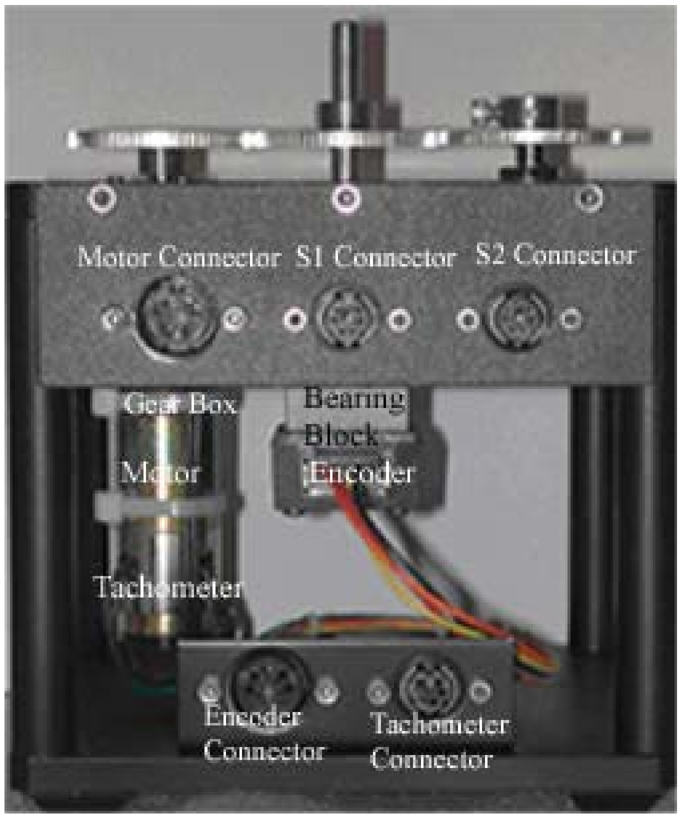
EE-371 CONTROL SYSTEMS LABORATORY

Session 6
Speed control Design Project
(Velocity Feedbacks)

Purpose

In this project you design and implement a speed control system for low frequency square wave input. The objectives of this project are:

- To design a feedback controller that regulates the speed of the output shaft and reduces the closed-loop steady-state error to zero.
- 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

Electric motor-driven servomechanism for position and speed control are used in many areas from low power applications to heavy-duty servos on steel rolling mills. When speed rather than position is the variable to be controlled the output is shaft speed instead of position. Accordingly, feedback sensor must be used to measure speed. This could be a tachometer providing a voltage proportional to speed.

1. Servo plant modeling

In the position control experiment the motor-load transfer function with speed as output found in (5.7) is

$$\frac{\Omega_o(s)}{V_i(s)} = \frac{\dfrac{\eta K_m K_g}{R_a J_{eq}}}{s + \dfrac{B_{eq}}{J_{eq}} + \dfrac{\eta K_m^2 K_g^2}{R_a J_{eq}}} \tag{6.1}$$

or

$$\frac{\Omega_o(s)}{V_i(s)} = \frac{a_m}{s + b_m} \tag{6.2}$$

Where

$$a_m = \frac{\eta K_m K_g}{R_a J_{eq}}, \qquad b_m = \frac{B_{eq}}{J_{eq}} + \frac{\eta K_m^2 K_g^2}{R_a J_{eq}} \tag{6.3}$$

The open-loop block diagram with the s-domain speed $\Omega_i(s)$ as input and simple gain controller K_o is shown in Figure 6.1.

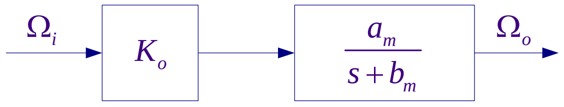


Figure 6.1 Open-loop plant transfer function.

For the open-loop system, the gain K_o can be made equal to $\dfrac{b_m}{a_m}$, so that with no load torque and disturbance the final value of the step response is equal to the input. The gain must be made equal to $K_o = \dfrac{R_a B_{eq}}{\eta K_m K_g} + K_m K_g$. This means that the open-loop gain is very sensitive to the plant parameters. We may not have accurate values and a change in the operating condition or temperature changes will cause the plant gain a_m / b_m to drift from it nominal value. Furthermore the system is very sensitive to disturbance and with constant K_o speed would change greatly with load. A suitable closed-loop system would reduce or eliminate some of these problems. A simple closed-loop with proportional controller is shown in Figure 6.2.

