

❖ 6. Design and Tuning of Single-Loop Control Systems

We will go through a whole bundle of tuning methods. We only need to "pick" three numbers for a PID controller, but this is one of the most confusing parts of learning control. Different tuning techniques give similar but not identical results. There are no "best" or "absolutely correct" answers. The methods all have pros and cons, and working together, they complement each other. We need to make proper selection and sound judgment—very true to the act (and art) of design.

What are we up to?

- Tune a controller with empirical relations
- Tune a controller with internal model control relations

6.1 Tuning controllers with empirical relations

Let's presume that we have selected the valves, transducers and even installed a controller. We now need to determine the controller settings—a practice which we call tuning a controller. Trial-and-error tuning can be extremely time consuming (and dumb!), to the extent that it may not be done. A large distillation column can take hours to reach steady state. A chemical reactor may not reach steady state at all if you have a reactor "runaway." Some systems are unstable at high and low feedback gains; they are stable only in some intermediate range. These are reasons why we have to go through all the theories to learn how to design and tune a controller with well educated (or so we hope) guesses.

Empirical tuning roughly involves doing either an open-loop or a closed-loop experiment, and fitting the response to a model. The controller gains are calculated on the basis of this fitted function and some empirical relations. When we use empirical tuning relations, we cannot dictate system dynamic response specifications. The controller settings are seldom optimal and most often require field tuning after installation to meet more precise dynamic response specifications. Empirical tuning may not be appealing from a theoretical viewpoint, but it gives us a quick-and-dirty starting point. Two remarks before we begin.

- Most empirical tuning relations that we use here are based on open-loop data fitted to a **first order with dead time** transfer function. This feature is unique to process engineering where most units are self-regulating. The dead time is either an approximation of multi-stage processes or a result of transport lag in the measurement. With large uncertainties and the need for field tuning, models more elaborate than the first order with dead time function are usually not warranted with empirical tuning.
- Some empirical tuning relations, such as Cohen and Coon, are developed to achieve a one-quarter decay ratio response in handling disturbances. When we apply the settings of these relations to a servo problem, it tends to be very oscillatory and is not what one considers as slightly underdamped.¹ The controller design depends on the specific problem at hand. We certainly need to know how to tune a controller after using empirical tuning relations to select the initial settings.²

¹ If we assume that an oscillatory system response can be fitted to a second order underdamped function. With Eq. (3-29), we can calculate that with a decay ratio of 0.25, the damping ratio ζ is 0.215, and the maximum percent overshoot is 50%, which is *not* insignificant. (These values came from Review Problem 4 back in Chapter 5.)

² By and large, a quarter decay ratio response is acceptable for disturbances but not desirable for set point changes. Theoretically, we can pick any decay ratio of our liking. Recall Section 2.7 (p. 2-17) that the position of the closed-loop pole lies on a line governed by $\theta = \cos^{-1}\zeta$. In the next chapter, we will locate the pole position on a root locus plot based on a given damping ratio.

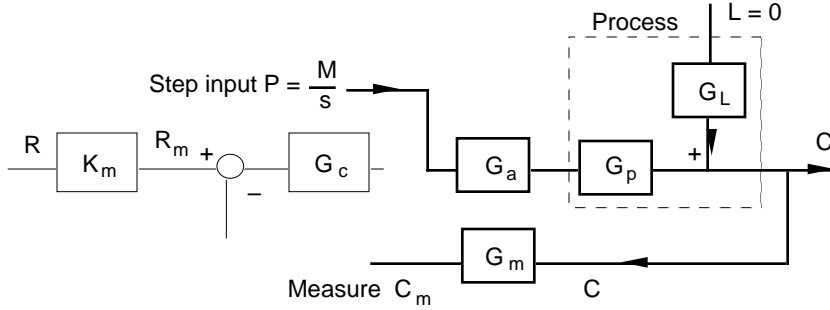


Figure 6.1. Block diagram illustration of an open-loop step test.

6.1.1 Controller settings based on process reaction curve

To make use of empirical tuning relations, one approach is to obtain the so-called *process reaction curve*. We disable the controller and introduce a step change to the actuator. We then measure the **open-loop step response**. This practice can simply be called an open-loop step test. Although we disconnect the controller in the schematic diagram (Fig. 6.1), we usually only need to turn the controller to the “manual” mode in reality. As shown in the block diagram, what we measure is a lumped response, representing the dynamics of the blocks G_a , G_p , and G_m . We denote the lumped function as G_{PRC} , the process reaction curve function:

$$G_{PRC} = \frac{C_m}{P} = G_a G_p G_m \quad (6-1)$$

From the perspective of doing the experiment, we need the actuator to effect a change in the manipulated variable and the sensor to measure the response.

The measurement of G_{PRC} is how we may design a system if we know little about our process and are incapable of constructing a model (What excuse!). Even if we know what the functions G_a and G_m should be, we do not need them since the controller empirical tuning relations were developed for the lumped function G_{PRC} . On the other hand, if we know precisely what the functions G_a , G_p and G_m are, we may use them to derive G_{PRC} as a reduced-order approximation of the product of $G_a G_p G_m$.

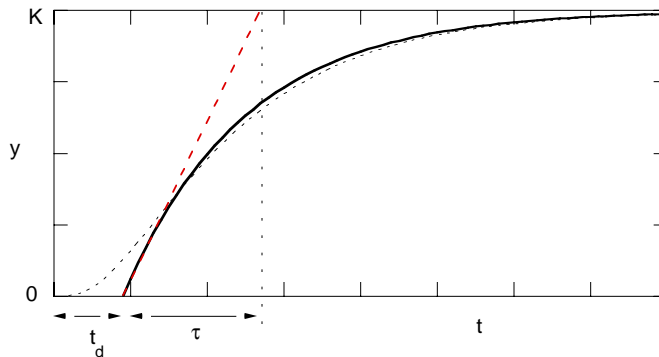


Figure 6.2. Illustration of fitting Eq. (6-2, solid curve) to open-loop step test data representative of self-regulating and multi-capacity processes (dotted curve). The time constant estimation shown here is based on the initial slope and a visual estimation of dead time. The Ziegler-Nichols tuning relation (Table 6.1) also uses the slope through the inflection point of the data (not shown). Alternative estimation methods are provided on our *Web Support*.

The real time data (the process reaction curve) in most processing unit operations take the form of a sigmoidal curve, which is fitted to a first order with dead time function (Fig. 6.2):¹

$$G_{\text{PRC}} = \frac{C_m}{P} \approx \frac{K e^{-t_d s}}{\tau s + 1} \quad (6-2)$$

One reason why this approximation works is that process unit operations are generally open-loop stable, and many are multi-capacity in nature. Reminder: Underdamped response of the system is due to the controller, which is taken out in the open-loop step test.

Using the first order with dead time function, we can go ahead and determine the controller settings with empirical tuning relations. The most common ones are the **Ziegler-Nichols** relations. In process unit operation applications, we can also use the **Cohen and Coon** or the **Ciancone and Marlin** relations. These relations are listed in the Table of Tuning Relations (Table 6.1).

6.1.2 Minimum error integral criteria

The open-loop test response fitted to a first order with dead time function G_{PRC} can be applied to other tuning relations. One such possibility is a set of relations derived from the minimization of error integrals. Here, we just provide the basic idea behind the use of error integrals.

To derive the tuning equations, we would use the theoretical time-domain closed-loop system response as opposed to a single quantity, such as the decay ratio. The time-domain solution is dependent on the type of controller and the nature of input (set point or disturbance changes) and, in our case, a “process” function that is first order with dead time. We can also calculate the error—the difference between the set point and the controlled variable. We then find controller settings which may minimize the error over time (the error integral), using for instance, Lagrange multipliers as in introductory calculus. Of course, we are not doing the work; the actual analysis is better left for a course in optimal control.

There are different ways to define the error function to be minimized. A few possibilities are as follows:

- (1) Integral of the square error (ISE)

$$\text{ISE} = \int_0^{\infty} [e'(t)]^2 dt \quad (6-3)$$

The ISE magnifies large errors—squaring a small number (< 1) makes it even smaller. Thus minimization of this integral should help to suppress large, initial errors. The resulting controller setting tends to have a high proportional gain and the system is very underdamped.

- (2) Integral of the absolute error (IAE)

$$\text{IAE} = \int_0^{\infty} |e'(t)| dt \quad (6-4)$$

The IAE simply integrates the absolute value and puts equal weight to large and small errors.

- (3) Integral of time-weighted absolute error (ITAE)

¹ The first order function with dead time is only appropriate for self-regulating and multi-capacity processes. In other controller design methods, we should choose different functions to fit the open-loop test response.

$$\text{ITAE} = \int_0^{\infty} t |e'(t)| dt \quad (6-5)$$

The time weighting function puts a heavy penalty on errors that persist for long periods of time. This weighting function also helps to derive controller settings which allow for low settling times.

Before we move on, a few comments and reminders:

- As far as we are concerned, using the error integral criteria is just another empirical method. We are simply using the results of minimization obtained by other people, not to mention that the first order with dead time function is from an open-loop test.
- The controller setting is different depending on which error integral we minimize. Set point and disturbance inputs have different differential equations, and since the optimization calculation depends on the time-domain solution, the result will depend on the type of input. The closed-loop poles are the same, but the zeros, which affect the time-independent coefficients, are not.
- The time integral is from $t = 0$ to $t = \infty$, and we can only minimize it if it is bounded. In other words, we cannot minimize the integral if there is a steady state error. Only PI and PID controllers are applicable to this design method.¹
- Theoretically, we can minimize the integral using other criteria. On the whole, the controller settings based on minimizing ITAE provide the most conservative controller design, and are highly recommended. This is the only set of tuning relations included in Table 6.1.

6.1.3 Ziegler-Nichols ultimate-cycle method

This empirical method is based on **closed-loop testing** (also called on-line tuning) of processes which are inherently stable, but where the system may become unstable. We use only proportional control in the experiment. If it is not possible to disable the integral and derivative control modes, we set the integral time to its maximum value and the derivative time to its minimum. The proportional gain is slowly increased until the system begins to exhibit sustained oscillations with a given small step set point or load change. The proportional gain and period of oscillation at this point are the **ultimate gain**, K_{cu} , and **ultimate period**, T_u . These two quantities are used in a set of empirical tuning relations developed by Ziegler and Nichols—again listed in Table 6.1.

