QUANSER
INNOVATE·EDUCATE

Lab Procedure
Lead Compensator

Introduction

Ensure the following:

1. You have reviewed the Application Guide – Lead Compensator.
2. The Qube-Servo 3 has been previously tested, is ON and connected to the PC.
3. Inertia disc load is attached to the Qube-Servo 3.
4. Launch MATLAB and browse to the working directory that includes the Simulink models for this lab.

This experiment is about designing a lead compensator to control the speed of the DC motor on the Qube-Servo 3 base. Recall that the transfer function from the input voltage to the speed of the Qube-Servo 3 is given by

$$P(s) = \frac{K}{\tau s + 1}.$$

The classic lead compensator given above may not achieve zero steady-state error. To ensure we have zero steady-state error, it is necessary to include an integrator in the controller and so we will use a controller of the form of Figure 1:

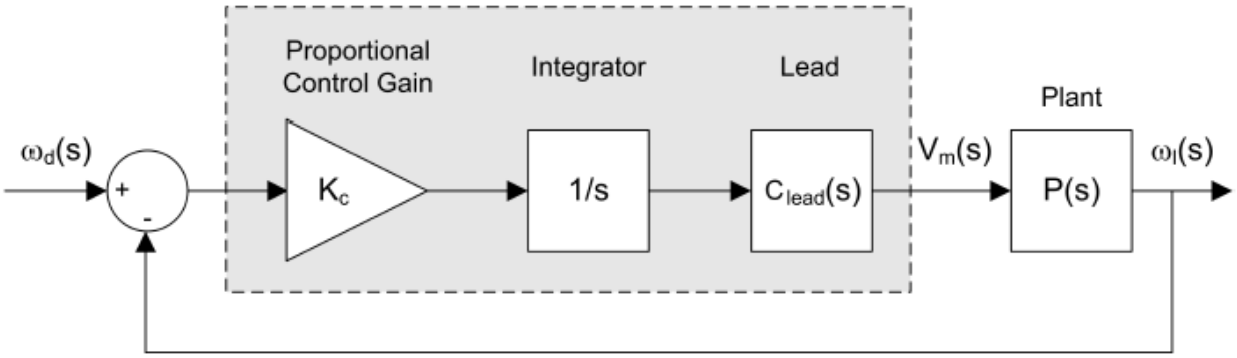

Figure 1: Lead compensator with integrator

For convenience, we will assume that the integrator is part of the plant model that is,

$$P_i(s) = P(s)\frac{1}{s} = \frac{K}{(\tau s + 1)s}.$$

The transfer function of a Lead compensator is given by

$$C(s)_{\text{lead}} = K_c \frac{1 + \alpha T s}{1 + T s}$$

where $\alpha > 1$. The lead compensator should fulfill the following design specifications:

$$e_{ss} = 0,$$

$$t_p = 0.05\text{s},$$

$$PO \leq 7.5\%,$$

$$PM \geq 75^0,$$

$$\omega_m \geq 45.0 \text{ rad/s},$$

where e_{ss} is the steady-state error, t_p is the peak time, PO is the percentage overshoot, PM is the phase margin, and ω_m is the system bandwidth. Since the process of designing a lead compensator is an iterative one, we will perform hand calculations to find the first iteration of the lead compensator and a MATLAB script will be developed alongside the hand calculations so that multiple iterations can be performed quickly.

Lead Compensator Design

1. Find the magnitude of the frequency response of the plant-integrator transfer function, ($|P_i(s)|$), in terms of the frequency ω.

2. The system has a gain of 1 (or 0 dB) at the crossover frequency ω_g. Find the expression for the crossover frequency in terms of the model parameters K and τ for $P_i(s)$ and calculate the value of the crossover frequency. In the MATLAB script lead_design.m, complete Section 1 and 2 of the script.

   Hint: Use the parameters found in the Step Response Modeling Lab or use the default model parameters $K = 24.3$ and $\tau = 0.11$.

3. Run the command `margin(Pi)` in the MATLAB command prompt to generate the bode plot of $P_i(s)$ and provide a screenshot of the plot. Validate that the crossover frequency calculated in Step 2 is accurate.

