Lecture 12

The Youla Parametrization. Internal Model Control

In the last part of the course, we will use numerical optimization to design optimal controllers for linear systems. The general framework (already introduced in Lecture 9) is shown in Figure 12.1. As we recall, the "Plant" block is a generalized process description that includes all dynamics and input–output relations that are relevant for the design problem at hand. The vector w contains all exogenous signals, including disturbances and reference signals. The vector z contains all performance outputs (error signals, control signals, etc.) and should in general be minimized. The "Controller" block takes a vector y as input that contains any signals that the controller could use, including measurements and reference signals. Finally the vector u contains the control signals as well as signals that should be included in the performance vector z.

In this lecture we will characterize *all stabilizing controllers* for a generalized plant using the *Youla parametrization* (also known as the *Q parametrization*). In Lecture 13 we will optimize over the family of stabilizing controllers, and in Lecture 14 we will study model reduction of the obtained controllers. The Youla parametrization is closely connected to the design method known as *internal model control* (IMC), which will be studied at the end of this lecture.

12.1 The Youla parameterization

The basic idea of this lecture is illustrated in Figure 12.2. Even in the simplest case, with a stable SISO plant P, it is nontrivial to know how to adjust the controller C to achieve the desired properties of the closed-loop system, $\frac{PC}{1+PC}$. By introducing the parameter

$$Q = \frac{C}{1+PC},$$

the closed-loop system can instead be written as

$$\frac{PC}{1+PC} = PQ.$$

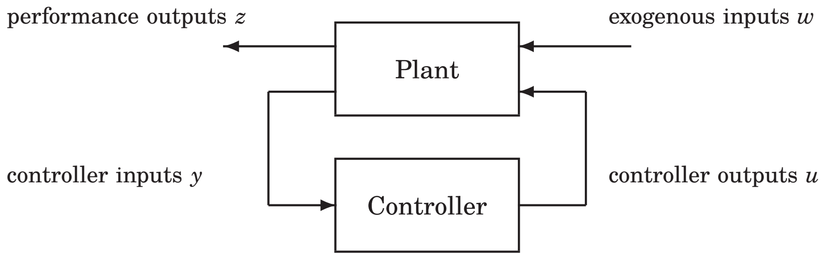


Figure 12.1 A general controller optimization framework.

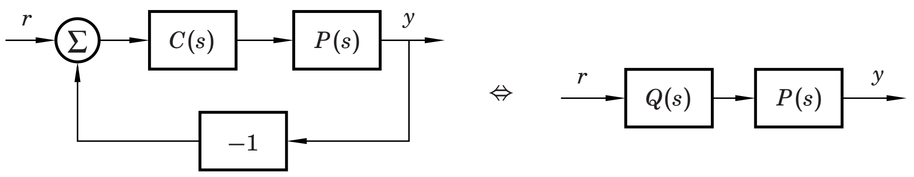


Figure 12.2 The basic idea of the Youla parametrization and IMC.

In the latter expression, the closed loop is linear in Q, and it is much easier to design Q to get the desired closed loop. Then one can simply obtain the controller as

$$C = \frac{Q}{1 - QP}.$$

The simple idea above can be generalized to MIMO systems. In Figure 12.1, let the controller transfer function be C (here the positive feedback convention is used) and let the stable plant P be described by the transfer matrix

$$P = \begin{bmatrix} P_{zw} & P_{zu} \\ P_{yw} & P_{yu} \end{bmatrix}.$$

The closed-loop transfer function from w to z can then be calculated as

$$\begin{aligned} u &= Cy \\ y &= P_{yu}u + P_{yw}w = P_{yu}Cy + P_{yw}w \quad \Rightarrow \quad y = (I - P_{yu}C)^{-1}P_{yw}w \\ z &= P_{zw}w + P_{zu}u = P_{zw}w + P_{zu}Cy = \underbrace{\left(P_{zw} + P_{zu}C(I - P_{yu}C)^{-1}P_{yw}\right)}_{G_{zw}}w \end{aligned}$$

Analogous to the SISO case above, we can introduce the parameter

$$Q = C\left(I - P_{yu}C\right)^{-1}.$$

The closed-loop transfer matrix from w to z then becomes an affine function of Q:

$$G_{zw} = P_{zw} + P_{zu}QP_{yw}.$$

Once Q has been designed, the controller can be retrieved via

$$C = \left(I + QP_{yu}\right)^{-1}Q.$$

Note that Q has the same number of inputs and outputs as C.

