Laboratory Activity #2:
DC Motor Control

ENGR 4220/5220: Control Systems
Professor Hill
University of Detroit Mercy, Summer 2013

---

1 Introduction

In this lab exercise we will apply some analysis and controller design techniques that we have learned in class to the control of a DC motor. In particular, it is desired to design a PI compensator to control the motor's speed so that the following requirements are achieved:

2% settle time	less than 0.30 seconds
peak time	less than 0.15 seconds
percent overshoot	less than 10%
steady-state error	approximately 0

Additionally, it is desired that the controller be able to reject load disturbances and that the controller use the least amount of control effort (power) possible while still satisfying the above goals. The diagram given in Figure 1 represents the system for which we are attempting to design the controller.

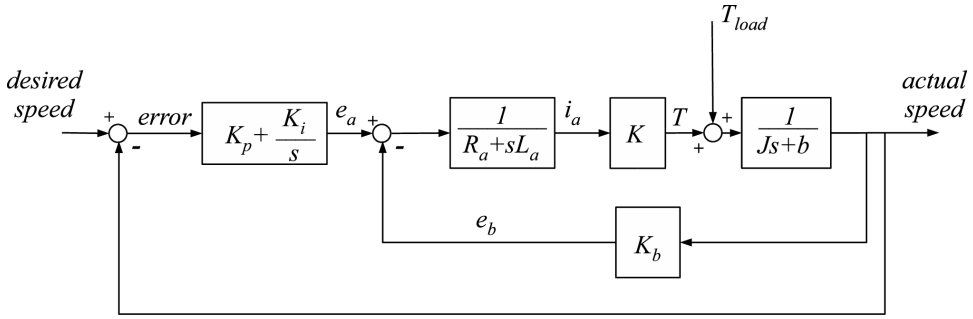

Figure 1: DC motor speed feedback control with disturbance

For the purposes of design, we will assume a simplified first-order motor model based on the results of the first laboratory experiment. This simplified system is shown in Figure 2.

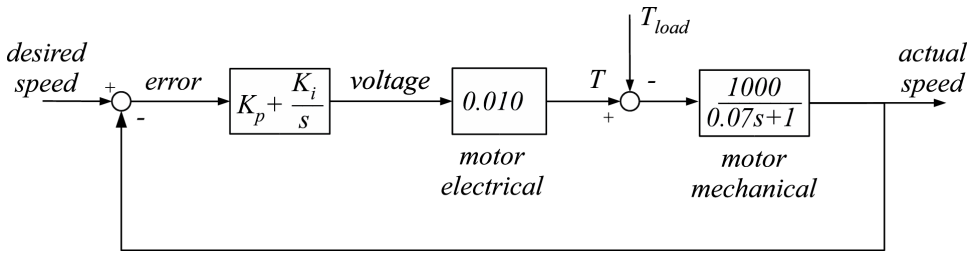

Figure 2: Simplified model of DC motor speed feedback control

2 Controller design and analysis

In this section we will vary the gains of our controller to meet the above goals and to attempt to better understand some concepts learned from class.

2.1 Experimental Set-up

We begin our experiment by setting up the system that we will use for commanding and observing the DC motor. The set-up is very similar to the one used in the first experiment.

1. *Connect Hardware* - For this experiment, connect the positive terminal of the DC-generator/motor armature to connector **PHASE A2** on the power electronics board and the negative terminal to connector **PHASE B2**. Next connect **CURR.A2** (motor armature current) of the power electronics board to the analog input channel **ADCH5** of the I/O board. Also attach the encoder output (mounted on the DC-generator/motor) to the **INC1 9-Pin D SUB** connector of the I/O board. Power is supplied to the electronics board from a variable DC power supply. Set the DC power supply to provide 42 V and set the current limit sufficiently high to source what is needed by the motor. Ask for instructor permission before switching on the output of the power supply and switching on the **Analog Power** on the power electronics board.

2. *Simulink set-up* - Start the MATLAB software (version R2007a) on the desktop computer. When prompted identify the model of dSPACE board being used as 1104. Within MATLAB, set the working directory to **C:/Documents and Settings/Student/Desktop/5020LAB2/**. Once in this directory, open the Simulink model **expPartA.mdl**. Also set the global variables for DC voltage being supplied and sample time within MATLAB. This is done by typing `Vd = 42` and `Ts = 0.0001` at the MATLAB command line. Explore the Simulink model to get a sense of what is being done in the program.

3. *Build the executable files* - Pressing **CTRL+B** will build the control system real-time executable code. Specifically, the C-code that will run on the controller board is generated by Simulink, while a second .sdf "variables file" to be used by the dSPACE ControlDesk software is also generated.

4. *Control interface set-up* - Start the dSPACE ControlDesk software on the desktop computer. Open the .sdf file containing the variables needed by the control software by clicking **File>Open Variable File**. This file can be found in the same working directory used above and is entitled **exp parta.sdf**. Next open the layout file **expPartA.lay** from the same directory by clicking **File>Open**. Click the **PLAY** button (green arrow) to begin the control program (if it is not already selected), then switch to **Animation Mode** as shown in the Figure 3 below. Gently experiment with the controls to command the drive motor. Observe the resulting angular speed, armature voltage, and armature current. In order to stop the system, revert the system to **Edit Mode** and click the **STOP** button. If the **Capture Settings Window** is not open, you can display it clicking **View>Controlbars>Capture Settings Window**. Within this window, set the **Length** to 10 seconds. The Capture Settings Window may initially open collapsed against the right side of the ControlDesk.

