Automatic Control – Basic Course

Laboratory Exercise 3

Position Control of Elastically Connected Masses with Network Delays

Department of Automatic Control
Lund University
Latest updated May 2019

1. Introduction

In previous labs we have studied a process, which has been relatively simple. We have been able to control it satisfactory using a simple PID-controller. In this lab we shall examine a process which is a bit more complicated and which requires a more advanced controller. The controller which we will use is based on state feedback and state estimation. We shall also examine how time delays in the control loop, introduced by the communications network, influence the control and experiment with time delay compensation.

Preparations

- Read through this manual.
- Review the lectures on state feedback, Kalman filtering and time delay compensation. At the beginning of the lab you should be able to answer the following questions:
  - What does state feedback mean? Explain in words!
  - Why is an observer often used in connection with state feedback?
  - How does the Smith predictor work? Explain in words!
- Solve the preparatory assignments 4.2, 5.2, 6.1 and 7.1.
- Study the MATLAB files `define_process.m`, `design1.m`, `design2.m` and `design3.m` which are found at the end of the manual (the files in this manual are commented in English, whereas the real files are commented in Swedish). Try to relate their content to the assignments in the manual.

2. The Process

A picture of the flexible servo to be controlled is shown in Figure 1. The process consists of two masses which are interconnected by a spring. A conceptual drawing of the process is shown in Figure 2. The mass at one end of the spring can be moved by a motor. We call this end the motor end and the other end the load end. Note that of the two dampers d_1 and d_2, only d_2 is present as a discrete part of the real process. Damping at other locations in the process can, however, be added and modeled according to Figure 2.

The purpose is to control the position p_2 of the mass on the load side. On the lab process, the position of both masses can be measured, but in the lab we will assume that only p_2 is measurable. The remaining states will be estimated by a Kalman filter.

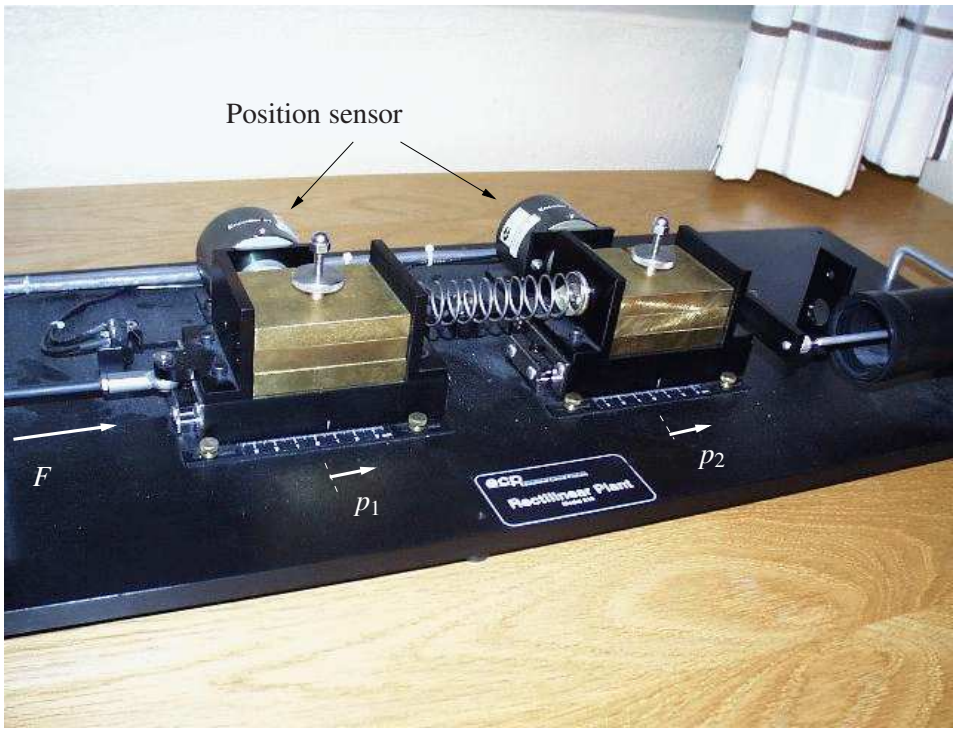


Figure 1 The flexible servo.

Circuit diagram

The D/A-converter should be connected to the process as shown in Figure 3.

A Linear Model of the Process

The two masses are m_1 and m_2. The interconnecting spring has the spring constant k. The damping of the masses are d_1 and d_2, respectively.