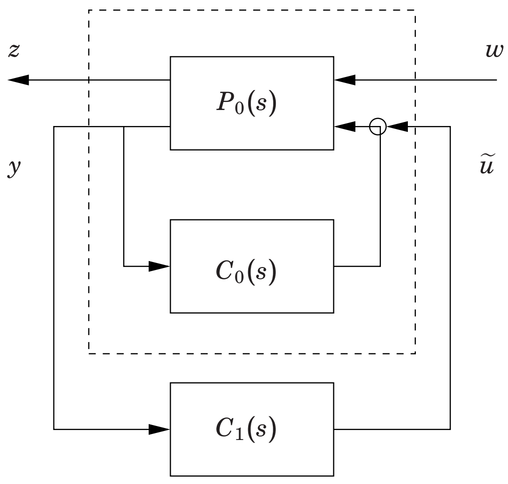
Characterization of all stabilizing controllers

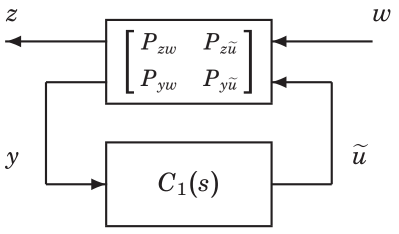
The simple idea above actually gives a straightforward way to characterize all stabilizing controllers for a stable plant $P(s)$. If $P(s)$ is stable, then stability of $Q(s)$ implies stability of the closed loop $G_{zw}(s) = P_{zw}(s) + P_{zu}(s)Q(s)P_{yw}(s)$. Conversely, if $Q(s)$ is unstable, then the closed loop will also be unstable. Hence, *all stabilizing controllers* for $P(s)$ are given by

$$C(s) = \left(I + Q(s)P_{yu}(s)\right)^{-1}Q(s)$$

where $Q(s)$ is an *arbitrary stable transfer function*. This key property will be exploited for optimization in the next lecture.

For unstable plants, the characterization is slightly more complicated. The plant must first be stabilized using some controller. Then the previous argument can be applied to the stabilized system. The procedure however also involves calculating a new generalized plant description for the stabilized system and possibly also translating the design specifications. The method is illustrated in Figure 12.3, where the unstable plant $P_0(s)$ is first stabilized using the inner controller $C_0(s)$. The stabilized plant is then controlled by the outer controller $C_1(s)$. Once $C_1(s)$ has been designed, the total controller is given by $C(s) = C_0(s) + C_1(s)$.





z
w
$\begin{bmatrix} P_{zw} & P_{z\tilde{u}} \\ P_{yw} & P_{y\tilde{u}} \end{bmatrix}$
y
$\tilde{u}$
$C_1(s)$

Figure 12.3 Unstable plants can be handled by an inner stabilizing controller $C_0(s)$ (left). The stabilized plant is then controlled by an outer controller $C_1(s)$ (right).

Example 12.1—DC-motor

For the simple control loop in Figure 12.4, assume that we want to optimize the closed-loop transfer matrix from (w_1, w_2) to (z_1, z_2),

$$G_{zw}(s) = \begin{bmatrix} \frac{P}{1-PC} & \frac{PC}{1-PC} \\ \frac{1}{1-PC} & \frac{C}{1-PC} \end{bmatrix}$$

when the plant $P = \frac{20}{s(s+1)}$ describes a simple DC-servo. As a first step, we would like to characterize all stable closed-loop maps G_{zw} using the Youla parametrization. The plant is not stable, so according to Figure 12.3 we write the controller as

$$C = C_0 + C_1,$$

where $C_0 = -1$ stabilizes the plant. We then have the situation shown in Figure 12.5 (left). The output signals from the stabilized plant are given by

$$\begin{aligned} z_1 &= Pw_1 - Py + P\tilde{u} \\ z_2 &= w_1 - y + \tilde{u} \\ y &= Pw_1 + w_2 - Py + P\tilde{u} \end{aligned}$$

Solving for the output signals we obtain

$$\begin{aligned} z_1 &= \tfrac{P}{1+P}w_1 - \tfrac{P}{1+P}w_2 + \tfrac{P}{1+P}\tilde{u} \\ z_2 &= \tfrac{1}{1+P}w_1 - \tfrac{1}{1+P}w_2 + \tfrac{1}{1+P}\tilde{u} \\ y &= \tfrac{P}{1+P}w_1 + \tfrac{1}{1+P}w_2 + \tfrac{P}{1+P}\tilde{u} \end{aligned}$$

