FACULTY OF ENGINEERING, UNIVERSITY OF JAFFNA

CONTROL SYSTEMS – EC5030

LABORATORY SESSION 2

MODELING CONTROL SYSTEMS IN MATLAB SIMULINK

REGISTRATION NUMBER: 13/E/

AIM: Modelling and analysing a second order closed loop system in Simulink

Modeling a DC motor control system in MATLAB and Simulink

SOFTWARE: MATLAB and Simulink

PART 1 – MODELING AND ANALYSING A SIMPLE CONTROL SYSTEM USING MATLAB SIMULINK

Here, you will be modelling a simple control system using the in-built blocks available in the Matlab Simulink.

PROCEDURE:

1. Initially click to the "Start" button of Matlab and open the Simulink Library Browser. Find the "math", "Source", "Sink" and "continuous" option in the left hand side of the dialog box. Familiarize with the following blocks.

    a. Sum (Math Library)

    A dialog box obtained by double-clicking on the SUM block performs the configuration of the SUM block, allowing any number of inputs and the sign of each. The sum block can be represented in two ways in Simulink, by a circle or by a rectangle. Both choices are shown in Fig. 1.

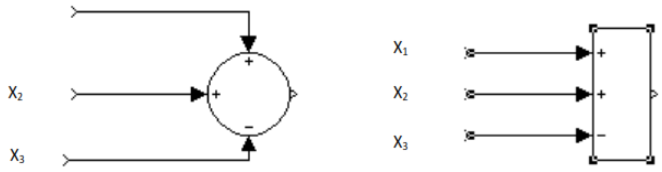


Fig. 1. Sum block

    b. Gain (Math Library)

    A gain block is shown by a triangular symbol as in Fig. 2, with the gain expression written inside if it will fit. If not, the symbol - k - is used. The value

used in each gain block is established in a dialog box that appears if the user double-clicks on its block.

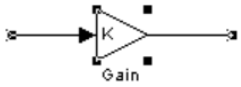


Fig. 2. Gain Block

c. Integrator (Continuous Library)

The block for an integrator as shown in Fig. 3a as a quantity 1/s coming from the Laplace transform expression for integration. When double-clicked on the symbol for an integrator, a dialog box appears allowing the initial condition for that integrator to be specified.

Alternatively, a second input to the block can be displayed to supply the initial condition explicitly, as in Fig. 3b. Initial conditions may be specific numerical values, literal variables, or algebraic expressions.

Fig. 3. Integrator with (a) implicit initial conditions (b) explicit initial conditions

d. Constants (Source Library)

Constant block creates a constant as in Fig. 4. Double-clicking on the symbol opens a dialog box to establish the constant's value.

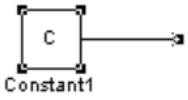


Fig. 4. Constant Block

e. Step (Source Library)

A Simulink block is provided for a Step input, a signal that changes (usually from zero) to a specified new, constant level at a specified time. These levels and time can be specified through the dialog box, obtained by double-clicking on the Step block.

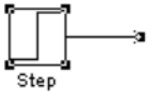


Fig. 5. Step Input Block

f. Scope (Sink Library)

The system response can be examined graphically, as the simulation runs, using the Scope block in the sinks library. It is possible to include several Scope blocks. Also it is possible to display several signals in the same scope block using a MUX block in the signals routing library. The Scope normally chooses its scales automatically to best display the data.

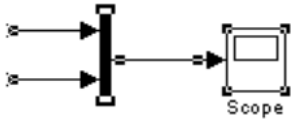


Fig. 6. Scope Block with MUX

g. Now create a Simulink "Model". Save it in the name of "Name_Regno.mdl".

h. Model a second order system as shown in Fig. 7. You can drag the blocks and drop it in the Simulink model. Consider the second order system transfer function as in (1). Provide a step input to the system and obtain the output waveform from the scope by running the program.

Note: Use the button "Autoscale" in the plot to obtain appropriately scaled plot.

$$H(s) = \frac{1}{s^2+10s+20} \qquad (1)$$

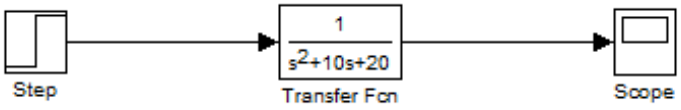


Fig. 7. Second Order Open Loop system

i. Model the second order closed loop system as in Fig. 8 with a proportional controller. Provide the proportional gain $K_p = 100$.

