FRTN55 Automatic Control, Advanced Course

Laboratory Session 1

Loop Shaping for a Flexible Linear Servo[1]

Department of Automatic Control
Lund University

1. Introduction

The purpose of this laboratory is to design a controller for a flexible linear servo using frequency-domain methods. We will use the technique of *loop shaping*, where we shape the Bode diagram of the open-loop system using different compensators. The final aim is to fulfill all of the requirements on the closed-loop system.

Most of the work for this lab will be done at home. This is possible since the process used in this lab can be effectively simulated in software. In the physical lab session you will only fine-tune the controller for the physical process and discuss the underlying theory with the lab supervisor. Of course you can find support on the forum in Piazza and at the exercise sessions.

1.1 Before the physical laboratory

Before the physical lab session you have to:

- Read about Bode diagrams and lead/lag compensation: lecture 11 from the basic control course or chapter 11 *Feedback Systems: An Introduction for Scientists and Engineers*[2].
- Go through the laboratory prerequisites below.
- Do the assignments marked as **HOME ASSIGNMENT** in this manual. Those include large part of the work of this laboratory. *With those assignments you will already achieve a good control of the process basically fulfilling all of the specifications.*

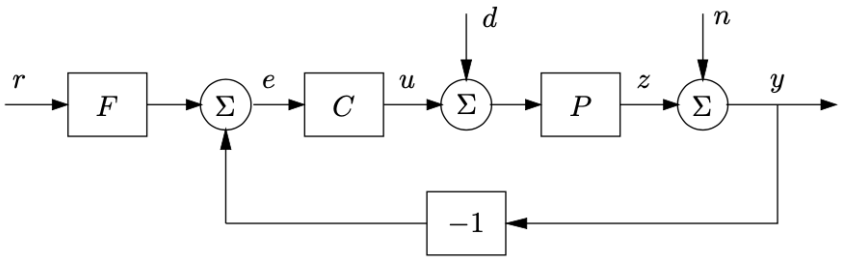
[1] Written by Tomas Olsson, updated by Claudio Mandrioli in 2020 for COVID pandemic, latest update August 31, 2022.

[2] Available at: `http://www.cds.caltech.edu/~murray/amwiki/index.php?title=Main_Page`

Laboratory prerequisites

Prepare for the lab by doing the following exercises, and make sure you are familiar with the related concepts. **NOTE**: those are exercises that should help you getting ready for the lab. Before the physical session you have also to do the assignments marked as home assignments.

1. Solve Problems 3.4 and 3.5 from Exercise Session 3.
2. Explain the different parts of the block diagram. What do the different signals correspond to in the real lab process?



3. Make connections between the items below (not necessarily one-to-one):

Loop design choice	CL response change
Increase feedback gain	Improve stability margins
Add a lead-filter	Reduce stationary error
Add a lag-filter	Get faster response
Introduce integral action	Worsen stability margins

4. Sketch a typical desired Bode diagram of an open-loop system $L(s) = C(s)P(s)$. Mark the amplitude and phase margins and the cross-over frequency.
5. The flexible linear servo is a resonant system. How can this be seen in the Bode diagram of $P(s)$?

The process

The flexible linear servo process consists of two masses connected by a spring, see Figure 1. You may recall the process from lab 3 in the basic control course. The mass on one side is driven by a DC motor, which exerts a force F on the mass. Note that the only damper that is visible on the real process is d_2. However, other dampings present in the system (e.g. friction in the two carts) can be lumped together and represented as d_1 and d_2 in the model according to Figure 1.

In this lab you will control the position p_2 of the mass m_2. Both positions of the masses can be measured in the real process, but during the lab we will only use the position measurement p_2.

Linear model

There are two masses m_1 and m_2. The spring between the masses has the spring constant k. The dampings are d_1 and d_2.