Two more comments:

- A preview: We can derive the ultimate gain and ultimate period (or frequency) with stability analyses. In Chapter 7, we use the substitution $s = j\omega$ in the closed-loop characteristic equation. In Chapter 8, we make use of what is called the Nyquist stability criterion and Bode plots.
- One may question the meaning of “sustained oscillations.” We may gather that the ultimate gain and ultimate period are associated with marginal stability—the instance when the system is just about to become unstable. Of course, we never want to push that far in real life. With large uncertainties involved in empirical tuning and field tuning, it is not necessary to have accurate measurements of K_{cu} and T_u . When we do an experiment, just increase the proportional gain until we achieve a fairly underdamped response.

¹ If you come across a proportional controller here, it is only possible if the derivation has ignored the steady state error, or shifted the reference such that the so-called offset is zero.

✎ **Example 5.7A:** What would be the PID controller settings for the dye mixing problem in Example 5.7 (p. 5-17)?

Based on what we have obtained in Example 5.7, if we did an open-loop experiment as suggested in Eq. (6-1), our step response would fit well to the function:

$$G_{\text{PRC}} = G_a G_p G_m = \frac{(0.8)(0.6)(2.6)e^{-0.725s}}{(4s+1)(0.2s+1)}$$

However, to use the empirical tuning relations, we need to fit the data to a first order function with dead time. Thus at this stage, we probably would have obtained the approximation:

$$G_{\text{PRC}} \approx \frac{1.25e^{-0.9s}}{4s+1}$$

Here, we assume that the data fitting allows us to recover the time constant of the dominant pole reasonably well, and the dead time is roughly 0.9 s. We are not adding exactly 0.2 to 0.725 as a way to emphasize that in reality, we would be doing data fitting and the result will vary. How good an approximation is depends very much on the relative differences in the time constants. (Try with MATLAB simulation to see how good the approximation is. For the numbers chosen in this example, it is easy to obtain a good approximation.)

Now, with Table 6.1,¹ we can calculate the following PID controller settings:

	K_c	τ_I	τ_D
Cohen-Coon	4.9	2.0	0.31
Ziegler-Nichols	4.3	1.8	0.45
ITAE (Set point)	2.8	3.1	0.31
Ciancone-Marlin (Set point)	1.2	4.4	0.07

All tuning relations provide different results. Generally, the Cohen and Coon relation has the largest proportional gain and the dynamic response tends to be the most underdamped. The Ciancone-Marlin relation provides the most conservative setting, and it uses a very small derivative time constant and a relatively large integral time constant. In a way, their correlation reflects a common industrial preference for PI controllers.

We'll see how they compare in time response simulations when we come back to this problem later in Example 5.7C. A point to be made is that empirical tuning is a very imprecise science. There is no reason to worry about the third significant figure in your tuning parameters. The calculation only serves to provide us with an initial setting with which we begin to do field or computational tuning.

¹ Really calculated with our M-file recipe.m, which can be found on our *Web Support*.

While the calculations in the last example may appear as simple plug-and-chug, we should take a closer look at the tuning relations. The Cohen and Coon equations for the proportional gain taken from Table 6.1 are:

$$\text{P:} \quad K_c K = \left(\frac{\tau}{t_d} + \frac{1}{3} \right) \quad (6-6)$$

$$\text{PI:} \quad K_c K = \left(0.9 \frac{\tau}{t_d} + \frac{1}{12} \right) \quad (6-7a)$$

$$\text{PID:} \quad K_c K = \left(\frac{4}{3} \frac{\tau}{t_d} + \frac{1}{4} \right) \quad (6-8a)$$

The choice of the proportional gain is affected by two quantities: the product $K_c K$, and the ratio of dead time to time constant, t_d/τ . It may not be obvious why the product $K_c K$ is important now, but we shall see how it arises from direct synthesis in the next section and appreciate how it helps determine system stability in Chapter 8.

Under circumstances where the dead time is relatively small, only the first term on the right is important in the three tuning equations. When dead time becomes larger (or τ/t_d smaller), we need to decrease the proportional gain, and this is how the tuning relations are constructed. When we add integral control, we need to decrease K_c . Indeed, in Eq. (6-7a), the τ/t_d term is decreased by 10%, and the constant term is reduced to 1/12. With the implementation of PID control, we can afford to have a larger K_c . This is reflected in (6-8a). We can make similar observations with the Ziegler-Nichols relations in Table 6.1. Furthermore, we may also see in Table 6.1 that if the dead time increases, we should raise the integral time constant.

Table 6.1. Table of tuning relations ¹**A. Tuning relations based on open-loop testing and response fitted to a first order with dead time function**

$$G_{\text{PRC}} = \frac{K e^{-t_d s}}{\tau s + 1}$$

Controller	Cohen-Coon	Ziegler-Nichols
P	$K_c K = \left(\frac{\tau}{t_d} + \frac{1}{3} \right) \quad (6-6)$	$K_c K = \frac{\tau}{t_d} \quad (6-9)$
PI	$K_c K = \left(0.9 \frac{\tau}{t_d} + \frac{1}{12} \right) \quad (6-7a)$	$K_c K = 0.9 \frac{\tau}{t_d} \quad (6-10a)$
	$\tau_I = t_d \frac{30 + 3(t_d/\tau)}{9 + 20(t_d/\tau)} \quad (6-7b)$	$\tau_I = 3.3 t_d \quad (6-10b)$
PID	$K_c K = \left(\frac{4}{3} \frac{\tau}{t_d} + \frac{1}{4} \right) \quad (6-8a)$	$K_c K = 1.2 \frac{\tau}{t_d} \quad (6-11a)$
	$\tau_I = t_d \frac{32 + 6(t_d/\tau)}{13 + 8(t_d/\tau)} \quad (6-8b)$	$\tau_I = 2 t_d \quad (6-11b)$
	$\tau_D = t_d \frac{4}{11 + 2(t_d/\tau)} \quad (6-8c)$	$\tau_D = 0.5 t_d \quad (6-11c)$

Minimum ITAE criterionFor **load** change:

$$K_c = \frac{a_1}{K} \left(\frac{\tau}{t_d} \right)^{b_1}, \quad \tau_I = \frac{\tau}{a_2} \left(\frac{t_d}{\tau} \right)^{b_2} \quad \text{and} \quad \tau_D = a_3 \tau \left(\frac{t_d}{\tau} \right)^{b_3} \quad (6-12)$$

Controller	a ₁	b ₁	a ₂	b ₂	a ₃	b ₃
PI	0.859	0.977	0.674	0.680	–	–
PID	1.357	0.947	0.842	0.738	0.381	0.995

¹ All formulas in Table 6.1, and the PID settings in Table 6.2 later, are implemented in the M-file recipe.m, available from our *Web Support*. The Ciancone and Marlin tuning relations are graphical, and we have omitted them from the tables. The correlation plots, explanations, and the interpolation calculations are provided by our M-file ciancone.m, which is also used by recipe.m.

For **set point** change:

$$K_c = \frac{a_1}{K} \left(\frac{\tau}{t_d} \right)^{b_1}, \quad \tau_I = \frac{\tau}{a_2 - b_2 (t_d/\tau)} \quad \text{and} \quad \tau_D = a_3 \tau \left(\frac{t_d}{\tau} \right)^{b_3}$$

(6-13)

Controller	a ₁	b ₁	a ₂	b ₂	a ₃	b ₃
PI	0.586	0.916	1.03	0.165	–	–
PID	0.965	0.855	0.796	0.147	0.308	0.929

B. Tuning relations based on closed-loop testing and the Ziegler-Nichols ultimate-gain (cycle) method with given ultimate proportional gain K_{cu} and ultimate period T_u .

Ziegler-Nichols ultimate-gain method

Controller						
P	$K_c = 0.5 K_{cu}$	(6-14)				
PI	$K_c = 0.455 K_{cu}$	(6-15a)				
	$\tau_I = 0.833 T_u$	(6-15b)				
PID	Quarter decay		Just a bit of overshoot		No overshoot	
	$K_c = 0.6 K_{cu}$	(6-16a)	$K_c = 0.33 K_{cu}$	(6-17a)	$K_c = 0.2 K_{cu}$	(6-18a)
	$\tau_I = 0.5 T_u$	(6-16b)	$\tau_I = 0.5 T_u$	(6-17b)	$\tau_I = 0.5 T_u$	(6-18b)
	$\tau_D = 0.125 T_u$	(6-16c)	$\tau_D = 0.333 T_u$	(6-17c)	$\tau_D = 0.333 T_u$	(6-18c)

6.2 Direct synthesis and internal model control

We now apply a different philosophy to controller design. Up until now, we have had a preconceived idea of what a controller should be, and we tune it until we have the desired system response. On the other hand, we can be more proactive: we define what our desired closed-loop response should be and design the controller accordingly. The resulting controller is not necessarily a PID controller. This is acceptable with computer based controllers since we are not restricted to off-the-shelf hardware.

In this chapter, however, our objective is more restricted. We will purposely choose simple cases and make simplifying assumptions such that the results are PID controllers. We will see how the method helps us select controller gains based on process parameters (*i.e.*, the process model). The method provides us with a more rational controller design than the empirical tuning relations. Since the result depends on the process model, this method is what we considered a **model-based** design.

6.2.1 Direct synthesis

We consider a servo problem (*i.e.*, $L = 0$), and set $G_m = G_a = 1$. The closed-loop function is the familiar

$$\frac{C}{R} = \frac{G_c G_p}{1 + G_c G_p} \quad (6-19)$$

which we now rearrange as

$$G_c = \frac{1}{G_p} \left[\frac{C/R}{1 - C/R} \right] \quad (6-20)$$

The implication is that if we define our desired system response C/R , we can derive the appropriate controller function for a specific process function G_p .

A couple of quick observations: First, G_c is the reciprocal of G_p . The poles of G_p are related to the zeros of G_c and vice versa—this is the basis of the so-called *pole-zero cancellation*.¹ Second, the choice of C/R is not entirely arbitrary; it must satisfy the closed-loop characteristic equation:

$$1 + G_c G_p = 1 + \left[\frac{C/R}{1 - C/R} \right] = 0 \quad (6-21)$$

From Eq. (6-20), it is immediately clear that we cannot have an ideal servo response where $C/R = 1$, which would require an infinite controller gain. Now Eq. (6-21) indicates that C/R cannot be some constant either. To satisfy (6-21), the closed-loop response C/R must be some function of s , meaning that the system cannot respond instantaneously and must have some finite response time.

