DEPARTMENT OF MECHANICAL & MECHATRONIC ENGINEERING

Simulation and Analysis of Second-Order System Responses

By

NDIDZULAFHI AGREEMENT MOSEYA 230492134

Subject: Control Systems CSS260S

Lecturer: Dr Esmail

Date: 18 SEPTEMBER 2025

Practical 2: LAB 2

Signed_____

Abstract

This lab introduced MATLAB as a tool for basic control system analysis and simulation. Matrix operations, polynomial expressions, and signal plotting were explored. Transfer functions and closed-loop feedback systems were modelled and analysed. A spring–mass–damper system was simulated to study position and velocity responses.

Introduction

The study of control systems requires tools that can perform mathematical modelling, simulation, and analysis of dynamic behaviour. MATLAB provides a powerful environment with built-in functions that allow engineers to represent systems using matrices, polynomials, and transfer functions while also visualizing responses through plots and simulations. In this laboratory, MATLAB was used to practice basic operations such as matrix multiplication, polynomial representation, signal generation, and plotting. The experiment further demonstrated how to model open-loop and closed-loop systems, as well as how to analyse the spring–mass–damper system, highlighting MATLAB's role in bridging theoretical concepts with practical system analysis.

Objectives

- To understand the basic application of MATLAB in control systems
- To familiarise with different MATLAB operations used in control systems

Theory

Control systems are analysed using mathematical models that describe how a system responds to inputs. MATLAB is a useful tool for expressing these models in matrix form, polynomials, transfer functions, and differential equations. Each of these concepts is explained below.

1. Matrix Operations

A **matrix** is a rectangular arrangement of numbers. In control systems, vectors and matrices are often used to represent signals and state variables.

- **Element-wise multiplication** a.* b multiplies each element of vector a with the corresponding element of vector b.
- **Matrix multiplication** a * b follows the rules of linear algebra, where the inner dimensions must agree. For example, multiplying a 4-row vector with a 4×1 column vector produces a scalar result, which represents the **dot product**.

2. Polynomial Representation

A **polynomial** is an expression made up of powers of a variable. In MATLAB, polynomials are often used to describe system equations such as transfer functions or characteristic equations.

For example: $7x^3 + 2x + 10$

Where:

$x = input$
$7x^3 = cubic\ terms$
$2x = linear\ term$
$10 = constant\ term$

3. Transfer Functions

A **transfer function** expresses the ratio between the Laplace transform of the output and input of a system, assuming zero initial conditions. It is written in terms of the Laplace variable ss.

General form:

$$H(S) = \frac{N(S)}{D(S)}$$

Where:

- H(s)H(s) = transfer function
- N(s)N(s) = numerator polynomial (represents zeros of the system)
- D(s)D(s) = denominator polynomial (represents poles of the system)

4. Feedback Systems

Feedback is used in control to modify system behaviour:

- **Negative feedback** stabilizes the system by subtracting output from the input.
- **Positive feedback** reinforces the output, which may cause instability.

5. Spring–Mass–Damper System

A mechanical system with a mass (mm), damping coefficient (cc), and spring stiffness (kk) follows Newton's second law:

The dynamics of a mass-spring-damper system are governed by the second-order differential equation below:

$$m\ddot{x}(t) + c\dot{x}(t) + kx(t) = f(t)$$

where:

- $x(t)$ = displacement of the mass (output)
- $\dot{x}(t)$ = velocity
- $\ddot{x}(t)$ = acceleration
- $f(t)$ = external force applied (input)
- m = mass (kg)
- c = damping coefficient (Ns/m)
- k = spring stiffness (N/m)

For $m = 1, c = 0.1$, and $k = 10$:

$$\ddot{x}(t) + 0.1\dot{x}(t) + 10x(t) = f(t)$$

The transfer function of displacement $X(s)$ over force $F(s)$ is:

$$\frac{X(s)}{F(s)} = \frac{1}{ms^2 + cs + k} = \frac{1}{s^2 + 0.1s + 10}$$

This shows how the physical system can be studied using control system theory and MATLAB/Simulink simulations.

