Simulation of Higher Order Differential Equations using MATLAB & Simulink

LAB # 05

Fall 2024

CSE-310L Control Systems Lab

Submitted by: **Ali Asghar**

Registration No.: **21PWCSE2059**

Class Section: **C**

"On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work."

Submitted to:

Dr. Muniba Ashfaq

Date:

2nd **November 2024**

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

Objectives:

The objective of this lab is to learn about:

- solving high order differential equation using Matlab.

Introduction:

MATLAB provides thousands of mathematical tools that allow users to solve complex mathematical problems and visualize results. In this case, we are solving a system represented by differential equations using both MATLAB and Simulink. Differential equations describe the rate of change of variables with respect to another, often time, and are foundational in modeling dynamic systems.

MATLAB Command Used:

To solve initial value problems for ordinary differential equations (ODEs), we use the ode45 function in MATLAB, which is particularly useful for solving systems of higher-order derivatives. The command syntax is as follows:

```
[t, Y] = ode45(fun, tspan, y0)
```

• **ode45:** This function numerically solves initial value problems for ODEs using the Runge Kutta method, which is efficient for systems with non-stiff differential equations. It is widely used for its balance of accuracy and computational efficiency.

• **fun:** This represents the function defining the ODE system. The function must be user defined in MATLAB. Once defined, the function can be called in the command window using ode45. Ensure the function file is saved with the same name as the function itself.

• **tspan:** A vector specifying the interval over which to integrate, given as [t0, tf]. The solver starts at t0, imposing initial conditions there, and integrates until tf. If specific time points are desired, they can be provided in tspan = [t0, t1, ..., tf]. The time values must be in ascending or descending order.

• **y0:** A vector specifying the initial conditions for the differential equations.

Simulink:

Simulink is a powerful, icon-driven simulation environment that allows users to create block diagram representations of dynamic systems. Each section of a block diagram corresponds to a specific function or part of the dynamic system, such as inputs, state-space models, transfer functions, or outputs. Simulink is highly visual, allowing users to "draw" connections between blocks, making it especially useful for modeling and understanding complex systems.

Key Steps in Simulink

1. Building the Block Diagram:

- Select and drag blocks from the library to represent various components of the system.
- Components may include integrators (for calculating derivatives), summation blocks, product blocks, and scopes for visualizing outputs.
- Connect the blocks by drawing lines between them to define relationships and data flow.

2. Setting Block Parameters:

- Each block requires specific parameters, such as gain values in transfer functions or initial conditions for integrators.

3. Configuring Simulation Parameters:

- In the simulation settings, define the integration method, step size, start and end times for the simulation.

Task 01:

Simulate the below mentioned system which is represented by differential equations in Matlab.

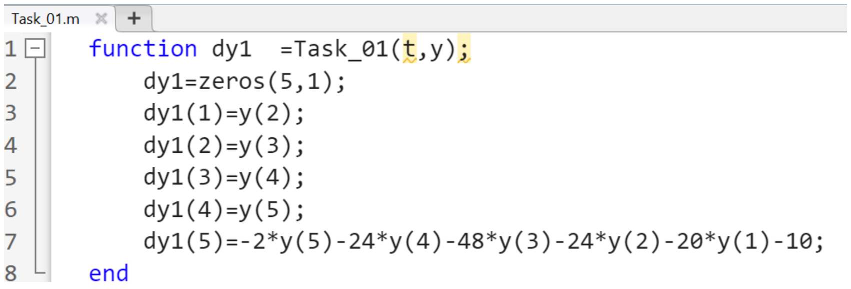
$$\frac{d^5y}{dt^5} + 2\frac{d^4y}{dt^4} + 24\frac{d^3y}{dt^3} + 48\frac{d^2y}{dt^2} + 24\frac{dy}{dt} + 20y + 10 = 0$$

$$y(0) = 2, y'(0) = 5, y''(0) = 10, y'''(0) = -4, y^{iv}(0) = -7$$

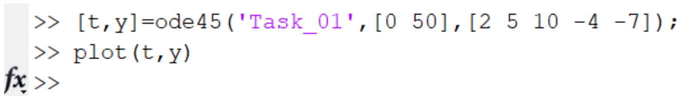
Also simulate it in Simulink and match the results.

