

# Simulation of Mechatronic Systems

## Exercise 3 protocol

Alessandro Rotondi - 27445613

Frederic Heil - 28742953

Nithin Badri - 17546426

Sudeesh Suragari - 14646468

December 19, 2025

### Exercise 3a): Simulating the step response of a transfer function

In the first sub task of this exercise, we want to simulate the step response of the transfer-function **PT1** which can be defined as follows:

$$G(s) = \frac{K}{1 + T \cdot s} \quad (1)$$

Hereby, variables  $K$  and  $T$  are to be defined as  $K = 5$  and  $T = 4$ . Furthermore, the differential equation of **PT1** is given as follows:

$$\dot{x} = \frac{1}{T}(K \cdot u - x) \quad (2)$$

Now that this has been established, we can build both equations 1 and 2 within Simulink which yields the following models:

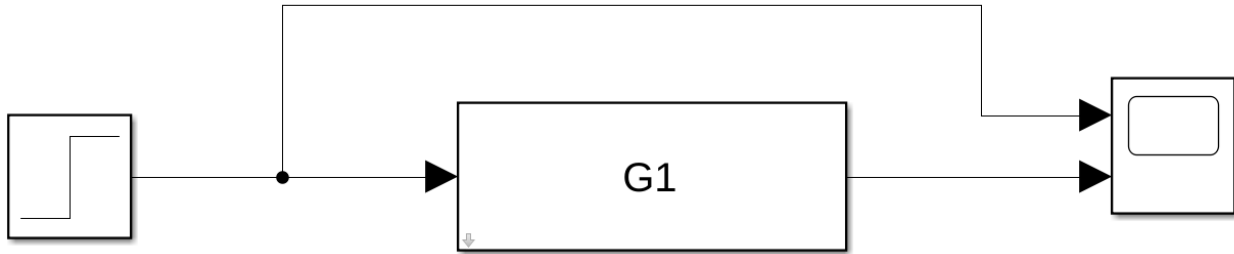


Figure 1: Step equation of **PT1** conveyed as Simulink model.

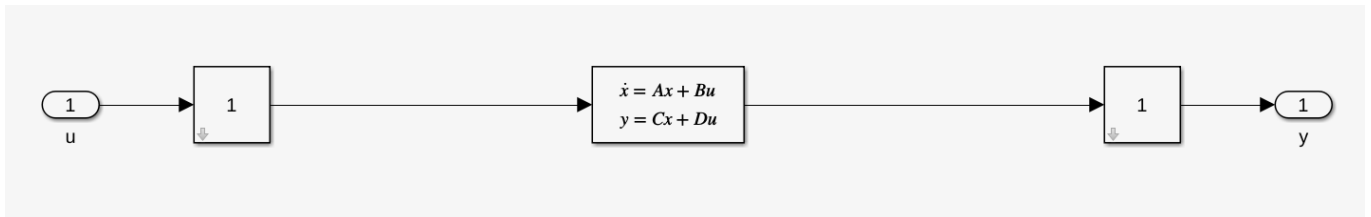


Figure 2: Inner workings of the G1 block of the step equation Simulink model as shown in figure 1.

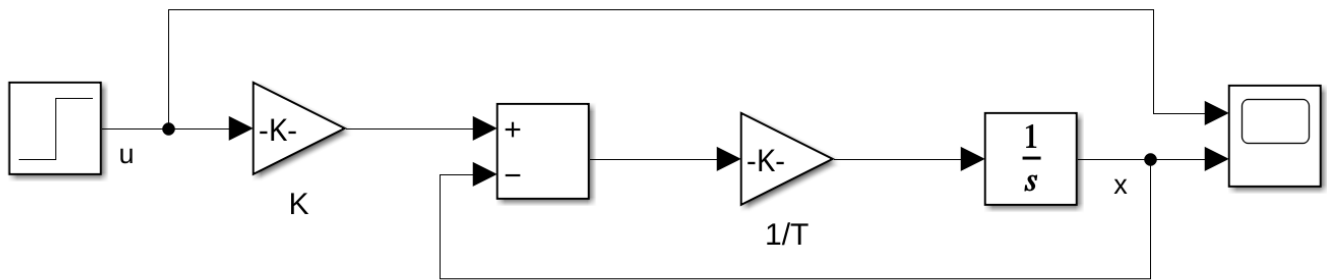


Figure 3: Differential equation of **PT1** conveyed as Simulink model.

