EE-371 CONTROL SYSTEMS LABORATORY

Session 7
Position Control Design Project
(PD controller – Root-locus Design)

Purpose

In this project you design and implement a position control system with a cascade PD controller using root-locus design. The objectives of this project are:

- To design a PD compensator using root-locus designs such that the output angle tracks a commanded position.
- To build the compensated servo plant in SIMULINK and simulate offline to obtain the response to a square wave input and verify the design.
- To build the WinCon application, and implement and test the system on the real-time hardware

Introduction

A common compensation technique is placing a cascade controller in the forward path. Typical compensators are phase-lead, phase-lag, and PID controller. In Lab 5 the

servomotor position control was achieved by means of position and rate feedback. In this project a PD controller is designed using root-locus method.

The root-locus trajectories show the location of the roots of the characteristic equation as one or more parameters are changed. If the root locations are not satisfactory, adding additional poles and zeros via a cascade compensator reshapes the root locus. The introduction of a compensator $G_c(s)$ results in the characteristic equation

$$1+G_c(s)G_p(s)H(s)=0$$

Writing $G_c(s)G_p(s)H(s)$ as $KG(s)$, we have

$$1+KG(s)=0$$

If s_1 is the desired location of a closed-loop pole, it must satisfy the above equation, which results in the following angle and magnitude criteria:

$$\sum\theta_{zi}-\sum\theta_{pi}=-180$$

$$K=\frac{\text{product of vectror lengths from finite poles}}{\text{product of vector lengths from finite zeros}}$$

The above equations can be used for graphical root-locus design.

1. Servo plant modeling

In the position control experiment (Lab 5) the motor-load transfer function with position as output was found to be

$$\frac{\Omega_o(s)}{V_i(s)}=\frac{\dfrac{\eta K_m K_g}{R_a J_{eq}}}{s\left(s+\dfrac{B_{eq}}{J_{eq}}+\dfrac{\eta K_m^2 K_g^2}{R_a J_{eq}}\right)} \tag{7.1}$$

or

$$\frac{\theta_o(s)}{V_i(s)}=\frac{a_m}{s(s+b_m)} \tag{7.2}$$

Where

$$a_m=\frac{\mu K_m K_g}{R_a J_{eq}},\qquad b_m=\frac{B_{eq}}{J_{eq}}+\frac{\mu K_m^2 K_g^2}{R_a J_{eq}} \tag{7.3}$$

The open-loop block diagram with the s-domain voltage $V_i(s)$ as input and $\theta_o(s)$ as output is shown in Figure 7.1.

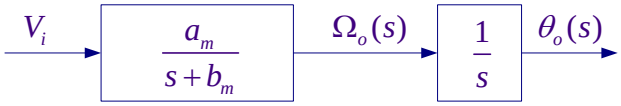

Figure 7.1 Open-loop plant transfer function.

The s-domain unit step response is

$$\theta_o(s) = \frac{a_m}{s(s+b_m)}\frac{1}{s}$$

The final value of the response is $\lim_{t\to\infty}\theta_o(t) = \lim_{s\to 0} s\theta_o(s) = \infty$. That is, the response is unbounded.

1.1 Position control

In order to control the output position to follow an input command, consider the addition of a PD controller $G_c(s) = K_P + K_D s$

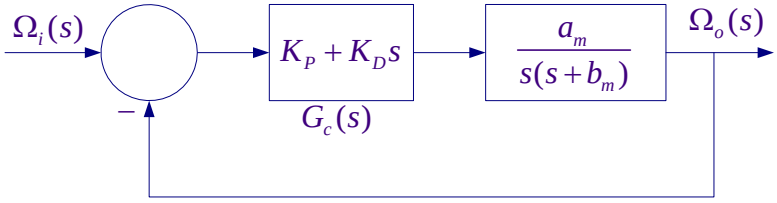

Figure 7.2 closed-loop control system with PD controller.

The PD controller is equivalent to the addition of a simple zero at $s = -\frac{K_P}{K_D}$ to the open-loop transfer function. This would shift the loci towards the left-half s-plane which improves the transient response .A problem with PD compensation is that the compensator gain continues to increase with frequency. If high-frequency noise is present in the system, the PD compensator will amplify this noise and a phase-lead controller may be used in place of the PD controller.

2. Pre Laboratory Assignment

Evaluate a_m, and b_m of the plant transfer function as given by (7.3). The servo plant parameters are given in the position control experiments (Lab 5). If you have performed Lab 5 you have these values. If you have not performed the position control project (Lab 5) you must derive the plant transfer function, verify the above equations and include modeling with this project. Otherwise give reference to Lab 5.

