CONTROL SYSTEMS LABORATORY

ECE410F-Fall 2003

LAB0: LABORATORY INTRODUCTION

The purpose of this laboratory is to introduce you to the equipment, so that you are familiar with the servomotor system, the computer hardware and software, and your TA before attempting Lab 1. There is no lab report, and nothing to submit, but you must demonstrate to your TA that you know:

- how to wire the servomotor system
- how to compile and run the real-time software; and
- how to load experimental data into MATLAB and plot it.

0.1 Preparation:

If you have not done so, read the *Introduction to Simulink, init.m*, and *Control.mdl*, before beginning the Laboratory.

0.2 Laboratory Procedure:

Note: Please do not attempt to connect the ribbon cable from the bench-top interface board to the computer. If the ribbon cable is disconnected, please ask your TA to connect it.

1- Study Appendix 1 to learn how to work with WinCon server.

2- Compile and run control.mdl. It is not necessary for the power amplifier to be driving the motor here, i.e. you may disconnect the output of the power amplifier from the servomotor input, although the power amplifier must be “on”, so the POT is biased. While the software is running, try to calibrate your DC-motor set following the procedure given in LAB1.

3- Start the control program using the wincon server and move the shaft of the motor randomly in order to generate a signal. Stop the program after approximately 5 seconds. (To stop a real time program in an exact time, you can drag and drop the block ``stop simulation" from “sinks” tool and apply a unit step to it by a “step” block at the time you wish to stop the simulation. (Note that by setting the stop time in the “simulation parameters” of simulink, you cannot stop the program, since it is running in “external” mode.)

4- Save the data collected in your workspace, and then go to workspace and plot the signal which have you generated. You can also save the collected data as a .mat file for later use such as in printing plots for your reports.

0.3 Report:

There is no laboratory report for this lab. Although Lab0 is not counted toward your final grade, it is important to become familiar with the equipment. Otherwise, you will have a difficult time completing Lab1.

CONTROL SYSTEMS LABORATORY

ECE410F-Fall 2003

LAB1: DC MOTOR CALIBRATION AND IDENTIFICATION

1- DC MOTOR CALIBRATION:

1.1 Introduction:

The output of the DC-Motor system in the lab is the angle θ of its shaft. This angle is measured by a potentiometer. This potentiometer has a gain *Gpot* and an offset *Opot*:

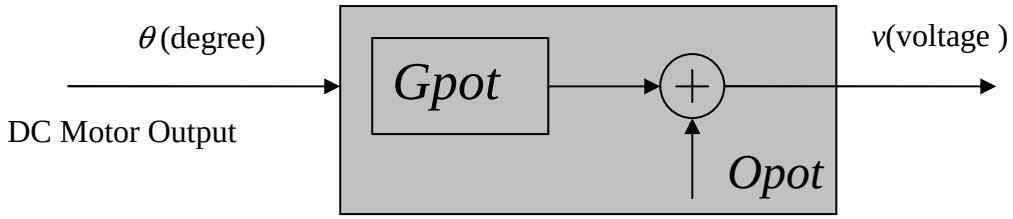

Figure 1.1: Mathematical model of a potentiometer

so that $v = Opot + Gpot \cdot \theta$

The values of *Gpot* and *Opot* are related to the voltage of the internal power supply and mechanical coupling. To have a correct value of θ for control purpose, you have to compensate for the effect of the potentiometer.

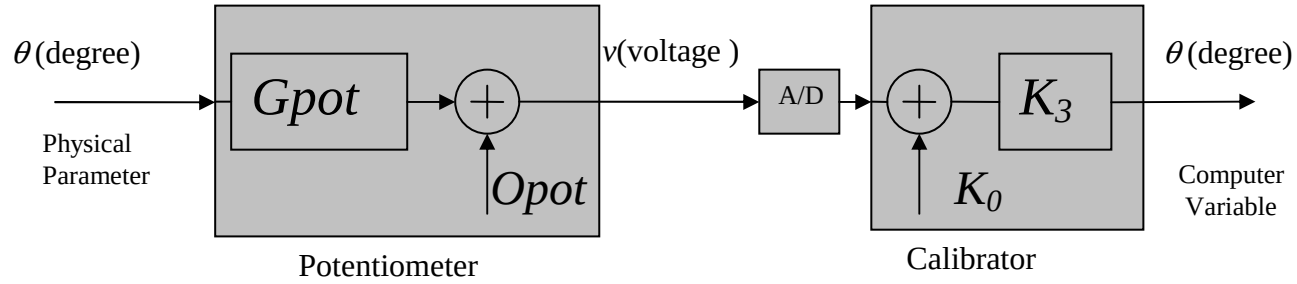

Figure 1.2: Compensation of the potentiometer characteristic by the calibrator block

Here if you choose $K_3 = 1/Gpot$ and $K_0 = -Opot$, then you will obtain the value of the physical variable θ as a variable in your computer.

