# FRTN10 Exercise 4. Matlab, Loop Shaping, Preparations for Lab 1

### Matlab and Control System Toolbox

In Control System Toolbox, the basic data structure is the linear time-invariant (LTI) model. There are a large number of ways to create, manipulate and analyze models (see `help ctrlmodels`). Some operations are best done on the LTI system, and others directly on the matrices of the model.

**4.1** Consider the following state-space model describing the linearized dynamics of an inverted pendulum:

$$\begin{aligned} \frac{dx(t)}{dt} &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} x(t) + \begin{pmatrix} 1 \\ 0 \end{pmatrix} u(t) \\ y(t) &= \begin{pmatrix} 0 & 1 \end{pmatrix} x(t) \end{aligned} \tag{4.1}$$

**a.** Enter the system matrices $A$, $B$, and $C$ in Matlab and use the command `eig` to determine the poles of the system.

**b.** Define an LTI state-space model of the pendulum with the command `ss`. Use the commands `tf` and `zpk` to determine the transfer function of the system.

**c.** Zeros, poles and static gain of an LTI model are computed with the commands `zero`, `pole` and `dcgain`, respectively. Use these commands on the pendulum model. Compare the output of `pole` with the results from `eig`.

The command `tf` can be used to create an LTI transfer function model. This is done by specifying the coefficients of the numerator and denominator polynomials. E.g., to specify the transfer function $P(s) = 1/(2s + 1)$ you can use

```
>> P = tf(1, [2 1]);
```

Often it is more convenient to first define the Laplace variable `s` using either of the following (the second option is numerically more robust):

```
>> s = tf('s');
>> s = zpk('s');
```

and then define the transfer functions in terms of `s`, i.e.,

```
>> P = 1/(2*s+1);
```

To include a delay of 3 time units in $P(s)$ you can use

```
>> P = 1/(2*s+1)*exp(-3*s);
```

To display all properties of an LTI model and their respective values, type

```
>> get(P)
```

**4.2** Consider the process

$$P(s) = \frac{1}{s^2 + 0.6s + 1} e^{-0.5s} \tag{4.2}$$

**a.** Define the process as an LTI system in Matlab. Use the commands `step`, `nyquist`, and `bode` to plot time and frequency responses of the system. According to the Nyquist criterion, will the closed-loop system be stable if unit negative feedback is applied? (To permanently change the gain unit for Bode diagrams from dB to absolute values, try `ctrlpref`.)

**b.** Draw a singular value plot of $P(s)$ using the command `sigma` and read off the $L_2$ gain of the system. Verify your result using `norm(P,inf)`.

LTI systems can be interconnected in a number of ways. For example, you may add and multiply systems (or constants) to achieve parallel and series connections, respectively.

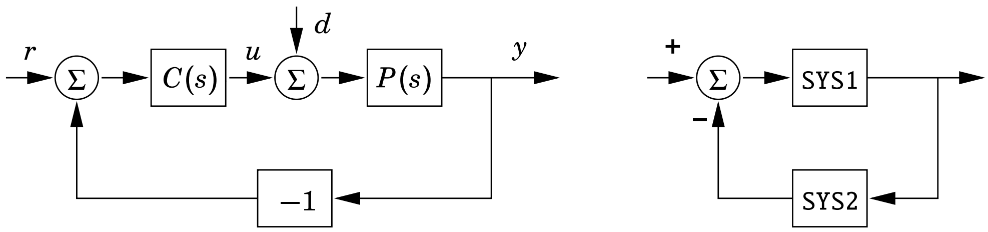


**Figure 4.1** A closed-loop control system.

To obtain the closed-loop transfer function for a feedback interconnection of systems, the best option is to use the command `feedback`. To obtain the transfer function in the block diagram to the right in Figure 4.1 you write `feedback(SYS1,SYS2)`. Note the sign conventions. It it also possible to directly calculate `SYS1/(1+SYS1*SYS2)`; however, it is numerically better to use `feedback`.