The first mass is driven by a brush-less DC motor which is driven by a current-feedback amplifier. Motor and amplifier dynamics are neglected. The force of the motor becomes proportional to the input voltage u of the amplifier according to

$$F = k_m u.$$

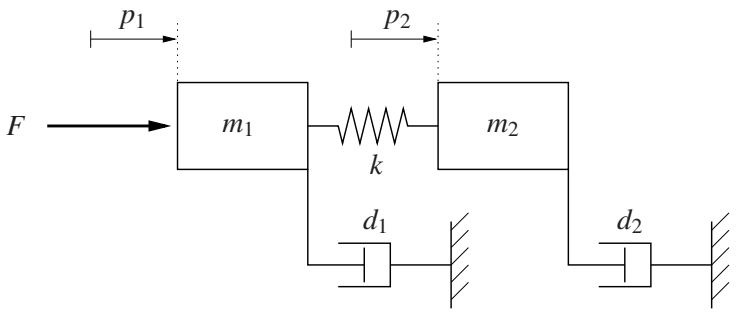


Figure 2 Conceptual drawing of the process.

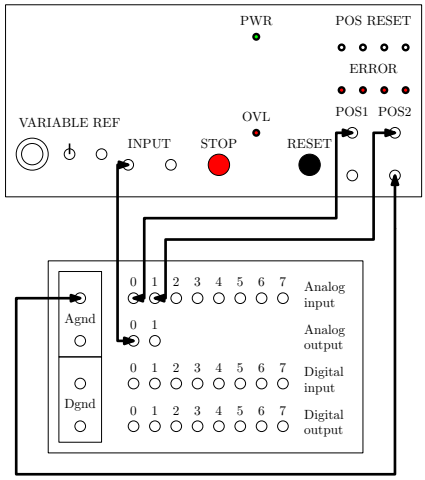


Figure 3 Circuit diagram for the process.

A force balance gives the following dynamic model:

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

Introduce the state vector $x = \begin{pmatrix} p_1 & \dot{p}_1 & p_2 & \dot{p}_2 \end{pmatrix}^T$. The process can then be expressed in state space form as

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \tag{1}$$

where

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -\frac{k}{m_1} & -\frac{d_1}{m_1} & \frac{k}{m_1} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{k}{m_2} & 0 & -\frac{k}{m_2} & -\frac{d_2}{m_2} \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ \frac{k_m}{m_1} \\ 0 \\ 0 \end{pmatrix} \tag{2}$$
$$C = \begin{pmatrix} 0 & 0 & k_y & 0 \end{pmatrix}$$

For the real lab process the following values of constants and coefficients have been measured and estimated

$$\begin{aligned} m_1 &= 2.29 \text{ kg} \\ m_2 &= 2.044 \text{ kg} \\ d_1 &= 3.12 \text{ N/m/s} \\ d_2 &= 3.73 \text{ N/m/s} \\ k &= 400 \text{ N/m} \\ k_m &= 2.96 \text{ N/V} \\ k_y &= 280 \text{ V/m} \end{aligned}$$

Examination of the Process

Assignment 2.1 Log in and start MATLAB by opening a terminal window and typing `cd;./labstart`. Define the process `Gp` by executing the file `define_process.m`:

```
>> define_process
```

Calculate the poles of the system by typing

```
>> pole(Gp)
```

Where are the poles located? Is the system stable? Asymptotically stable? Push the real process with your hand and study its stability. Does the real behavior agree with the analysis?

Assignment 2.2 Draw the Bode plot of the process

```
>> bode(Gp)
```

If the amplitude in the Bode plot is shown in decibel, it can be changed by executing the command

```
>> ctrlpref
```

and change "Magnitude" from dB to absolute.

Note the resonance peak in the amplitude curve. At what frequency is it located? What relation holds between the location of the resonance peak and the locations of the poles? Excite the process by hand and try to estimate its natural frequency. Does it coincide with that of the model?

3. Specifications

The specifications are the requirements which the controlled system shall fulfill. In this case we have chosen to specify the closed loop system in the time domain. We wish to have a well damped step response with a rise time between 0.2 and 0.4 seconds, see Figure 4. In the meanwhile, the control signal shall not be to violent, because this would wear the motor.

Additionally, to obtain a certain robustness against model uncertainties and time delays we want a phase margin of at least 30°.

The oscillative properties of the process make it impossible to fulfill the specifications with a PID-controller. An attempt using a PD-controller is shown in Figure 5. The step response is fast enough, but the controller cannot damp out the oscillations.