Experimental procedures

1. Matrix Multiplication:

- Perform element-wise multiplication of vectors a= [4 4 4 4] and b= [2 4 6 8] using the command a.*b.

- Perform standard matrix multiplication using the command a*b'.

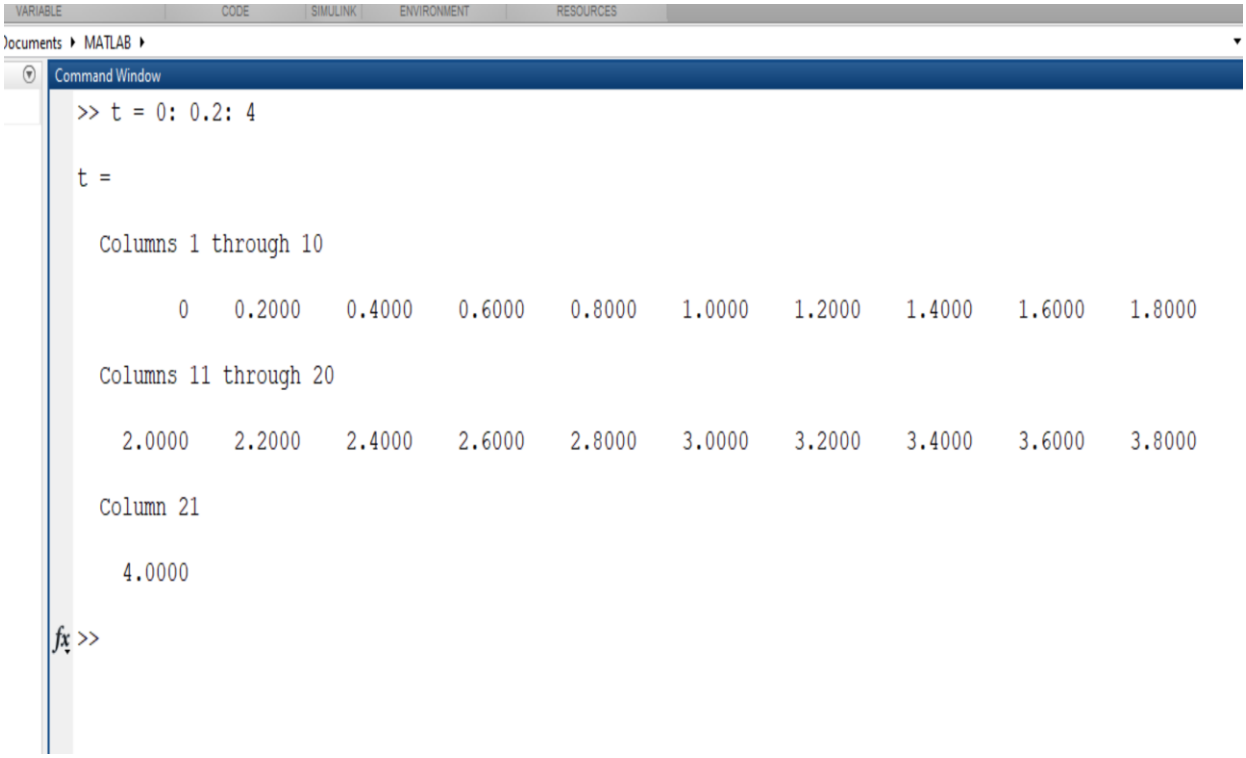
- Explain the results of both operations.