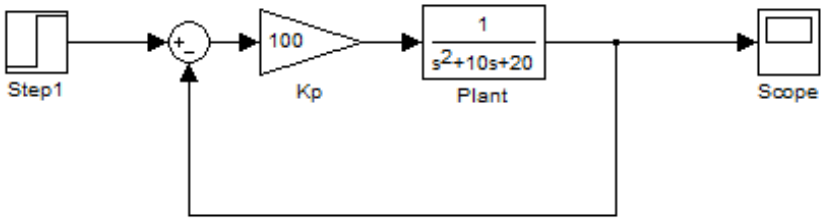


Fig. 8. Plant with Proportional controller

j. Change the value of K_p to 300 and plot the variation in the output waveform.

k. Now create a PI controller as in Fig. 9 to the closed loop system. Observe the output waveform by changing the K_i value from 70 to 100.

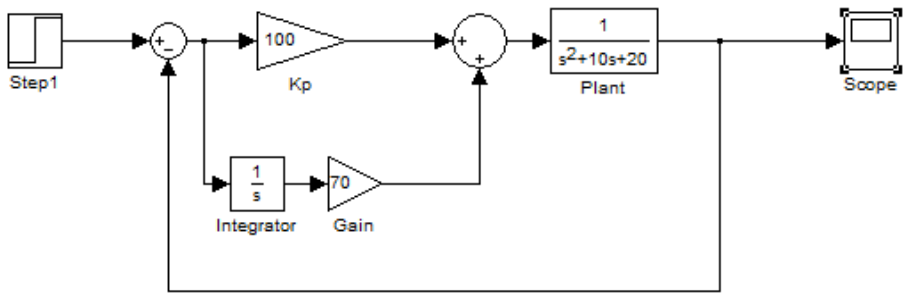


Fig. 9. Closed loop system with PI controller

l. Now prepare a PD controller for a closed loop system as in Fig. 10 and plot the waveform for different K_d values starting from 10 to 20.

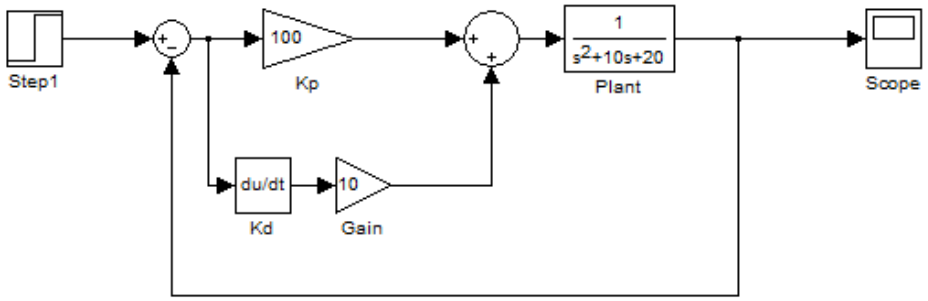


Fig. 10. Closed loop system with PD controller

m. Model a PID controller with K_p=100, K_d=10 and K_i=70 for the second order closed loop system in Fig. 11 and plot the outputs. Here, MUX is used to plot both input and output in the same graph.