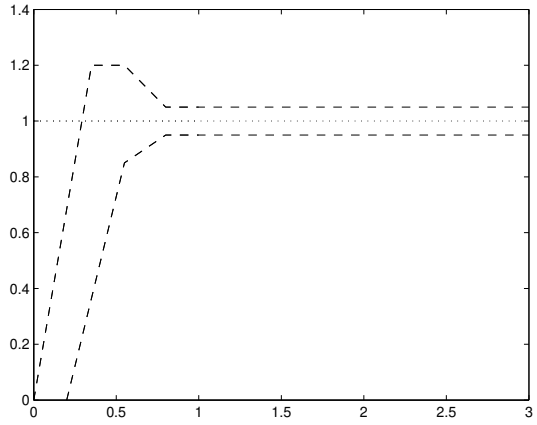


Figure 4 The step response shall lie within the marked region.

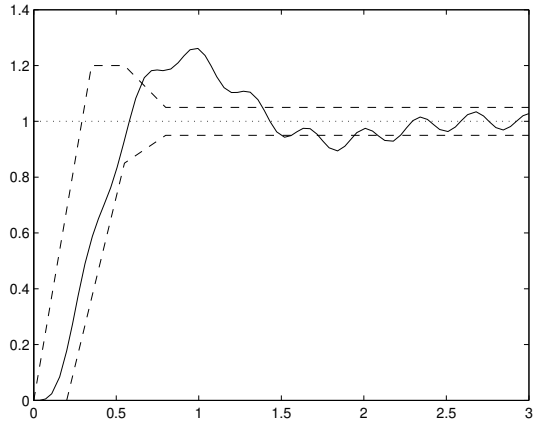


Figure 5 Step response using a PD controller with $K = 0.07$ and $T_d = 0.1$.

4. State Feedback

To be able to change the process dynamics arbitrarily we make use of state feedback. First we confirm that the system is controllable.

Assignment 4.1 Find and calculate the rank (i.e. the number of linearly independent columns) of the system's controllability matrix:

```
>> Wc = [ ... ]
>> rank(Wc)
```

(Note that the matrices `A` and `B` are already defined.)

Is the system controllable?

If we start out under the assumption that the entire state vector x can be measured, the following control law can be used

$$u = -Lx + l_r r \tag{3}$$

Here L is a row vector, r is the reference value and l_r is a scalar, see Figure 6.

Assignment 4.2 (Preparation) Show that the closed loop system can be written on the form

$$\dot{x} = A_{cl}\, x + B_{cl}\, r$$
$$y = C_{cl}\, x$$

when the control law (3) is used on the process. (1). What are A_{cl}, B_{cl} and C_{cl}? How should l_r be chosen to obtain static gain 1 from r to y? Express your answers with matrix notations used in the state model (1), i.e., do **not** use the explicit expressions in (2). ◇

By means of state feedback we can place the poles of the closed loop system arbitrarily. Practically, there are, however, limitations, e.g. limitations of the control signal. Somewhat simplified one can state that the further a pole is moved from its original location, the more control action will be required.

The desired pole placement can be expressed using a forth order characteristic polynomial (see Figure 7):

$$\left(s^2 + 2\zeta_a\omega_a s + \omega_a^2\right)\left(s^2 + 2\zeta_b\omega_b s + \omega_b^2\right)$$

Given a pole placement, the feedback vector L is easily computed using the command `place` (see `design1.m`).

Assignment 4.3 Edit the file `design1.m` by in MATLAB executing

```
>> edit design1.m
```

Place the poles of the closed loop system by inserting suitable values of ω_a, ζ_a, ω_b and ζ_b. Then calculate the controller in MATLAB by typing:

```
>> design1
```

Open the Simulink model `model1.mdl` by typing

```
>> model1
```

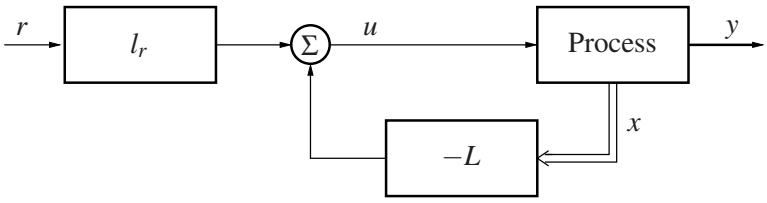


Figure 6 State feedback.

