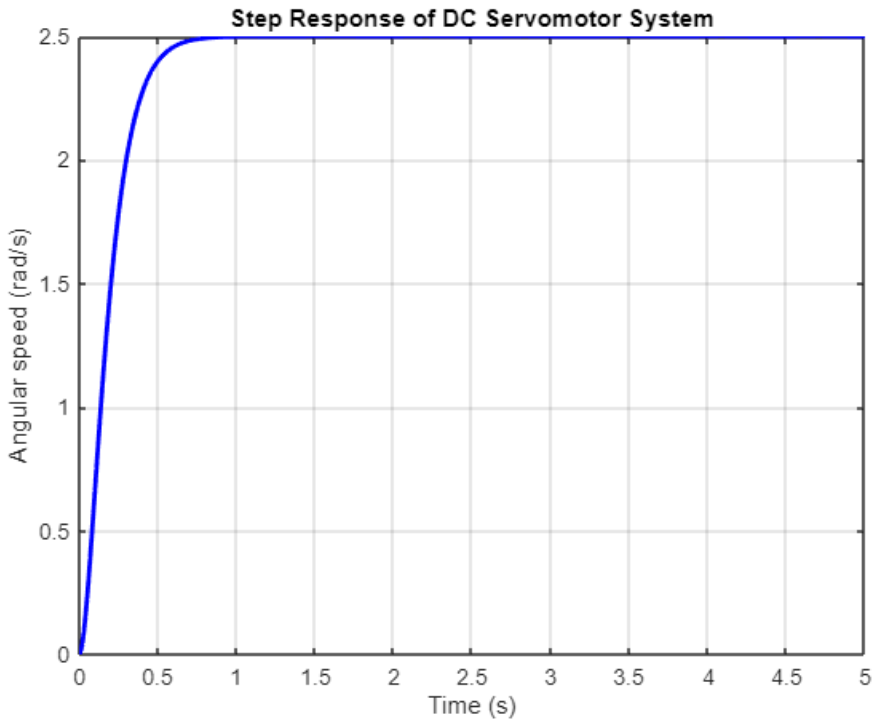
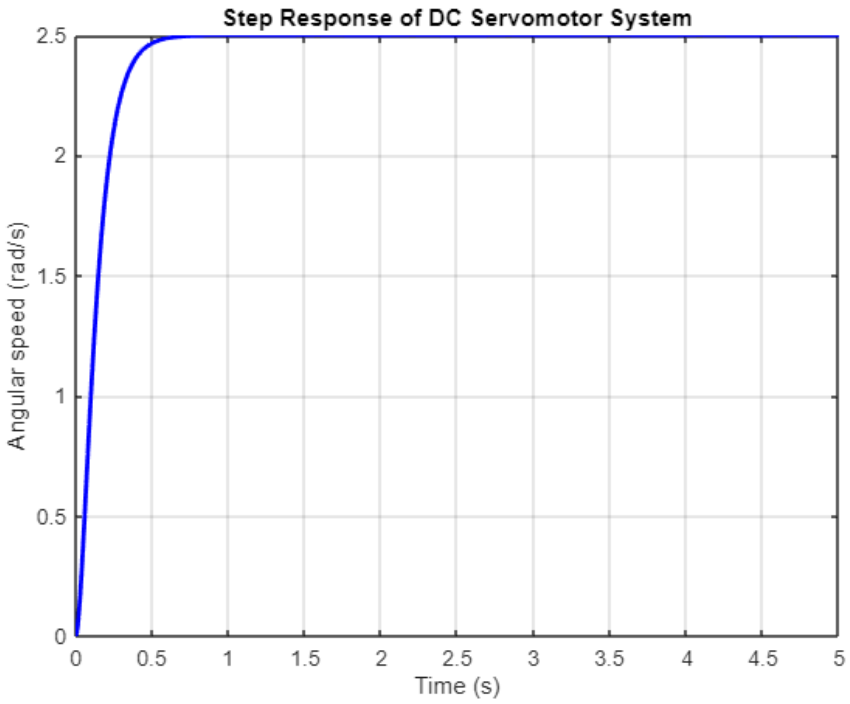