Assuming a simple proportional controller K, draw the root locus to scale for

$$KG(s) = \frac{a_m K}{s(s+b_m)}$$

Give all pertinent characteristics of the loci such as number of asymptotes, breakaway and/or re-entry points. Indicate the loci directions for increasing *K* by arrows. Specify range of *K* for closed loop system stability.

2.1 Position control design

Using root-locus graphical method design a PD controller to meet the following time-domain specifications:

- Step response dominant poles damping ratio $\zeta = 0.707$
- Step response dominant poles time constant $\tau = 0.02$ sec

Since the characteristic equation is a second order, K_P and K_D can be found by equating the coefficients of the characteristic equation with the standard second-order characteristic equation. However, to reinforce the root-locus design method for higher order systems, you must design the PD controller using root-locus design technique (see Example 2 in Tutorial III Root Locus Design). Check your design using **rldesigngui** program.

The SIMULINK simulation diagram named "Lab7_Sim.mdl" is constructed as shown in Figure 7.3.

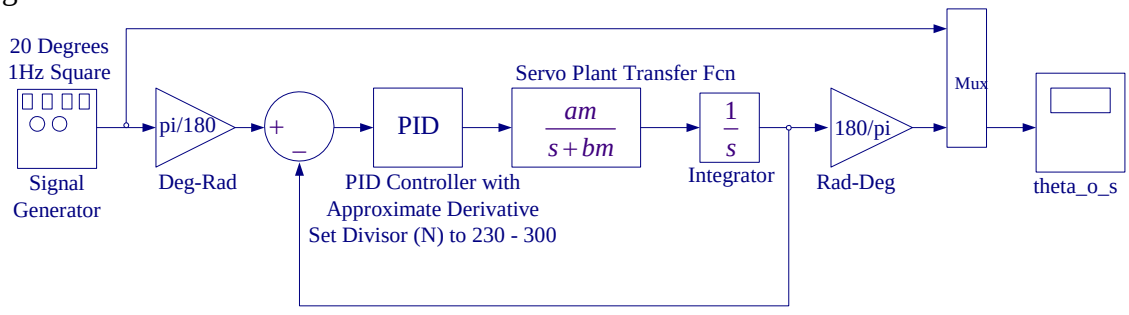

Figure 7.3 Simulink diagram for the servo plant position control

Use a Signal Generator with Amplitude 20 degrees and Frequency 1 Hz. Get the PID Controller with **Approximate Derivative** for the PD controller. This is found in Simulink Extras /Additional Linear library. In the Simulink block diagram you can replace am, bm, KP, and KD with their values and set ***KI* to zero**, and the **Derivative Divisor (N) to 300**. From the Simulink/Simulation Parameters select the Solver page and for Solver option Type, select Fixed-step and ode4 (Runge-Kutta) and set the fixed step size to 0.001. Actually a Variable-step selection with auto or a smaller Max step size would produce more accurate results, but because in the implementation diagram we are using a uniform sampling rate of 0.001 second, we use the same fixed-step size of 0.001 second for integration.

Simulate and use 'plotscope' function to capture the scope plot and produce a Figure plot. To do this, type **plotscope** at the MATLAB prompt, then click on the Scope Figure

(outside the plot area) and hit return you will have a Figure print. You can add label and legend commands or edit the graph, label as Figure 7.4.

All the prelab calculations, design and simulation must be completed prior to the laboratory session. The plants transfer function and the controller values must be checked and verified by your instructor. Also complete the implementation diagram as outlined in section 3.1 and 3.2.

3. Laboratory Procedure

When you have finished testing your model in SIMULINK, it has to be prepared for implementation on the real-time hardware. This means the plant model has to be replaced by the I/O components that form the interfaces to the real plant.

3.1 Creating the Subsystem Block for the Interface to SRV02

The Encoder_Tach.mdl subsystem, which was created in the position control experiment (Lab 5), is shown in Figure 7.5(a). If you if you don't have this subsystem block follow the procedure in Section 3.1 Lab 5 to configure the required interfaces to SRV02.

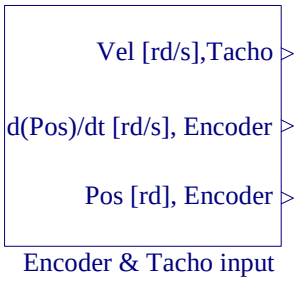

Figure 7.5(a) Interface to the SRV02 Feedback Signals.

If you double-click on the above subsystem it will display the underlying system as shown in Figure 7.5(b).

