FRTN10 - Multivariable Control

Laboratory Session 1

Loop Shaping for a Flexible Linear Servo[1]

Department of Automatic Control
Lund University

1. Introduction

The purpose of this laboratory session 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.

Pre-lab assignments

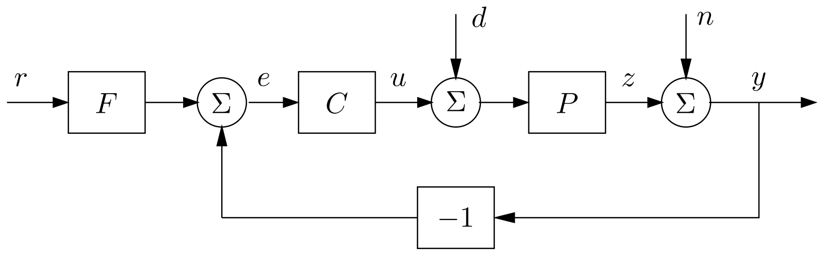
Before the lab session you should:

- Solve Problems 4.4 and 4.5 from Exercise Session 4.
- Read about Bode diagrams and lead/lag compensation. Good resources for this are the textbook from the basic control course and *Feedback Systems: An Introduction for Scientists and Engineers*[2].

At the beginning of the lab, you should also be able to discuss the points below:

Discussion points

1. Explain the different parts of the block diagram. What do the different signals correspond to in the real lab process?



2. Make connections between the items below:

Increase feedback gain	Improve stability
Add a lead-filter	Reduce stationary error
Add a lag-filter	Get faster response
Introduce integral action	

3. 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.
4. The flexible linear servo is a resonant system. How can this be seen in the Bode diagram of $P(s)$?

[1]Written by Tomas Olsson, latest update September 20, 2019.
[2]Available at: `http://www.cds.caltech.edu/~murray/amwiki/index.php?title=Main_Page`

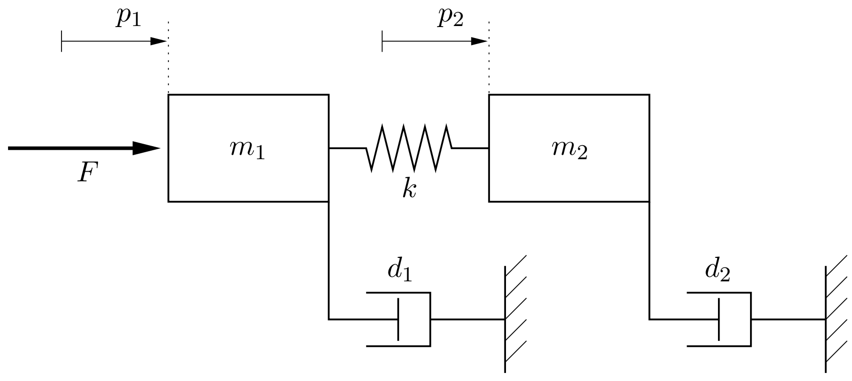


Figure 1 The flexible linear servo process.

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 can be lumped together and represented as d_1 and d_2 in the model according to Figure 1.

The purpose is to 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 u, that is

$$F = k_m \cdot u$$

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

$$\begin{aligned} m_1 \frac{d^2 p_1}{dt^2} &= -d_1 \frac{dp_1}{dt} - k(p_1 - p_2) + F(t) \\ m_2 \frac{d^2 p_2}{dt^2} &= -d_2 \frac{dp_2}{dt} + k(p_1 - p_2) \end{aligned}$$

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,

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned}$$

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 between 0.2 and 0.6 seconds.
  - Settling time ≤ 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 $(= 6 \text{ dB})$

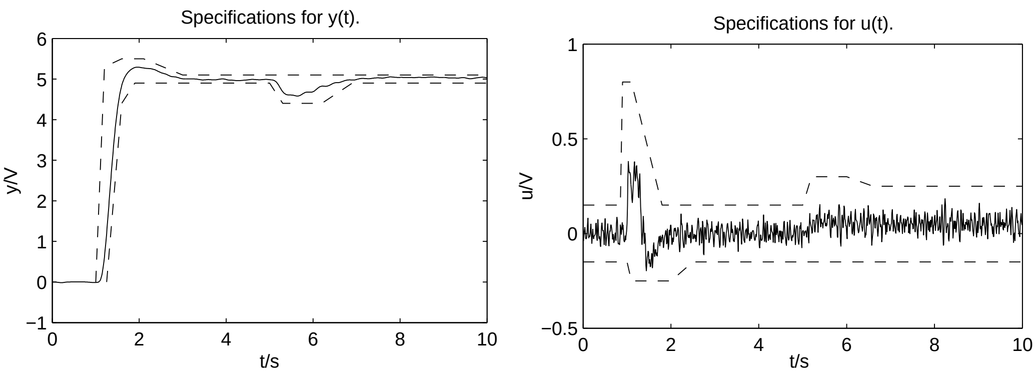


Figure 2 Specifications in the time domain: response to reference step at $t = 1$ and load disturbance at $t = 3$ (left) and the corresponding control signal (right).