Since the characteristic of the potentiometer and its offset differs from set to set, you have to determine K_3 and K_0 for your own DC-Motor set. Since you are going to compensate a linear potentiometer, a two-point calibration can be used to find K_3 and K_0.

$$\begin{cases} \theta_1 = K_3(v_1 + K_0) \\ \theta_2 = K_3(v_2 + K_0) \end{cases} \Rightarrow \begin{cases} K_0 = \dfrac{\theta_2 v_1 - \theta_1 v_2}{\theta_1 - \theta_2} \\ K_3 = \dfrac{\theta_1 - \theta_2}{v_1 - v_2} \end{cases} \qquad (1.1)$$

1.2 Potentiometer Calibration Procedure:

Note: Please do not attempt to connect the ribbon cable from the bench-top interface board to the computer. If the ribbon cable is disconnected, please ask your TA to connect it.

1- Supply the potentiometer with a voltage of +12*v* and -12*v*. Disconnect the motor wiring, or apply a zero voltage to the D/A so that the motor does not move. Connect the output of the potentiometer to an A/D channel (Channel 0 is recommended since it is a default channel).

2- In matlab, open *init.m* and run it to initiate variables. Then open *control.mdl* and build it by choosing *build* under *wincon* in the Simulink pull down menu. (See Appendix 1)

3- If there is no error, a Wincon server will be launched automatically and you can choose values or signals to view from this server. Open the subsystem *Calibrator Solver*. Go to *wincon* and select the four display signals showing K_0, K_3, *Voltage*, and *Angle*. Then Start the program by pushing *Start* button. Now you should be able to read a reasonable voltage value of A/D when you move the shaft of the potentiometer.

4- Move the shaft to a specific angle say +60 degrees, then set the value of "Theta1" inside the calibrator block to the same value you have set physically. Then apply a rising edge at "Set1" by forcing it to zero and then force it to one. (At this point, you have latched the information for one of the calibration points.)

5- Move the shaft to another angle say -60 degrees, then set the value of "Theta2" inside the calibrator block to the same physical value and apply a rising edge signal to "Set2" to latch the information of the second point for calibration.

6- Now you can observe the appropriate values for K_0 and K_3 inside the display boxes. To confirm that these values are correct, you must observe the same physical angle of the shaft in the *Angle* display box when you move the rotor. Record the values of K_0 and K_3 right away to use for your next experiments. These values must be updated in the workspace. To do, stop the wincon, copy these numbers in the *init.m* file, and run *init.m* to update the workspace. Keep in mind that this recording will not be done automatically and if you lose this information, you have to calibrate the potentiometer again.

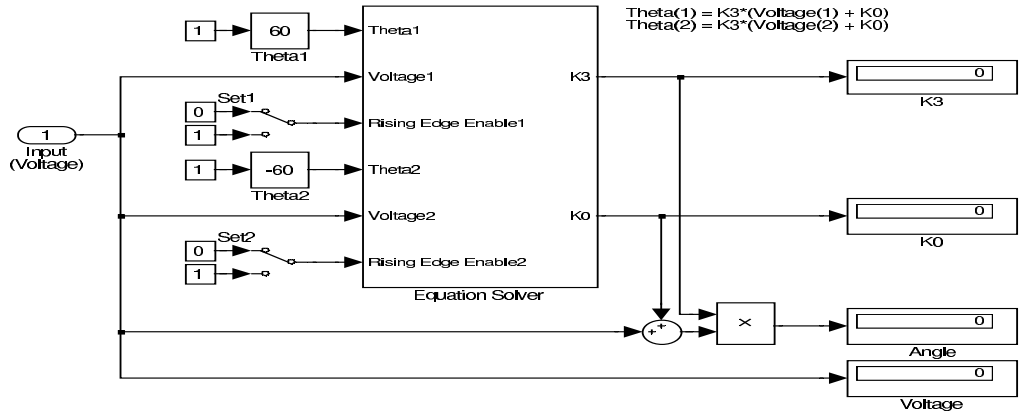

Figure 1.3: *Calibrator Solver* subsystem

2- DC MOTOR MODEL IDENTIFICATION:

In this experiment, you will identify the parameters of a servomotor system, i.e. compute numerical values for the two parameters *Kp* and *T* that completely characterize the system. These values will be used for control design.