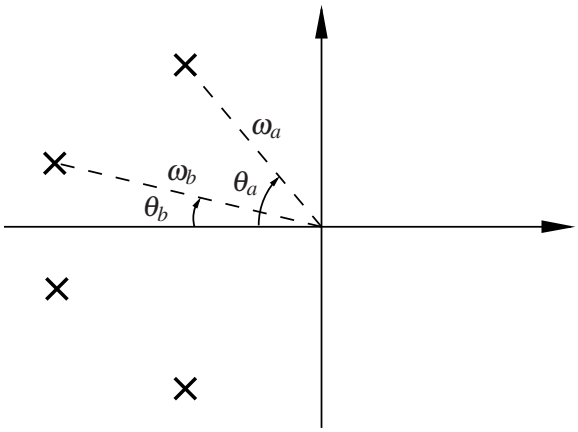


Figure 7 Pole placement according to the characteristic polynomial $\left(s^2+2\zeta_a\omega_a s+\omega_a^2\right)\left(s^2+2\zeta_b\omega_b s+\omega_b^2\right)$, dr $\zeta_a=\cos\theta_a$ och $\zeta_b=\cos\theta_b$.

Simulate the closed loop system and see if it fulfills the specification on the step response. The command `specs` can be used after a simulation to compare the results to specification in Figure 3. Change the design parameters and repeat the procedure until a desired behavior is obtained. What is a suitable pole placement for the state feedback?

Assignment 4.4 Why don't we try this controller on the real process?

5. Observers

Practically, we cannot measure all states of the process, but only its output y. Instead we acquire a model of the process and feed the model with the same input as the real process. The difference between the outputs of the model and real process is used to correct the state of the model so that it converges to the state of the process. Such a device is called an *observer* or *Kalman filter*.

The observer is described by

$$\frac{d\hat{x}}{dt}=A\hat{x}+Bu+K\left(y-C\hat{x}\right) \tag{4}$$

where $\hat{x}$ denotes the estimated states. The column vector K can be chosen such that the states of the observer converge to the states of the process arbitrarily fast, given that the system is observable.

Assignment 5.1 Find and calculate the rank of the observability matrix for the system:

```
>> Wo = [ ... ]
>> rank(Wo)
```

Is the system observable?

Using the Kalman filter we can establish feedback from the estimated states instead of the real states, see Figure 8. The new control law becomes

$$u = -L\hat{x} + l_r r \tag{5}$$

Assignment 5.2 (Preparation) Starting out with (4) and (5), show that the controller based on state feedback from the estimated states can be written on the form

$$\frac{d\hat{x}}{dt} = A_R\hat{x} + B_{R_y}y + B_{R_r}r$$
$$u = C_R\hat{x} + D_{R_y}y + D_{R_r}r$$

What are A_R, B_{R_y}, B_{R_r}, C_R, D_{R_y} and D_{R_r}? Express your answers with the matrix notations used in (1), (4) and (5). ◇

Because the process has four states, the observer will also have four states. We specify the poles of the observer according to the following characteristic polynomial

$$\left(s^2 + 2\zeta_c\omega_c s + \omega_c^2\right)\left(s^2 + 2\zeta_d\omega_d s + \omega_d^2\right)$$

A suitable choice of poles depend, among other things, on the amount of measurement noise, the size of modeling inaccuracies and wether the initial condition is known. Fast poles mean high amplification of measurement noise, whereas slow poles give slow convergence of the estimate. As starting point, a rule of thumb stating that the observer poles should be 1.5-2 times faster than the state feedback, could be used.

Assignment 5.3 Edit the file `design2.m` and enter the values of ω_a, ζ_a, ω_b and ζ_b from section 4. Then enter some suitable values for ω_c, ζ_c, ω_d and ζ_d. Calculate the entire controller (state feedback + observer) by executing the file:

```
>> design2
```

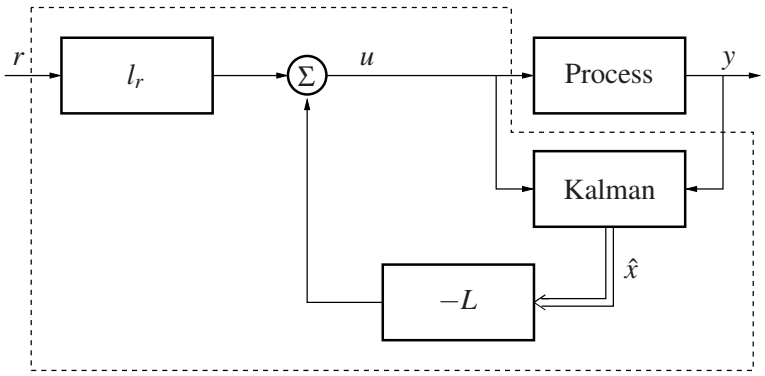


Figure 8 Feedback from estimated states. The controller consists of the blocks within the dashed line.