One of the masses is driven by a DC motor. Here we neglect the internal dynamics of the motor. The force from the motor on the mass is proportional to the voltage

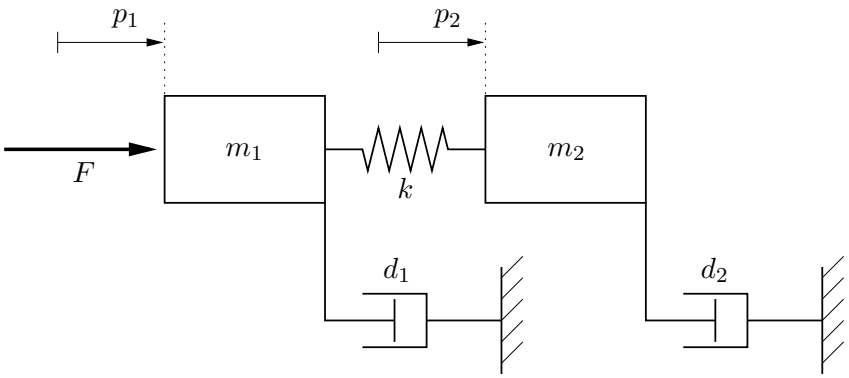


Figure 1 The flexible linear servo process.

u, that is

$$F = k_m \cdot u$$

Balance equations for the forces in the system gives us the following model:

$$m_1 \frac{d^2 p_1}{dt^2} = -d_1 \frac{dp_1}{dt} - k(p_1 - p_2) + F(t)$$

$$m_2 \frac{d^2 p_2}{dt^2} = -d_2 \frac{dp_2}{dt} + k(p_1 - p_2)$$

Introducing the state vector $x = [\, p_1 \quad \dot{p}_1 \quad p_2 \quad \dot{p}_2 \,]^T$ and the output $y = p_2$, the system can be written in state-space form,

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t)$$

where

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k}{m_1} & -\frac{d_1}{m_1} & \frac{k}{m_1} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{k}{m_2} & 0 & -\frac{k}{m_2} & -\frac{d_2}{m_2} \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ \frac{k_m}{m_1} \\ 0 \\ 0 \end{bmatrix}, \quad C = [\, 0 \quad 0 \quad k_y \quad 0 \,]$$

For a real lab process we have measured and estimated the following constants and coefficients:

$$\begin{aligned} m_1 &= 2.3 \text{ kg} \\ m_2 &= 2.1 \text{ kg} \\ d_1 &= 3.2 \text{ N/m/s} \\ d_2 &= 8.6 \text{ N/m/s} \\ k &= 400 \text{ N/m} \\ k_m &= 2.95 \text{ N/V} \\ k_y &= 280 \text{ V/m} \end{aligned}$$

The transfer function from u to y is given by

$$P(s) = \frac{68406}{s(s + 2.695)(s^2 + 2.791s + 362.6)}$$

Design specifications

A good control design should satisfy a number of requirements, e.g., good reference tracking, fast rejection of load disturbances, low sensitivity to measurement noise, robustness to modeling errors, and not use too big control signals.

In our case, the closed-loop specifications are the following: (see Figure 2 for an illustration)

- Reference step response:
  - Well-damped.
  - Rise-time (from 10% to 90% of the step response) between 0.2 and 0.6 seconds.
  - Settling time $\leq$ 2 seconds.
- Rejection of constant load disturbances in at most 2 seconds.

We also have the following requirements in order to ensure robustness to process variations:

- Phase margin $\geq 35°$.
- Gain margin ≥ 2 (≈ 6 dB)