For example, to calculate $G_{yr}(s)$ for the system to the left in Figure 4.1, you identify that `SYS1` corresponds to $P(s)C(s)$ and `SYS2` corresponds to 1, so the command to use is `feedback(P*C,1)`. To calculate $G_{yd}(s)$, use `feedback(P,C)`, etc.

**4.3** Assume that the process (4.2) is controlled by a PD controller,

$$C(s) = K\big(1 + sT_d\big), \quad K = 0.8,\ T_d = 1.2$$

according to the block diagram to the left in Figure 4.1.

**a.** Define an LTI model of the controller $C(s)$ and find the amplitude and phase margins for the loop transfer function $L(s) = P(s)C(s)$. Use the command `margin`.

**b.** Calculate the closed-loop transfer functions from $r$ to $y$ (output response to reference change) and from $d$ to $y$ (output response to load disturbance) using `feedback`, and plot their respective step responses. What is the static gain from $r$ to $y$? Can the system reject a constant load disturbance?

**c.** (*) Introduce integral action in the controller and repeat **a.** and **b.**.

**Loop shaping, preparations for Lab 1**

Exercises 4.4 and 4.5 are preparatory exercises for Laboratory Session 1. In these exercises, we will design a feedback compensator using loop shaping for a process described by the transfer function

$$P(s) = \frac{15}{(s^2 + s + 1)(s + 3)}$$

**4.4** Define the process as an LTI system in Matlab. Use the commands `step`, `nyquist`, and `margin` to plot time and frequency responses of the system. According to the Nyquist criterion, will the closed-loop system be stable if unit negative feedback is applied? Verify your conclusion by explicitly calculating the poles of the closed-loop system.

The structure of the control system is given in Figure 4.2. In this exercise, we set $F(s) = 1$ and focus on designing a feedback controller $C(s)$ to achieve a fast enough, robust closed-loop system that can reject slow load disturbances. For our system, this translates to the following frequency-domain requirements:

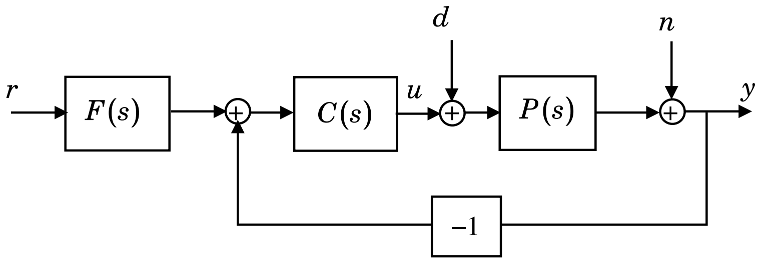


**Figure 4.2** Our control loop with reference signal $r$, load disturbance $d$, measurement noise $n$ and output $y$.

- Cross-over frequency: $\omega_c \approx 1.4$ rad/s.
- Low-frequency disturbance rejection: $|C(i\omega)P(i\omega)| \geq 25$ ( = 28 dB) for $\omega \leq 0.01$ rad/s.
- Stability margins: $\varphi_m \geq 45°$, $A_m \geq 2$ ( = 6 dB), $M_s \leq 2$.

**4.5** Design a feedback compensator that achieves the above specifications. A recommended workflow is the following:

1. Start with a simple $P$ controller, $C(s) = K$. Adjust the gain to achieve $\omega_c \approx 1.4$ rad/s and evaluate the other requirements.
2. Add a compensator to fulfill the disturbance rejection requirement. Evaluate the other requirements, and, if necessary, adjust the other controller parameters.
3. Add a second compensator to fulfill the stability margin requirements. Evaluate the other requirements, and, if necessary, go back and adjust the other controller parameters.