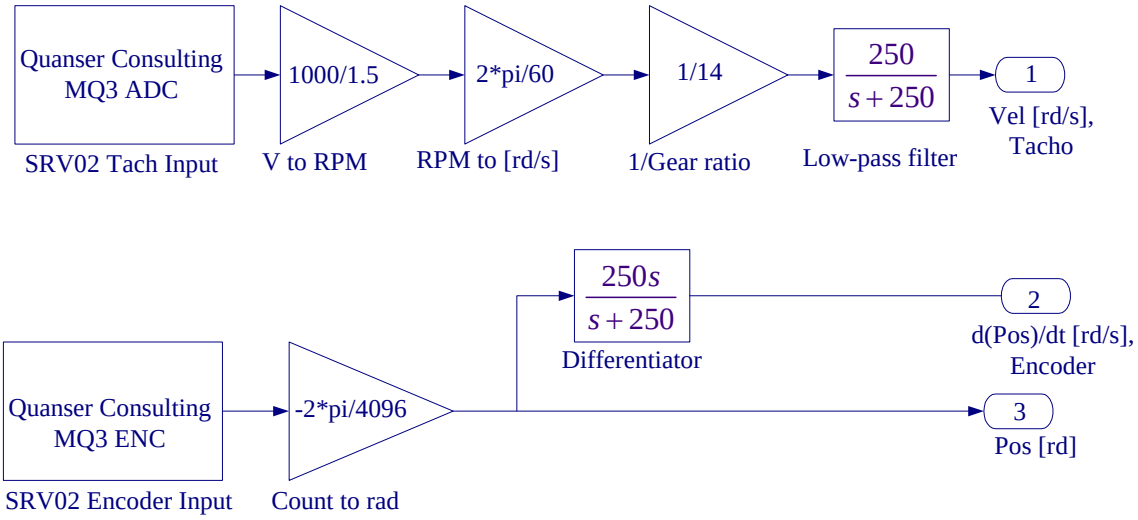

Figure 7.5(b) Content of the subsystem Encoder-Tach.

3.2 Creating the Implementation model

The implementation model can be added on the previously constructed SIMULINK model. This would enable you to obtain the simulation and actual results simultaneously. Open Lab7_Sim.mdl (your simulation model), save it under a new name (say Lab7_Imp.mdl). Remove the MUX block.

Start constructing the implementation diagram below the simulation diagram. Copy the Encoder_tach.mdl (constructed in part 3.1) to the clipboard and paste it on your implementation model as shown in Figure 7.6. Get the Analog Output block from the Quanser MultiQ3 and set the Channel Use to 0 as is in the wiring diagram. Use a Signal Generator with Amplitude 20 degree and Frequency 1 Hz. Add the PID controller with **Approximate Derivative** to the signal coming from the Motor Encoder, complete the feedback loops and connect the resulting signal to the Quanser Analog output.
Place as many Scopes as you like to monitor the phase angle, velocity etc. Your completed model should be the same as shown in Figure 7.6. Set **KI to zero**, the **Derivative Divisor (N) to 300,** and gains KP and KD to the values found in part (2.1), or run the m-file that returns the values of KP and KD. The gains in Simulink Simulation diagram are renamed to KP1 and KD1, so that if the value of the variables KP and KD are changed at the MATLAB prompt for fine tuning, the values in the Simulation diagram are not changed. You may save the augmented model under the file name say, “Lab7_Imp.mdl”.