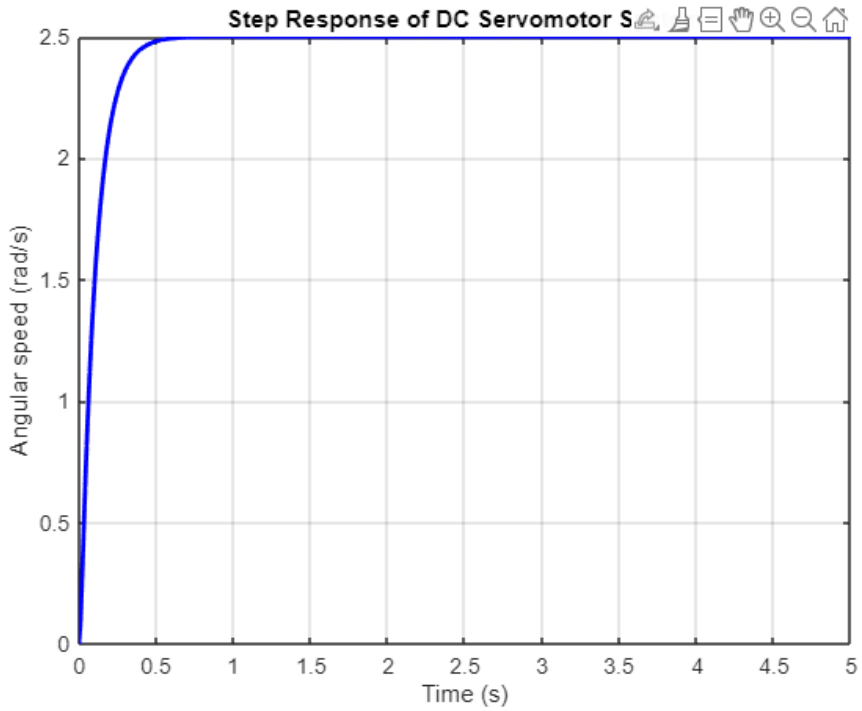
b. Now we apply varying armature electric inductances, La, from 0.5 H, 0.3 H, 0.1 H, 0.05 H down to 0.01 H to our model. Change your MATLAB routine so that it is suitable for such conditions.

Using the similar Matlab Code from 4.3a, replace the L_a value from 0H to 0.5 H, 0.3 H, 0.1 H, 0.05 H and 0.01H.

$L_a\ value$	output

0.5 H	 Step Response of DC Servomotor System Angular speed (rad/s) Time (s) 0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5
0.3 H	 Step Response of DC Servomotor System Angular speed (rad/s) Time (s) 0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5

0.1 H	 Step Response of DC Servomotor System Angular speed (rad/s) Time (s) 0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5
0.05 H	 Step Response of DC Servomotor System Angular speed (rad/s) Time (s) 0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5

0.01 H	

c. Write your conclusions about the investigated DC servomotor model.

The DC servomotor behaves like a first-order system when the armature inductance is very small, meaning its response to changes is smooth and without oscillations. The speed at which the motor reaches its final angular velocity is determined by the motor's inertia on how hard it is to get it moving and the damping on how much resistance there is to motion. The amplifier gain helps increase the output but doesn't change how fast the motor responds. This model shows a gradual increase in speed after a step input, with no overshooting or oscillations, indicating it is well-damped.

However, since this is an open-loop system with no feedback, the motor's performance may be affected by disturbances or changes in load, meaning it might not always reach the exact desired speed without external control. In real applications, feedback control would be needed to ensure the motor can adjust to such changes and perform more accurately. The simplified model gives a good representation of how the motor works in ideal conditions.

4.4 TORSIONAL MASS-SPRING-DAMPER SYSTEM

Consider the torsional mass-spring-damper system in Fig. 4.5.

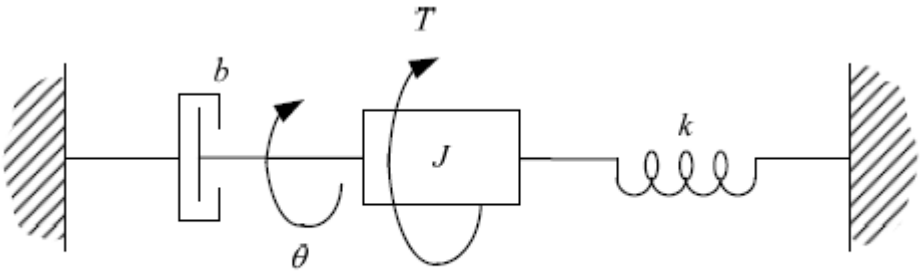


Figure 4.5: Diagram of torsional mass-spring-damper system.

Figure 4.5: Diagram of torsional mass-spring-damper system.

The system variables are

T= external torque applied on rotor (Nm)

Θ= angular position of rotor (rad)

'\dot{\theta}' angular speed of rotor (rad/s).