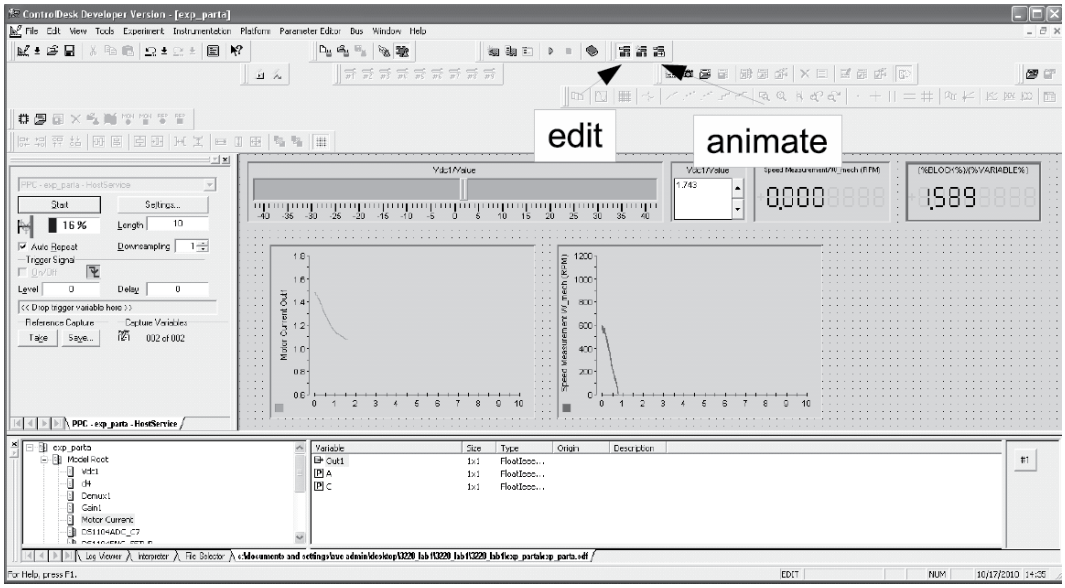

Figure 3: Various control buttons for starting and stopping the system

2.2 Controller Tuning and Experimentation

Refer to the controller you designed for this system in Problem 2(b) of Homework 9. Set the proportional gain K_p to the value chosen in this homework problem and set K_i initially to zero. With the system initially at 0 rpm, command a step reference of 250 rpm (or larger if it won't trip the breaker). Save the resulting data as a **.mat** file. Now add the integral control as chosen in your Homework problem and repeat the previous step response experiment. Note the change in behavior and again save the resulting data. How does the resulting time response behavior compare to that indicated by the closed-loop pole locations, as calculated in your homework for a canonical 2nd order system? Does the observed difference make sense? You may need to import your

step response data into MATLAB so that you can zoom in close enough to identify the necessary parameters. Recall that after importing your .mat file to the MATLAB workspace (for example **control.mat**), the time data can be accessed from **control.X.Data** and the speed, voltage, and current data can be accessed from **control.Y(i).Data** where **i** represents the corresponding index (1 = voltage, 2 = current, 3 = speed).

Now attempt to retune your gains to reduce the overall armature voltage command while still meeting the necessary requirements. You may open the **SISO Design Tool** within MATLAB to assist you. Once you have settled on a controller design, save an example closed-loop step response.

For your newly designed controller, run the system at some steady speed. Then add a load disturbance by slowing the spindle of the load motor with your hand. How does the system behave when you apply the disturbance? Does your observation agree with your expectations? Save data capturing your system's disturbance rejection behavior.

3 Lab Assignment

In the interest of saving time, you may perform this assignment entirely in simulation employing the system model shown in Figure 2. The lab activity will then just be performed to reinforce what you see in simulation. If you choose this option, you must work individually. If you wish to work in a group of two or three, then you must answer the following questions with support from actual experimental data. The assignment is due Friday, August 2nd.

1. (20 points) Present graphs of your system's step response for the original K_p gain alone, and then with the addition of the original K_i gain. Explain what you observed and if it is what you expected. For the PI controller, determine from your data the resulting settle time, peak time, and percent overshoot. Are the values consistent with what you expected based on your results from Problem 2 of Homework 9? Explain.

2. (40 points) Present a graph of your system's step response for the modified PI controller. From your data, determine the resulting step response parameters and describe how you used the root locus and/or Bode plot to help you tune your controller. Compare the plot of armature voltage for the step response using this controller as compared to the original controller. (This is basically a repeat of Problem 2(c) from Homework 9.)

3. (10 points) Compare and contrast a pole placement (root locus) approach to controller design and a frequency response approach. Be specific to this example.

4. (15 points) Present a graph exemplifying your system's disturbance rejection properties. Does the observed behavior agree with the theory you have learned in class and the model you are assuming? Explain.

5. (15 points) Use MATLAB to generate the open-loop bode plot for this system with original and modified PI controller. Explain what the two Bode plots imply about your system's performance using these two different controllers.