As shown in figure 3 above, the value of  $u$  gets simulated by a step block. Both simulated values of  $u$  and  $x$  get forwarded to the scope block, which lets us examine their values within a Matlab script. Something that is not directly apparent from the graphic is the simulation run time, which has been set to 50 seconds.

Within our Matlab code, we can set and access the Simulink simulation as demonstrated in the code snippet below:

#### Matlab Code Snippet:

```

1 clear; clf; clc;
2 % PARAMETERS EXERCISE 1
3 par1.K = 5;           % Gain
4 par1.T = 4;           % Time constant
5 par1.u = 1;           % step value
6 par1.t.start = 0;     % start simulation time
7 par1.t.stop = 50;     % stop simulation time
8
9 out1 = sim('exercise3_SIM');           % simulate the system with
    simulink and get the scope datas
10
11 figure()
12 plot(out1.plotG1_Tf.time,out1.plotG1_Tf.signals(1).values,...           % Plot the result of
    the simulation in simulink (transfer function)
13     out1.plotG1_Tf.time,out1.plotG1_Tf.signals(2).values)
14 grid on, xlabel('Time [sec]'); legend('step','out'); title('step response transfer function
    SIMULINK');
15
16 figure()
17 plot(out1.plotG1_De.time,out1.plotG1_De.signals(1).values,...           % Plot the result of
    the simulation in simulink (differential equation)
18     out1.plotG1_De.time,out1.plotG1_De.signals(2).values)
19 grid on, xlabel('Time [sec]'); legend('step','out'); title('step response differential
    equation SIMULINK');

```

Hereby, we can access concrete scope variables using `plotG1.Tf` and `plotG1.De`, a logging value which has been defined within our Simulink models and is not directly apparent from any graphic.

The plot code of the step response yields the following graphics 4 and 5:

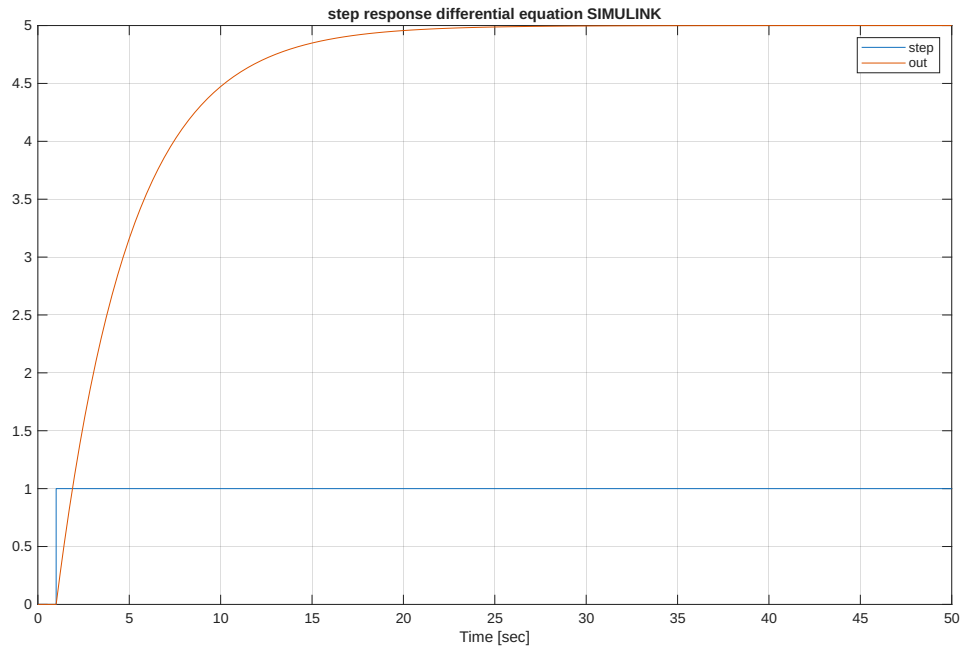


Figure 4: Step response of our Simulink model plotted within Matlab.