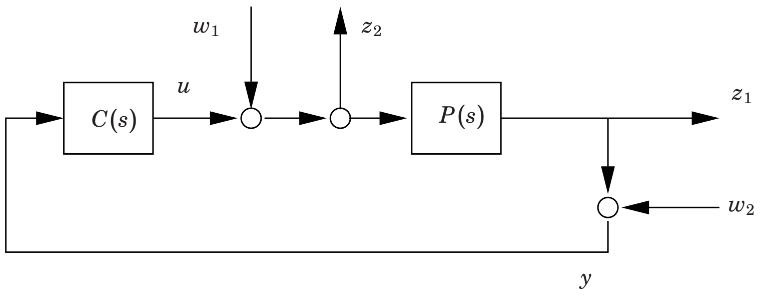


Figure 12.4 Simple control loop for control of a DC-motor.

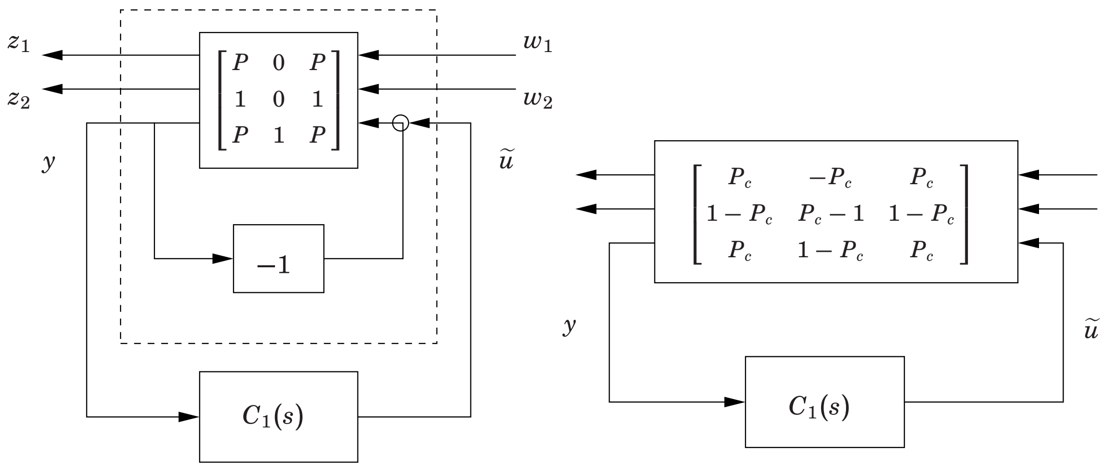

z_1
z_2
y
w_1
w_2
$\tilde{u}$
$\begin{bmatrix} P & 0 & P \\ 1 & 0 & 1 \\ P & 1 & P \end{bmatrix}$
-1
$C_1(s)$
$\begin{bmatrix} P_c & -P_c & P_c \\ 1-P_c & P_c-1 & 1-P_c \\ P_c & 1-P_c & P_c \end{bmatrix}$
y
$\tilde{u}$
$C_1(s)$

Figure 12.5 Redrawn diagram for stabilized DC-motor

Introducing

$$P_c = \frac{P}{1+P} = \frac{20}{s^2+s+20}$$

all stable closed-loop maps are found to be characterized by (see Figure 12.5 (right))

$$G_{zw}(s) = \underbrace{\begin{bmatrix} P_c & -P_c \\ 1-P_c & P_c-1 \end{bmatrix}}_{P_{zw}} + \underbrace{\begin{bmatrix} P_c \\ 1-P_c \end{bmatrix}}_{P_{z\tilde{u}}} Q \underbrace{\begin{bmatrix} P_c & 1-P_c \end{bmatrix}}_{P_{yw}},$$

where Q is an arbitrary scalar, stable transfer function. Once Q has been determined, the complete controller is given by

$$C = C_0 + C_1 = -1 + \frac{Q}{1+QP_c}.$$

□

12.2 Internal Model Control (IMC)

The Youla parametrization of all stabilizing controllers is suitable for optimization-based design. For simpler cases, like stable SISO plants, essentially the same idea can be used to design controllers by hand. The principle is called *internal model control* (IMC) and is illustrated in Figure 12.6 (a). A plant model $P_m(s)$ is simulated in parallel with the real plant $P(s)$, both driven by the same control signal $u(t)$. Feedback to the controller $Q(s)$ is only done on the difference between the plant output $y(t)$ and the model output. Note that negative feedback is normally assumed in conjunction with IMC. Redrawing the block diagram we obtain Figure 12.6 (b), which clearly shows the connection to the Youla parametrization. After designing $Q(s)$, the controller is given by

$$C(s) = \frac{Q(s)}{1-Q(s)P_m(s)}$$

With $P(s) = P_m(s)$, the transfer function from r to y becomes $P(s)Q(s)$.

