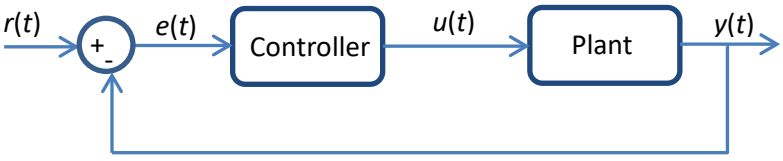
Homework Assignment #2 (ME-190, Fall 2017)

Due date: Tuesday, Sept. 19, 2017, 1:30 pm.

Objective: Develop a simulink model of a PID controller for a free mass system.

Background: Block diagram of a basic control system is shown below:



Where the "Plant" block represents the system to be controlled, and the "Controller" block contains the control logic, $r(t)$ is the reference trajectory, $e(t)$ is the tracking error, $u(t)$ is the plant input, and $y(t)$ is the plant output. A good controller will force $y(t)$ to converge to and stay on $r(t)$, keeping $e(t)$ near zero.

Problem Statement: In this exercise, we would like to simulate and tune a proportional-integral-derivative (PID) controller for positioning a free mass system using Matlab and Simulink. PID controller is one of the most widely used control structures in practice. To do this, take the plant as:

$$m\ddot{y}(t) = u(t) \tag{1}$$

Which simply is the equation of motion for a free mass m, with position $y(t)$, subject to force $u(t)$.

Take the controller as:

$$u(t) = k_P e(t) + k_I \int_0^t e(\tau)d\tau + k_D \dot{e}(t) \tag{2}$$

where k_P, k_I, and k_D are the proportional, integral, and derivative gains to be chosen and tuned by the control designer.

Finally, the positioning (tracking) error is given by:

$$e(t) = r(t) - y(t) \tag{3}$$

Exercise 1: Combine Eq.(1)-(3) into a single differential equation with $y(t)$ and $r(t)$ being the only variables. Your final equation should not include $e(t)$ or $u(t)$, or their derivatives or integrals.

Exercise 2: Develop a simulink model for the differential equation of the system obtained in Exercise 1 for the following parameter values and solver setting:

m = 0.5 [kg],

k_P = 200, k_I = 1 , and k_D = 5

Solver: “Fixed Step” OED 5.

Duration: 0-1 s, with 0.01 s time step.

Let $r(t)$ be the unit step function at time 0.

- ➔ Develop a Matlab script that sets the parameter values, and calls the Simulink model.
- ➔ Plot $r(t)$ and $y(t)$ vs time on the same plot after the Simulink model is called. For this, use “To Workspace” block to take r and y to Matlab workspace. Double click and modify the variable names, and set the format to “Array”. Also, send time vector to workspace using the “Clock” and “To Workspace” blocks.

Exercise 3: Keeping the values of k_I and k_P at their initial values (i.e., $k_P = 200$, $k_I = 1$) find the (integer) value of k_D that brings the overshoot just below 5%. This means the maximum value of $y(t)$ should be less than but close to 1.05 m. Use trial and error to find k_D. This is a common practice in controller tuning.

Exercise 4: Replace the “step” function with a “sine” function, and set its frequency to 1Hz (2*pi*1 rad/s). Keep the rest of the parameters at the default values. Plot $r(t)$ and $y(t)$ vs time on the same plot. Can the system track the reference trajectory precisely? Plot the error vs. time by plotting $r(t) - y(t)$ to see the amplitude of the tracking error.

Exercise 5: Change the value of k_P to 1000, (keep $k_I = 1$) and find the (integer) value of k_D which leads to less than 5% overshoot in response to the unit step input. Now, repeat Exercise 4 (sine wave tracking) and check if the amplitude of the tracking error is reduced.

Exercise 6 (Optional): To see what the downside of choosing large control gains is, plot $u(t)$, the force applied to the mass, for both low and high gain values. What difference do you observe? Which one would be more practical if there is a limit on $u(t)$? To access $u(t)$ get $\ddot{y}(t)$ and multiple it by m.

Document your codes and results to the above problems, and turn in the hard copy including all your scripts, plots, simulink diagrams, and conclusions. Make sure your plots include representative labels and titles. Use Matlab’s “Publish” feature to create the report.