2. Polynomial Representation:

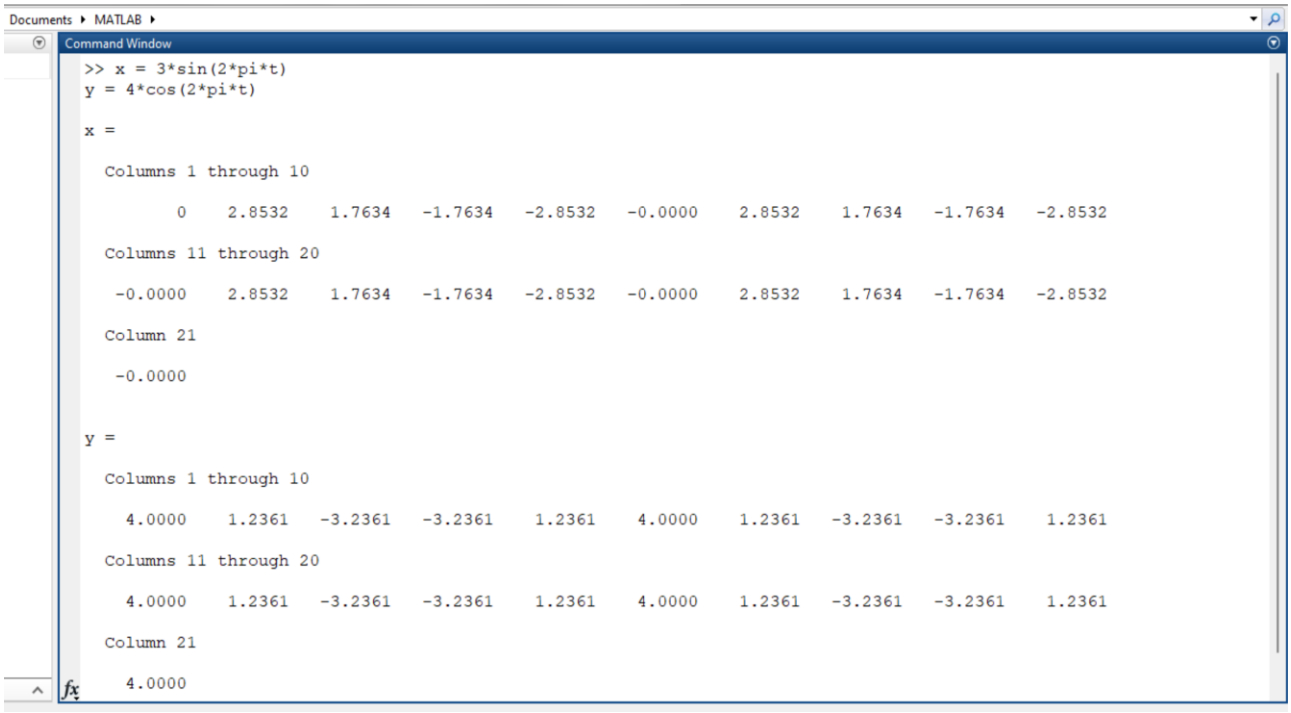
- Represent the polynomial (y =$7.x^3 + 2.x + 10$) in MATLAB using the vector format: y = [7 2 10].

3. Define:

- time starting from t = 0 to t = 4 sec in steps of 0.2 where t = 0:0.2:4.



- ii. x = 3*sin(2*pi*t)
- iii. y = 4*cos(2*pi*t)



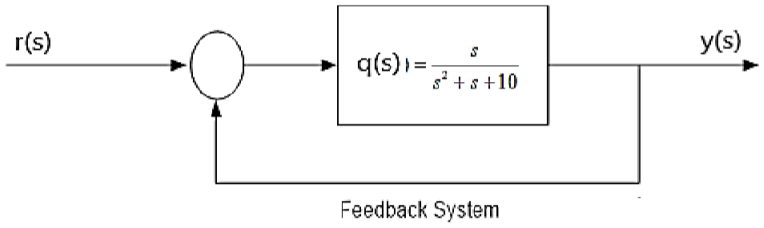
(4) Plot the two graphs and display them in a single screen using subplot function, giving appropriate titles e.g. “Sine wave (frequency 1Hz)” for x and y labels e.g. xlabel and ylabel as “Time (sec)” and “Amplitude (volts)” respectively

(5) Write the transfer function H(S) = $\frac{S}{S^2+2S+10}$ in MATLAB format.