For each step, also plot the closed-loop step responses of $G_{yr}(s)$ and $G_{yd}(s)$. The following Matlab script can be used as a starting point:

```
s = zpk('s');               % define Laplace variable
P = 15/((s^2+s+1)*(s+3)); % define process transfer function

% 1. Try a simple P controller
K = 0.5;
C = K;                      % define controller transfer function
figure(1)
margin(P*C)                 % plot loop gain with stability margins
figure(2)
Gyr = feedback(P*C,1);      % calculate closed loop r -> y
Gyd = feedback(P,C);        % calculate closed loop d -> y
step(Gyr,Gyd)               % plot closed-loop step responses
legend('Gyr','Gyd')
S = feedback(1,P*C);        % calculate sensitivity function
Ms = norm(S,inf)            % calculate maximum sensitivity
```

Write down the transfer function of your final controller $C(s)$ and bring this to the lab session. Also note the resulting stability margins $\varphi_m$, $A_m$ and $M_s$.

**4.6** (*) Consider the control system in Figure 4.2, where the plant is described by

$$P(s) = \frac{1}{(s+1)(s+0.02)}$$

and $F(s) = 1$. An unexperienced engineer has designed the controller

$$C(s) = \frac{(s+a)}{s}$$

with $a = 0.02$, but the resulting control system reacts extremely slowly to step disturbances in $d$. The reason is that the slow pole in $-0.02$ is canceled by the controller zero. The Bode diagrams of the plant, the controller, and the open-loop system are shown in Figure 4.3.

**a.** The load disturbance $d$ is typically most significant at low frequencies, so we are interested in keeping the magnitude of the transfer function $G_{yd}$ from $d$ to $y$ significantly smaller than 1 in a frequency range $[0, \omega_b]$. What is (approximately) $\omega_b$ if you use the given controller? Use the Bode diagram in Figure 4.3.

**b.** To reject the disturbance $d$ faster, $\omega_b$ should be increased. For noise reasons, we want the cross-over frequency of the system to be the same.

How should the value of $a$ in the controller be changed to achieve this? Motivate your design by showing that:

- The range $[0, \omega_b]$ where you get good disturbance rejection of $d$ is increased.
- The cross-over frequency of the system is still approximately the same.

Exact proofs are not required; some Bode-diagram reasoning will do.

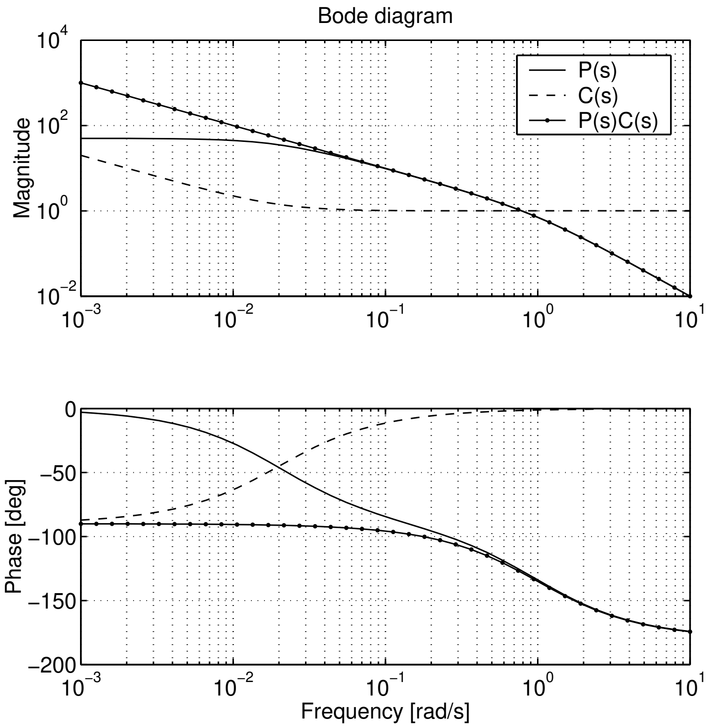


**Figure 4.3** The Bode diagrams of $P(s)$, $C(s)$ and the open loop $P(s)C(s)$ when $a = 0.02$.

# Solutions to Exercise 4. Matlab, Loop Shaping, Preparations for Lab 1

**4.1** **a.**

```
>> A = [0 1; 1 0];
>> B = [1 0]';
>> C = [0 1];
>> D = 0;
>> eig(A)
ans =
    -1
     1
```

**b.**

```
>> sys = ss(A,B,C,D);
>> tf(sys)
ans =
     1
  -------
  s^2 - 1
>> zpk(sys)
ans =
       1
  -----------
  (s+1) (s-1)
```