Then open the Simulink model `model2.mdl` by typing

```
>> model2
```

Simulate the closed loop system and see if it fulfills the specifications. Change the design parameters and iterate the procedure until desired behavior is obtained. (If necessary, also change the pole placement for the state feedback.) Also draw the Bode plot of the open loop system using

```
>> margin(Gp*Gr)
```

and verify that the specification on the phase margin is fulfilled. Also in this assignment you should try to keep the phase margin around 40°. What is a suitable pole placement for the observer?

Assignment 5.4 Draw the Bode plot by typing

```
>> bode(Gr)
```

What gain does the controller have for low frequencies? What does this mean to the controller's ability to suppress constant load disturbances?

Assignment 5.5 Try the controller on the real process, see section A. How does the real step response differ from the simulated one? What is limiting the performance?

Assignment 5.6 On the real process it is actually possible to measure both x_1 (the position of the first mass) and x_3 (the position of the second mass), even though we assume in the lab that only x_3 is available for feedback. We can, however, use both measurement signals to examine how well the Kalman filter works. Study how well the estimated states $\hat{x}_1$ and $\hat{x}_3$ agree with the real process states x_1 and x_3. For which state do we have the best estimate?

6. Integral Action

To eliminate stationary errors integral action is introduced in the controller, see Figure 9. The integrator is introduced as an extra state x_i according to

$$\begin{aligned} x_i &= \int (r-y)\,dt \\ \dot{x}_i &= r-y = r-Cx \end{aligned} \tag{6}$$

If the extended state vector

$$x_e = \begin{pmatrix} x \\ x_i \end{pmatrix}$$

is introduced, the extended system (i.e. the process and the integrator) can be written

$$\begin{aligned} \dot{x}_e &= \underbrace{\begin{pmatrix} A & 0 \\ -C & 0 \end{pmatrix}}_{A_e} x_e + \underbrace{\begin{pmatrix} B \\ 0 \end{pmatrix}}_{B_e} u + \underbrace{\begin{pmatrix} 0 \\ 1 \end{pmatrix}}_{B_r} r \\ y &= \underbrace{\begin{pmatrix} C & 0 \end{pmatrix}}_{C_e} x_e \end{aligned}$$

If we, for the moment, reassume that the entire state vector is measurable, we can establish feedback from both the states of the process and the integral state according to

$$u = -Lx - l_i x_i + l_r r = -L_e x_e + l_r r$$

where

$$L_e = \begin{pmatrix} L & l_i \end{pmatrix}$$

The closed loop system becomes

$$\begin{aligned} \dot{x}_e &= (A_e - B_e L_e)x_e + (B_e l_r + B_r)r \\ y &= C_e x_e \end{aligned}$$

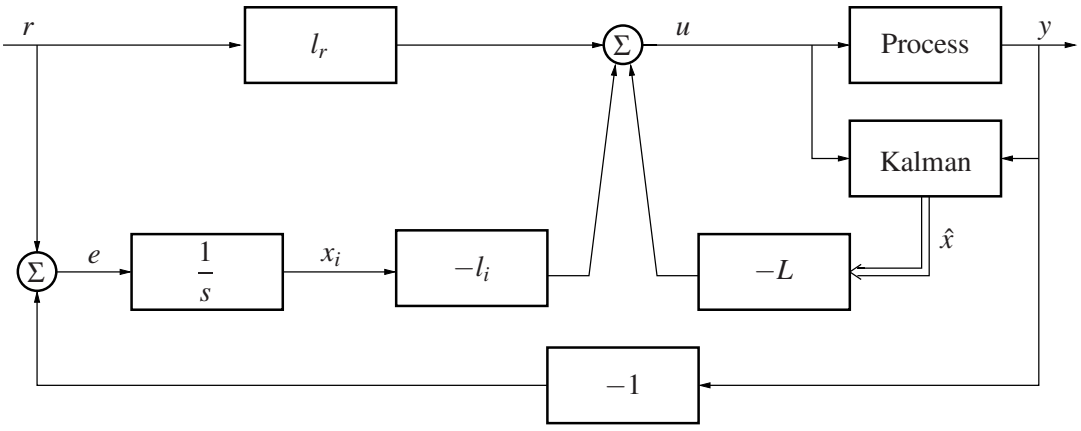


Figure 9 Feedback from estimated states with integral action.