The parameters of the system in Fig. 4.5 include

J= rotor's moment of inertia (kg m2)

B= torsional damping coefficient (Nms/rad)

K= angular stiffness (Nm/rad).

Modelling torsional mass-spring-damper

For modelling the torsional mass-spring-damper system (Fig. 4.5) the following parameters are:

• Simulation interval, 0 ≤ - ≤ 0.4 s

• Time step (for simulation), Dt = 0.1 ms

• Rotor's moment of inertia, J = 0.1 kg m2

• Angular stiffness, k = 100,000 Nm/rad

• Torsional damping coefficient, b = 0.1 Nms/rad

• Electromechanical gain, Kx = 0.2 As

• Amplifier gain, Kh = 25

• Voltage-to-angular-deflection ratio, Kd = 5,000 V/rad

• Voltage-to-angular-deflection ratio, K1 = 0.1 – 1,000 V/rad (varying)

• Voltage-to-angular-speed ratio, K2 = 0.5 – 15 Vs/rad (varying)

a. Write a MATLAB program to find the step response of the above torsional mass-spring-damper

system with θ_d(s) as input and θ (s) as output. Investigate the system behaviour by

varying the gains K1 and K2. For each combination of K1 and K2 find the system's

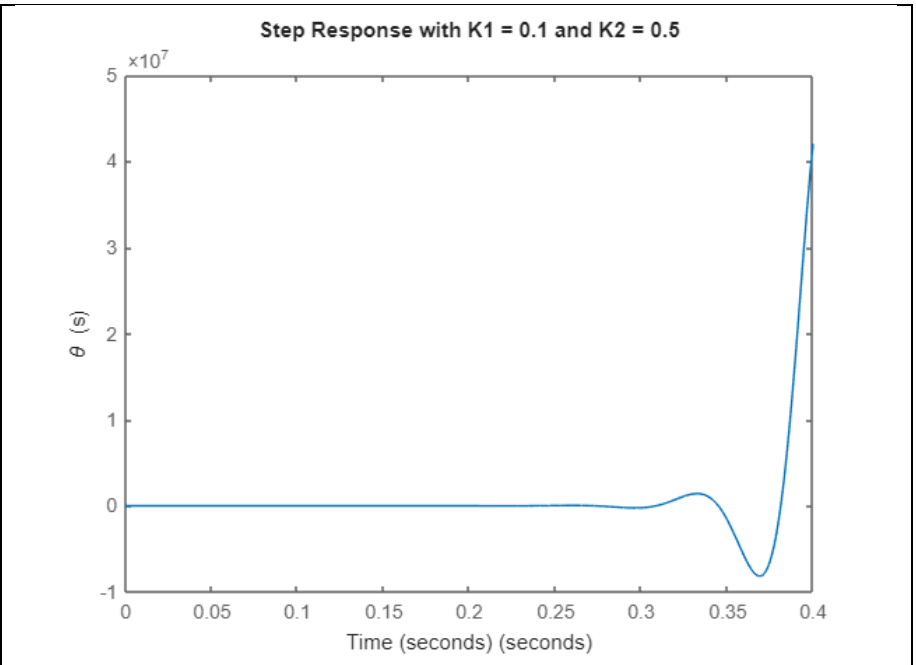
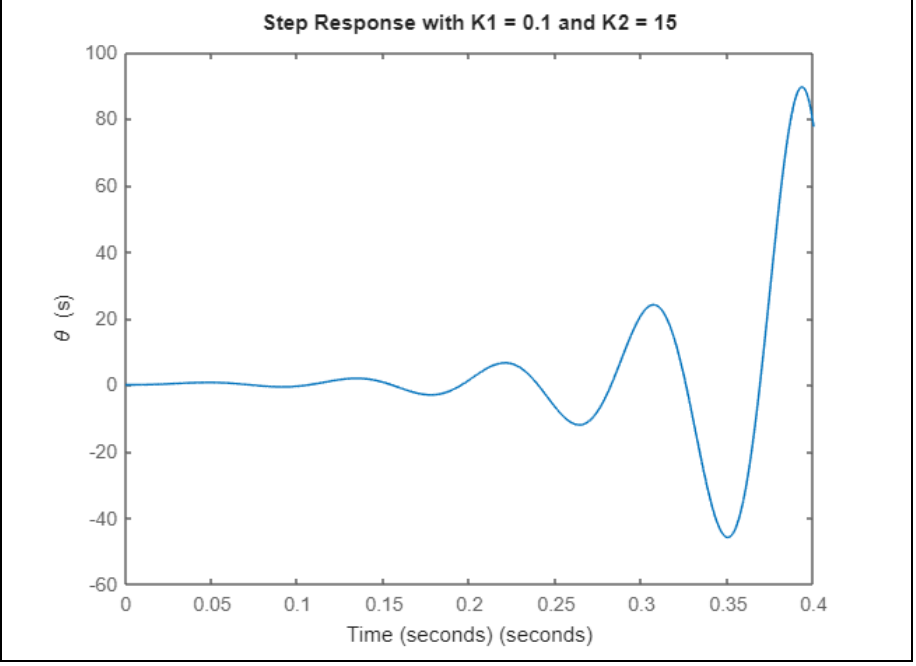
Eigenfrequency ωn and damping ratio £. Describe the effect of K1, and K2 respectively, on the system.