Let's select a more realistic system response, say, a simple first-order function with unity steady state gain

$$\frac{C}{R} = \frac{1}{\tau_c s + 1} \quad (6-22)$$

¹ The controller function will take on a positive pole if the process function has a positive zero. It is not desirable to have an inherently unstable element in our control loop. This is an issue which internal model control will address.

where τ_c is the system time constant, a design parameter that we specify. The unity gain means that we should eliminate offset in the system. Substitution of Eq. (6-22) in (6-20) leads to the controller function:

$$G_c = \frac{1}{G_p} \left[\frac{1}{\tau_c s} \right] \quad (6-23)$$

The closed-loop characteristic equation, corresponding to Eq. (6-21), is

$$1 + \frac{1}{\tau_c s} = 0 \quad (6-24)$$

which really is $1 + \tau_c s = 0$ as dictated by (6-22). The closed-loop pole is at $s = -1/\tau_c$. This result is true no matter what G_p is—as long as we can physically build or program the controller on a computer. Since the system time constant τ_c is our design parameter, it appears that direct synthesis magically allows us to select whatever response time we want. Of course this cannot be the case in reality. There are physical limitations such as saturation.

✎ **Example 6.1:** Derive the controller function for a system with a **first order process** and a system response dictated by Eq. (6-22).

The process transfer function is $G_p = \frac{K_p}{\tau_p s + 1}$, and the controller function according to Eq. (6-23) is

$$G_c = \frac{(\tau_p s + 1)}{K_p} \frac{1}{\tau_c s} = \frac{\tau_p}{K_p \tau_c} \left(1 + \frac{1}{\tau_p s} \right) \quad (E6-1)$$

which is obviously a PI controller with $K_c = \tau_p/K_p \tau_c$, and $\tau_I = \tau_p$. Note that the proportional gain is inversely proportional to the process gain. Specification of a small system time constant τ_c also leads to a large proportional gain.

A reminder: the controller settings K_c and τ_I are governed by the process parameters and the system response, which we choose. The *one and only tuning parameter* is the system response time constant τ_c .

✎ **Example 6.2:** Derive the controller function for a system with a **second order overdamped process** and system response as dictated by Eq. (6-22).

The process transfer function is $G_p = \frac{K_p}{(\tau_1 s + 1)(\tau_2 s + 1)}$, and the controller function according to Eq. (6-23) is

$$G_c = \frac{(\tau_1 s + 1)(\tau_2 s + 1)}{K_p} \frac{1}{\tau_c s}.$$

We may see that this is a PID controller. Nevertheless, there are two ways to manipulate the function. One is to expand the terms in the numerator and factor out $(\tau_1 + \tau_2)$ to obtain

$$G_c = \frac{(\tau_1 + \tau_2)}{K_p \tau_c} \left[1 + \frac{1}{(\tau_1 + \tau_2)} \frac{1}{s} + \left(\frac{\tau_1 \tau_2}{\tau_1 + \tau_2} \right) s \right] \quad (E6-2)$$

The proportional gain, integral time and derivative time constants are provided by the respective terms in the transfer function. If you have trouble spotting them, they are summarized in Table 6.2.

The second approach is to consider the controller function as a series-PID such that we write

$$G_c = \frac{\tau_1}{K_p \tau_c} \left(1 + \frac{1}{\tau_1 s} \right) (\tau_2 s + 1), \text{ with } \tau_1 > \tau_2, \quad (\text{E6-3})$$

We can modify the derivative term to be the “real” derivative action as written in Eqs. (5-9a and b) on page 5-7.

Based on experience that the derivative time constant should be smaller than the integral time constant, we should pick the larger time constant as the integral time constant. Thus we select τ_1 to be the integral time constant and τ_2 the derivative time constant. In the limit $\tau_1 \gg \tau_2$, both arrangements (E6-2 and 3) of the controller function are the same.

When dead time is inherent in a process, it is difficult to avoid dead time in the system. Thus we define the system response as

$$\frac{C}{R} = \frac{e^{-\theta s}}{\tau_c s + 1} \quad (\text{6-25})$$

where θ is the dead time in the system. The controller function, via Eq. (6-20), is hence

$$G_c = \frac{1}{G_p} \left[\frac{e^{-\theta s}}{(\tau_c s + 1) - e^{-\theta s}} \right] \approx \frac{1}{G_p} \left[\frac{e^{-\theta s}}{(\tau_c + \theta) s} \right] \quad (\text{6-26})$$

To arrive at the last term, we have used a simple Taylor expansion ($e^{-\theta s} \approx 1 - \theta s$) of the exponential term. This is purposely done to simplify the algebra as shown in the next example. (We could have used the Padé approximation in Eq. (6-26), but the result will not be the simple PI controller.)

Example 6.3: Derive the controller function for a system with a **first order process with dead time** and system response as dictated by Eq. (6-25).

The process transfer function is $G_p = \frac{K_p e^{-t_d s}}{\tau_p s + 1}$. To apply Eq. (6-26), we make an assumption about the dead time, that $\theta = t_d$. The result is a PI controller:

$$G_c = \frac{\tau_p}{K_p (\tau_c + \theta)} \left(1 + \frac{1}{\tau_p s} \right) \quad (\text{E6-4})$$

Even though this result is based on what we say is a process function, we could apply (E6-4) as if the derivation is for the first order with dead time function G_{PRC} obtained from an open-loop step test.

This is a question that invariably arises: what is a reasonable choice of the system time constant τ_c ? Various sources in the literature have different recommendations. For example, one

guideline suggests that we need to ensure $\tau_c > 1.7\theta$ for a PI controller, and $\tau_c > 0.25\theta$ for a PID controller. A reasonably conservative choice has been programmed into the M-file `reciepe.m` available from our *Web Support*. The important reminder is that we should have a habit of checking the τ_c setting with time response simulation and tuning analysis.

In contrast to Eq. (6-22), we can dictate a second order underdamped system response:

$$\frac{C}{R} = \frac{1}{\tau^2 s^2 + 2\zeta\tau s + 1} \quad (6-27)$$

where τ and ζ are the system natural period and damping ratio yet to be determined. Substitution of (6-27) in Eq. (6-20) leads to

$$G_c = \frac{1}{G_p} \left[\frac{1}{\tau^2 s^2 + 2\zeta\tau s} \right] \quad (6-28)$$

which is a slightly more complicated form than (6-23). Again, with simplified cases, we can arrive at PID type controllers.

✎ **Example 6.4:** Derive the controller function for a system with a **second order overdamped process** but an **underdamped system** response as dictated by Eq. (6-27).

The process transfer function is $G_p = \frac{K_p}{(\tau_1 s + 1)(\tau_2 s + 1)}$, and the controller function according to Eq. (6-28) is

$$G_c = \frac{(\tau_1 s + 1)(\tau_2 s + 1)}{K_p \tau s (\tau s + 2\zeta)}.$$

We now define $\tau_f = \tau/2\zeta$, and G_c becomes

$$G_c = \frac{(\tau_1 s + 1)(\tau_2 s + 1)}{2\zeta K_p \tau s (\tau_f s + 1)}$$

Suppose that τ_2 is associated with the slower pole ($\tau_2 > \tau_1$), we now require $\tau_f = \tau_2$ such that the pole and zero cancel each other. The result is a PI controller:

$$G_c = \frac{1}{2\zeta K_p \tau} \frac{(\tau_1 s + 1)}{s}$$

With our definition of τ_f and the requirement $\tau_f = \tau_2$, we can write $\tau = 2\zeta\tau_2$, and the final form of the controller is

$$G_c = \frac{\tau_1}{4K_p \zeta^2 \tau_2} \left(1 + \frac{1}{\tau_1 s} \right) = K_c \left(1 + \frac{1}{\tau_1 s} \right) \quad (E6-5)$$

The integral time constant is $\tau_I = \tau_1$, and the term multiplying the terms in the parentheses is the proportional gain K_c . In this problem, the system damping ratio ζ is the only tuning parameter.

6.2.2 Pole-zero cancellation

We used the term “pole-zero cancellation” at the beginning of this section. We should say a few more words to better appreciate the idea behind direct synthesis. Pole-zero cancellation is also referred to as **cancellation compensation** or dominant pole design. Of course, it is unlikely to

have perfect pole-zero cancellation in real life, and this discussion is more toward helping our theoretical understanding.

The idea is that we may cancel the (undesirable open-loop) poles of our process and replace them with a desirable closed-loop pole. Recall in Eq. (6-20) that G_c is sort of the reciprocal of G_p . The zeros of G_c are by choice the poles of G_p . The product of $G_c G_p$ cancels everything out—hence the term pole-zero cancellation. To be redundant, we can rewrite the general design equation as

$$G_c G_p = \left[\frac{C/R}{1 - C/R} \right] \quad (6-20a)$$

That is, no matter what G_p is, we define G_c such that their product is dictated entirely by a function (the RHS) in terms of our desired system response (C/R). For the specific closed-loop response as dictated by Eq. (6-22), we can also rewrite Eq. (6-23) as

$$G_c G_p = \left[\frac{1}{\tau_c s} \right] \quad (6-23a)$$

Since the system characteristic equation is $1 + G_c G_p = 0$, our closed-loop poles are only dependent on our design parameter τ_c . A closed-loop system designed on the basis of pole-zero cancellation has drastically different behavior than a system without such cancellation.

Let's try to illustrate using a system with a PI controller and a first order process function, and the simplification that $G_m = G_a = 1$. The closed-loop characteristic equation is

$$1 + G_c G_p = 1 + K_c \left(\frac{\tau_1 s + 1}{\tau_1 s} \right) \frac{K_p}{\tau_p s + 1} = 0 \quad (6-29)$$

Under normal circumstances, we would pick a τ_1 which we deem appropriate. Now if we pick τ_1 to be identical to τ_p , the zero of the controller function cancels the pole of the process function. We are left with only one open-loop pole at the origin. Eq. (6-29), when $\tau_1 = \tau_p$, is reduced to

$$1 + \frac{K_c K_p}{\tau_p s} = 0, \text{ or } s = -\frac{K_c K_p}{\tau_p}.$$

There is now only one real and negative closed-loop pole (presuming $K_c > 0$). This situation is exactly what direct synthesis leads us to.

Recall from Example 6.1 that based on the chosen C/R in Eq. (6-22), the PI controller function is

$$G_c = K_c \left(\frac{\tau_1 s + 1}{\tau_1 s} \right) = \frac{\tau_p}{K_p \tau_c} \left(\frac{\tau_p s + 1}{\tau_p s} \right)$$

where $\tau_1 = \tau_p$ and $K_c = \tau_p / K_p \tau_c$. Substitution of K_c one step back in the characteristic equation will show that the closed-loop pole is indeed at $s = -1/\tau_c$. The product $G_c G_p$ is also consistent with Eq. (6-23a) and τ_c .