4. Using the Bode plot for $P_i(s)$, find the proportional gain K_c necessary such that $K_c P_i(s)$ has a crossover frequency of 20 rad/s and generate the Bode plot of $K_c P_i(s)$ and take a screenshot of the plot. The gain will need to be converted from decibels (dB) to absolute units. Complete Section 3 of the MATLAB script, lead_design.m.

   Hint: To find the K_c that corresponds to the desired crossover frequency use the command `inv(abs(freqresp(Pi, cross_freq_des)))` in MATLAB. Also note that 20 rad/s was picked as the initial guess for crossover frequency since it is less than half of the desired closed-loop bandwidth (this is a general guideline).

5. Calculate how much phase lead ϕ_m, needs to be added to the system $K_c P_i(s)$ by the lead compensator.

Hint: When trying to increase the phase margin, it is useful to add $10°$ to $15°$ to the design requirements (i.e. add $10°$ to PM_{des}). This is because adding phase will increase the crossover frequency, which will reduce the predicted increase in phase margin.

6. Calculate α.

7. Determine the new crossover frequency, ω_m. Complete Section 4 of the MATLAB script.

   Hint: For the MATLAB script, the function `allmargin()` will return the information from the bode plot in a struct format.

8. Does the crossover frequency meet the design requirement? If not, mention what can be done to solve this problem.

9. Derive the transfer function of the lead compensator. Start by evaluating T.

10. Calculate the pole and zero locations of the lead compensator. Generate the Bode plot of only the lead compensator transfer function using `margin()`. Validate that the desired phase margin occurs at the target frequency. Take a screenshot of the plot. Complete Section 5 of the MATLAB script, lead_design.m.

11. Generate a Bode plot of fully compensated loop $K_c C_L(s) P_i(s)$. Take a screenshot of the Bode plot. Are the system requirements met? Notice how the phase margin has not increased to PM_{des}.

12. How does modifying the variables `cross_freq_des` and `PM_des` within the MATLAB script lead_design.m, affect the system requirements?

13. If the system requirements were not met in the previous steps, adjust the variables `cross_freq_des` and `PM_des` until the system requirements are met.

14. Simulate the step response of the completed system loop using Section 6 of the MATLAB script, lead_design.m. Take a screenshot of the plot and the performance metrics. Are the system requirements met?
    Note: It is okay if the system requirements are not met at this stage, they will be adjusted in the implementation stage.

15. Use Section 7 to generate a Bode diagram showing both the uncompensated and compensated system plots overlayed. Take a screenshot of the response. Comment on how the phase margin and system bandwidth have changed from the uncompensated Bode plot.

Lead Compensator Implementation

The **Hardware Interfacing** and **Filtering** labs explained the basic blocks to read and write from the Qube-Servo 3. For simplicity, all labs forward will use a Qube-Servo 3 block that sets up the system beforehand and outputs the available information from the Qube.

The Simulink model shown in Figure 2 implements the lead compensator designed in the previous section for testing on the physical Qube-Servo 3. You will observe that Figure 2 implements the scheme shown in Figure 1.

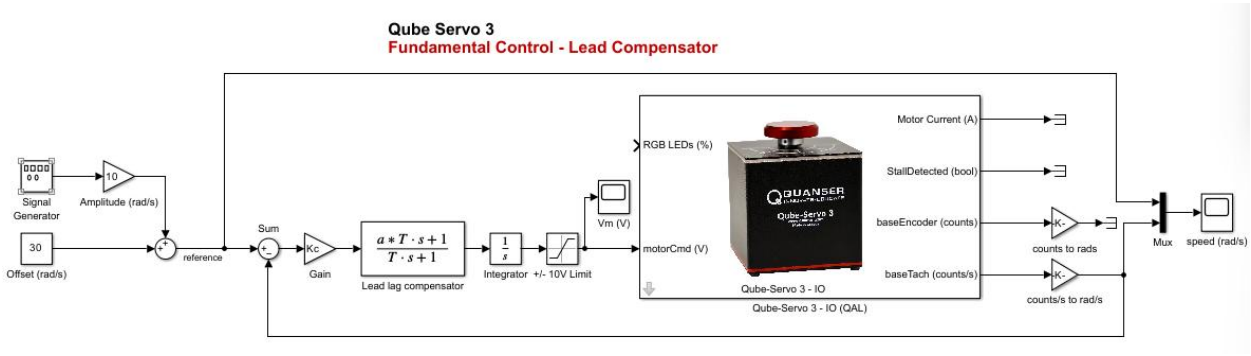

Figure 2: Simulink implementation of lead compensator for Qube-Servo 3.