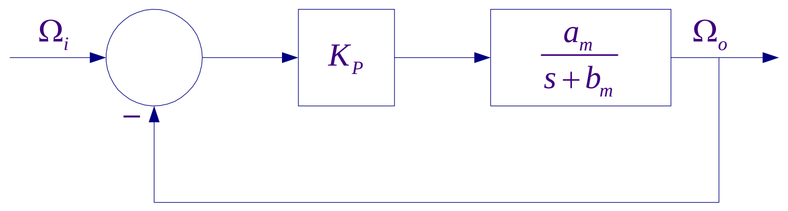


Figure 6.2: closed-loop control system with proportional controller.

The closed-loop transfer function is

$$\frac{K_P a_m}{s+b_m+K_P a_m} \tag{6.4}$$

For large K_P, the closed-loop time constant is reduced and the response final value is not very sensitive to plant gain. The system is type zero and the closed-loop introduces a steady-state error given by

$$e_{ss}=1-\frac{K_P a_m}{b_m+K_P a_m}=\frac{1}{1+\frac{K_P a_m}{b_m}}=\frac{1}{1+K_p} \tag{6.5}$$

Where $K_p=\frac{K_P a_m}{b_m}$ (K_p with lower case subscript is known as position error constant)

2. Pre Laboratory Assignment

Evaluate a_m, and b_m of the plant transfer function as given by (6.3). 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.

For the given parameters and $K_P=1$, what is the steady-state error in degrees/s for a step input of 428 degree/s.

The steady-state error is $e_{ss}=$________________ .

The velocity control can be improved by introducing an integrator in order to increase the system to type one and reduce the steady-state error to zero. This is shown in Figure 6.3.

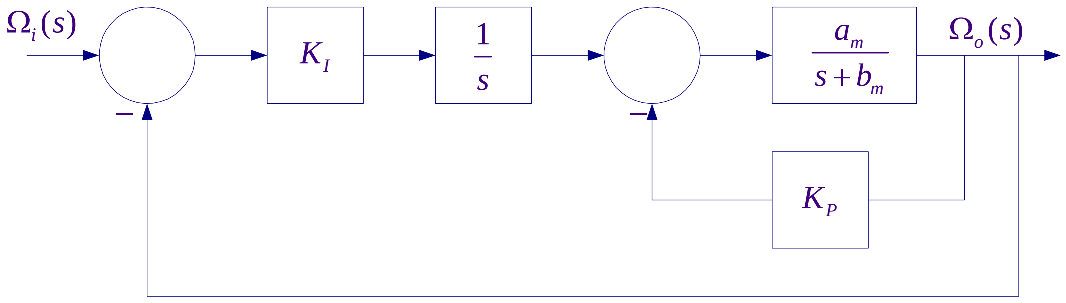


Figure 6.3 closed-loop control system with proportional and integral controller.

Applying the Mason's gain formula, the overall transfer function becomes

$$\frac{\Omega_o(s)}{\Omega_i(s)} = \frac{K_I a_m}{s^2 + \left(b_m + K_P a_m\right)s + K_I a_m} \tag{6.6}$$

2.1 Speed control design

The essential requirement in a speed control is to design a feedback controller that regulates the speed of the output shaft and reduces the steady-state error to zero. For a discussion of assumptions, approximations made in modeling the servomechanism, and the sensors refer to the position control experiment.

2.2 Time-domain specifications

Let $\omega_i(t)$ be a square wave of amplitude 1000 RPM and a frequency of 0.5 Hz. Design a control system and determine the gains K_P and K_I such that the following time-domain specifications are met:

- Step response damping ratio of $\zeta = 0.707$
- Peak time of $t_p = 0.0875$ second.

The second-order response peak time t_p, is given by

$$t_p = \frac{\pi}{\omega_n \sqrt{1-\zeta^2}} \tag{6.7}$$

and the theoretical peak value of the step response is

$$M_{pt} = \left(1 + e^{-\zeta\pi / \sqrt{1-\zeta^2}}\right)\omega_{i(\text{Amplitude})} \tag{6.8}$$

For $\zeta = 0.707$, we find ω_n.

The servo motor transfer function has the same form as the standard second-order transfer function

$$\frac{\Omega_o(s)}{\Omega_i(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{6.9}$$

Comparing the plant characteristic equation given in (6.6) with the standard second-order characteristic equation in (6.9), find two equations for K_I and K_P in terms of a_m, b_m, ζ, and ω_n. Substitute for the parameters and obtain the values of K_I, and K_P for the above design specifications

Construct the SIMULINK simulation diagram as shown in Figure 6.4 and save it as "Lab6_Sim.mdl".