6.2.3 Internal model control (IMC)

A more elegant approach than direct synthesis is internal model control (IMC). The premise of IMC is that in reality, we only have an approximation of the actual process. Even if we have the correct model, we may not have accurate measurements of the process parameters. Thus the imperfect model should be factored as part of the controller design.

In the block diagram implementing IMC (Fig. 6.3a), our conventional controller G_c consists of the (theoretical) model controller G_c^* and the approximate function \tilde{G}_p . Again, our objective is limited. We use the analysis in very restrictive and simplified cases to arrive at results in Example 6.5 to help us tune PID controllers as in Fig. 6.3b.

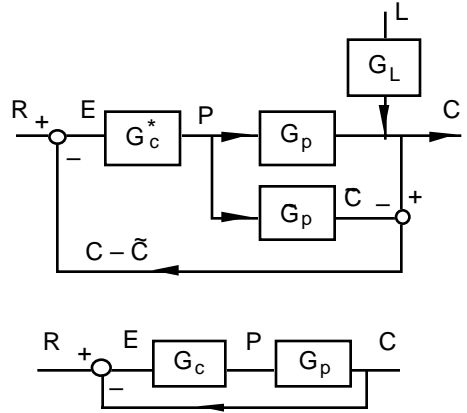


Figure 6.3. A system with IMC (upper panel) as compared with a conventional system in the lower panel.

We first need to derive the closed-loop functions for the system. Based on the block diagram, the error is

$$E = R - (C - \tilde{C})$$

and the model controller output is

$$P = G_c^* E = G_c^* (R - C + \tilde{C})$$

If we substitute $\tilde{C} = \tilde{G}_p P$, we have

$$P = G_c^* (R - C + \tilde{G}_p P) \quad (6-30)$$

from which we can rearrange to obtain

$$P = \frac{G_c^*}{1 - G_c^* \tilde{G}_p} (R - C) \quad (6-28a)$$

The gist of this step is to show the relationship between the conventional controller function G_c and the other functions:

$$G_c = \frac{G_c^*}{1 - G_c^* \tilde{G}_p} \quad (6-31)$$

This is an equation that we will use to retrieve the corresponding PID controller gains. For now, we substitute Eq. (6-28a) in an equation around the process,

$$C = G_L L + G_p P = G_L L + \frac{G_p G_c^*}{1 - G_c^* \tilde{G}_p} (R - C)$$

From this step, we derive the closed-loop equation:

$$C = \left[\frac{(1 - G_c^* \tilde{G}_p) G_L}{1 + G_c^* (G_p - \tilde{G}_p)} \right] L + \left[\frac{G_p G_c^*}{1 + G_c^* (G_p - \tilde{G}_p)} \right] R \quad (6-32)$$

The terms in the brackets are the two closed-loop transfer functions. As always, they have the same denominator—the closed-loop characteristic polynomial.

There is still one unfinished business. We do not know how to choose G_c^* yet. Before we make this decision, we may recall that in direct synthesis, the poles of G_c are “inherited” from the zeros of G_p . If G_p has positive zeros, it will lead to a G_c function with positive poles. To avoid that, we “split” the approximate function as a product of two parts:

$$\tilde{G}_p = \tilde{G}_{p+} \tilde{G}_{p-} \quad (6-33)$$

with \tilde{G}_{p+} containing all the positive zeros, if present. The controller will be designed on the basis of \tilde{G}_{p-} only. We now define the model controller function in a way similar to direct synthesis: ¹

$$G_c^* = \frac{1}{\tilde{G}_{p-}} \left[\frac{1}{\tau_c s + 1} \right]^r, \quad \text{where } r = 1, 2, \text{ etc.} \quad (6-34)$$

Like direct synthesis, τ_c is the closed-loop time constant and our *only* tuning parameter. The first order function raised to an integer power of r is used to ensure that the controller is physically realizable. ² Again, we would violate this intention in our simple example just so that we can obtain results that resemble an ideal PID controller.

▮ **Example 6.5:** Repeat the derivation of a controller function for a system with a **first order process with dead time** using IMC.

Say we model our process (read: fitting the open-loop step test data) as a first order function with time delay, and expecting experimental errors or uncertainties, our measured or approximate model function \tilde{G}_p is

$$\tilde{G}_p = \frac{K_p e^{-t_d s}}{\tau_p s + 1}$$

We use the first order Padé approximation for the dead time and isolate the positive zero term as in Eq. (6-33):

$$\tilde{G}_p \approx \frac{K_p}{(\tau_p s + 1)(\frac{t_d}{2} s + 1)} (-\frac{t_d}{2} s + 1) = \tilde{G}_{p-} \tilde{G}_{p+} \quad (E6-6)$$

where

$$\tilde{G}_{p+} = (-\frac{t_d}{2} s + 1)$$

If we choose $r = 1$, Eq. (6-34) gives

$$G_c^* = \frac{(\tau_p s + 1)(\frac{t_d}{2} s + 1)}{K_p} \frac{1}{(\tau_c s + 1)} \quad (E6-7)$$

¹ If the model is perfect, $G_p = \tilde{G}_p$, and Eq. (6-32) becomes simply $C = G_p G_c^* R$ if we also set $L = 0$. We choose C/R to be a first order response with unity gain, and we'd arrive at a choice of G_c^* very similar to the definition in (6-34).

² The literature refers the term as a first order filter. It only makes sense if you recall your linear circuit analysis or if you wait until the chapter on frequency response analysis.

Substitution of (E6-5) and (E6-6) into Eq. (6-31), and after some algebraic work, will lead to the tuning parameters of an ideal PID controller:

$$K_c = \frac{1}{K_p} \frac{2\frac{\tau_p}{t_d} + 1}{2\frac{\tau_c}{t_d} + 1} \quad ; \quad \tau_I = \tau_p + \frac{t_d}{2} \quad ; \quad \tau_D = \frac{\tau_p}{2\frac{\tau_p}{t_d} + 1} \quad (\text{E6-8})$$

▮ **Example 5.7B:** What would be the PID controller settings for the dye mixing problem if we use IMC-based tuning relations?

With the same first order with dead time approximation in Example 5.7A (p. 6-5), and the choice of τ_c being two-thirds the value of dead time, the IMC relations in (E6-8) provide the following PID settings (as computed with our M-file recipe.m):

	K_c	τ_I	τ_D
IMC	3.4	4.5	0.4

Compare this result using other tuning relations in Example 5.7A. The IMC proportional gain falls in between the Cohen-Coon and ITAE settings, but the integral time constant is relatively high. With less integrating action, we expect this IMC tuning to be less oscillatory. Indeed, we shall see that if we do Example 5.7C (or you can cheat and read the plotting result from our *Web Support*).

▮ **Example 5.7C:** How do the different controller settings affect the system time response in the dye mixing problem?

We can use the following MATLAB statements to do time response simulations (explanations are in MATLAB Session 5). Better yet, save them in an M-file. The plotting can be handled differently to suit your personal taste. (Of course, you can use Simulink instead.)

```

alfa=0.1; % Real PID
Gc=tf(kc*[taui*taud (taui+taud) 1],[alfa*taui*taud taui 0]);
td=0.725;
Gm=tf([-td/2 1],[td/2 1]); %Padé approximation for dead time
Km=2.6; %Move Km into the forward path
Gp=tf(0.8,[4 1]);
Ga=tf(0.6,[0.2 1]);
Gcl=feedback(Km*Gc*Ga*Gp,Gm); % The closed-loop function

step(Gcl) % Plotting...
```

We reset the three controller parameters each time we execute the M-file. For example, to use the Cohen-Coon results, we would take from Example 5.7A:

```
kc=4.9; taui=2; taud=0.31;
```

MATLAB calculation details and plots can be found on our *Web Support*. You should observe that Cohen-Coon and Ziegler-Nichols tuning relations lead to roughly 74% and 64% overshoot, respectively, which are more significant than what we expect with a quarter decay ratio criterion.

The ITAE, with about 14% overshoot, is more conservative. Ciancone and Marlin tuning relations are ultra conservative; the system is slow and overdamped.

With the IMC tuning setting in Example 5.7B, the resulting time response plot is (very nicely) slightly underdamped even though the derivation in Example 6.4 predicates on a system response without oscillations. Part of the reason lies in the approximation of the dead time function, and part of the reason is due to how the system time constant was chosen. Generally, it is important to double check our IMC settings with simulations.

At this point, one may be sufficiently confused with respect to all the different controller tuning methods. Use Table 6.3 as a guide to review and compare different techniques this chapter and also Chapters 7 and 8.

Table 6.2. Summary of PID controller settings based on IMC or direct synthesis

Process model	Controller	K_c	τ_I	τ_D
$\frac{K_p}{\tau_p s + 1}$	PI	$\frac{\tau_p}{K_p \tau_c}$	τ_p	—
$\frac{K_p}{(\tau_1 s + 1)(\tau_2 s + 1)}$	PID	$\frac{\tau_1 + \tau_2}{K_p \tau_c}$	$\tau_1 + \tau_2$	$\frac{\tau_1 \tau_2}{\tau_1 + \tau_2}$
	PID with $\tau_1 > \tau_2$	$\frac{\tau_1}{K_p \tau_c}$	τ_1	τ_2
	PI (underdamped)	$\frac{\tau_1}{4K_p \zeta^2 \tau_2}$	τ_1	—
$\frac{K_p}{\tau^2 s^2 + 2\zeta \tau s + 1}$	PID	$\frac{2\zeta \tau}{K_p \tau_c}$	$2\zeta \tau$	$\frac{\tau}{2\zeta}$
$\frac{K_p}{s(\tau_p s + 1)}$	PD	$\frac{1}{K_p \tau_c}$	—	τ_p
$\frac{K_p e^{-t_d s}}{\tau_p s + 1}$	PI	$\frac{\tau_p}{K_p (\tau_c + t_d)}$	τ_p	—
	PID	$\frac{1}{K_p} \frac{2\tau_p/t_d + 1}{2\tau_c/t_d + 1}$	$\tau_p + t_d/2$	$\frac{\tau_p}{2\tau_p/t_d + 1}$
$\frac{K_p}{s}$	P	$\frac{1}{K_p \tau_c}$	—	—

□ Review Problems

4. Repeat Example 6.1 when we have $G_p = \frac{K_p}{s(\tau_p s + 1)}$. What is the offset of the system?
5. What are the limitations to IMC? Especially with respect to the choice of τ_c ?
6. What control action increases the order of the system?
7. Refer back to Example 6.4. If we have a third order process

$$G_p = \frac{K_p}{(\tau_1 s + 1)(\tau_2 s + 1)(\tau_3 s + 1)}$$

what is the controller function if we follow the same strategy as in the example?