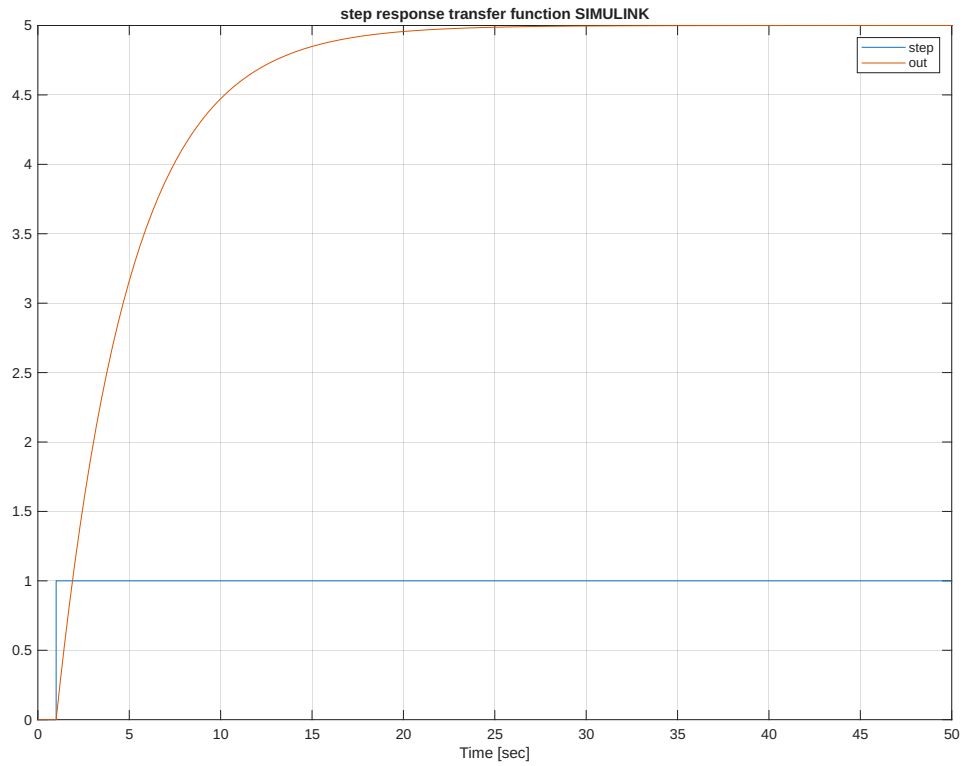


Figure 5: Step response of our Simulink model plotted within Matlab.

As expected, the output **out** of the step function **PT1** converges at a value of 5, which corresponds to the value of parameter  $K$ . Also, the output of both models is identical which also corresponds to the expected result.

## Exercise 3b): Building a one-mass system in Simulink

Next, we want to simulate a one mass spring system and investigate the influence of the physical parameters on the simulation result. The physical system is illustrated with the following graphic below:

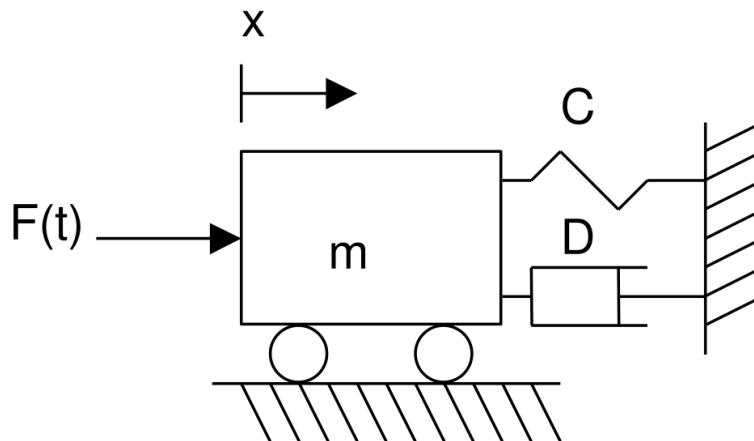


Figure 6: One-mass spring/damper system with some physical characteristics.

Hereby the movement can be described with the following differential equation which is already given by the exercise description as follow:

$$m \cdot \ddot{x} = -C \cdot x - D \cdot \dot{x} + F(t) \quad (3)$$

The initial values are defined as  $m = 10\text{kg}$ ,  $C = 100 \frac{\text{N}}{\text{m}}$  and  $D = 3 \frac{\text{Ns}}{\text{m}}$ . Using equation 3 we can build our model in Simulink which looks like this:

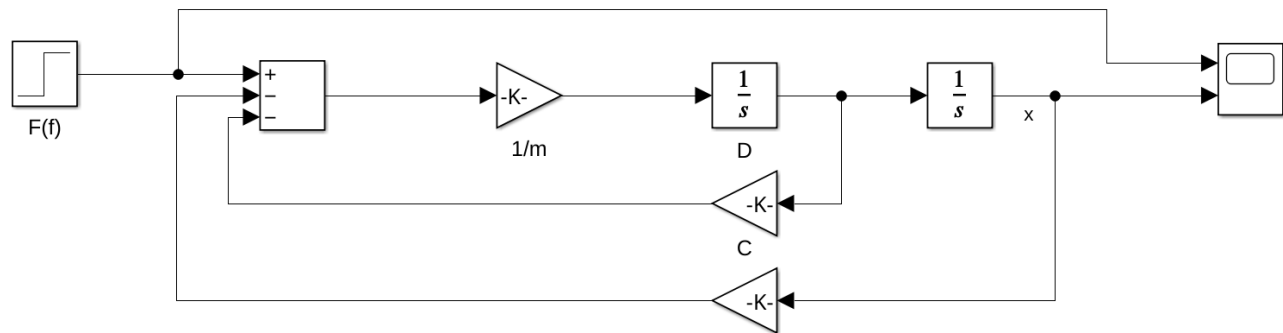


Figure 7: Differential equation of one-mass spring/damper system conveyed as Simulink model.

This model uses two integrator blocks to depict signals of  $\ddot{x}$ ,  $\dot{x}$  and  $x$  whose outputs get multiplied by their corresponding gain blocks. Furthermore, the initial force  $F(t)$  is simulated by a step block.

To investigate the influence of physical parameters on the simulation result, we are at first going to run the simulation with its initial values. Then, for each physical property we are going to execute the simulation twice, once with a decreased and once with an increased value, so that only one property changes at a time. For  $m$ ,  $D$  and  $C$  this yields the following plots:

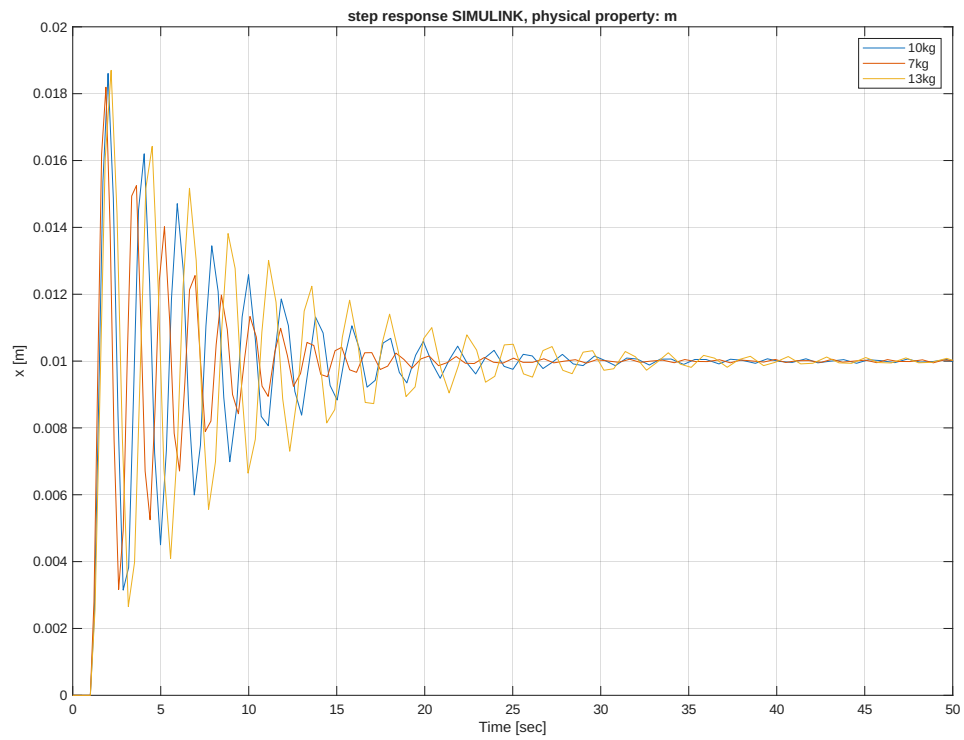


Figure 8: Step response with different values for mass.

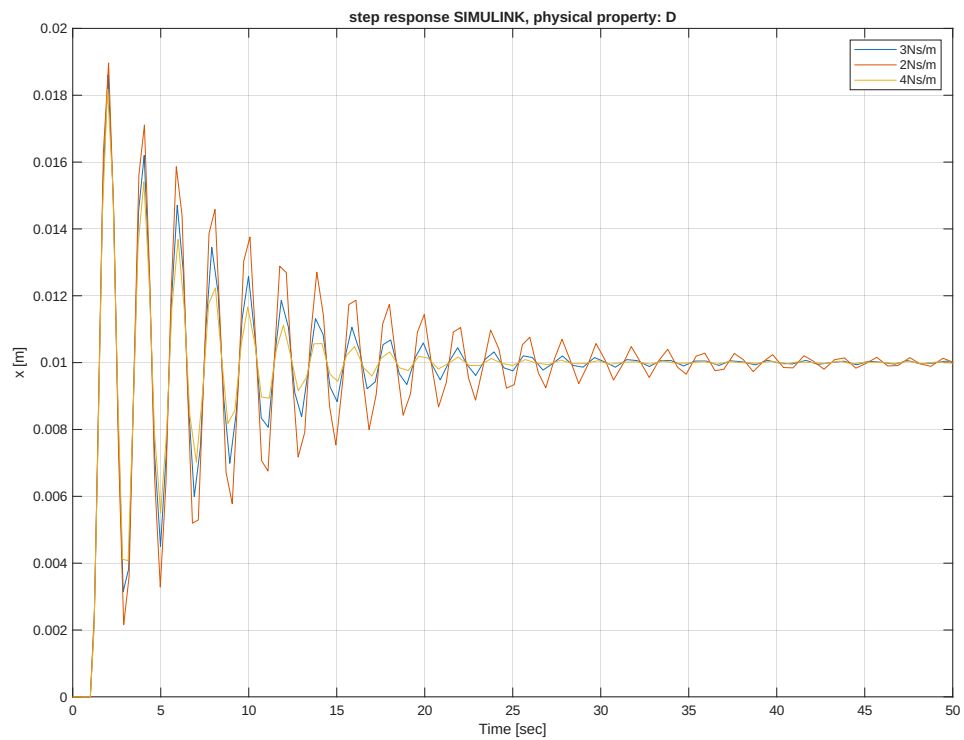


Figure 9: Step response with different values for the damper constant.