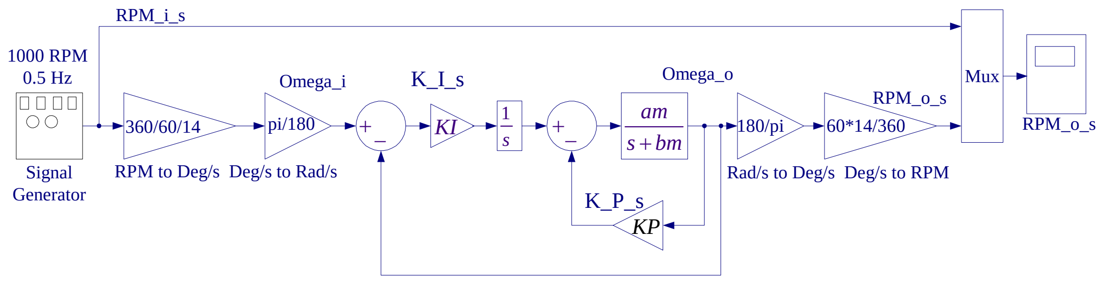


Figure 6.4: Simulink diagram for the servo plant speed control

Use a signal generator with amplitude 1000 RPM, and frequency 0.5 Hz. In the Simulink block diagram you can replace $am, bm, KP,$ and KI with their values. Alternatively, you can place equations (6.3), (6.10), and (6.11) in a script m-file to compute $am, bm, KP,$ and KI which is sent to the MATLAB Workspace, then simulate the Simulink diagram. 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 6.5. Check to see if the response meets the design requirements within the simulation numerical integration accuracy.

All the pre-lab 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 6.6(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.

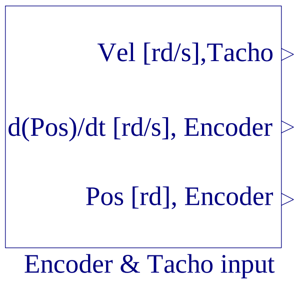


Figure 6.6(a) Interface to the SRV02 Feedback Signals.

If you double-click on the above subsystem it will display the underlying system as shown in Figure 6.6 (b).