8. Complete the time response simulations in Example 5.7C using settings in Example 5.7A.
9. (Optional) How would you implement the PID algorithm in a computer program?

Hints:

1. The result is an ideal PD controller with the choice of $\tau_D = \tau_p$. See that you can obtain the same result with IMC too. Here, take the process function as the approximate model and it has no parts that we need to consider as having positive zeros. There is no offset; the integrating action is provided by G_p .
2. Too small a value of τ_c means too large a K_c and therefore saturation. System response is subject to imperfect pole-zero cancellation.
3. Integration is $1/s$.
10. The intermediate step is

$$G_c = \frac{(\tau_1 s + 1)(\tau_2 s + 1)(\tau_3 s + 1)}{2\zeta K_p \tau s(\tau_c s + 1)}$$

where $\tau_f = \tau/2\zeta$, and now we require $\tau_f = \tau_3$, presuming it is the largest time constant. The final result, after also taking some of the ideas from Example 6.2, is an ideal PID controller with the form:

$$G_c = \frac{(\tau_1 + \tau_2)}{4K_p \zeta^2 \tau_3} \left(1 + \frac{1}{\tau_1 + \tau_2} \frac{1}{s} + \frac{\tau_1 \tau_2}{\tau_1 + \tau_2} s \right)$$

The necessary choices of K_c , τ_I , and τ_D are obvious. Again, ζ is the only tuning parameter.

5. See our *Web Support* for the simulations.
6. Use finite difference. The ideal PID in Eq. (5-8a) can be discretized as

$$p_n = p^s + K_c \left[e_n + \frac{\Delta t}{\tau_I} \sum_{k=1}^n e_k + \frac{\tau_D}{\Delta t} (e_n - e_{n-1}) \right]$$

where p_n is the controller output at the n -th sampling period, p^s is the controller bias, Δt is the sampling period, and e_n is the error. This is referred to as the *position form* algorithm. The alternate approach is to compute the change in the controller output based on the difference between two samplings:

$$\Delta p_n = p_n - p_{n-1} = K_c \left[(e_n - e_{n-1}) + \frac{\Delta t}{\tau_I} e_n + \frac{\tau_D}{\Delta t} (e_n - 2e_{n-1} + e_{n-2}) \right]$$

This is the *velocity form* algorithm which is considered to be more attractive than the position form. The summation of error is not computed explicitly and thus the velocity form is not as susceptible to reset windup.

Table 6.3. Summary of methods to select controller gains

Method	What to do?	What is evaluated?	Comments
<div><div>❑ Transient response criteria</div><div><ul style="list-style-type: none">Analytical derivation</div></div>	<p>Derive closed-loop damping ratio from a second order system characteristic polynomial. Relate the damping ratio to the proportional gain of the system.</p>	<p>Usually the proportional gain.</p>	<ul style="list-style-type: none">Limited to second order systems. No unique answer other than a P-controller.Theoretically can use other transient response criteria.1/4 decay ratio provides a 50% overshoot.
<div><div>❑ Empirical tuning with open-loop step test</div><div><ul style="list-style-type: none">Cohen-CoonZiegler-NicholsCiacone-Marlin</div></div>	<p>Measure open-loop step response, the so-called process reaction curve. Fit data to first order with dead-time function. Apply empirical design relations.</p>	<p>Proportional gain, integral and derivative time constants to PI and PID controllers.</p>	<ul style="list-style-type: none">Cohen-Coon was designed to handle disturbances by preventing a large initial deviation from the set point. The one-quarter decay ratio response is generally too underdamped for set point changes.
<div><div></div><div><ul style="list-style-type: none">Time integral performance criteria (ISE, IAE, ITAE)</div></div>	<p>Apply design relations derived from minimization of an error integral of the theoretical time-domain response.</p>	<p>Proportional gain, integral and derivative time constants to PI and PID controllers.</p>	<ul style="list-style-type: none">Different settings for load and set point changes.Different settings for different definitions of the error integral.The minimum ITAE criterion provides the least oscillatory response.
<div><div>❑ Ziegler-Nichols Continuous Cycling (empirical tuning with closed loop test)</div></div>	<p>Increase proportional gain of only a proportional controller until system sustains oscillation. Measure ultimate gain and ultimate period. Apply empirical design relations.</p>	<p>Proportional gain, integral and derivative time constants of PID controllers.</p>	<ul style="list-style-type: none">Experimental analog of the $s = j\omega$ substitution calculation.Not necessarily feasible with chemical systems in practice.Tuning relations allow for choices from 1/4 decay ratio to little oscillations.
<div><div>❑ Stability analysis methods</div><div><ul style="list-style-type: none">Routh-Hurwitz criterion</div></div>	<p>Apply the Routh test on the closed-loop characteristic polynomial to find if there are closed-loop poles on the right-hand-plane.</p>	<p>Establish limits on the controller gain.</p>	<ul style="list-style-type: none">Usually applies to relatively simple systems with the focus on the proportional gain.Need be careful on interpretation when the lower limit on proportional gain is negative.
<div><div></div><div><ul style="list-style-type: none">Direct substitution</div></div>	<p>Substitute $s = j\omega$ in characteristic polynomial and solve for closed-loop poles on Im-axis. The Im and Re parts of the equation allow the ultimate gain and ultimate frequency to be solved.</p>	<p>Ultimate gain and ultimate period ($P_u = 2\pi/\omega_u$) that can be used in the Ziegler-Nichols continuous cycling relations.</p>	<ul style="list-style-type: none">Result on ultimate gain is consistent with the Routh array analysis.Limited to relatively simple systems.

Summary (continued) Method	What to do?	What is evaluated?	Comments
<ul style="list-style-type: none"> • Root-locus 	With each chosen value of proportional gain, plot the closed-loop poles. Generate the loci with either hand-sketching or computer.	The loci of closed-loop poles reveal the effect of controller gain on the probable closed-loop dynamic response. Together with specifications of damping ratio and time constant, the loci can be a basis of selecting proportional gain.	<ul style="list-style-type: none"> • Rarely used in the final controller design because of difficulty in handling dead-time. • Method is instructive and great pedagogical tool.
<input type="checkbox"/> (Model-based) Direct synthesis	For a given system, synthesize the controller function according to a specified closed-loop response. The system time constant, τ_c , is the only tuning parameter.	Proportional gain, integral and derivative time constants where appropriate.	<ul style="list-style-type: none"> • The design is not necessarily PID, but where the structure of a PID controller results, this method provides insight into the selection of the controller mode (PI, PD, PID) and settings. • Especially useful with system that has no dead time.
<ul style="list-style-type: none"> • Internal model control 	Extension of direct synthesis. Controller design includes an internal approximation process function.	For a first order function with dead-time, the proportional gain, integral and derivative time constants of an ideal PID controller.	
<input type="checkbox"/> Frequency-domain methods			<ul style="list-style-type: none"> • Can handle dead-time easily and rigorously. • The Nyquist criterion allows the use of open-loop functions in Nyquist or Bode plots to analyze the closed-loop problem. • The stability criteria have no use for simple first and second order systems with no positive open-loop zeros.
<ul style="list-style-type: none"> • Nyquist plot • Bode plot 	Nyquist plot is a frequency parametric plot of the magnitude and the argument of the open-loop transfer function in polar coordinates. Bode plot is magnitude vs. frequency and phase angle vs. frequency plotted individually.	Calculate proportional gain needed to satisfy the gain or phase margin.	<ul style="list-style-type: none"> • These plots address the stability problem but need other methods to reveal the probable dynamic response.
<ul style="list-style-type: none"> • Nichols chart 	Nichols chart is a frequency parametric plot of open-loop function magnitude vs. phase angle. The closed-loop magnitude and phase angle are overlaid as contours.	With gain or phase margin, calculate proportional gain. Can also estimate the peak amplitude ratio, and assess the degree of oscillation.	<ul style="list-style-type: none"> • Nichols chart is usually constructed for unity feedback loops only.
<ul style="list-style-type: none"> • Maximum closed-loop log modulus 	A plot of the magnitude vs. frequency of the closed-loop transfer function.	The peak amplitude ratio for a chosen proportional gain.	

❖ 10. Multiloop Systems

There are many advanced strategies in classical control systems. Only a limited selection of examples is presented in this chapter. We start with cascade control, which is a simple introduction to a multiloop, but essentially SISO, system. We continue with feedforward and ratio control. The idea behind ratio control is simple, and it applies quite well to the furnace problem that we use as an illustration. Finally, we address a multiple-input multiple-output system using a simple blending problem as illustration, and use the problem to look into issues of interaction and decoupling. These techniques build on what we have learned in classical control theories.

What are we up to?

- Apply classical controller analysis to cascade control, feedforward control, feedforward-feedback control, ratio control, and the Smith predictor for time delay compensation.
- Analyze a MIMO system with relative gain array, and assess the pairing of manipulated and controlled variables.
- Attempt to decouple and eliminate the interactions in a two-input two-output system.

10.1 Cascade control

A very common design found in process engineering is cascade control. This is a strategy that allows us to handle load changes more effectively with respect to the manipulated variable.

To illustrate the idea, we consider the temperature control of a gas furnace, which is used to heat up a cold process stream. The fuel gas flow rate is the manipulated variable, and its flow is subject to fluctuations due to upstream pressure variations.

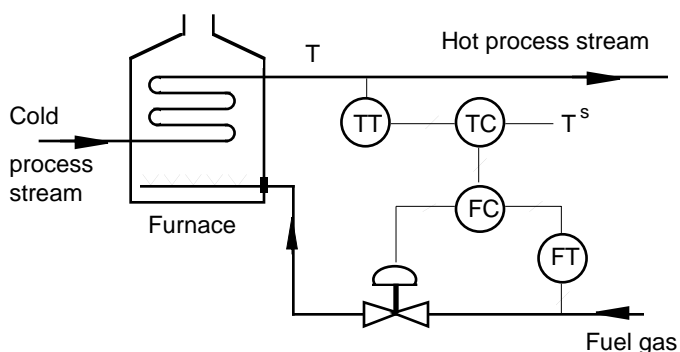


Figure 10.1. Cascade control of the temperature of a furnace, which is taken to be the same as that of the outlet process stream. The temperature controller does not actuate the regulating valve directly; it sends its signal to a secondary flow rate control loop which in turn ensures that the desired fuel gas

In a simple single-loop system, we measure the outlet temperature, and the temperature controller (TC) sends its signal to the regulating valve. If there is fluctuation in the fuel gas flow rate, this simple system will not counter the disturbance until the controller senses that the temperature of the furnace has deviated from the set point (T^s).