Because the extended system has five states, the poles of the state feedback are now specified using a fifth order characteristic polynomial:

$$\left(s^2+2\zeta_a\omega_a s+\omega_a^2\right)\left(s^2+2\zeta_b\omega_b s+\omega_b^2\right)\left(s+\omega_e\right)$$

As before, we cannot measure the states of the process. Consequently, feedback is established from the estimated states and the integrator according to the control law

$$u=-L\hat{x}-l_i x_i+l_r r \tag{7}$$

Assignment 6.1 (Preparation) Starting out with (4), (6) and (7), show that a controller with integral action based on state feedback from estimated states can be written on the form

$$\begin{pmatrix}\frac{d\hat{x}}{dt}\\ \frac{dx_i}{dt}\end{pmatrix}=\underbrace{\begin{pmatrix}* & *\\ * & *\end{pmatrix}}_{A_R}\begin{pmatrix}\hat{x}\\ x_i\end{pmatrix}+\underbrace{\begin{pmatrix}*\\ *\end{pmatrix}}_{B_{Ry}}y+\underbrace{\begin{pmatrix}*\\ *\end{pmatrix}}_{B_{Rr}}r$$

$$u=\underbrace{\begin{pmatrix}* & *\end{pmatrix}}_{C_R}\begin{pmatrix}\hat{x}\\ x_i\end{pmatrix}+\underbrace{\begin{pmatrix}*\end{pmatrix}}_{D_{Ry}}y+\underbrace{\begin{pmatrix}*\end{pmatrix}}_{D_{Rr}}r$$

What are A_R, B_{R_y}, B_{R_r}, C_R, D_{R_y} and D_{R_r}? ◇

Assignment 6.2 Edit the file `design3.m` and insert your values on ω_a, ζ_a, ω_b, ζ_b, ω_c, ζ_c, ω_d and ζ_d. Also insert suitable values of ω_e. Calculate the entire controller (state feedback + integrator + observer) by executing the file:

```
>> design3
```

Open the Simulink model `model3.mdl` by typing

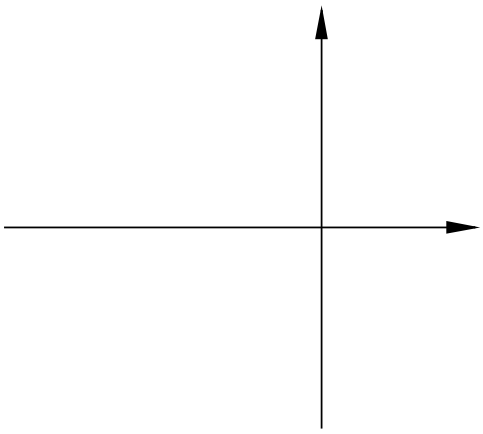
```
>> model3
```

Simulate the closed loop system and see wether it fulfills the specifications. Change the design parameters and iterate the procedure until the step response specification is fulfilled. Also draw the Bode plots of the open loop system by typing

```
>> margin(Gp*Gr)
```

and verify that the specification on the phase margin is fulfilled. What is a suitable pole placement?

Assignment 6.3 Insert *all* poles of the closed loop system in the pole zero plot below:



Assignment 6.4 Draw the Bode plot of the controller using

```
>> bode(Gr)
```

How is it seen that the controller has integral action?

Assignment 6.5 Try the controller on the real process. How do the results differ from assignment 5.5.?

When integral action is introduced, the term $l_r r$ is no longer needed in the control law to obtain the correct static gain – this is handled by the integrator. Instead l_r can be chosen to trim the step response at reference value changes. As seen in (7), $l_r \neq 0$ means a direct connection between reference value and control signal. A value $l_r > 0$ can also be used to give the process an extra ”push” at a reference value step. (This could be especially useful when controlling the real process, which has unmodelled friction.)

Assignment 6.6 What value does l_r have now? Change the value of `lr` in `design3.m` and conduct new experiments on the real process. What is a suitable value for l_r?