2.1 Preparation:

The servomotor system is given as follows:

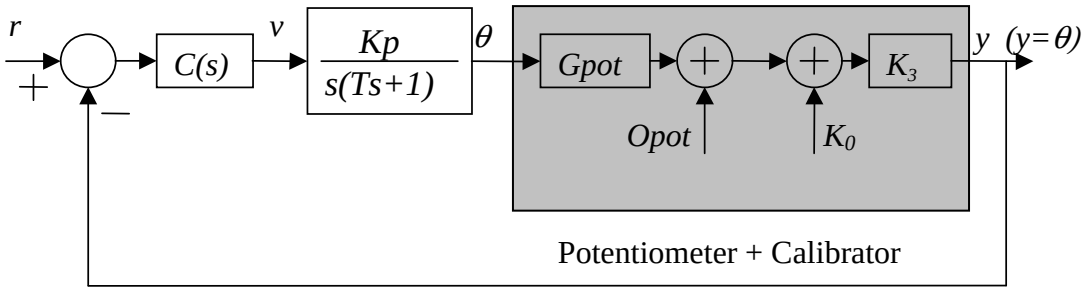

Figure 2.1 Simplified servomotor block diagram.

1- Let the controller $C(s)$ in the figure above be a positive gain: $C(s) = K$. Write down the closed-loop transfer function $G(s) = \frac{Y(s)}{R(s)}$ in terms of K, Kp, and T.

2- Since Kp and T are positive constants, $G(s)$ is a stable second-order system with no zeros. It can in fact be written in the form $G(s) = \frac{G_0 \omega_n^{\,2}}{s^2 + 2\delta\omega_n s + \omega_n^{\,2}}$ for positive constants G_0, δ, and ω_n. Determine the relationship between the triples (δ, ω_n, G_0) and (K, K_p, T).

3- Suppose that the damping ratio δ for the closed loop servomotor system is less than 1 (for a certain value of K). Given a step response, we know how to compute ω_n and δ from the maximum overshoot and peak time (refer if necessary to the document on transient response of second order systems). Combining with the results of (2), you can determine the values of K_p and T from the step response. Write down the formulas for Kp and T. You can also show the details of the step response on a plot.

4- Sketch a root locus of $G(s)$ as a function of $K>0$, for fixed, positive values of Kp, and T. What happens to the poles of the closed-loop system as K is increased from 0?

5- Read the Laboratory Procedure so that you know what to do before coming to the lab.

2.2 DC Motor Model Identification Laboratory Procedure:

Note: Please do not attempt to connect the ribbon cable from the bench-top interface board to the computer. If the ribbon cable is disconnected, please ask your TA to connect it.

1- Modify your *control.mdl* file such that the controller of the dc motor be a simple gain *K*.

2- Build and run your *control.mdl* simulink file to close the control loop. (You should have already calibrated the potentiometer and your K_0 and K_3 parameters must be valid in the workspace).

3- Experimentally determine a value of $K > 0$ (beginning from a small value such as 0.01) for which the step response has a percent overshoot of approximately 4*0%* and collect the data. Because each of the servomotors has a different amount of inertia, gain *K* will differ from each servomotor to another one. (Don't copy your neighbor's value!)

4- Plot the step response data collected in Step 4 into MATLAB, and determine ω_n and δ, by measuring peak-time and percentage-overshoot from your graph. Then determine the physical values of K_p, and *T*. Do a good job here, because your measurements will be used in subsequent labs, and your numbers are unique to your workstation. Save the collected data from your Simulink model in your workspace.

5- Plot the simulated response of *G*(*s*) for the identified values of ω_n and δ on the same axes that you have plotted the experimental data and keep the plot for your report. The two plots should be close to one another (see the figure below).