A cascade control system can be designed to handle fuel gas disturbance more effectively (Fig. 10.1). In this case, a *secondary loop* (also called the *slave loop*) is used to adjust the regulating valve and thus manipulate the fuel gas flow rate. The temperature controller (the master or primary controller) sends its signal, in terms of the desired flow rate, to the secondary flow control loop—in essence, the signal is the set point of the secondary flow controller (FC).

In the secondary loop, the flow controller compares the desired fuel gas flow rate with the measured flow rate from the flow transducer (FT), and adjusts the regulating valve accordingly. This inner flow control loop can respond immediately to fluctuations in the fuel gas flow to ensure

that the proper amount of fuel is delivered.

To be effective, the secondary loop must have a faster response time (smaller time constant) than the outer loop. Generally,

we use as high a proportional gain as feasible. In control jargon, we say that the inner loop is tuned very tightly.

We can use a block diagram to describe Fig. 10.1. Cascade control adds an inner control loop with secondary controller function G_{c2} (Fig. 10.2a). This implementation of cascade control requires two controllers and two measured variables (fuel gas flow and furnace temperature). The furnace temperature is the controlled variable, and the fuel gas flow rate remains the only manipulated variable.

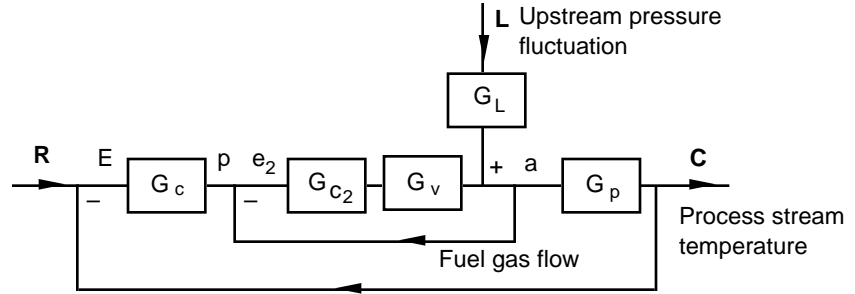


Figure 10.2a. Block diagram of a simple cascade control system with reference to the furnace problem.

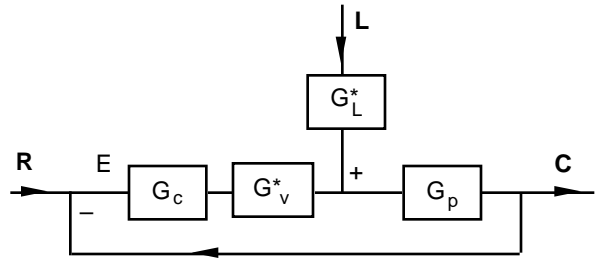


Figure 10.2b. Reduced block diagram of a cascade control system.

For cleaner algebra, we omit the measurement transfer functions, taking $G_{m1} = G_{m2} = 1$. Disturbance, such as upstream pressure, which specifically leads to changes in the fuel gas flow rate is now drawn to be part of the secondary flow control loop. (A disturbance such as change in the process stream inlet temperature, which is not part of the secondary loop, would still be drawn in its usual location as in Section 5.2 [on page 5-7](#).)

We now reduce the block diagram. The first step is to close the inner loop so the system becomes a standard feedback loop (Fig. 10.2b). With hindsight, the result should be intuitively obvious. For now, we take the slow route. Using the lower case letter locations in Fig. 10.2a, we write down the algebraic equations

$$e_2 = p - a$$

and

$$a = G_{c2}G_v e_2 + G_L L$$

Substitution of e_2 leads to

$$a = G_{c2}G_v(p - a) + G_L L$$

and the result after rearrangement is a form that allows us to draw Fig. 10.2b:

$$a = \left[\frac{G_{c2}G_v}{1 + G_{c2}G_v} \right] p + \left[\frac{G_L}{1 + G_{c2}G_v} \right] L = G_v^* p + G_L^* L$$

where

$$G_v^* = \left[\frac{G_{c2}G_v}{1 + G_{c2}G_v} \right] \quad \text{and} \quad G_L^* = \left[\frac{G_L}{1 + G_{c2}G_v} \right] \quad (10-1)$$

The remaining task to derive the closed-loop transfer functions is routine. Again, slowly, we can write the relation in Fig. 10.2b as

$$C = G^*_L G_p L + G_c G^*_v G_p E$$

and substituting $E = R - C$, we have, after rearrangement,

$$C = \left[\frac{G_c G_v^* G_p}{1 + G_c G_v^* G_p} \right] R + \left[\frac{G_p G_L^*}{1 + G_c G_v^* G_p} \right] L \quad (10-2)$$

The closed-loop characteristic polynomial of this cascade system is

$$1 + G_c G_v^* G_p = 0 \quad (10-3)$$

If we now substitute G_v^* from (10-1), the characteristic polynomial takes the form ¹

$$1 + G_{c2} G_v + G_c G_{c2} G_v G_p = 0 \quad (10-3a)$$

So far, we know that the secondary loop helps to reduce disturbance in the manipulated variable. If we design the control loop properly, we should also accomplish a faster response in the actuating element: the regulating valve. To go one step further, cascade control can even help to make the entire system more stable. These points may not be intuitive. We'll use a simple example to illustrate these features.

✎ **Example 10.1:** Consider a simple cascade system as shown in Fig. 10.2a with a PI controller in the primary loop, and a proportional controller in the slave loop. For simplicity, consider first order functions

$$G_p = \frac{0.8}{2s + 1}, \quad G_v = \frac{0.5}{s + 1}, \quad \text{and} \quad G_L = \frac{0.75}{s + 1}.$$

- (a) How can proper choice of K_{c2} of the controller in the slave loop help to improve the actuator performance and eliminate disturbance in the manipulated variable (e.g., fuel gas flow in the furnace temperature control)?

If we substitute $G_{c2} = K_{c2}$, and $G_v = \frac{K_v}{\tau_v s + 1}$ into G_v^* in Eq. (10-1), we should find

$$G_v^* = \left[\frac{K_{c2} K_v}{(\tau_v s + 1) + K_{c2} K_v} \right] = \frac{K_v^*}{\tau_v^* s + 1}, \quad (E10-1)$$

where

$$K_v^* = \left[\frac{K_{c2} K_v}{1 + K_{c2} K_v} \right], \quad \text{and} \quad \tau_v^* = \left[\frac{\tau_v}{1 + K_{c2} K_v} \right]. \quad (E10-2)$$

Similarly, substituting $G_L = \frac{K_L}{\tau_L s + 1}$ in G_L^* should give

$$K_L^* = \left[\frac{K_L}{1 + K_{c2} K_v} \right]. \quad (E10-3)$$

Thus as the proportional gain K_{c2} becomes larger, K_v^* approaches unity gain, meaning there

¹ If we remove the secondary loop, this characteristic equation should reduce to that of a conventional feedback system equation. It is not obvious from (10-3) because our derivation has taken the measurement function G_{m2} to be unity. If we had included G_{m2} in a more detailed analysis, we could get the single loop result by setting $G_{c2} = 1$ and $G_{m2} = 0$.

is a more effective change in the manipulated variable, and K^*_L approaches zero, meaning the manipulated variable is becoming less sensitive to changes in the load. Furthermore, the effective actuator time constant τ^*_v will become smaller, meaning a faster response.

- (b) The slave loop affords us a faster response with respect to the actuator. What is the proportional gain K_{c2} if we want the slave loop time constant τ^*_v to be only one-tenth of the original time constant τ_v in G_v ?

From the problem statement, $K_v = 0.5$ and $\tau_v = 1$ s. Thus $\tau^*_v = 0.1$ s, and substitution of these values in τ^*_v of (E10-2) gives

$$0.1 = \left[\frac{1}{1 + 0.5 K_{c2}} \right], \text{ or } K_{c2} = 18.$$

The steady state gain is

$$K_v^* = \frac{(18)(0.5)}{1 + (18)(0.5)} = 0.9$$

The slave loop will have a 10% offset with respect to desired set point changes in the secondary controller.

- (c) So far, we have only used proportional control in the slave loop. We certainly expect offset in this inner loop. Why do we stay with proportional control here?

The modest 10% offset that we have in the slave loop is acceptable under most circumstances. As long as we have integral action in the outer loop, the primary controller can make necessary adjustments in its output and ensure that there is no steady state error in the controlled variable (e.g., the furnace temperature).

- (d) Now, we tackle the entire closed-loop system with the primary PI controller. Our task here is to choose the proper integral time constant among the given values of 0.05, 0.5, and 5 s. We can tolerate underdamped response but absolutely not a system that can become unstable. Of course, we want a system response that is as fast as we can make it, *i.e.*, with a proper choice of proportional gain. Select and explain your choice of the integral time constant.

Among other methods, root locus is the most instructive in this case. With a PI primary controller and numerical values, Eq. (10-3) becomes

$$1 + K_c \left(\frac{\tau_I s + 1}{\tau_I s} \right) \left(\frac{0.9}{0.1 s + 1} \right) \left(\frac{0.8}{2 s + 1} \right) = 0$$

With MATLAB, we can easily prepare the root locus plots of this equation for the cases of $\tau_I = 0.05, 0.5$, and 5 s. (You should do it yourself. We'll show only a rough sketch in Fig E10.1. Help can be found in the Review Problems.)

From the root locus plots, it is clear that the system may become unstable when $\tau_I = 0.05$ s. The system is always stable when $\tau_I = 5$ s, but the speed of the system response is limited by the dominant pole between the origin and -0.2 . The proper choice is $\tau_I = 0.5$ s in which case the system is always stable but the closed-loop poles can move farther, loosely speaking, away from the origin.