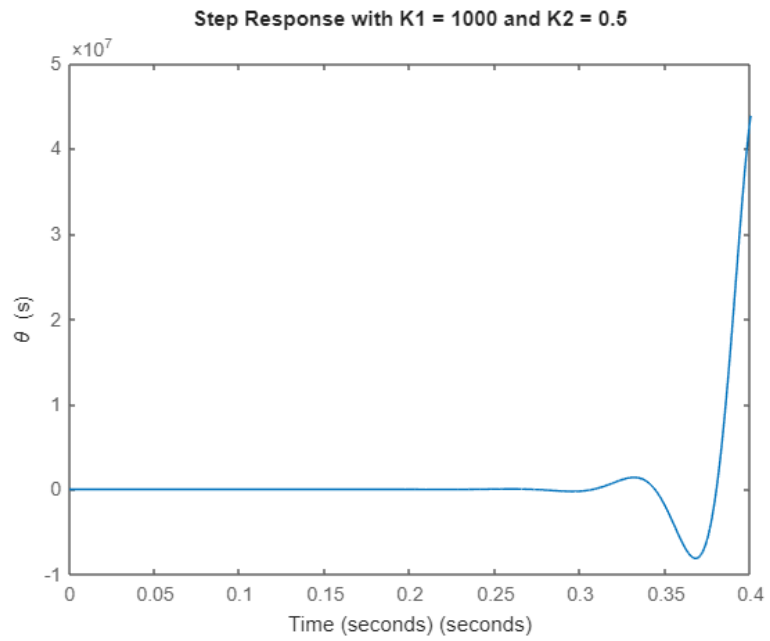
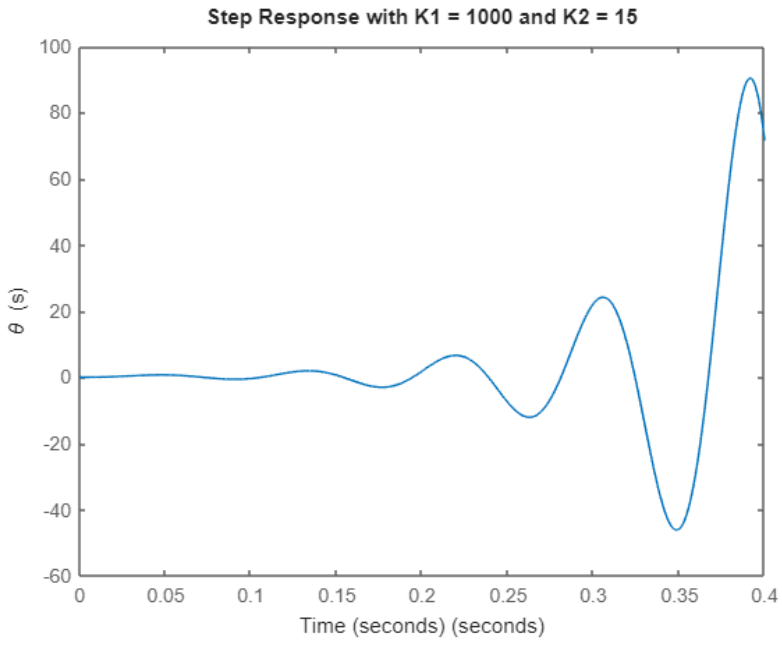
The MATLAB code:

```
% MATLAB code for torsional mass-spring-damper system step response
clc;
clear;

% Given parameters
J = 0.1;  % Rotor's moment of inertia (kg*m^2)
k = 100000;  % Angular stiffness (Nm/rad)
b = 0.1;  % Torsional damping coefficient (Nms/rad)
Kx = 0.2;  % Electromechanical gain (As)
Kh = 25;  % Amplifier gain
Kd = 5000;  % Voltage-to-angular-deflection ratio (V/rad)
theta_d = 1;  % Desired angular deflection (step input)

% Varying gains for investigation
K1_values = [0.1, 1000];  % Vary K1 from 0.1 to 1000 V/rad
K2_values = [0.5, 15];    % Vary K2 from 0.5 to 15 Vs/rad

% Time settings
t_step = 0.0001;  % Simulation time step (Dt = 0.1 ms)
t_sim = 0: t_step: 0.4;  % Simulation interval 0 <= t <= 0.4 s

% Loop over different K1 and K2 combinations
for K1 = K1_values
   for K2 = K2_values

      % Transfer function of the system: theta(s)/theta_d(s)
      num = Kh * Kd * Kx;
      den = [J K2 b K1 + k];  % J*s^2 + K2*s + b*s + (K1 + k)
      sys = tf(num, den);  % Define the transfer function

      % Step response
      figure;
      step(sys, t_sim);
      title(['Step Response with K1 = ', num2str(K1), ' and K2 = ', num2str(K2)]);
      xlabel('Time (seconds)');
      ylabel('\theta (s)');

      % Calculate natural frequency and damping ratio
```

```
        [wn, zeta] = damp(sys);  % Get eigenfrequency (wn) and damping ratio (zeta)
        fprintf('For K1 = %.2f and K2 = %.2f, \n', K1, K2);
        fprintf('Natural Frequency (wn) = %.2f rad/s\n', wn(1));
        fprintf('Damping Ratio (zeta) = %.2f\n', zeta(1));
        fprintf('-------------------------\n');
    end
end
```