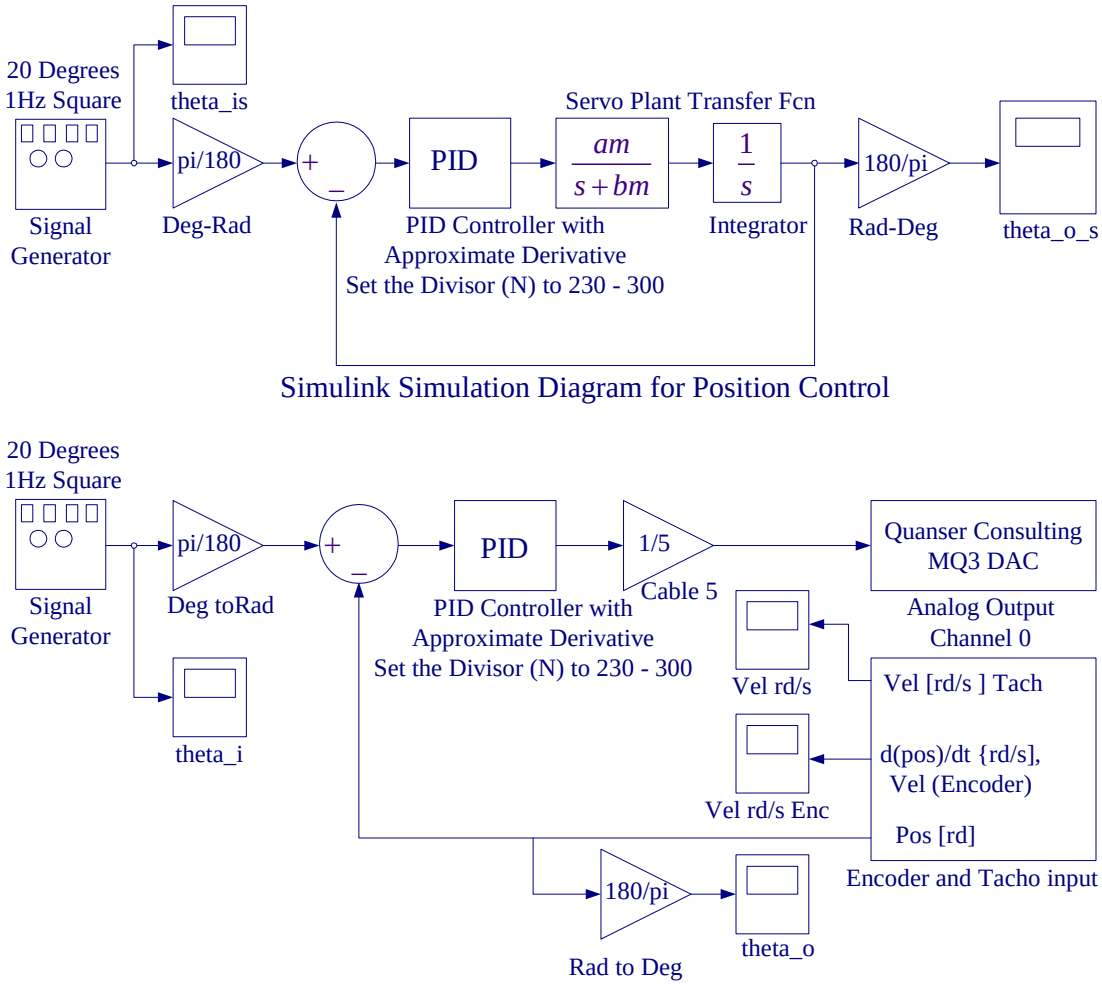

Figure 7.6 SIMULINK Simulation and Implementation diagram

3.3 Wiring diagram

Using the set of leads, universal power module (UPM), SRV02 Servomotor, and the connecting board of the MultiQ3 data acquisition board complete the wiring diagram shown in Figure 7.7 as follows:

From	To	Cable
Encoder on SRV02	MultiQ/Encoder 0	5 pin Din to 5 pin Din
Motor on SRV02	UPM/To Load	6 pin to 4 pin Din, **Gain 5 Cable**
D/A #0 on MultiQ	UPM – From D/A	RCA to 5 pin Din
A/D # 0, 1, 2, 3, on MultiQ	UPM- TO A/D	5 pin Din to 4xRCA

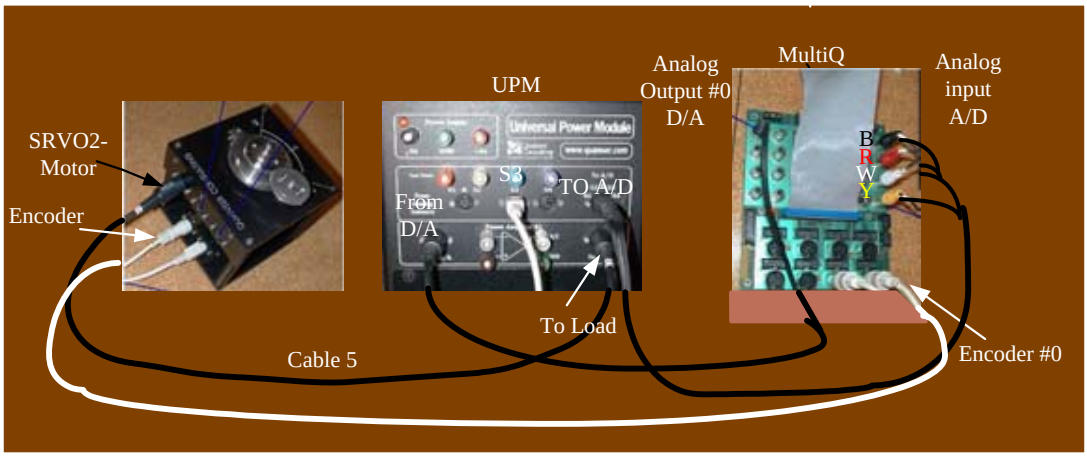

Figure 7.7 Wiring diagram for servo motor position control system.