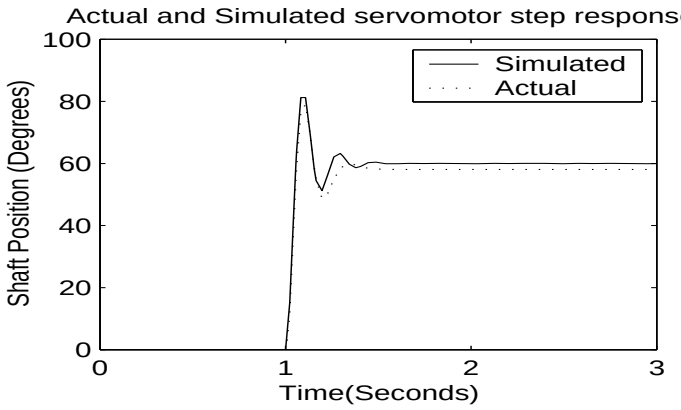

Figure 2.2 : Actual and simulated servomotor position step response

2.3 Questions:

1- Why use a value of *K* that gives overshoot of 40%? Can other values work? What if you use a value of *K* so that the closed-loop system is over-damped? What if you use a value that gives 100% overshoot?

2- What are some the reasons for discrepancy between the measured step response and the simulated one?. For example, there is static friction in the motor, which is not modeled by the second - order system. Why and how does static friction make the responses differ?

2.4 Report:

Please refer to the *Lab1 report format template* for your write-up.

CONTROL SYSTEMS LABORATORY

ECE410F-Fall 2003

LAB2: DC MOTOR CONTROLLER DESIGN

3. LAG COMPENSATOR DESIGN FOR DC MOTOR:

In this experiment, you will design a dynamic compensator for the servomotor system, using classical frequency-domain techniques. The controller design is to be based upon the model you obtained in previous session.

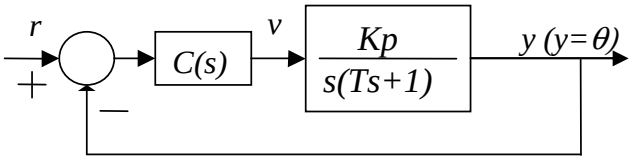

Figure 3.1 Simplified servomotor block diagram

3.1 Preparation:

The preparation involves designing the controller $C(s)$ so that the closed-loop system satisfies certain specifications, and then implementing it in discrete-time. Begin with your transfer function $G(s)$ from v in volts to θ in degrees. The main constraint concerns the plant input; so as not to saturate the A/D board, v must satisfy

$$|v(t)| \le 5 \text{ volts.} \qquad (3.1)$$

1. Begin with $C(s) = k$, a pure gain. By simulation, find the largest k so that spec (3.1) is satisfied when $r(t)$ is a step of 45 deg. For this k, record the gain crossover frequency and the phase margin.

2. Add to the controller a phase lag compensator, so it's now $C(s) = kC_1(s)$, where $C_1(s)$ achieves the specs:
   (a) phase margin about 45 deg.,
   (b) velocity error constant the same as that of $kG(s)$,

3. Let $r(t)$ be a unit step of 45 deg. Simulate the closed-loop system, plotting $y(t)$ and $v(t)$. Verify that (3.1) is satisfied.

4. Compute a discrete-time approximation $C_d(z)$ to $C(s)$. Assume the controller sampling rate is 1000 Hz.

5. Generate a Simulink model to implement your controller.

3.2 Controller Design Laboratory Procedure:

Note: Please do not attempt to connect the ribbon cable from the bench-top interface board to the computer. If the ribbon cable is disconnected, please ask your TA to connect it.

1. Compile and run your version of Simulink model. Use the same step input $r(t)$ as in preparation. Save the actual and simulated angle $\theta(t)$ and the reference $r(t)$ to a data file. Save the control signal $v(t)$ in both actual and simulated system. Verify that they satisfy (3.1).

2. Show the results to your TA.

3. Apply a triangle wave form with the frequency 0.25 Hz and the amplitude 45 degrees (± 45) to the closed loop system. To generate this wave form, you can use the block "Repeating Sequence" under "Sources" and enter [0 2 4] for time values and [-45 45 -45] for output values. Now you can observe the ramp response. Compare the ramp response of the closed loop system in two cases $C(s) = k$ and $C(s) = kC_1(s)$.