The output:

K1= 0.1 K2= 0.5	
K1= 0.1 K2=15	

K1= 1000 K2= 0.5	
K1= 1000 K2= 15	

b. Use Simulink to study the model behaviour of torsional mass-spring-damper (Fig. 4.5).

Compare the result with Problem 4.4a.

The step response plots from Simulink exhibit similar trends to the MATLAB results. Increasing K1 lead to an increase in stiffness, making the system respond faster and more oscillatory. Meanwhile increasing K2 lead to increase in damping which leading to less oscillation and a slower response.

c. Conclusion

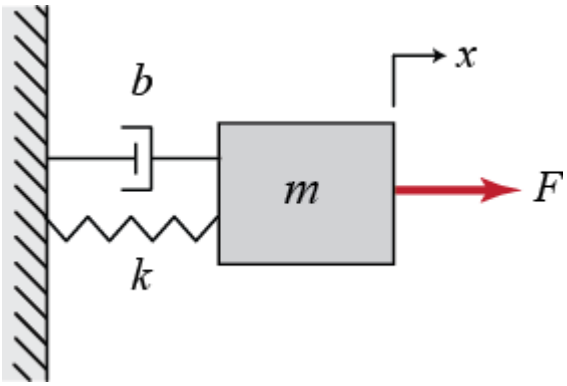
In both the MATLAB and Simulink simulations of the torsional mass-spring-damper system, the system's behaviour is significantly influenced by the gains K1 and K2. As K1 increases, the system's natural frequency ωn increases, making the system stiffer and more responsive but also more oscillatory. This results in a higher overshoot and faster rise time, which is evident in both the MATLAB step response plots and Simulink model. On the other hand, increasing K2 enhances the damping in the system, which reduces the overshoot and dampens oscillations, but also slows down the overall response. Both tools show that larger K2 values stabilize the system by increasing the damping ratio £, leading to a smoother response with a longer settling time.

The comparison between the MATLAB program and the Simulink model shows consistent behaviour across both approaches, confirming the accuracy of the results. MATLAB allowed precise calculation of key system parameters like the natural frequency and damping ratio, while Simulink provided a more visual and interactive approach to studying the system dynamics. The consistency in results between the two methods demonstrates that both can effectively be used to study and tune torsional systems. Overall, increasing K1 leads to a faster but more oscillatory system, while increasing K2 promotes stability at the cost of a slower response.

4.5 SHOCK ABSORBER

a. A shock absorber is a mechanical device designed to smooth out or damp shock impulse, and dissipate kinetic energy. Sketch a mass-spring-damper model of a spring-based shock absorber (Fig. 4.8) and derive a second-order differential system from it.

Figure 4.8: Spring-based shock absorber.