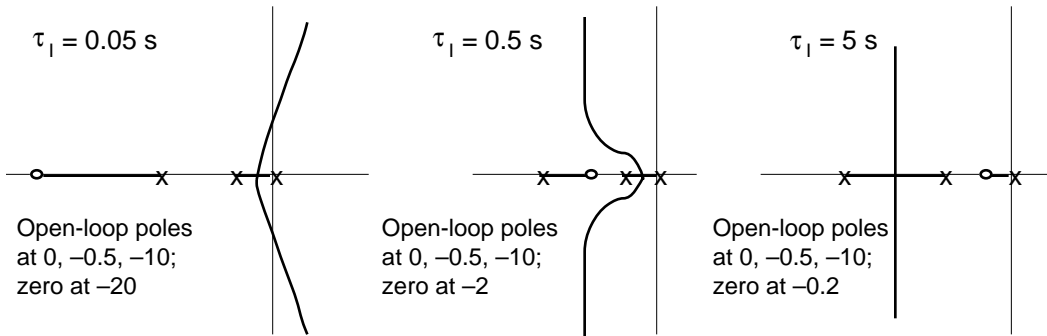


Figure E10.1

- (e) Take the case *without* cascade control, and using the integral time constant that you have selected in part (d), determine the range of proportional gain that we can use (without the cascade controller and secondary loop) to maintain a stable system. How is this different from when we use cascade control?

With the choice of $\tau_i = 0.5 \text{ s}$, but without the inner loop nor the secondary controller, the closed-loop equation is

$$1 + G_c G_v G_p = 1 + K_c \left(\frac{0.5s+1}{0.5s} \right) \left(\frac{0.5}{s+1} \right) \left(\frac{0.8}{2s+1} \right) = 0$$

which can be expanded to

$$s^3 + 1.5s^2 + (0.5 + 0.2K_c)s + 0.4K_c = 0$$

With the Routh-Hurwitz analysis in Chapter 7, we should find that to have a stable system, we must keep $K_c < 7.5$. (You fill in the intermediate steps in the Review Problems. Other techniques such as root locus, direct substitution or frequency response in Chapter 8 should arrive at the same result.)

With cascade control, we know from part (d) that the system is always stable. Nevertheless, we can write the closed-loop characteristic equation

$$1 + K_c \left(\frac{0.5s+1}{0.5s} \right) \left(\frac{0.9}{0.1s+1} \right) \left(\frac{0.8}{2s+1} \right) = 0$$

or

$$0.1s^3 + 1.05s^2 + (0.5 + 0.36K_c)s + 0.72K_c = 0$$

A Routh-Hurwitz analysis can confirm that. The key point is that with cascade control, the system becomes more stable and allows us to use a larger proportional gain in the primary controller. The main reason is the much faster response (smaller time constant) of the actuator in the inner loop.²

² If you are skeptical of this statement, try do the Bode plots of the systems with and without cascade control and think about the consequence of changing the break frequency (or bandwidth) of the valve function. If you do not pick up the hint, the answer can be found on our *Web Support* on the details of Example 10.1.

10.2 Feedforward control

To counter probable disturbances, we can take an even more proactive approach than cascade control, and use feedforward control. The idea is that if we can make measurements of disturbance changes, we can use this information and our knowledge of the process model to make proper adjustments in the manipulated variable *before* the disturbance has a chance to affect the controlled variable.

We will continue with the gas furnace to illustrate feedforward control. For simplicity, let's make the assumption that changes in the furnace temperature (T) can be effected by changes in the fuel gas flow rate (F_{fuel}) and the cold process stream flow rate (F_s). Other variables such as the process stream temperature are constant.

In Section 10.1, the fuel gas flow rate is the manipulated variable (M) and cascade control is used to handle its fluctuations. Now, we consider also changes in the cold process stream flow rate as another disturbance (L). Let's presume further that we have derived diligently from heat and mass balances the corresponding transfer functions, G_L and G_p , and we have the process model

$$C = G_L L + G_p M \quad (10-4)$$

where we have used the general notation C as the controlled variable in place of furnace temperature T.

We want the controlled variable to track set point changes (R) precisely, so we substitute the ideal scenario $C = R$, and rearrange Eq. (10-4) to

$$M = \frac{1}{G_p} R - \frac{G_L}{G_p} L \quad (10-5)$$

This equation provides us with a model-based rule as to how the manipulated variable should be adjusted when we either change the set point or face with a change in the load variable. Eq. (10-5) is the basis of what we call *dynamic* feedforward control because (10-4) has to be derived from a time-domain differential equation (a transient model).³

In Eq. (10-5), $1/G_p$ is the *set point tracking* controller. This is what we need if we install only a feedforward controller, which in reality, we seldom do.⁴ Under most circumstances, the change in set point is handled by a feedback control loop, and we only need to implement the second term of (10-5). The transfer function $-G_L/G_p$ is the *feedforward* controller (or the *disturbance rejection* controller). In terms of disturbance rejection, we may also see how the feedforward controller arises if we command $C = 0$ (i.e., no change), and write (10-4) as

$$0 = G_L L + G_p M$$

To see how we implement a feedforward controller, we now turn to a block diagram (Fig. 10.3).⁵ For the moment, we omit the feedback path from our general picture. With the

³ In contrast, we could have done the derivation using steady state models. In such a case, we would arrive at the design equation for a *steady state* feedforward controller. We'll skip this analysis. As will be shown later, we can identify this steady state part from the dynamic approach.

⁴ The set point tracking controller not only becomes redundant as soon as we add feedback control, but it also unnecessarily ties the feedforward controller into the closed-loop characteristic equation.

⁵ If the transfer functions G_L and G_p are based on a simple process model, we know quite well that they should have the same characteristic polynomial. Thus the term $-G_L/G_p$ is nothing but a ratio of the steady state gains, $-K_L/K_p$.

expectation that we'll introduce a feedback loop, we will not implement the set point tracking term in Eq. (10-5). Implementation of feedforward control requires measurement of the load variable.

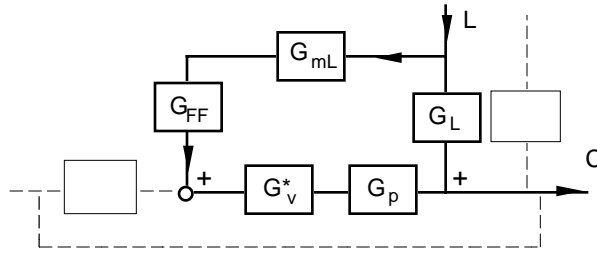


Figure 10.3. A feedforward control system on a major load variable with measurement function G_{mL} and feedforward controller G_{FF} .

If there is more than one load variable, we theoretically could implement a feedforward controller on each one. However, that may not be good engineering. Unless there is a compelling reason, we should select the variable that either undergoes the most severe fluctuation or has the strongest impact on the controlled variable.

Here, we use L to denote the major load variable and its corresponding transfer function is G_L . We measure the load variable with a sensor, G_{mL} , which transmits its signal to the feedforward controller G_{FF} . The feedforward controller then sends its decision to manipulate the actuating element, or valve, G_v . In the block diagram, the actuator transfer function is denoted by G_v^* . The idea is that cascade control may be implemented with the actuator, G_v , as we have derived in Eq. (10-1). We simply use G_v^* to reduce clutter in the diagram.

With the feedforward and load path shown, the corresponding algebraic representation is

$$C = [G_L + G_{mL}G_{FF}G_v^*G_p] L \quad (10-6)$$

The ideal feedforward controller should allow us to make proper adjustment in the actuator to achieve perfect rejection of load changes. To have $C = 0$, the theoretical feedforward controller function is

$$G_{FF} = -\frac{G_L}{G_{mL}G_v^*G_p} \quad (10-7)$$

which is a slightly more complete version of what we have derived in Eq. (10-5).

Before we blindly try to program Eq. (10-7) into a computer, we need to recognize certain pitfalls. If we write out the transfer functions in (10-7), we should find that G_{FF} is not physically realizable—the polynomial in the numerator has a higher order than the one in the denominator.⁶

If we approximate the composite function $G_{mL}G_v^*G_p$ as a first order function with dead time, $Ke^{-\theta s}/(\tau s + 1)$, Eq. (10-7) would appear as

⁶ If we go by the book, G_L and G_p are the transfer functions to a process and their dynamic terms (characteristic polynomial) in Eq. (10-7) must cancel out. The feedforward transfer function would be reduced to something that looks like $(-K_L/K_{mL}K_v^*K_p)(\tau_{mL}s + 1)(\tau_v^*s + 1)$ while the denominator is just 1.

In the simplest scenario where the responses of the transmitter and the valve are extremely fast such that we can ignore their dynamics, the feedforward function consists of only the steady state gains as in Eq. (10-9).

$$G_{FF} = - \frac{K_L}{K} \frac{\tau_s + 1}{\tau_p s + 1} e^{\theta s}$$

Now the dead time appears as a positive exponent or an advance in time. We cannot foresee future and this idea is not probable either.⁷

The consequence is that most simple implementation of a feedforward controller, especially with off-the-shelf hardware, is a lead-lag element with a gain:

$$G_{FF} = K_{FF} \frac{\tau_{FLD}s + 1}{\tau_{FLG}s + 1} \quad (10-8)$$

Based on Eq. (10-7), the gain of this feedforward controller is

$$K_{FF} = - \frac{K_L}{K_m K_v^* K_p} \quad (10-9)$$

This is the *steady state compensator*. The lead-lag element with lead time constant τ_{FLD} and lag time constant τ_{FLG} is the *dynamic compensator*. Any dead time in the transfer functions in (10-7) is omitted in this implementation.

When we tune the feedforward controller, we may take, as a first approximation, τ_{FLD} as the sum of the time constants τ_m and τ_v^* . Analogous to the "real" derivative control function, we can choose the lag time constant to be a tenth smaller, $\tau_{FLG} \approx 0.1 \tau_{FLD}$. If the dynamics of the measurement device is extremely fast, $G_m = K_m$, and if we have cascade control, the time constant τ_v^* is also small, and we may not need the lead-lag element in the feedforward controller. Just the use of the steady state compensator K_{FF} may suffice. In any event, the feedforward controller must be tuned with computer simulations, and subsequently, field tests.

⁷ If the load transfer function in Eq. (10-7) had also been approximated as a first order function with dead time, say, of the form $K_L e^{-t_d s} / (\tau_p s + 1)$, the feedforward controller would appear as

$$G_{FF} = - \frac{K_L}{K} \frac{\tau_s + 1}{\tau_p s + 1} e^{-(t_d - \theta)s}.$$

Now, if $t_d > \theta$, it is possible for the feedforward controller to incorporate dead time compensation.