2. The lab interface

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, you define your controller by simply entering the continuous-time transfer functions `C` (the feedback compensator) and `F` (the feedforward filter) in the MATLAB workspace. Example:

```
s = zpk('s');
C = 1/s;     % A pure I-controller
F = 1;       % No feedforward filter before Assignment 6
```

To save time, you can define your controller directly in `loop_shaping.m` and run this script 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.

Before you start you should download and extract the lab files from the course webpage. After you have started Matlab, ensure that your current folder is where you have extracted the files.

Simulation model

The control system can be simulated in the Simulink model *servo_simulated* (type `servo_simulated` to open it). The model is shown in Figure 3. It is possible to change the process parameters by double-clicking on the button *Change process parameters.* The button *Check specifications* will plot the results from the simulation together with the specifications. Using the switches we can enable/disable the measurement noise and change the load disturbances.

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, you will also need to press the "RESET" button.

3. Introductory experiments

Assignment 1. Load the process model into Matlab by running the script `servo_model`. Type `edit servo_model` in Matlab and see what the script does. Plot the Bode diagram (see Figure 4 — change the gain unit of the Bode diagrams using `ctrlpref`). Where are the poles and zeros located (use `pzmap` and set `axis equal`)? Simulate a push on the first mass with the command `impulse`. Try the same thing on the real process by tapping the first mass. Will the process be difficult to control?

Assignment 2. What is the cross-over frequency and phase margin of the system when it is controlled by a proportional controller with $K = 1$? Is the closed-loop system stable?

Try finding a P-controller that gives a phase margin $\geq 35^\circ$ and a gain margin $\geq 2\,(= 6\text{dB})$. To do this, enter your P-controller in the script `loopshaping.m`, and then run the script to produce a Bode-plot of the loop transfer function. Will the resulting P-controller also satisfy the specifications on the time response? To check this, run the commands

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

These commands runs the simulation model `servo_simulated` with the controller `C` that you have specified, and then plots the specifications together with the time response from the simulation. The commands are also present in the script `loopshaping.m`, you just need to uncomment them.

Now consider a PI controller,

$$C(s) = K\left(1 + \frac{1}{sT_i}\right) = K\left(\frac{sT_i + 1}{sT_i}\right)$$

Do you think a PI controller would work better? (Look at the Bode phase plot!) How about a PID?

4. Loop-shaping design

In this section, we will design different controllers $C(s)$ to shape the frequency response of the open-loop system $L(s) = C(s)P(s)$. We will ignore the feedforward filter $F(s)$ until Assignment 5.

In the following assignments, you should work with the Bode plot of $L(s)$ until you get the desired shape. The command `margin` plots a Bode diagram and also shows the amplitude and phase margins. Also simulate the system to check your design against the time-domain specifications.

Assignment 3. Use loop shaping to find a controller $C(s)$ that fulfills the design specifications as well as possible.

Note: You do not have to fulfill the specification on the overshoot in the reference step, since the overshoot can be reduced later by using a feedforward filter.

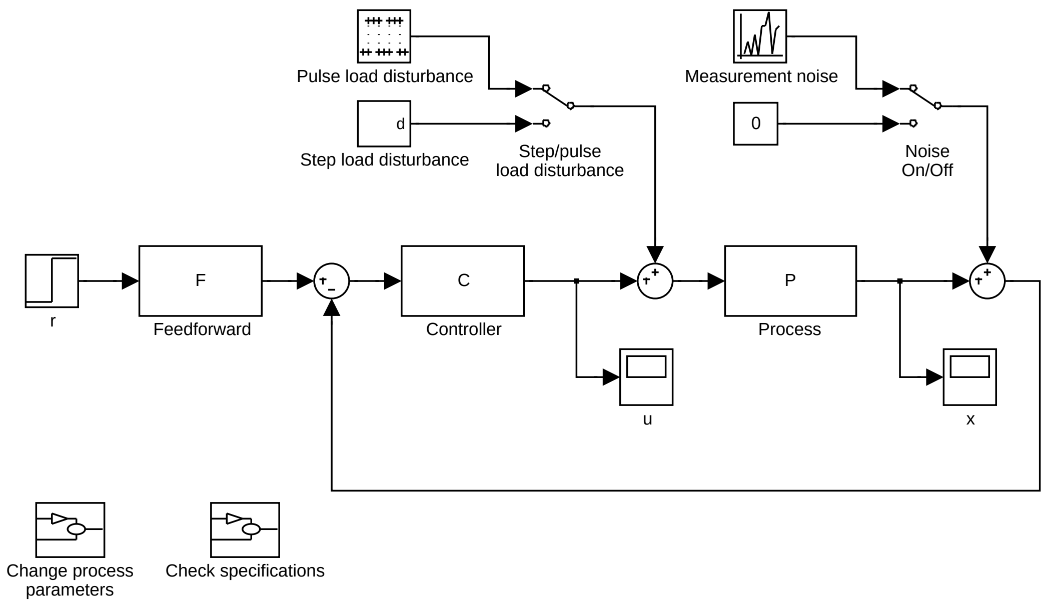


Figure 3 Simulink model `servo_simulated`.