MATLAB:

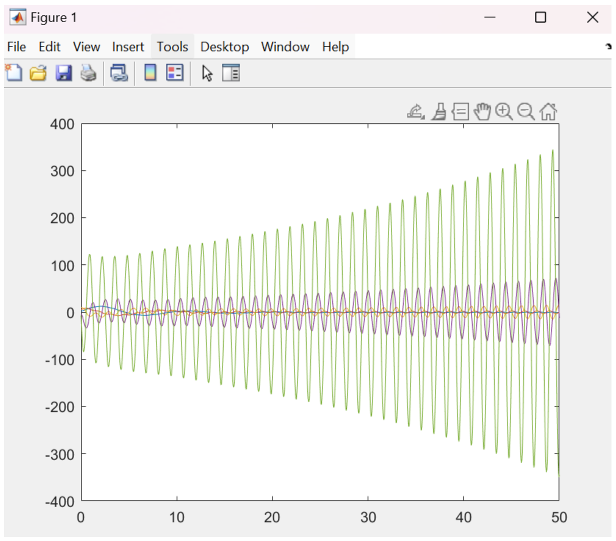
Code:


```
function dy1  =Task_01(t,y);
    dy1=zeros(5,1);
    dy1(1)=y(2);
    dy1(2)=y(3);
    dy1(3)=y(4);
    dy1(4)=y(5);
    dy1(5)=-2*y(5)-24*y(4)-48*y(3)-24*y(2)-20*y(1)-10;
end
```

Function Call:

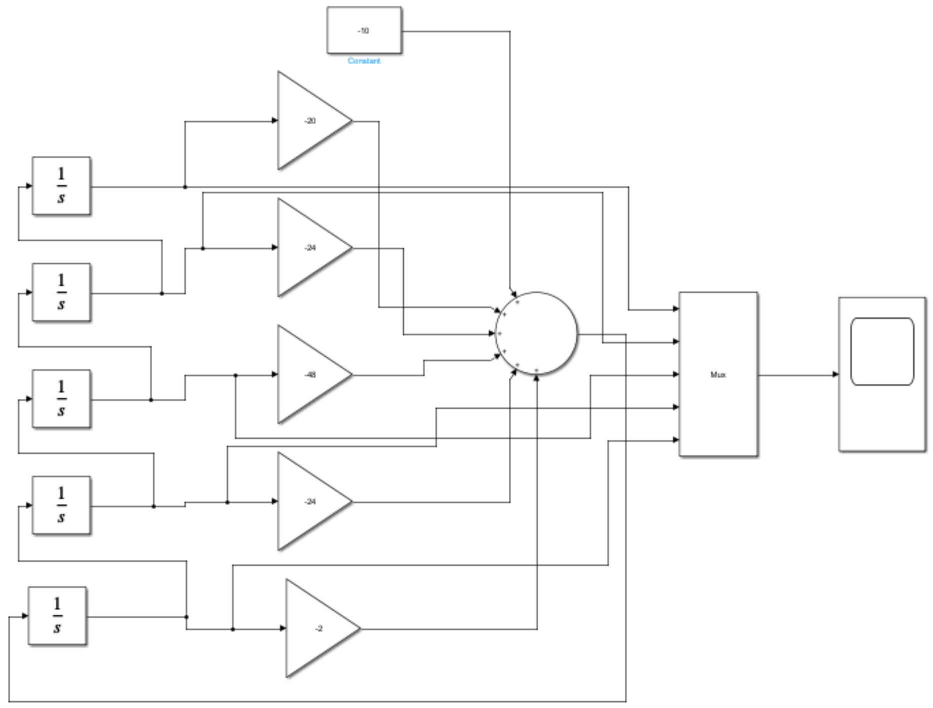

```
>> [t,y]=ode45('Task_01',[0 50],[2 5 10 -4 -7]);
>> plot(t,y)
>>
```

Output:

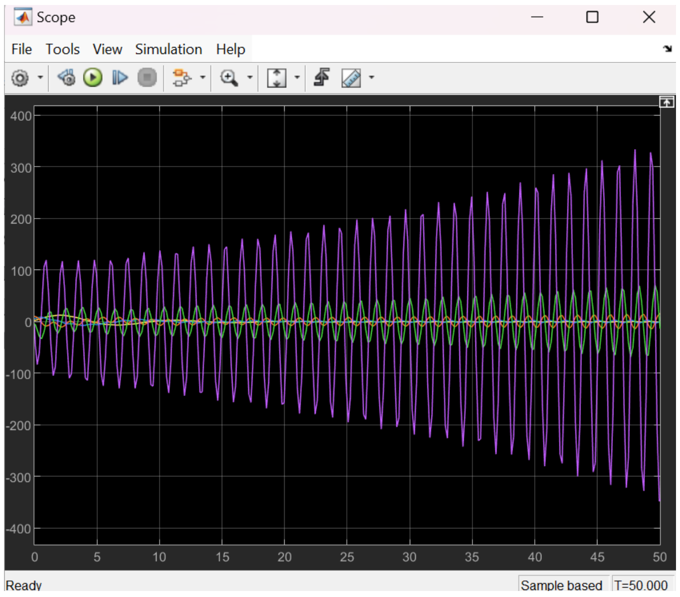


Simulink:

Block Diagram:



Output:



Conclusion:

In this lab, I learned how to implement low order differential equations in MATLAB and Simulink.