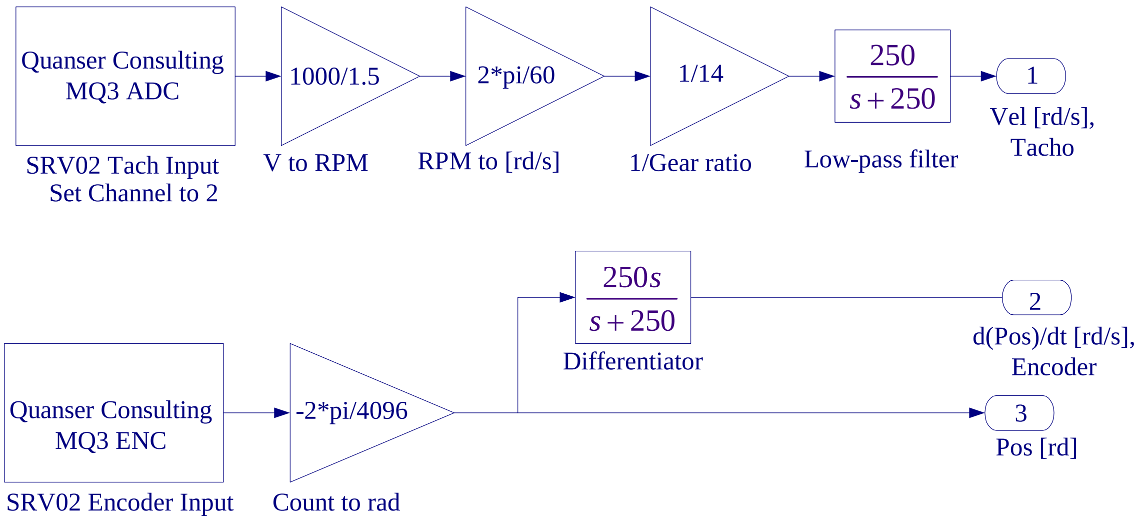


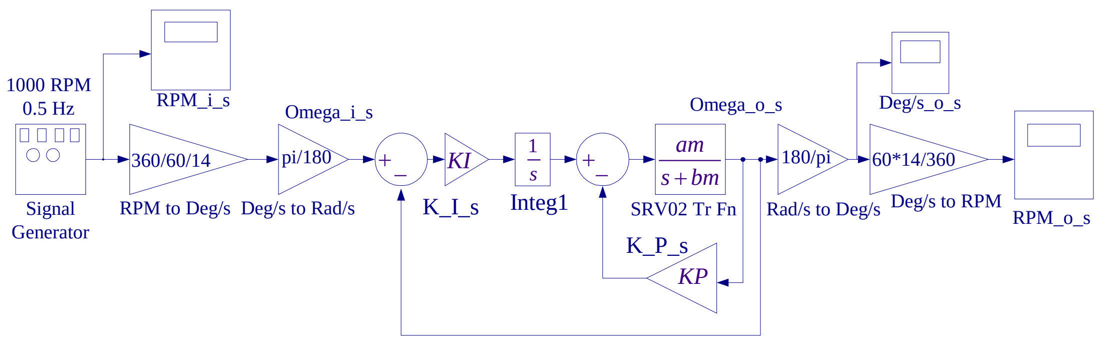
Figure 6.6(b) Content of the subsystem Encoder-Tach.

3.2 Creating the Implementation model

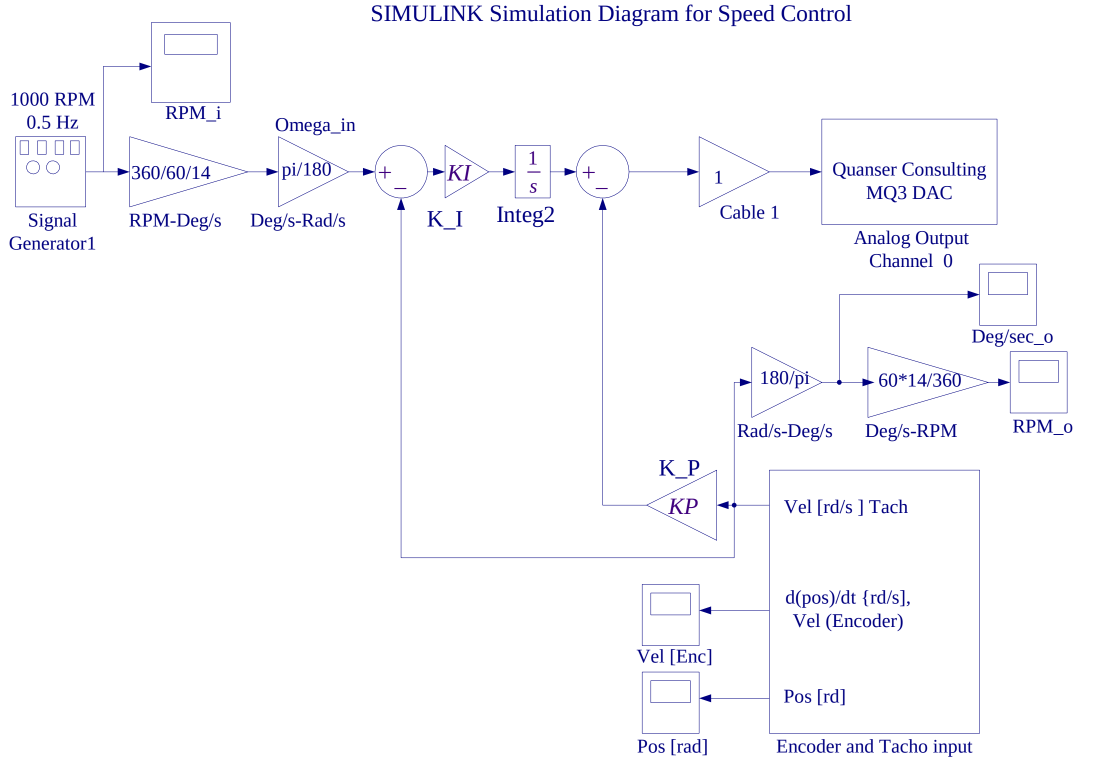
The implementation model can be added on the previously constructed SIMULINK model. This would enable you to obtain the simulation and actual results simultaneously.