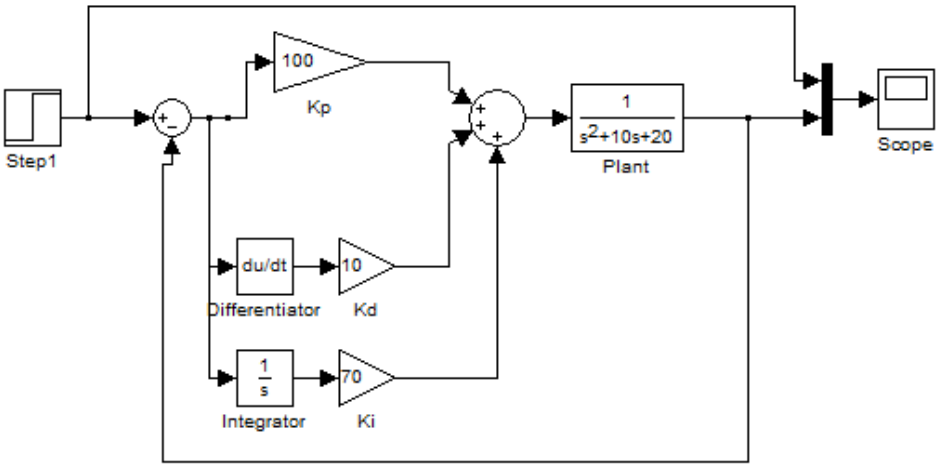


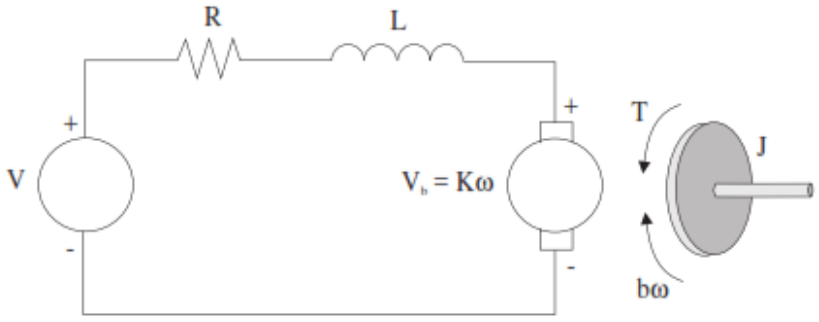
Fig. 11. Closed loop system with PID controller

PART 2: MODELING A DC MOTOR AND CONTROLLER

In this example we will learn how to develop a linear model for a DC motor, how to analyze the model under MATLAB (poles and zeros, frequency response, time-domain response, etc.), how to design a controller, and how to simulate the open-loop and closed-loop systems under SIMULINK.

PROCEDURE:

1. Consider the following physical system of a DC motor, whose electric circuit of the armature and the free body diagram of the rotor are shown in Fig. 12.



Fig, 12. Schematic Representation of a DC motor

2. Consider the following values for the physical parameters. The input is the armature voltage V in Volts (driven by a voltage source). Measured variables are the angular velocity of the shaft is ω in radians per second, and the shaft angle ϑ in radians. The block diagram for the DC motor is obtained as in Fig. 13. The transfer function from the input voltage, $V(s)$, to the output angle, $\Theta(s)$ is given in (2).

moment of inertia of the rotor	$J = 0.01$ kg· m^2
damping (friction) of the mechanical system	$b = 0.1$ Nms
(back-)electromotive force constant	$K = 0.01$ Nm/A
electric resistance	$R = 1\ \Omega$
electric inductance	$L = 0.5$ H

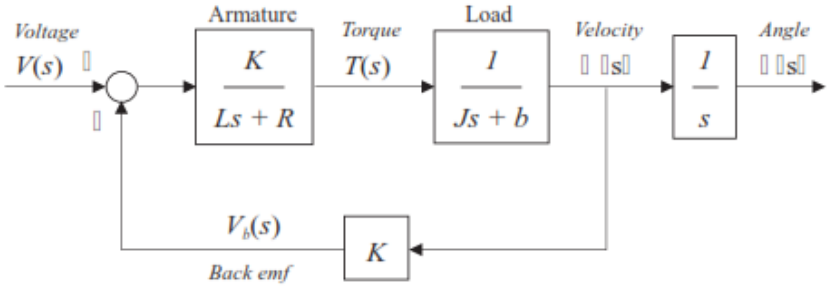


Fig. 13. Block diagram representing DC motor

$$G_a(s) = \frac{\theta(s)}{V(s)} = \frac{K}{s[(R+Ls)(Js+b)+K^2]}. \quad (2)$$

3. Obtain the MATLAB representation of the DC motor as follows. Initially use M-file for coding.

```
%% DC Motor Model
J=0.01;
b=0.1;
K=0.01;
R=1;
L=0.5;

%% Method(1)
aux = tf(K,conv([L R],[J b]))   % or aux = tf(K,[L R])*tf(1,[J b]);
Gv = feedback(aux,K);   % transfer function for voltage and angle
Ga = tf(1,[1 0])*Gv;    % transfer function for voltage and angular
velocity

%% Method (2)
s = tf([1 0],1);
Gv = K/((L*s + R)*(J*s + b) + K^2);   % transfer function for voltage
and angle
Ga = Gv/s;     % transfer function for voltage and angular velocity

% Labeling the input and output
Gv.InputName = 'Voltage';
Gv.OutputName = 'Velocity';
Ga.InputName = 'Voltage';
Ga.OutputName = 'Angle';
```

4. Convert G_v and G_a into their respective state-space (function ss) and zero-pole-gain (function zpk) representations.