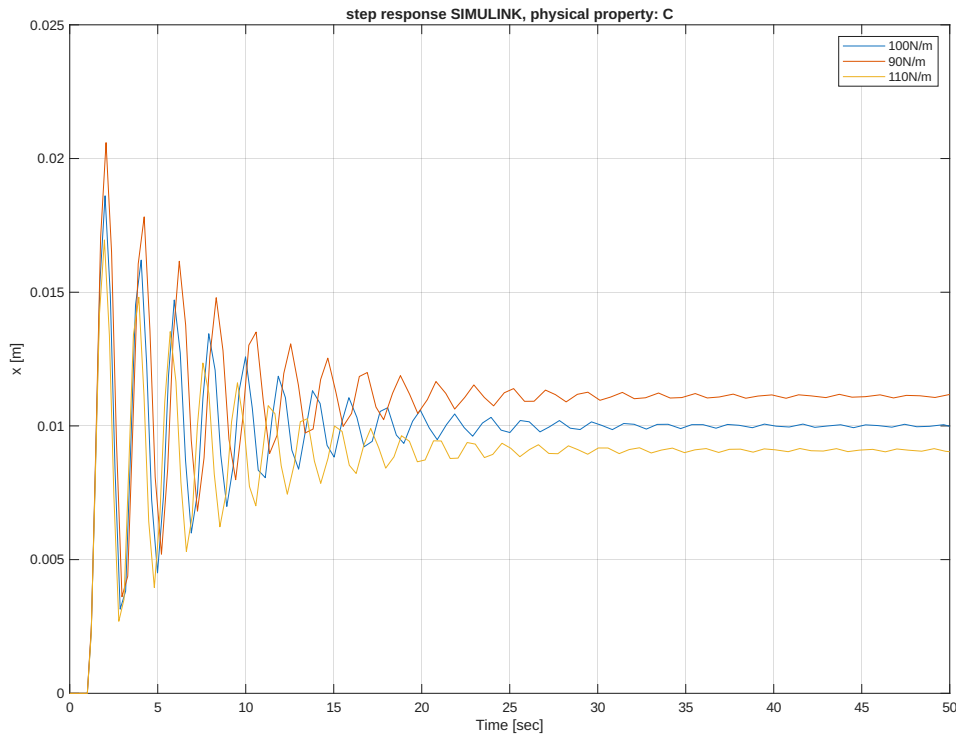


Figure 10: Step response with different values for the spring constant.

**Influence of mass (m):** As shown in figure 8, changing the mass influences oscillation amplitude and interval. A reduced mass leads to shorter interval and smaller amplitude, while a mass increase leads to longer interval and larger amplitude.

**Influence of damper constant (D):** As shown in figure 9, changing the damper constant influences only the amplitude, more specifically the rate of amplitude decrease over time. A reduced damper constant leads to a slower decrease of amplitude over time, while an increased damper constant leads to a faster decrease of amplitude over time. The oscillation interval is not being influenced by the damper value.

**Influence of spring constant (C):** As shown in figure 10, changing the spring constant influences the value to which  $x$  converges towards. A reduced spring constant leads to a larger convergent value of  $x$ , while an increased spring constant leads to a smaller convergent value of  $x$ .

The Matlab code for generating these plots and running the simulation is provided in the snippet below.

**Matlab code snippet:**

```

1 % EXERCISE B -----
2 out2 = sim("exercise3_SIM.slx"); % step response with initial parameters
3 timeInitial = out2.plotE2.time;
4 stepResponseInitial = out2.plotE2.signals(2).values;
5
6 par2.m = 7; % reduced mass m
7 out2 = sim("exercise3_SIM.slx");
8 timeReducedM = out2.plotE2.time;
9 stepResponseReducedM = out2.plotE2.signals(2).values;
10
11 par2.m = 13; % increased mass m
12 out2 = sim("exercise3_SIM.slx");
13 timeIncreasedM = out2.plotE2.time;
14 stepResponseIncreasedM = out2.plotE2.signals(2).values;
15 par2.m = 10; % revert m to initial
16
17 par2.D = 2; % reduced damper constant D
18 out2 = sim("exercise3_SIM.slx");
19 timeReducedD = out2.plotE2.time;