For the mass-spring-damper system, the governing differential equation is derived from Newton’s second law of motion:

$$F(t)=mx\ddot{}(t)$$

Where:

- $mx\ddot{}(t)$ is the inertial force due to the mass,
- $bx\dot{}(t)$ is the damping force proportional to velocity,
- $kx(t)$ is the restoring force exerted by the spring proportional to displacement,
- $F(t)$ is the external force applied to the system.

The total forces acting on the mass include:

- Damping force $F_{damping}=bx\dot{}(t)$
- Spring restoring force $F_{spring}=kx(t)$

Thus, using Newton's second law:

$$mx\ddot{}(t)=-bx\dot{}(t)-kx(t)+F(t)$$

Rearranging the equation:
$$mx\ddot{}(t)+bx\dot{}(t)+kx(t)=F(t)$$

This is a second-order differential equation describing the motion of the mass-spring-damper system.

Model of shock absorber

For the model of shock absorber, the following parameters are available:

• Mass, m = 2 kg

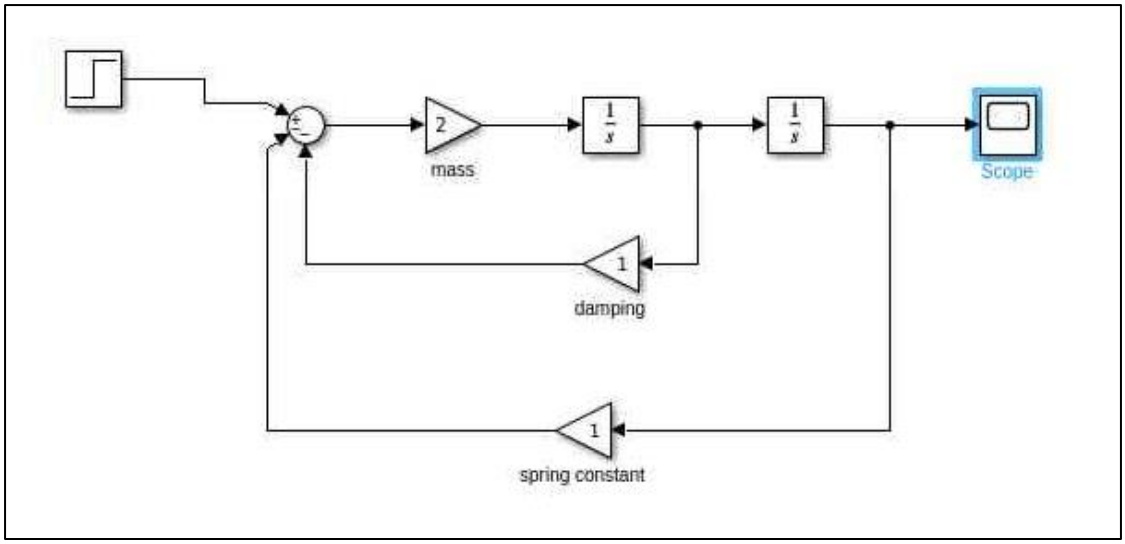
• Spring constant, k = 300 – 3,000 N/m (varying)

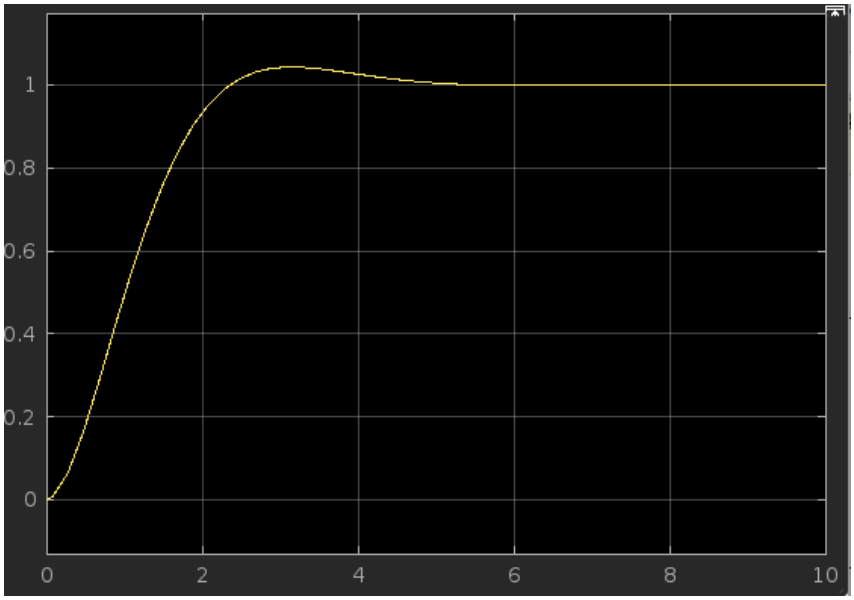
• Damping ratio, b = 10 – 200 kg/s (varying)

• Simulation interval, 0 ≤ - ≤ 3 s

• Time step (for simulation), Dt = 0.1 ms

b. Draw the block diagram of your model (Fig. 4.8) and simulate the impulse response of the system using Simulink.