16. Open the qs3_lead.slx Simulink model, design the Simulink model to be like Figure 2 by inserting and connecting the relevant blocks as in Figure 2.

17. Make sure your gains to convert counts/s to rad/sec is set properly based on previous labs.

18. Ensure that the variables K_c, a, and T have been loaded into the MATLAB workspace based on your lead compensator design.

19. Click `Monitor & Tune` in the `Hardware` or `QUARC` tab to deploy and run the model.

20. Take a screenshot of `Speed (rad/s)` and `Vm (V)` scopes. A sample response using $K_c = 1$ is shown in 3 below:

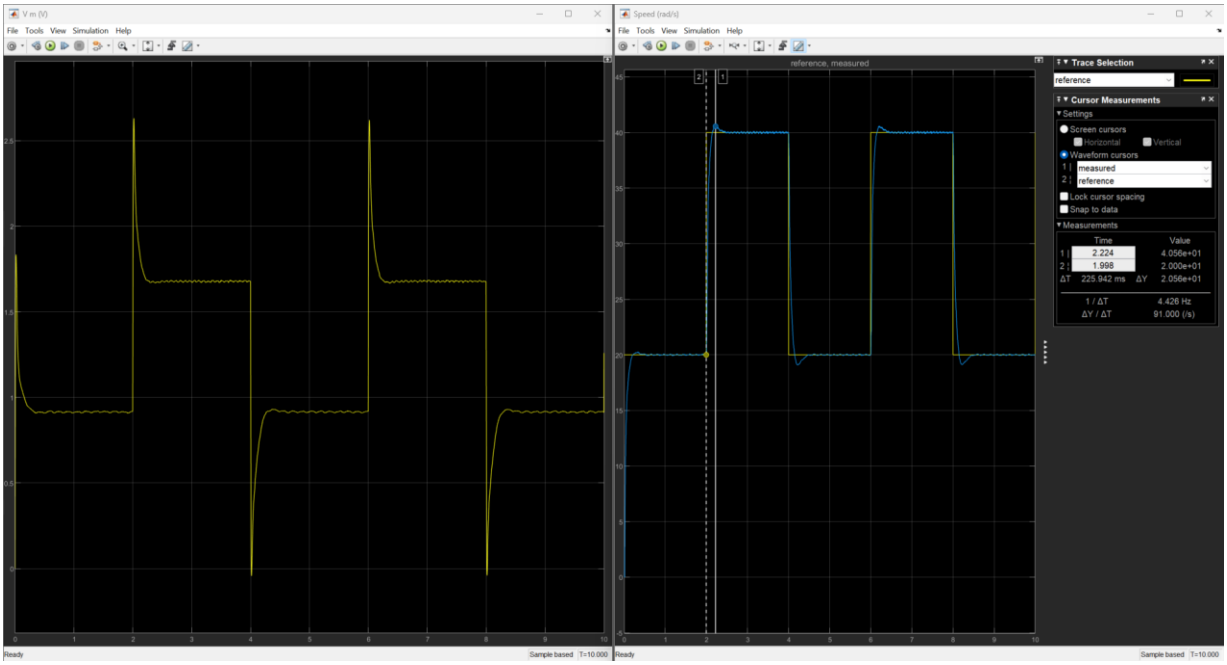

Figure 3: Sample response of the lead compensator using Kc = 1.

21. Use the measurement tool as shown in figure 3 to measure the percentage overshoot and peak time of the response. Does the response match the predicted step response from the previous section?

22. Does the response meet the requirements for the system? If it does not, adjust K_c until the requirements are met.

    Note: This can be done by overwriting the K_c `Gain` block with a new value.

23. Ensure the Simulink model has been stopped.

24. Turn OFF the Qube-Servo 3.