```

```

20 stepResponseReducedD = out2.plotE2.signals(2).values;
21
22 par2.D = 4;    % increased damper constant D
23 out2 = sim("exercise3_SIM.slx");
24 timeIncreasedD = out2.plotE2.time;
25 stepResponseIncreasedD = out2.plotE2.signals(2).values;
26 par2.D = 3; % revert D to initial
27
28 par2.C = 90;    % reduced spring constant C
29 out2 = sim("exercise3_SIM.slx");
30 timeReducedC = out2.plotE2.time;
31 stepResponseReducedC = out2.plotE2.signals(2).values;
32
33 par2.C = 110;    % increased spring constant C
34 out2 = sim("exercise3_SIM.slx");
35 timeIncreasedC = out2.plotE2.time;
36 stepResponseIncreasedC = out2.plotE2.signals(2).values;
37 par2.C = 100; % revert C to initial
38
39 % plot comparing different masses
40 figure(21);
41 plot(timeInitial, stepResponseInitial, ...
42      timeReducedM, stepResponseReducedM, ...
43      timeIncreasedM, stepResponseIncreasedM)
44 grid on, xlabel('Time [sec]'), ylabel('x [m]');
45 legend('10kg', '7kg', '13kg');
46 title('step_response_SIMULINK, physical_property: m');
47
48 % plot comparing different damper constants
49 figure(22);
50 plot(timeInitial, stepResponseInitial, ...
51      timeReducedD, stepResponseReducedD, ...
52      timeIncreasedD, stepResponseIncreasedD)
53 grid on, xlabel('Time [sec]'), ylabel('x [m]');
54 legend('3Ns/m', '2Ns/m', '4Ns/m');
55 title('step_response_SIMULINK, physical_property: D');
56
57 % plot comparing different spring constants
58 figure(23);
59 plot(timeInitial, stepResponseInitial, ...
60      timeReducedC, stepResponseReducedC, ...
61      timeIncreasedC, stepResponseIncreasedC)
62 grid on, xlabel('Time [sec]'), ylabel('x [m]');
63 legend('100N/m', '90N/m', '110N/m');
64 title('step_response_SIMULINK, physical_property: C');

```

### Exercise 3c): Building and optimizing a two-mass system in Simulink

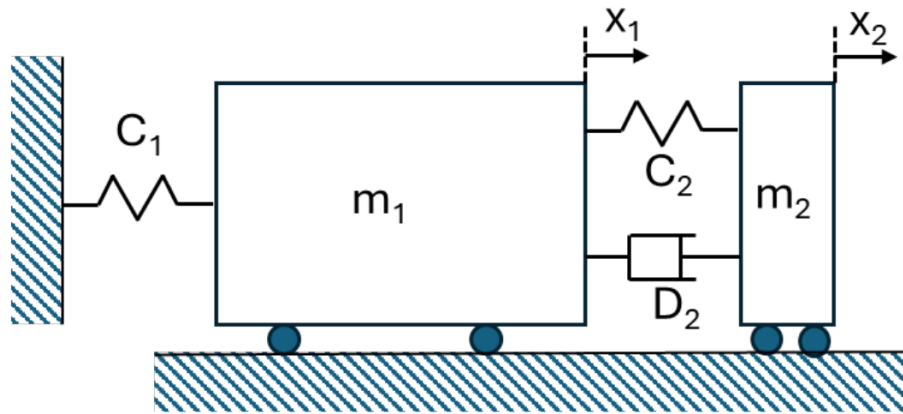


Figure 11: Two-mass spring/damper system with some physical characteristics.

The model to be simulated is illustrated in figure 11 above.