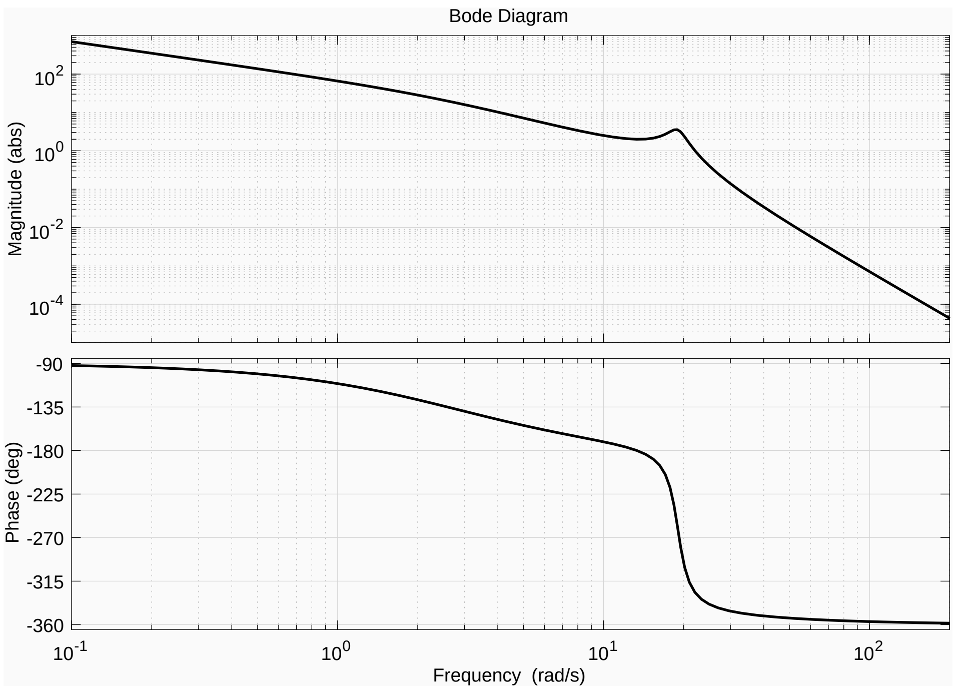


Figure 4 Bode plot for the process.

One way is to start with a notch filter to reduce the effect of the resonant peak to be able to get a cross-over frequency that is high enough. To do so, you can use an element of the form

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

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. Try to find suitable values of the notch filter parameters to smoothen out the resonant peak. Now by adding zeros and poles, you may try to improve the low-frequency gain and the phase margin while aiming for a cross-over frequency at around 5–8 rad/s. A single lead or lag filter will not be enough to fulfill the specifications.

Try to fulfill both the time domain specifications and the specifications on the phase and amplitude margin stated on page 3. The time domain specifications are easily checked using “Check specifications” in the Simulink model.

Enter your controller design in the script `loop_shaping.m`. As previously mentioned, it is convenient to run the Simulink model and check the specifications directly from the script with

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

When you find a controller that works well in simulations, try to control the real process. Does it work? If not, try to explain why! 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 4. 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.

Feedforward filter

Sometimes rejection of disturbances and fast response to changes in the reference value are almost mutually exclusive properties of the control system. As you have seen in the previous assignment, it is difficult to satisfy all the specifications using a feedback controller. This is because it handles changes in the reference signal and changes in the process output in the same way, according to

$$U(s) = C(s)\left(R(s) - Y(s)\right).$$

Often it is better to first design a feedback controller to achieve good rejection of load disturbances, robustness to process variations, and small amplification of measurement noise. Then we can design the feedforward filter $F(s)$ to better handle changes in the reference signal.

The new controller can be written

$$U(s) = C(s)\left(F(s)R(s) - Y(s)\right).$$

The feedforward can be seen as a filter on the reference signal, and it does not affect the disturbance rejection properties of the system.

Assignment 5. Add a feedforward filter $F(s)$ to the design you obtained in Assignment 3. If you have an overshoot in the step response, a very simple option is to use a first-order filter,

$$F(s) = \frac{1}{1 + sT_f}.$$

A more advanced option 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.

After simulating the controller including the feedfoward filter, test it on the real process. What is different from the simulations?

Assignment 6 (*). Change the process model so that we instead measure and control the position of mass 1. How does this change the process poles, zeros and Bode plot? Try to design a controller for this process in the same way as for mass 2. Is it easier or more difficult than for mass 2? Test your controller on the real process.