IMC design rules

IMC is in itself a well-developed controller design method. Here we will only touch upon some basic design rules. Viewing the closed-loop system as $P(s)Q(s) = T(s)$, the problem of designing Q is very similar to feedforward design (see Lecture 4). For perfect reference following, one would like to put $Q(s) = P(s)^{-1}$. This leads to $T(s) = 1$, which we know is impossible to achieve for several reasons. We rather put forward the following practical design rules for IMC:

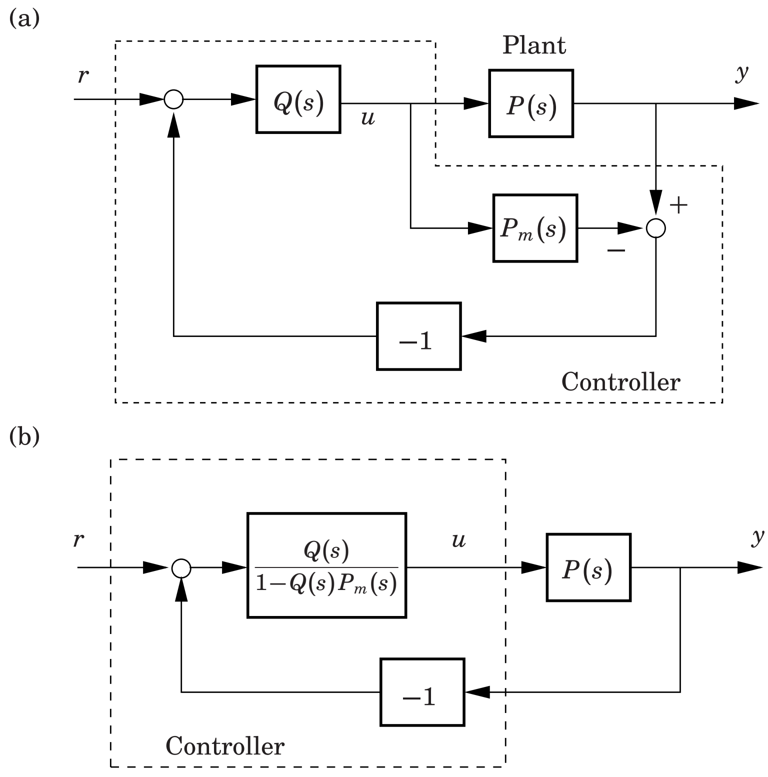


Figure 12.6 Internal Model Control (IMC)

- If $P(s)$ is strictly proper, the inverse would have more zeros than poles. Instead, one can choose
$$Q(s) = \frac{1}{(\lambda s + 1)^n} P(s)^{-1},$$
where n is large enough to make Q proper. The parameter λ determines the speed of the closed-loop system (i.e., the bandwidth of $T(s)$).
- If $P(s)$ has unstable zeros, the inverse would be unstable. Here two different options are available:
  - Remove every unstable factor $(-\beta s + 1)$ from the plant numerator before inverting.
  - Replace every unstable factor $(-\beta s + 1)$ with $(\beta s + 1)$ (mirroring the zero in the imaginary axis). With this option, only the phase is modified, not the amplitude function.
- If $P(s)$ includes a time delay, its inverse would have to predict the future, i.e., it would be non-causal. Instead, the time delay is removed before inverting.