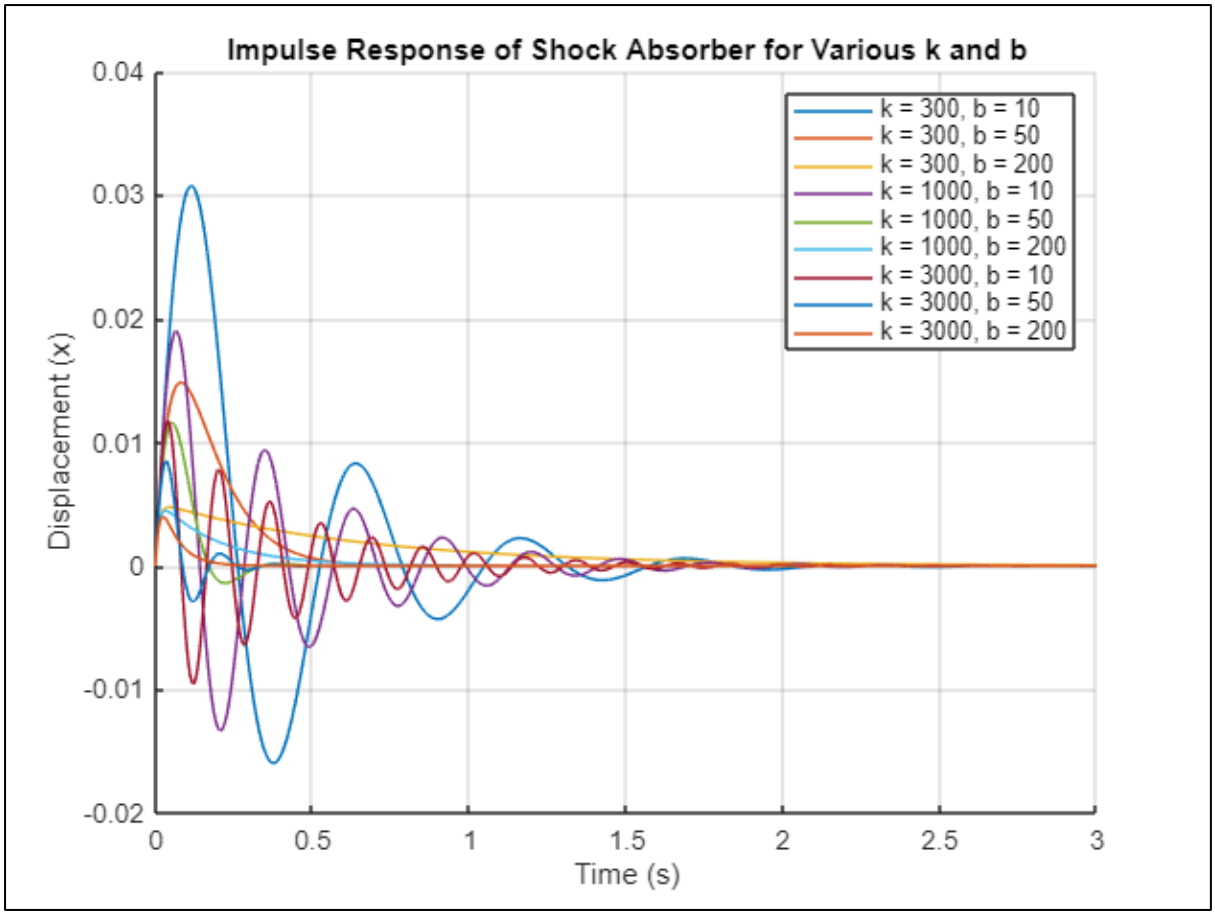
c. By using MATLAB/Simulink, investigate the dynamic system behaviour in response to a unit impulse. Plot the impulse response of your system for various k and b.