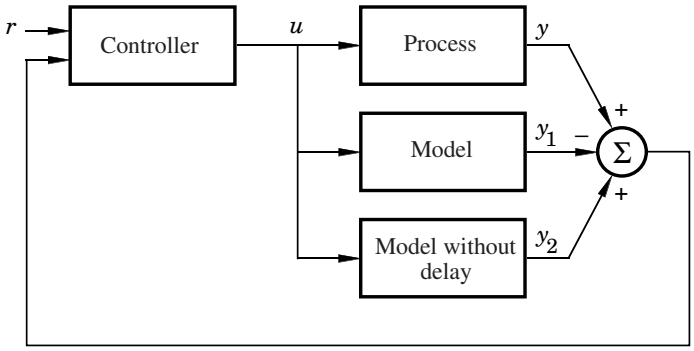


Figure 10 The Smith predictor.

7. Control with Time Delay

Modern control systems are often distributed, so that sensors, actuators and controllers are situated at different nodes in a network. The advantage with distributed systems is mainly their flexibility. A disadvantage, from the control point of view, are the time delays which arise when a control loop is closed over a network. If the delays are long compared to the speed of the closed loop system, performance can be significantly decreased.

In this section we continue working on the controller we designed in the previous section, but add a time delay in the control loop by delaying the control signal.

Assignment 7.1 (Preparation) What is the delay margin for a system with phase margin φ_m and cutoff frequency ω_c?

Assignment 7.2 Compute the delay margin of the controller which you designed in assignment 6.2 using the command

```
>> allmargin(Gp*Gr)
```

(If the amplitude curve of the open loop system has multiple crossings, there can exist several cutoff frequencies and phase margins. In this case the correct value is the *smallest* value of the delay margin reported by `allmargin`.)

Control with dead time

The controlled process is slow in comparison to the speed in typical local area networks intended for control. To obtain delays of significant influence on performance, we shall instead try to control the process through a detour.

We now assume that the process output is sampled with a period of $h = 10$ ms. The measurement signal is then delayed using a model of a network delay. You can decide the length of the delay in the network yourselves. In the model, noise is added to the desired delay, so that the delay will vary.

If the time delay is known and somewhat constant it can be compensated for by using a Smith predictor, see Figure 10. The concept of the Smith predictor is to control the process using a simulated model without delay. The real output is canceled by a simulated

model with delay. This obviously requires a good model of the real process. Additionally, it requires the process model to be stable.

Assignment 7.3 Open the Simulink model `model4.mdl` by typing

```
>> model4
```

We begin by investigating the performance without delay compensation. Enter a couple of dead times between 5 ms and 25 ms and simulate the system. Write down your result. How long delays can the system handle without becoming unstable? Does the result agree with the analysis in assignment 7.2?

(The sampling of the measurement signal and updating of the control signal is done in the beginning of every sampling period. This results in an upward rounding of the real delay to the closest multiple of 10 ms. Additionally, the sampling by itself can be considered to introduce an extra delay of a half sampling period, i.e. an additional 5 ms delay.)

Assignment 7.4 Enter a time delay of around 30–50 ms. Activate the Smith predictor by entering the estimated delay in the delay block. How well does the delay compensation work?

Assignment 7.5 Also try the Smith predictor on the real process. How does the real step response differ from the simulated one?

Assignment 7.6 Try some really long delays. How long delays can be compensated for using the real process?

8. Summary

This summary is intended to review relevant questions which you should be able to answer after finishing the experimental part. The lab assistant will go through your summary before you can pass the lab.

Assignment 8.1 Mention at least three limitations of the real process which are not captured by the mathematical model.

Assignment 8.2 The flexible servo is a strongly resonant process. How can this be seen in its Bode plot and pole-zero diagram, respectively?

Assignment 8.3 Why didn't we try the controller based on pure state feedback (section 4) on the real process?

Assignment 8.4 How can state feedback be used if all states are not measurable?

Assignment 8.5 When using state feedback from estimated states (section 5), how many poles does the closed loop system have?

Assignment 8.6 How many states did the controller with integral action (section 6) contain? Which?

Assignment 8.7 How is it seen in the Bode plot of the controller wether it has integral action or not?

Assignment 8.8 Why is it practically impossible to compensate for arbitrary long delays?

A. Handling of the Process

On the process there are buttons to reset the position sensors. Place the masses in position 0 and press RESET and the two POS buttons. If the masses hit either of the stop blocks a safety system is activated and the current is cut from the motor. The safety system is reset by pressing the RESET button. This can only be done when the input to the process is zero.