Example 12.2—IMC design for first-order plant
Given the stable first-order plant
$$P(s) = \frac{1}{Ts + 1}$$
we need one extra pole in Q to make it proper:
$$Q(s) = \frac{1}{\lambda s + 1} P(s)^{-1} = \frac{Ts + 1}{\lambda s + 1}$$
The controller then becomes
$$C(s) = \frac{Q(s)}{1 - Q(s)P(s)} = \frac{\frac{Ts+1}{\lambda s+1}}{1 - \frac{1}{\lambda s+1}} = \frac{T}{\lambda}\left(1 + \frac{1}{sT}\right)$$
This is actually a PI controller with the integral time $T_i = T$. This way of tuning a PI controller is known as *lambda tuning* and is very popular within the process industry. □

EXAMPLE 12.3—IMC DESIGN FOR NON-MINIMUM PHASE PLANT
Given the non-minimum phase plant

$$P(s) = \frac{-\beta s + 1}{Ts + 1}$$

we follow the second rule above and mirror the unstable plant zero when forming Q:

$$Q(s) = \frac{(-\beta s + 1)}{(\beta s + 1)} P(s)^{-1} = \frac{Ts + 1}{\beta s + 1}$$

Using this $Q(s)$ to control the time-delay plant $P(s)$ then gives the controller

$$C(s) = \frac{Q(s)}{1 - Q(s)P(s)} = \frac{\frac{Ts+1}{\beta s+1}}{1 - \frac{(-\beta s+1)}{(\beta s+1)}} = \frac{T}{2\beta}\left(1 + \frac{1}{sT}\right)$$

This is again a PI controller with integral time $T_i = T$. It is interesting to note that the gain is adjusted in accordance with the fundamental limitation imposed by the RHP zero in $1/\beta$. □

IMC design for deadtime processes

Deadtime processes are common in industry and are difficult to control due to their non-minimum-phase behaviour. In 1957, inventor Otto J.M. Smith proposed a prediction-based method to cope with long time delays, nowadays known as the Smith predictor. We will see that his solution follows the IMC principle. Let the deadtime process be given by

$$P(s) = P_0(s)e^{-sL},$$

where $P_0(s)$ is the delay-free part of the plant and L is the (known) delay. Let

$$C_0 = Q/(1 - QP_0)$$

be a controller designed for the delay-free plant P_0. Solving for Q gives

$$Q = C_0/(1 + C_0P_0).$$

Using this Q for the deadtime process, the controller then becomes

$$\begin{aligned} C(s) &= \frac{Q(s)}{1 - Q(s)P_0(s)e^{-sL}} = \frac{C_0(s)/(1 + C_0(s)P_0(s))}{1 - e^{-sL}P_0(s)C_0(s)/(1 + C_0(s)P_0(s))} \\ &= \frac{C_0(s)}{1 + (1 - e^{-sL})C_0(s)P_0(s)} \end{aligned}$$

The principle of the Smith predictor is illustrated in Figure 12.7. The predictor uses an internal model of the process (P_m with the delay and P_0 without the delay). Ideally y and y_m cancel each other and only feedback from y_0 "without delay" is used. If the model is perfect, i.e., if $P_m = P$, the transfer from r to y becomes

$$Y(s) = \frac{C_0(s)P_0(s)}{1 + C_0(s)P_0(s)} e^{-sL} R(s).$$

The closed loop hence behaves just like if the delay-free plant P_0 was controlled by C_0, except that the response is delayed by L. In reality, model errors and disturbances will limit the performance. Also, the fundamental limitation associated with the time delay cannot be ignored. According to the rule of thumb given in Lecture 7, we should not aim for a cross-over frequency larger than $1.6/L$.

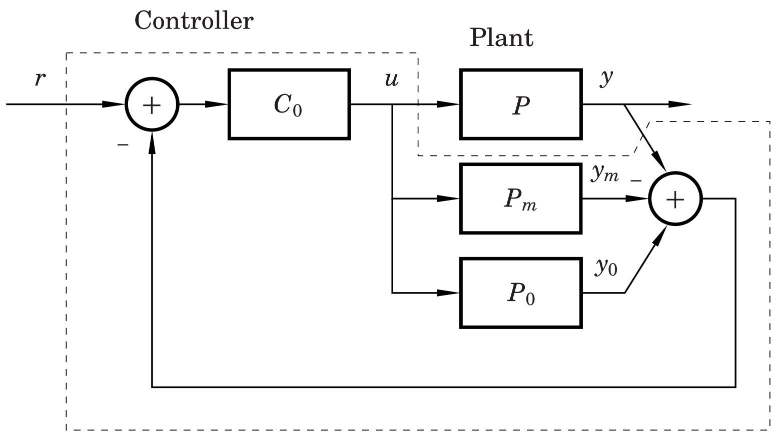


Figure 12.7 The Smith predictor

Example 12.4—Deadtime compensation
Consider control of the first-order plus deadtime process

$$P(s) = \frac{1}{s+1}e^{-s}.$$

As a nominal controller for the delay-free plant $P_0(s) = \frac{1}{1+s}$, a PI controller has been designed as