(6) Form a MATLAB command of a closed-loop arrangement of the system shown below for positive feedback as an example, clsys = feedback(q,1)

(7) Generate and plot step response for both H(s) and clsys [on one screen] using different colours of Red and Blue respectively. Example of impulse response using G(s) is impulse (G, 'g', clsys, 'w')

The diagram below relates to part (5), (6) and (7).



8. Consider the spring-mass-damper system shown in Figure below, with mass, m, spring stiffness, k, and damping coefficient, c. The deflection $x(t)$, of the mass is measured from its static equilibrium position (given by x = 0) and is the system's output. A force, $f(t)$, applied to the mass is the input

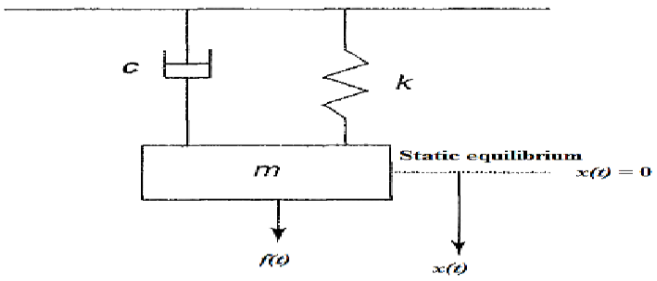


Figure 3: Spring-mass-damper system with input, *f(t)* and output, *x(t)*.

a. Find the differential equation of the mass-spring-damper system.

b. Use Simulink function blocks to plot the velocity and position responses of the differential equation obtained in question **4a.** if ***m*** =1 kg, ***k*** = 10 N/m, **c** =0.1 Ns/m and $f(t) = 400$ N, assuming zero initial conditions.

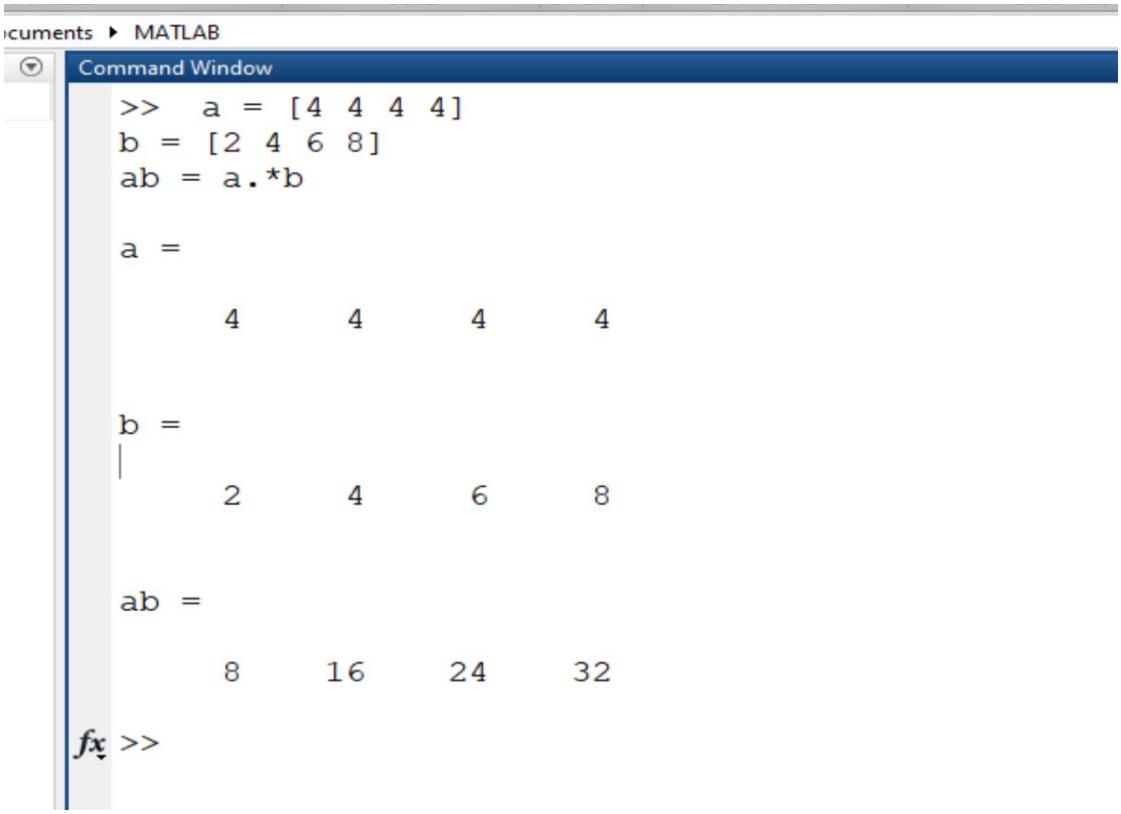
c. Plot both responses if now $f(t) = \text{impluse input}$ with bandwidth frequency of 0.001 Hz created from the input that given in question **4b.**