Always put the masses in the zero position and reset the position sensors before every experiment to decrease transient behavior at the beginning of the experiment.

Observe! The motor driving the masses is very strong and it is therefore important to handle the process with respect. Think about where you place your hands during experiments.

B. MATLAB-filer

define_process.m

```
% Create a linear model of the process

m1 = 2.29; m2 = 2.044;                  % masses
d1 = 3.12; d2 = 3.73;                   % damping constants
k = 400;                                % spring constant
km = 2.96;                              % motor constant
ky = 280;                               % measurement constant

A = [0 1 0 0; -k/m1 -d1/m1 k/m1 0; 0 0 0 1; k/m2 0 -k/m2 -d2/m2];
B = [0; km/m1; 0; 0];
C = [0 0 ky 0];
D = 0;

Gp = ss(A,B,C,D);                       % create state space model of the process
```

design1.m — Calculation of controller based on pure state feedback

```
% Design of state feedback

omegaa = ...;                           % speed of one pole pair
zetaa =  ...;                           % damping of one pole pair
omegab = ...;                           % speed of the other pole pair
zetab =  ...;                           % damping of the other pole pair

pc = conv([1 2*omegaa*zetaa omegaa^2],[1 2*omegab*zetab omegab^2]);
poles1 = roots(pc);

L  = place(A,B,poles1);                 % compute the state feedback vector L

lr = 1/(C*inv(-A+B*L)*B);               % compute lr such that the static gain
                                        % from r->y becomes 1
```

design2.m — Calculation of controller based on state feedback from observer

```
% Design of state feedback

omegaa = ...;                          % speed of one pole pair
zetaa =  ...;                          % damping of one pole pair
omegab = ...;                          % speed of the other pole pair
zetab =  ...;                          % damping of the other pole pair

pc = conv([1 2*omegaa*zetaa omegaa^2],[1 2*omegab*zetab omegab^2]);
poles1 = roots(pc);

L  = place(A,B,poles1);                % compute the state feedback vector L
lr = 1/(C*inv(-A+B*L)*B);              % compute lr such that the static gain
                                       % from r->y becomes 1
% Design of Observer

omegac = ...;                          % speed of one pole pair
zetac =  ...;                          % damping of one pole pair
omegad = ...;                          % speed of the other pole pair
zetad =  ...;                          % damping of the other pole pair

po = conv([1 2*omegac*zetac omegac^2],[1 2*omegad*zetad omegad^2]);
poles2 = roots(po);

K = place(A',C',poles2)';              % compute the Kalman gain K

% Computation of controller (observer + state feedback)

AR = A-B*L-K*C;
BRy = K;
BRr = B*lr;
CR = -L;
DRy = 0;
DRr = lr;

Gr = -ss(AR, BRy, CR, DRy);            % transfer function from -y to u
```

design3.m — Calculation of controller based on state feedback from observer with integral action

```
% Design of state feedback with integral action

Ae = [A zeros(4,1); -C 0];              % A-matrix for the extended system
Be = [B; 0];                            % B-matrix for the extended system

omegaa = ...;                           % speed of one pole pair
zetaa =  ...;                           % damping of one pole pair
omegab = ...;                           % speed of the other pole pair
zetab =  ...;                           % damping of the other pole pair
omegae = ...;                           % speed of the fifth pole

pc = conv([1 2*omegaa*zetaa omegaa^2],[1 2*omegab*zetab omegab^2]);
pc = conv(pc, [1 omegae]);
poles1 = roots(pc);

Le  = place(Ae,Be,poles1);              % compute the state feedback vector Le
L = Le(1:4);
li = Le(5);
lr = 0;                                 % direct connection from reference value

% Design of observer

omegac = ...;                           % speed of one pole pair
zetac =  ...;                           % damping of one pole pair
omegad = ...;                           % speed of the other pole pair
zetad =  ...;                           % damping of the other pole pair

po = conv([1 2*omegac*zetac omegac^2],[1 2*omegad*zetad omegad^2]);
poles2 = roots(po);

K = place(A',C',poles2)';               % compute the Kalman gain K

% Computation of controller (observer + state feedback with integral action)

AR = [A-B*L-K*C -B*li; zeros(1,4) 0];
BRy = [K; -1];
BRr = [B*lr; 1];
CR = [-L -li];
DRy = 0;
DRr = lr;

Gr = -ss(AR,BRy,CR,DRy);                % transfer function from -y to u
```