$$C_0(s) = K\left(1 + \frac{1}{s}\right).$$

Simulation of the controller on the deadtime process with $K = 0.4$ and $K = 1.0$ respectively is shown in Figure 12.8. It is seen that the deadtime makes the performance deteriorate as the controller gain is increased. (It can be calculated that the loop goes unstable for $K > 2.26$.) Keeping $K = 1$ and introducing a Smith predictor, the result can be dramatically improved as seen in Figure 12.9 (a). In this simulation a perfect plant model was assumed, which is of course unrealistic. Changing the real plant to

$$P(s) = \frac{1}{s+0.8}e^{-1.2s}$$

and keeping the same plant model as before, the performance deteriorates as shown in Figure 12.9 (b). The robustness will be worse the higher the bandwidth of the closed-loop system is compared to the time delay. □

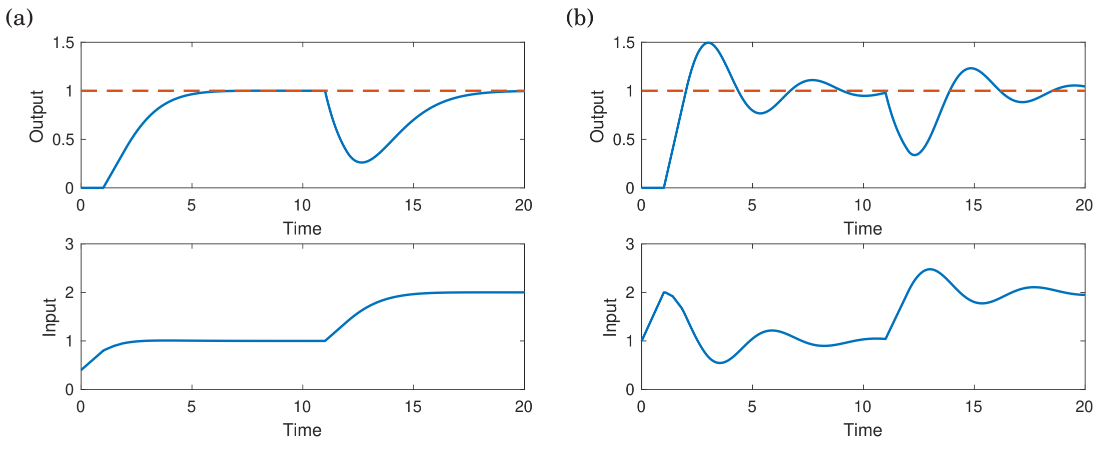


Figure 12.8 PI control of a deadtime process without Smith predictor: (a) $K = 0.4$, (b) $K = 1.0$.

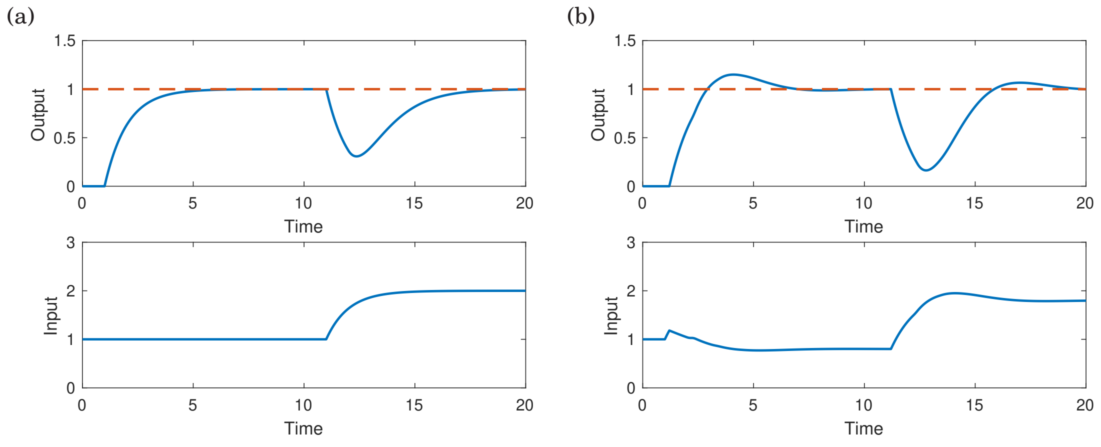


Figure 12.9 PI control of a deadtime process with Smith predictor: (a) Perfect model, (b) Imperfect model.