5. What are the poles and zeros of the system? Is the system stable? Why?

6. How can you use MATLAB to find out whether the system is observable and controllable?

7. Obtain the time and frequency domain responses of the transfer function of DC motor. Use the following code.

```
%% Plotting the time and frequency domain responses
figure(1);
step(Ga);
hold on
step(Gv);

figure(2);
impulse(Ga);
hold on
impulse(Gv);

figure(3);
bode(Ga);
```

```
hold on
bode(Gv)
```

8. Design a PID controller for the DC motor circuit as in Fig. 14. Use the code as follows.

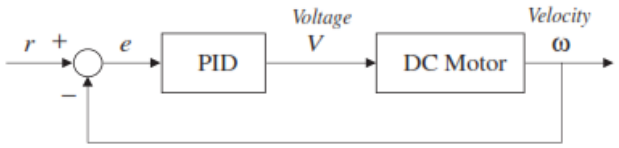


Fig. 14. PID controller for DC motor

```
%% Adding a PID controller
Kp = 1;
Ki = 0.8;
Kd = 0.3;
C = tf([Kd Kp Ki],[1 0]);

C = tf([Kd Kp Ki],[1 0]);   % (Kd*s^2+Kp*s+Ki)/s
Gc = feedback(Ga*C,1);
figure(4);
step(Gc)
```

9. Select the overall gain of the PID controller such that the controller is stable and has the desired location of the poles (within the defined ratio among the K_p, K_i and K_d constants). If the design is not satisfactory, this ratio can be changed. Plot the step response of the output.

```
figure(5)
rlocus(Ga*C);
Kp = rlocfind(Ga*C)

Gc1 = feedback(Ga*C*Kp,1);
figure(6);
step(Gc1)
```

10. Obtain a Simulink model for the motor as in Fig. 15. Plot the step response for angle and angular velocity.

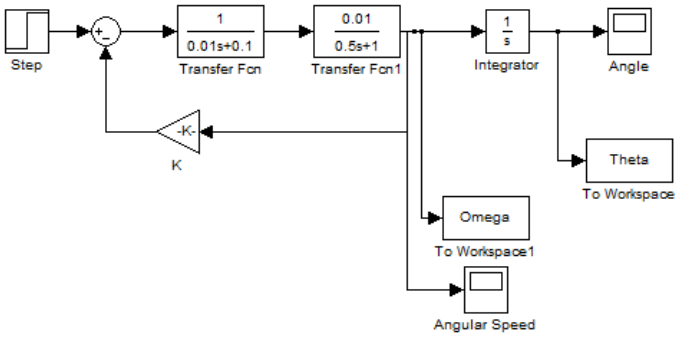


Fig. 15. Simulink model for DC motor

11. Create a Simulink file with PID controller to control the DC motor. Use “subsystem” to model an inbuilt DC motor as in Fig. 16.

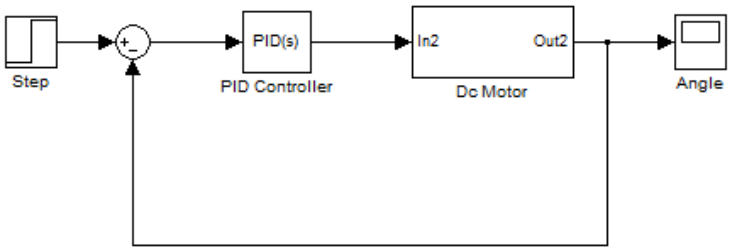


Fig. 16. DC Motor control in Simulink with PID controller

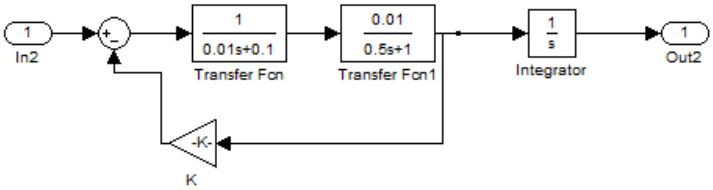


Fig. 17. DC Motor subsystem in Simulink