Before proceeding to the next part request the instructor to check your electrical connections and the implementation diagram.

3.4 Compiling the model

In order to run the implementation model in real-time, you must first build the code for it. Turn on the UPM. Start WinCon, Click on the MATLAB icon in WinCon server. This launches MATLAB. In the Command menu set the Current Directory to the path where your model Lab7_Imp.mdl is. Before building the model, you must set the simulation parameters. Pull down the Simulation dialog box and select Parameters. Set the Start time to 0, the Stop time to 5, for Solver Option use Fixed-step and ode4 (Runge-Kutta) method set the Fixed-step size, i.e., the sampling rate to 0.001. In the Simulation drop down menu set the model to **External**. Make sure all the controller gains are set. Start the WinCon **Server** on your laptop and then use **Client Connect**, in the dialog box type the proper Client workstation IP address. Generate the real-time code corresponding to your diagram by selecting the "**Build**" option of the WinCon menu from the Simulink window. The MATLAB window displays the progress of the code generation task. Wait until the compilation is complete. The following message then appears: "*### Successful completion of Real-Time Workshop builds procedure for model: Lab7_Imp*".

3.5 Running the code

Following the code generation, WinCon Server and WinCon Client are automatically started. The generated code is automatically downloaded to the Client and the system is ready to run. To start the controller to run in real-time, click on the **Start** icon from the WinCon Server window shown in Figure 7.8. It will turn red and display STOP. Clicking on the **Stop** icon will stop the real-time code and return to the green button.

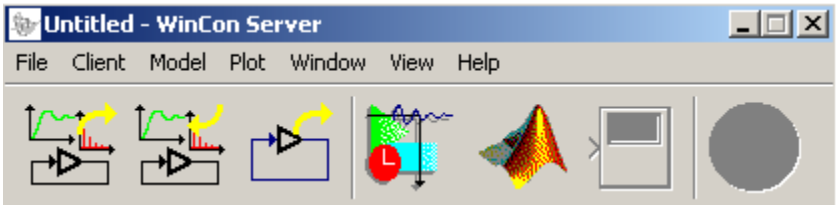

Figure 7.8 WinCon Server

If you hear a whining or buzzing in the motor you are feeding high frequency noise to the motor or motor is subjected to excessive voltage, immediately stop the motor. Ask the instructor to check the implementation diagram and the compensator gains before proceeding again.

You can also change the controller gains on the fly (i.e. while the controller is running in real-time). To do so, double click on the Simulink Implementation gain block, change to the desired value, and select Apply, or OK. Note the changes in the real-time plots. Increasing KP will increase the overshoot, but it would reduce the offset.

3.6 Plotting Data

You can now plot in real-time any variables (e.g. angles, velocities) of your diagram by clicking on the "**Plot/New/Scope**" button in the WinCon Server window and selecting the variable you wish to visualize. Select "theta_o" and click OK. This opens one real-time plot. To plot more variables in that same window, click on "**File/Variables…**" from the Scope window menu. The names of all blocks in the Simulink model diagram appear in a Multiple. Select Variable Tree. You can then select the variable(s) you want to plot. In this case, select, for example, "theta_i" and "theta_o_s. In the Scope pull-down menu, using Freeze you can freeze the plot, and Update/Buffer can be used to change the final time to display fewer cycles. From the File menu you can Save and Print the graph. Choose Save As M-File, and save the plot as M-file (say Figure7_9.m). Now at the MATLAB prompt type the file name to obtain the MATLAB Figure plot. You can type "grid" to place a grid on the graph or edit the Figure as you wish.

Zoom in to obtain the comparison between the simulated response and the actual response.

Use File Save, this saves the compiled controller including all plots as a **.wpc** (WinCon project) file. In case you want to run the experiment again, from WinCon Server use File/Open to reload this **.wcp** file, and run the project in real time independent of MATLAB/Simulink.

To prevent excessive wear to the motor and gearbox run the experiment for a short time.

4. Project Report

Discuss the assumption and approximations made in the modeling the servomotor. In the report show the closed-loop system block diagram and your analysis. Comment on your results; how does experimental response compare to simulated response? Discuss the reason for any deviation in the actual transient response and the simulated response. What is the system type, and what is the theoretical steady-state error? Estimate the actual steady-state error if any, and discuss the reason for the steady-state error. A brief discussion of frictional forces and the amplifier saturation is included in the Appendix at the end of Lab 5, which you may find helpful for preparing the discussion of results in your project report. After the completion of this lab you should be confident in tuning this type of controller to achieve a desired response.