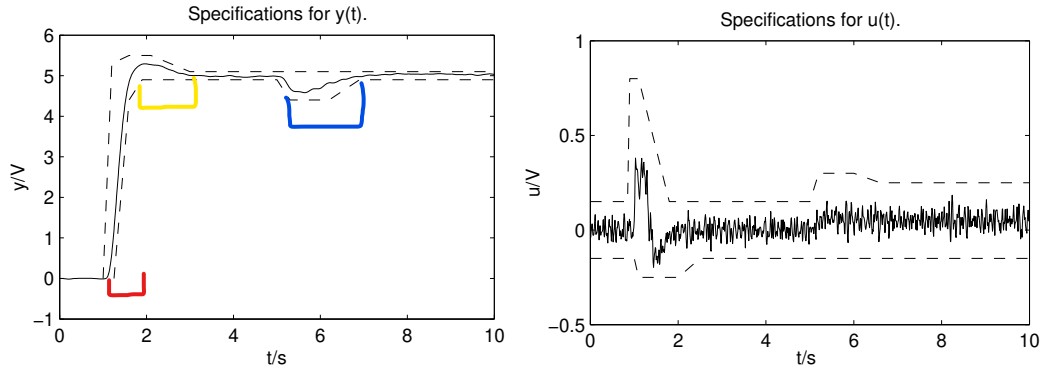


Figure 2 Specifications in the time domain: response to reference step at $t = 1$ and load disturbance at $t = 3$ (left) and the corresponding control signal (right).

2. The lab interface

Download and extract the lab files from the canvas webpage of the course. After you have started Matlab, ensure that your current folder is where you have extracted the files.

The controllers are designed and evaluated in MATLAB/Simulink. There is one Simulink model for simulation and another one for experiments on the real process. In both cases, the model uses as the controller the continuous-time transfer functions `C` (the feedback compensator) and `F` (the feedforward filter) from the MATLAB workspace. This means you don't need to change anything in the Simulink model, just define `C` and `F` in the workspace. Example:

```
s = zpk('s');
C = 1/s;     % A pure I-controller
F = 1;       % No feedforward filter
```

Define your controller directly in the script `loop_shaping.m` and run it before each simulation. The script creates a Bode plot with phase margin, amplitude margin and cross-over frequency. If needed, more plots can be added.

2.1 Simulink model

The control system can be simulated in the Simulink model *servo_simulated*. The model is also shown in Figure 3. It is possible to change the process parameters by double-clicking on the block *Change process parameters*. Double clicking on the block *Check specifications* will plot the results from the simulation together with the specifications. Using the switches (double click on them to change the signal routing) you can enable/disable the measurement noise and change the load disturbances.

You can also run the Simulink models with the following commands (it can be practical to integrate them in the `loop_shaping.m` script):

```
sim('servo_simulated');
specs;
```

2.2 Experiments on the real process

The Simulink model *servo_real* is similar to *servo_simulated*, but here we use the controller on the real process instead. Before each experiment you need to reset the current position measurement to zero by pressing the two buttons on the servo marked "POS RESET". If you get a **process overload** – marked by a red light on the servo, usually happens when the process goes unstable and one of the masses hits an end-point – , you will also need to (i) run a test with zero input ($u = 0$) and (ii) press the "RESET" button.

2.3 Workflow and learning outcome

The workflow of the lab should be that you iteratively:

1. design your controller in the `loop_shaping.m` script and study the loop transfer function, and
2. test the controller on the process and verify the specifications on the closed loop system.

The learning outcome should be that you are able to relate the shape of the *loop transfer function* with the behaviour of the *closed loop* system in the time-domain, and therefore be able to design it to meet the control system specifications.

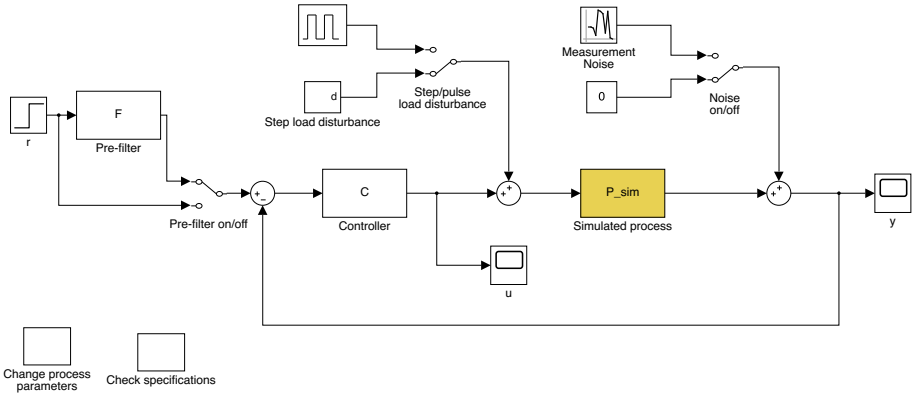


Figure 3 Simulink model `servo_simulated`.