Hint: Use Newton law to build your differential equation. Plot both the reference set-point and the output responses for better observation.

Experimental Results:

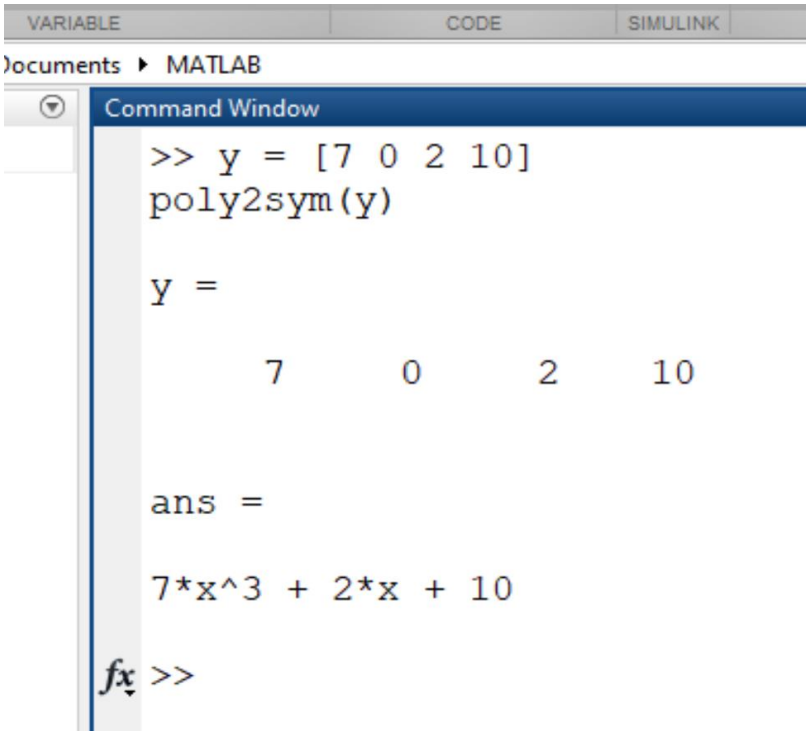
Matrix Multiplication

- Using a.*b, the element-wise product of [4 4 4 4] and [2 4 6 8] gave: [8 16 24 32]
- Using a*b produced an error because the dimensions do not match. After transposing b, the multiplication a*b.' produced the scalar (dot product): 80
- The outer product a.'*b produced a 4×4 times matrix.