3.3 Questions:

1- Compare the step response of the system with a "gain" controller and with a "lag compensator".

2- Compare the ramp response of the system with a "gain" controller and with a "lag compensator".

3.4 Report:

Please refer to the *Lab2 report format template* for your write-up.

Appendix 1: WINCON

Wincon is a COM (Component Object Model) server which allows Simulink to operate as a real-time controller. It connects the Simulink environment to the kernel level of the Windows operating system and thence to the Data Acquisition hardware using Real-Time Workshop.
In the Simulink library, you can locate the "Wincon" Tool box, and if you browse through it, you can locate the QUANSER Tool box. Browse through the "Quanser Consulting MultiQ-PCI series" blocks (This is the type of the Data Acquisition board installed on your PC). You can drag and drop "Analog input" and "Analog output" blocks into your model to read from the A/D and to write to the D/A channels.
For example, a simple model such as:

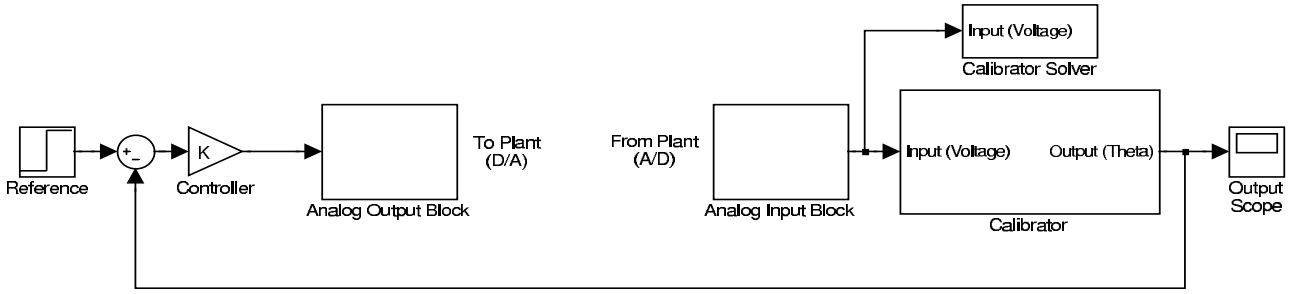

Figure A1.1 A sample control loop model in simulink which uses wincon to control a servomotor

can control your plant.
Note that Simulink has two modes of operations "Normal" and "External". If your model contains a physical I/O, you have to select the "External" mode to operate real-time experiments. To compile and run a model, one should select "Build" under "wincon" in the Simulink pull down menu. The model will then be compiled, and if there is no error, the compiler will then automatically download the compiled model to the real-time kernel. During the first download the wincon server will be automatically launched.
You can control your program through the wincon server. In particular, you can start/stop the controller, monitor variables, and plot signals. Note that when Simulink is in its "External" mode, some monitoring blocks such as "Display" and "Scope" and even "To Workspace" do not work properly; however, you can display and/or plot signals by selecting them from the wincon server list of variables. To collect and save data, you have to add one "Scope" block to where you wish to collect the data, and then use "save" under the "file" option of your "Scope". You can save information as an m-file, mat-file, or save to workspace. Here is the face plate of the wincon server.

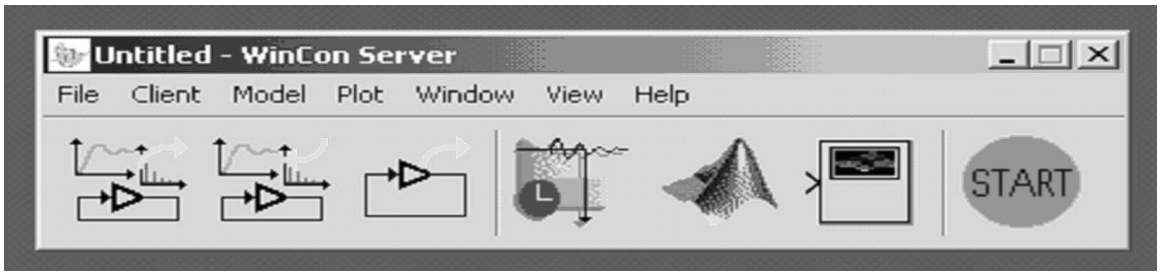

Figure A1.2: WinCon server