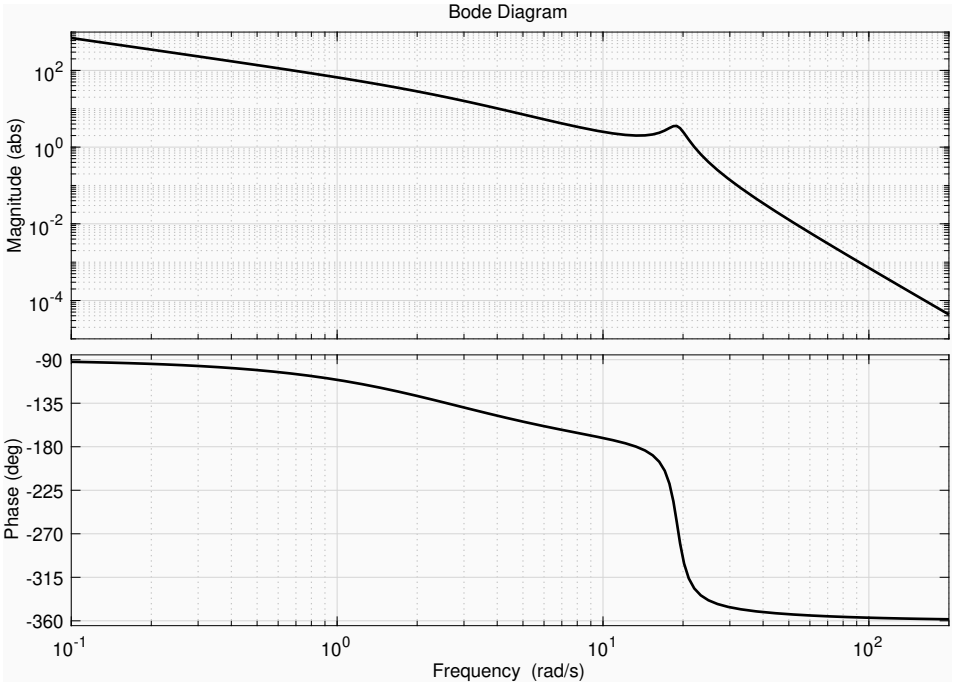


Figure 4 Bode plot for the process.

3. System Analysis

HOME Assignment 1. Run the script `servo_model` to load the process model into Matlab. Open the script and see what it does. Where are the poles and zeros located (use `pzmap`, type `axis equal` to scale the axes)? Plot the Bode diagram (see Figure 4 — you can change different plot settings using `ctrlpref`). Mark the frequencies of the poles on the bode plot and look at how they relate with the shape of the plot.

What is the cross-over frequency and phase margin of the system when it is controlled by a proportional controller with unit gain? Is the closed-loop system stable?

Simulate a push on the first mass with the command `impulse` (at the physical lab session you can try the same thing on the real process by tapping the first mass). Will the process be difficult to control? Why?

4. Loop-shaping design

You will now design a controller for the system so that it fulfills the specifications. The following assignments will make you follow the steps suggested in the lectures:

- Adjust gain to obtain the desired cross-over frequency,
- Add lag element to improve the low-frequency gain,
- Add lead element to improve the phase margin.

To achieve the desired performance you will also need to take care of the resonant peak with some extra trick (a notch filter). Also, to fully meet the performance, we will use a feedforward controller to smoothen the response and prevent overshoot.

Hint: do not expect to fix everything at the first iteration. Along the different steps, you will probably have to go back to the tuning of the previous parts of the controller to achieve the desired performance. This is valid especially for the gain, that you will likely need to re-adjust at every step to retain stability. However, the different steps should show you how to use and tune the different filters: focus on understanding this at the first attempts.