Open Lab6_Sim.mdl (your simulation model), save it under a new name (say Lab6_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 6.7. Get the Analog Output block from the Quanser MultiQ and set the Channel Use to 0 as is in the wiring diagram. Use a Signal Generator with Amplitude 1000 RPM, and Frequency 0.5 Hz. Add the position and velocity feedback gains, complete the feedback loops and connect the resulting signal to the Quanser Analog output. Place as many Scopes as you like to monitor the RPM, velocity etc. Your completed model should be the same as shown in Figure 6.7. Set the gains KP and KI to the values found in part (2.2), or run the m-file that returns the values of KP and KI. The gains in Simulink Simulation diagram are renamed to KP1 and KI1, so that if the value of the variables KP and KI 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, “Lab6_Imp.mdl”. In this project we are using the tachometer signal (alternatively one can use the derivative of the encoder signal). A low-pass filter of bandwidth 250 Hz is used in to filter the tachometer signal. The tachometer signal is inherently noisy, you would probably get a cleaner response if a 100 Hz or a 50 Hz low-pass filter is used instead of the 250 Hz low pass filter. Try it.



SIMULINK Simulation Diagram for Speed Control



SIMULINK Implementation Diagram for Speed Control

Figure 6.7 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 6.8 as follows:

From	**To**	**Cable**
Tach on SRV02	UPM/S3	6 pin Mini Din to 6 pin mini Din
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 1 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

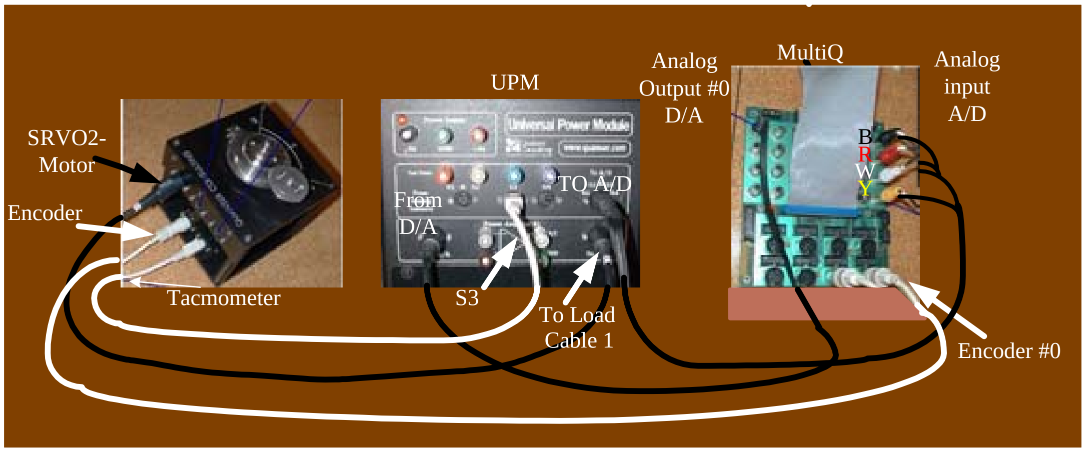


Figure 6.8 Wiring diagram for servo motor speed control system.

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 Lab6_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: Lab6_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 6.9. It will turn red and display STOP. Clicking on the **Stop** icon will stop the real-time code and return to the green button.

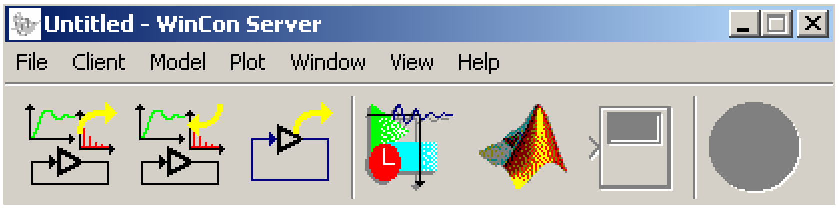


Figure 6.9 WinCon Server

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.

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 "RPM_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, "RPM_i" and "RPM_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 Figure6_10.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 assumptions and approximations made in the modeling the servomotor. Describe the effect of adding the velocity feedback and describe your design. In the report show the closed-loop system block diagram and your analysis, Comment on your results; how do experimental response compare to simulated response? Zoom in and estimate the peak value and the peak time for the experimental response and compare with the simulated value. Discuss the reason for any deviation in the actual transient response. What are the system type, and the theoretical steady-state error? Estimate the actual steady-state error if any, and discuss the reason for the steady-state error.