**c.**

```
>> zero(sys)
ans =
Empty matrix: 0-by-1
```

I.e., the system has no zeros.

```
>> pole(sys)
ans =
    -1
     1
```

These are the same as you computed with `eig(A)`.

```
>> dcgain(sys)
ans =
    -1
```

Note that `dcgain` does not check whether the system is stable; it just computes $G(0) = C(-A)^{-1}B + D$.

**4.2**

```
>> s = zpk('s');
>> P = 1/(s^2+0.6*s+1)*exp(-0.5*s)
P =
                        1
exp(-0.5*s) * ---------------
              s^2 + 0.6 s + 1
>> step(P)
>> nyquist(P)
>> bode(P)
>> pzmap(P)
```

As seen in the pole-zero map, the *open-loop* system is stable, as also indicated by the step response. The Bode and Nyquist plots both show that the *closed-loop* system will be stable as well.

**4.3** **a.**
```
>> C = 0.8*(1+s*1.2);
>> margin(P*C)
```
The amplitude margin is 2.79, and the phase margin is 50.7°.

**b.**
```
>> Gyr = feedback(P*C,1);
>> step(Gyr)
>> dcgain(Gyr)
ans =
    0.4444
>> Gyd = feedback(P,C);
>> step(Gyd)
>> dcgain(Gyd)
ans =
    0.5556
```
The static gain from $r$ to $y$ is 0.44, but it should ideally be 1. The system cannot reject a constant disturbance; the magnitude of the static error is 0.56.

**c.** Choosing e.g. $T_i = 1$, we obtain a PID controller as

```
>> C = 0.8*(1+s*1.2+1/s);
>> margin(P*C)
>> step(Gyr)
>> dcgain(Gyr)
ans =
     1
>> Gyd = feedback(P,C);
>> step(Gyd)
>> dcgain(Gyd)
ans =
     0
```

The closed loop now has the correct static gain and can reject constant load disturbances.

**4.4** *No solution provided.*

**4.5** *No solution provided.*

**4.6** **a.** The transfer function from $d$ to $y$ is given by

$$G_{yd}(s) = \frac{P}{1+PC}$$

For frequencies $\omega \leq 0.5$ (approximately), it can be seen in the Bode diagram that both $|P(i\omega)| \gg 1$ and $|P(i\omega)C(i\omega)| \gg 1$. Therefore $G_{yd}(s) \approx \frac{1}{C}$, and $|C(i\omega)|$ becomes larger than 1 for frequencies $\omega \leq 0.02$.

The magnitude of $G_{yd}(s)$ is thus smaller than 1 in a frequency range of approximately $[0, 0.02]$, thus $\omega_b = 0.02$ rad/s.

This can also be seen as the frequency point where $|PC|$ becomes larger than $|P|$ in the bode diagram.

**b.** To increase $\omega_b$, we would like to increase the gain of $C(i\omega)$ for frequencies $\omega > 0.02$. This is done by moving the zero in $C(s)$ (the break-point in the Bode diagram) from 0.02 to some higher frequency.

Choose, e.g., $a = 0.1$. Motivation:

- As $G_{yd}(s) \approx \frac{1}{C}$, and $|C(i\omega)|$ now becomes larger than 1 for frequencies $\omega \leq 0.1$, $\omega_b$ has been increased to about 0.1.
- The cross-over frequency for $a = 0.02$ is $\omega_c \approx 0.8$. As this frequency is higher than the new break-point 0.1, $C(i\omega_c) \approx 1$ still holds $\Rightarrow$ the cross-over frequency stays the same.