HOME Assignment 2 – P controller. Design a stabilizing P-controller that achieves the desired "response speed" (or gets close to it as much as possible). Are the other requirements (disturbance rejection, robustness, and performance) met?

To do this (as presented in Section 2.3), edit the script `loopshaping.m`. Run the script to produce a Bode-plot of the loop transfer function. Use the `servo_simulated` Simulink model to produce the step response.

HOME Assignment 3 – adding a lag filter The second step in the procedure is to add a lag filter (you are likely to have to adjust the gain to mantain stability). Think carefully on where to place the zero and pole. What do you expect it to improve? What will get worse instead? Verify your guesses with the bode plot and the simulation.

Hint: in the `loop_shaping.m` you are provided with templates for the different filters (lead, lag and notch). Use those to incrementally compose your controller.

HOME Assignment 4 – adding a lead filter The third step in the procedure is to add a lead filter (you are likely to have to adjust the gain to mantain stability). Think carefully on where to place the zero and pole. What do you expect it to improve? What will get worse instead? Verify your guesses with the bode plot and the simulation.

HOME Assignment 5 – adding a notch filter Now the resonant frequency is likely to be the main limiting factor in your controller. To overcome this limitation we can try to dampen the loop transfer function at said frequency with a notch filter. Design a notch filter to dampen the resonant frequency of the system. We suggest you to implement it in the following form:

$$N(s) = \frac{s^2 + \gamma b s + \omega_{cr}^2}{s^2 + b s + \omega_{cr}^2},$$

where $0 < \gamma < 1$ is the attenuation factor, $b > 0$ corresponds to the width of the rejected band and ω_{cr} is the central rejected frequency.

Feedforward design

Sometimes rejection of disturbances and fast response to changes in the reference value are almost mutually exclusive properties of the control system. A feedback controller handles in the same way changes in the reference and disturbances in the loop (detected through the feedback signal y). This can be seen in the feedback controller definition: $U(s) = C(s)\left(F(s)R(s) - Y(s)\right)$. The feedforward can be seen as a filter on the reference signal, and it does not affect the disturbance rejection properties of the system.

HOME Assignment 6 – adding a feedforward filter (**Note**: don't spend much time on this if you don't already have a good feedback controller.)

If you have overshoot or too strong control action in response to the reference step, add a feedforward filter $F(s)$ to your control design. A first order filter should be sufficient, but you can try also more advanced options.

One advanced possibility is to design the feedforward filter according to

$$F(s) = \frac{T(s)^{-1}}{(1 + sT_f)^d}$$

where $T(s) = \frac{PC}{1+PC}$ is the complementary sensitivity function. You must check that T does not have right half-plane zeros and choose d large enough to make sure that $F(s)$ is stable and proper.

If your controller is not yet meeting all the specification (both in the time-domain and in the frequency domain) iterate over the different filters that you have added so far and fine tune them. Always think carefully at what each parameter changes in the loop transfer function and what this will change in the simulation. You can also consider adding more filters (lead or lag).

To verify assignments from 1 to 6 at the physical lab session you will be asked to show your controller and motivate its design.

Assignment 7 – testing the controller on the real process Test your controller on the real process using the `servo_real.mdl` Simulink model. Does it work? If not, try to explain why! What do you think is the component of your controller that is most affected when switching to the real system? Do you see any significant differences when comparing the step response from the real process with the simulated response? Modify the controller to make it work better on the real process.

Note: The motor that drives the masses is very strong. Therefore it is important that you handle the process carefully!

Assignment 8 – model error robustness Try to change the mass m_2 in the simulation model, and try to control the modified process using the controller from the previous assignment. Use "Change process parameters" in the Simulink model. How good is the robustness towards process variations? Try to change other process parameters and see how the performance of the controller changes. You can also perform changes to the real process by removing some of the mass from either cart.

Assignment 9 – integral action So far we have not considered integral action in our controller, but it could be a good idea to do so. What filter in the controller serves the same purpose as the integral action? Try changing that filter for an integrator – this won't work right away: some other changes will be needed, e.g. changing the gain, but not exclusively.