the MATLAB code:

```
% Define system parameters
m = 2;  % mass in kg
k_values = [300, 1000, 3000];  % varying spring constants in N/m
b_values = [10, 50, 200];  % varying damping coefficients in N·s/m

% Time vector
t = 0:0.001:3;  % time from 0 to 3 seconds with 1 ms time step

% Loop through various k and b values
figure;
hold on;
for k = k_values
```

```
    for b = b_values
        % Define the transfer function: m * d^2x/dt^2 + b * dx/dt + k * x = F(t)
        num = 1;  % Impulse input is 1
        den = [m, b, k];  % Coefficients of d^2x/dt^2, dx/dt, and x

        % Create the transfer function
        sys = tf(num, den);

        % Compute the impulse response
        [y, t] = impulse(sys, t);

        % Plot the impulse response
        plot(t, y, 'DisplayName', sprintf('k = %d, b = %d', k, b));
    end
end

% Customize the plot
xlabel('Time (s)');
ylabel('Displacement (x)');
title('Impulse Response of Shock Absorber for Various k and b');
legend show;
grid on;
hold off;
```

The output with various value of k and b:



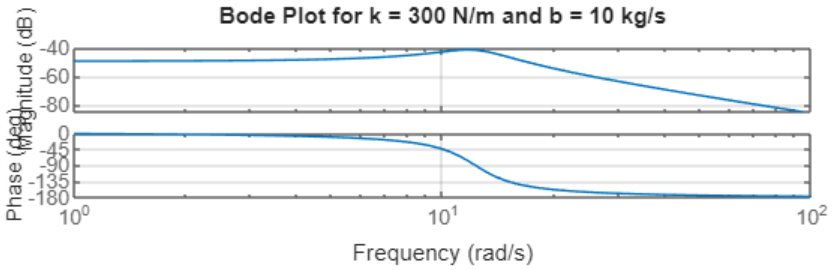
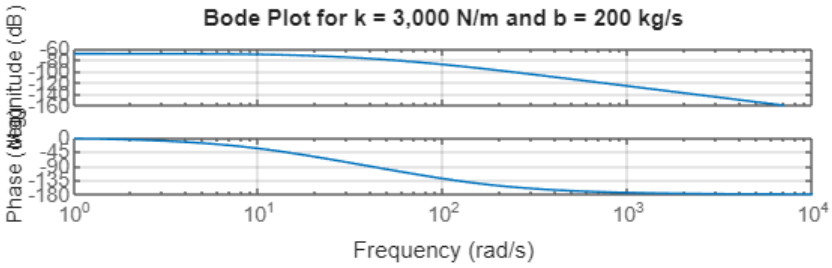
d. Find the system frequency response, i.e. G(s=jw), using Bode plot for the following.

The MATHLAB code:

```
% Define the system parameters
m = 2;  % Mass in kg

% First case: k = 300 N/m, b = 10 kg/s
k1 = 300;  % Spring constant in N/m
b1 = 10;   % Damping coefficient in N·s/m

% Second case: k = 3,000 N/m, b = 200 kg/s
k2 = 3000; % Spring constant in N/m
b2 = 200;  % Damping coefficient in N·s/m

% Define transfer functions for both cases
G1 = tf(1, [m, b1, k1]);  % Transfer function for k1 and b1
G2 = tf(1, [m, b2, k2]);  % Transfer function for k2 and b2

% Create Bode plots for both cases
figure;
subplot(2, 1, 1);
bode(G1);
title('Bode Plot for k = 300 N/m and b = 10 kg/s');
```

```
grid on;

subplot(2, 1, 2);
bode(G2);
title('Bode Plot for k = 3,000 N/m and b = 200 kg/s');
grid on;
```

The output:

i. k = 300 N/m and b = 10 kg/s	
ii. k = 3,000 N/m and b = 200 kg/s	

Examine these two Bode plots thoroughly. Write your conclusions.

System with k=300 N/m and b=10kg/s is more flexible and less damped. It shows a higher response at lower frequencies, and the Bode plot may exhibit a resonance peak, indicating that the system is more prone to oscillations. The lower damping makes the system more sensitive to input forces near its natural frequency, but it also means it takes longer for oscillations to die out. Meanwhile the system with k=3,000N/m and b=200 kg/s is much stiffer and more heavily damped, meaning it is more stable and less responsive to low-frequency inputs. The high damping prevents oscillations, making the system overdamped. The system responds more sluggishly but more smoothly, with very little chance of resonance. This is a desired behaviour for shock absorbers meant to minimize vibrations effectively.

In conclusion, increasing both k and b significantly alters the system's frequency response by reducing resonance and oscillations, making it more stable and less responsive to frequency inputs. A lower k and b result in a more flexible and oscillatory system, sensitive to low-frequency inputs and prone to resonance.