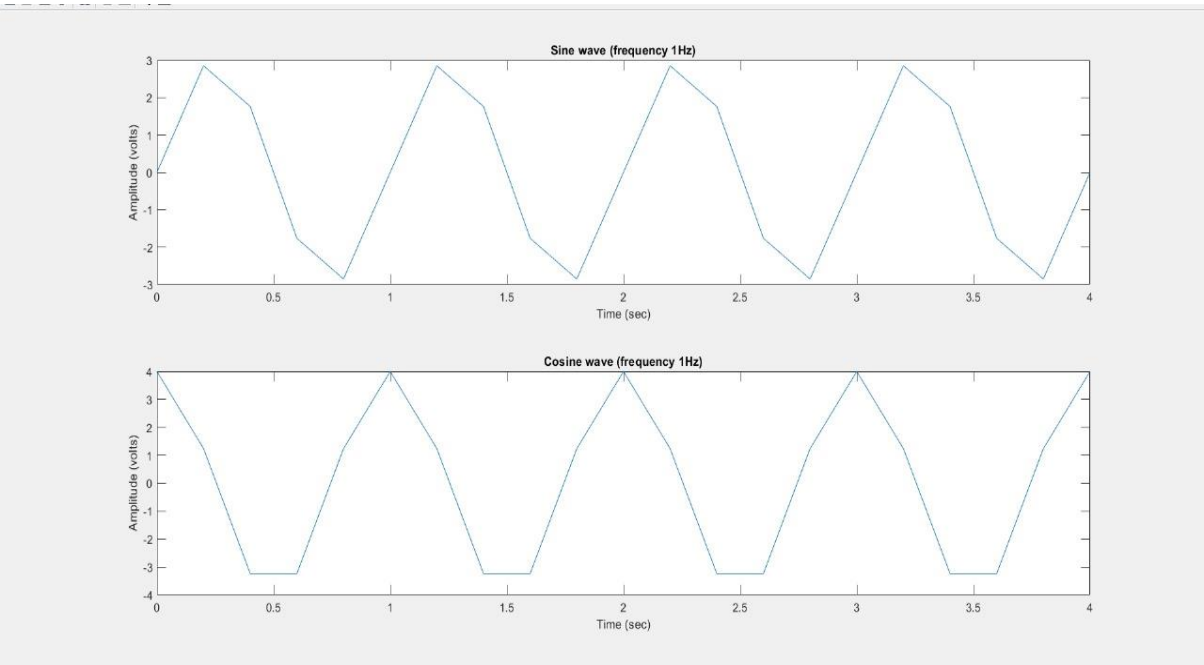
Polynomial Representation:

The resented polynomial $7x^3 + 2x + 10$ in MATLAB: y= [7 0 2 10]
poly2sym (Y)



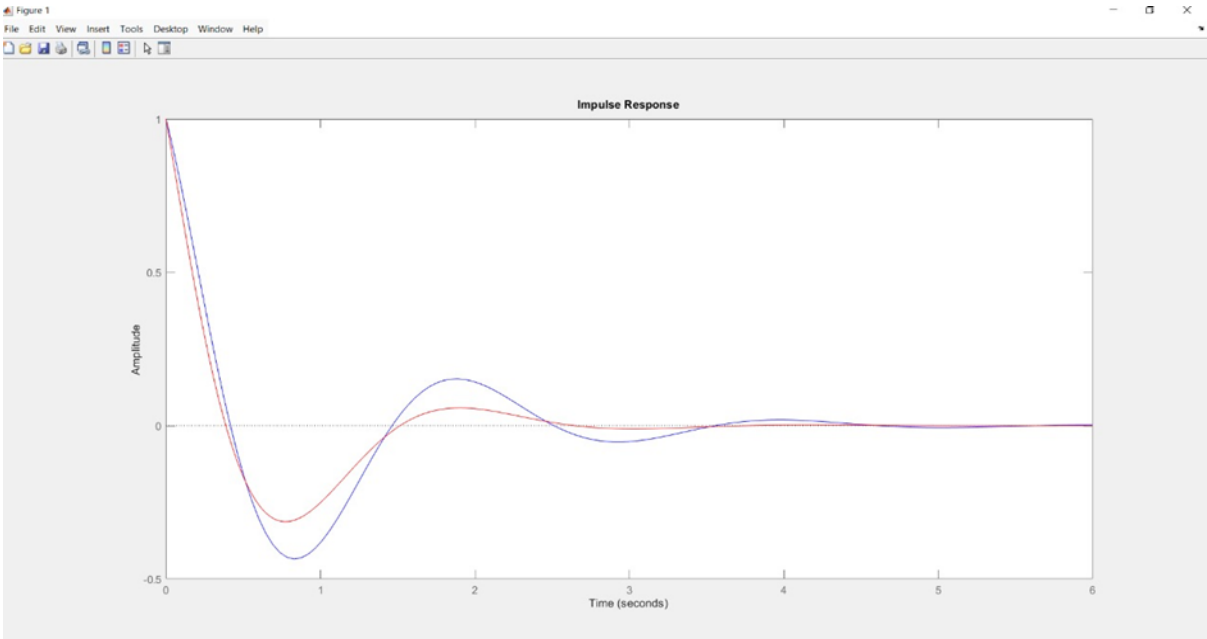
Generated Signal and Plotting:

- The sine and cosine waves plotted using the subplot function, with clear labels and titles is as follows:



Transfer Function

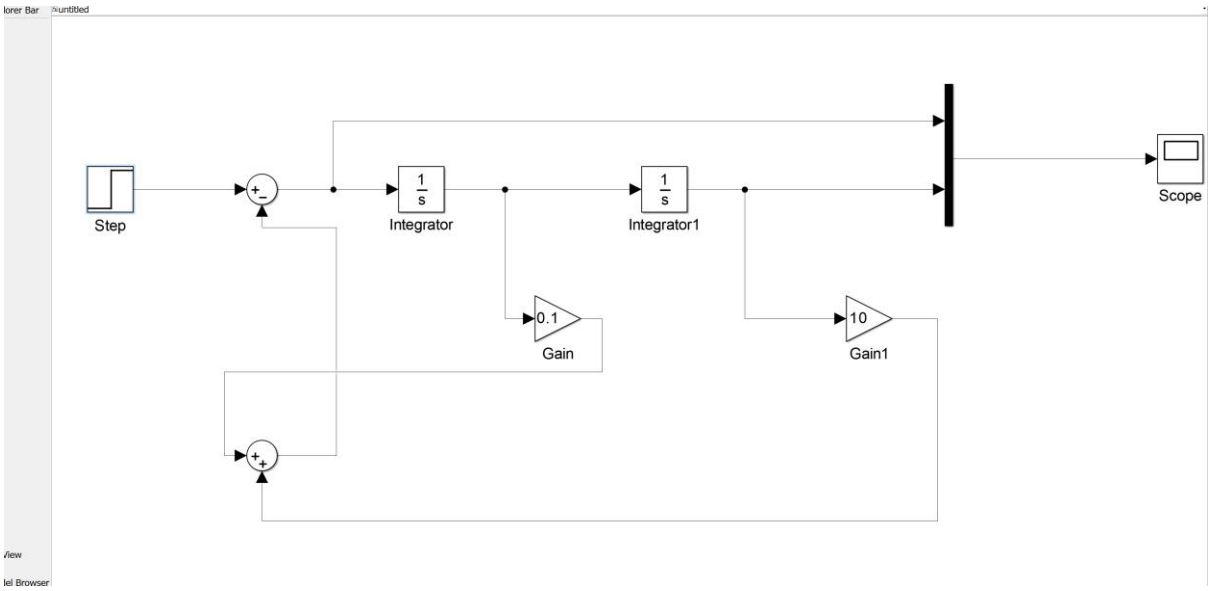
The represented transfer function H(s) and the step response of both the open-loop and closed-loop systems plotted below:

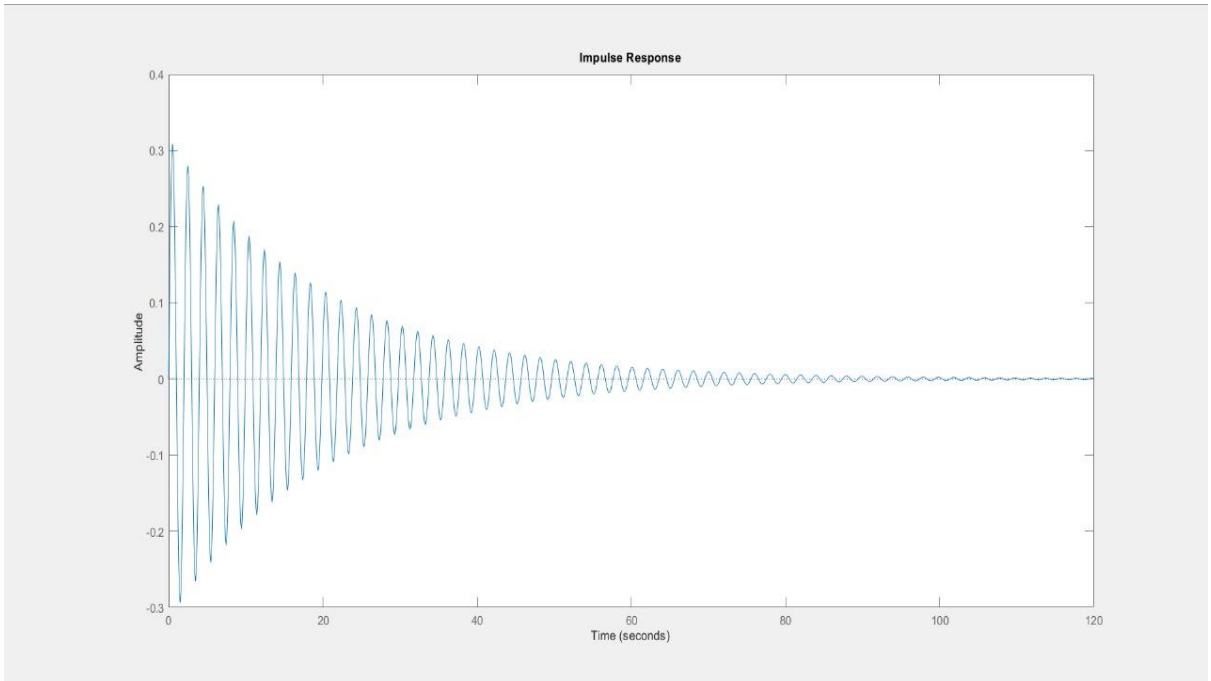


Mass-Spring-Damper Simulation:

- The differential equation is derived as $\frac{d^2x(t)}{dt^2} + \frac{0.1dx(t)}{dt} + 10x(t) = 400$

The following Simulink simulations shows the system's underdamped response to a step input and the rapid decay following an impulse input.





Discussion

The results from this lab confirmed the theory we studied in class. The element-wise multiplication showed how MATLAB handles simple comparisons between matching values in two vectors, while the dot product from matrix multiplication demonstrated how the same vectors can produce a single scalar result when treated as matrices. The polynomial expression was represented correctly in MATLAB, and the plotted signals of sine and cosine waves matched their expected shapes, confirming the correct use of time vectors and trigonometric functions.

The transfer function was successfully created, and the step response plots highlighted the difference between open-loop and closed-loop behaviour. As expected, feedback changed the response characteristics, showing the importance of feedback in controlling stability and performance.

For the spring–mass–damper system, the derived differential equation matched Newton's second law, and the MATLAB simulation produced realistic displacement and velocity responses. The constant force input showed how the mass moved to a steady position, while the impulse input revealed damped oscillations, which is typical for second-order systems with small damping. Overall, the experimental results closely followed the theoretical predictions, with no major differences observed.

Conclusion

This lab demonstrated how MATLAB can be used as a powerful tool for learning and applying control system concepts. We practiced essential operations such as matrix multiplication, polynomial representation, and plotting signals, and then extended these skills to more advanced tasks like building transfer functions and analysing feedback systems. The spring–mass–damper example showed how physical systems can be represented mathematically and simulated effectively. The exercises gave a clearer understanding of how theory connects to practical results, and highlighted the role of MATLAB in simplifying complex calculations and visualizing system behaviour.