The situation where we may find the load function dead time is larger than that in the feedforward path of $G_m G_v^* G_p$ is not obvious from our simplified block diagram. Such a circumstance arises when we deal with more complex processing equipment consisting of several units (*i.e.*, multicapacity process) and the disturbance enters farther upstream than where the controlled and manipulated variables are located.

10.3 Feedforward-feedback control

Since we do not have the precise model function G_p embedded in the feedforward controller function in Eq. (10-8), we cannot expect perfect rejection of disturbances. In fact, feedforward control is never used by itself; it is implemented in conjunction with a feedback loop to provide the so-called feedback trim (Fig. 10.4a). The feedback loop handles (1) measurement errors, (2) errors in the feedforward function, (3) changes in unmeasured load variables, such as the inlet process stream temperature in the furnace that one single feedforward loop cannot handle, and of course, (4) set point changes.

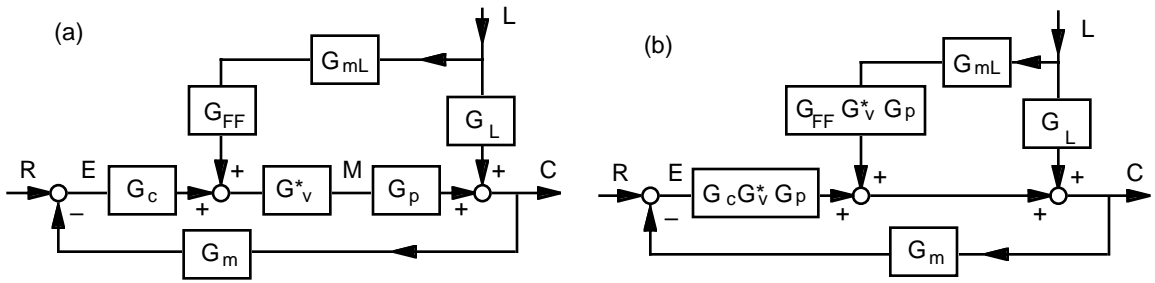


Figure 10.4. (a) A feedforward-feedback control system. (b) The diagram after moving $G^*_v G_p$.

Our next task is to find the closed-loop transfer functions of this feedforward-feedback system. Among other methods, we should see that we can "move" the $G^*_v G_p$ term as shown in Fig. 10.4b. (You can double check with algebra.) After this step, the rest is routine. We can almost write down the final result immediately. Anyway, we should see that

$$C = [G_L + G_{mL} G_{FF} G^*_v G_p] L + [G_c G^*_v G_p] E$$

and

$$E = R - G_m C$$

After substitution for E and rearrangement, we arrive at

$$C = \frac{G_L + G_{mL} G_{FF} G^*_v G_p}{1 + G_m G_c G^*_v G_p} L + \frac{G_c G^*_v G_p}{1 + G_m G_c G^*_v G_p} R \quad (10-10)$$

If we do not have cascade control, G^*_v is simply G_v . If we are using cascade control, we can substitute for G^*_v with Eq. (10-1), but we'll skip this messy algebraic step. The key point is that the closed-loop characteristic polynomial is

$$1 + G_m G_c G^*_v G_p = 0 \quad (10-11)$$

and the feedforward controller G_{FF} does not affect the system stability.

Example 10.2: Consider the temperature control of a gas furnace used in heating a process stream. The probable disturbances are in the process stream temperature and flow rate, and the fuel gas flow rate. Draw the schematic diagram of the furnace temperature control system, and show how feedforward, feedback and cascade controls can all be implemented together to handle load changes.

The design in Fig. E10.2 is based on our discussion of cascade control. The fuel gas flow is the manipulated variable, and so we handle disturbance in the fuel gas flow with a flow controller

(FC) in a slave loop. This secondary loop remains the same as the G_v^* function in (10-1), where the secondary transfer function is denoted by G_{c2} .

Of the other two load variables, we choose the process stream flow rate as the major disturbance. The flow transducer sends the signal to the feedforward controller (FFC, transfer function G_{FF}). A summer (Σ) combines the signals from both the feedforward and the feedback controllers, and its output becomes the set point for the secondary fuel gas flow rate controller (FC).

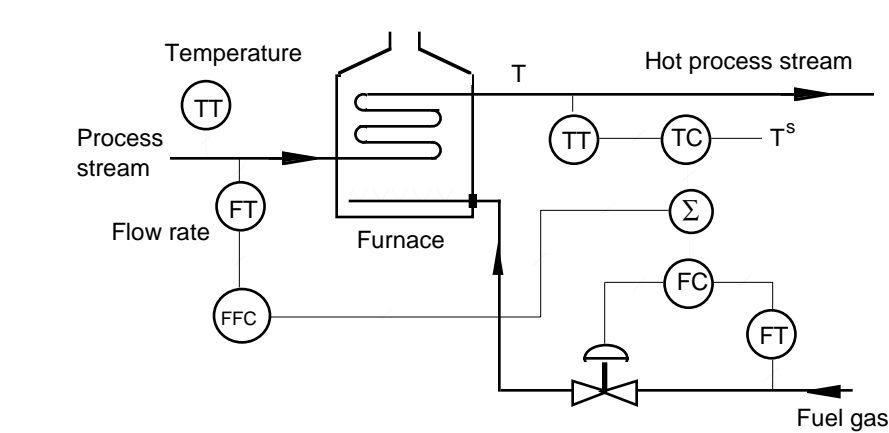


Figure E10.2

Handling of disturbance in the inlet process stream temperature is passive. Any changes in this load variable will affect the furnace temperature. The change in furnace temperature is measured by the outlet temperature transducer (TT) and sent to the feedback temperature controller (TC). The primary controller then acts accordingly to reduce the deviation in the furnace temperature.

10.4 Ratio control

We are not entirely finished with the furnace. There is one more piece missing from the whole picture—the air flow rate. We need to ensure sufficient air flow for efficient combustion. The regulation of air flow is particularly important in the reduction of air pollutant emission.

To regulate the air flow rate with respect to the fuel gas flow rate, we can use ratio control. Fig. 10.5 illustrates one of the simplest implementations of this strategy. Let's say the air to fuel gas flow rates must be kept at some constant ratio

$$R = \frac{F_A}{F_{FG}}$$

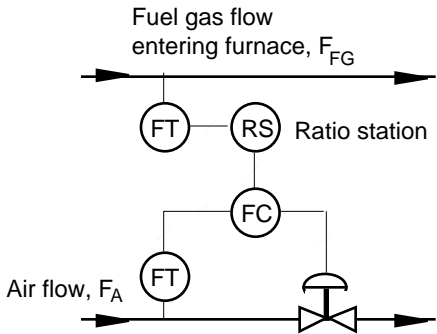


Figure 10.5. Simple ratio control of air flow rate. (10-14)

What we can do easily is to measure the fuel gas flow rate, multiply the value by R in the so-called ratio station, and send the signal as the set point to the air flow controller. The calculation can be based on actual flow rates rather than deviation variables.

A more sophisticated implementation is *full metering control* (Fig. 10.6). In this case, we send the signals from the fuel gas controller (FC in the fuel gas loop) and the air flow transmitter (FT) to the ratio controller (RC), which takes the desired flow ratio (R) as the set point. This controller calculates the proper air flow rate, which in turn becomes the set point to the air flow controller (FC in the air flow loop). If we take away the secondary flow control loops on both the fuel gas and air flow rates, what we have is called parallel positioning control. In this simpler case, of course, the performance of the furnace is subject to fluctuations in fuel and air supply lines.

We are skipping the equations and details since the air flow regulation should not affect the stability and system analysis of the fuel gas controller, and ratio control is best implemented with Simulink in simulation and design projects.

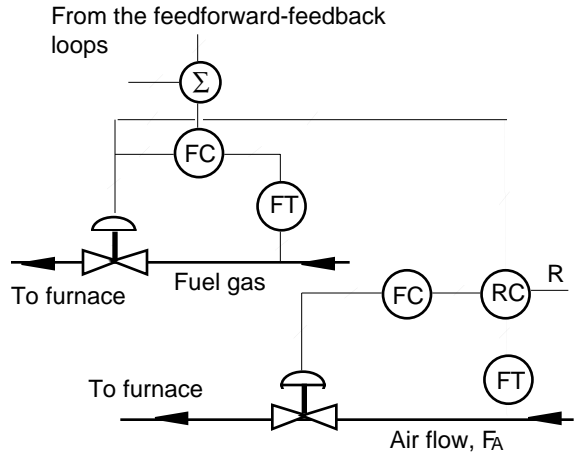


Figure 10.6. Full metering ratio control of fuel and air flows.

10.5 Time delay compensation—the Smith predictor

There are different schemes to handle systems with a large dead time. One of them is the Smith predictor. It is not the most effective technique, but it provides a good thought process.

Consider a unit feedback system with a time delay in its process function (Fig. 10.7). The characteristic polynomial is

$$1 + G_c(s) G(s) e^{-t_d s} = 0 \quad (10-13)$$

We know from frequency response analysis that time lag introduces extra phase lag, reduces the gain margin and is a significant source of instability. This is mainly because the feedback information is outdated.

If we have a model for the process, *i.e.*, we know $G(s)$ and t_d , we can predict what may happen and feedback this estimation. The way the dead time compensator (or predictor) is written (Fig. 10.8), we can interpret the transfer function as follows. Assuming that we know the process model, we feedback the "output" calculation based on this model. We also have to subtract out the "actual" calculated time delayed output information.

Now the error E also includes the feedback information from the dead time compensator:

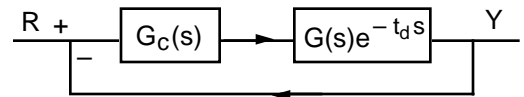


Figure 10.7. System with inherent dead time.

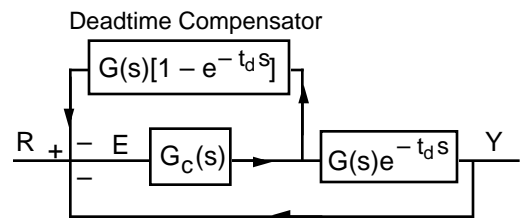


Figure 10.8. Implementation of the Smith predictor.

$$E = R - Y - E \left[G_c G (1 - e^{-t_d s}) \right],$$

and substituting

$$Y = G_c G e^{-t_d s} E$$

we have

$$E = R - E \left[G_c G e^{-t_d s} + G_c G (1 - e^{-t_d s}) \right]$$

where the exponential terms cancel out and we are left with simply

$$E = R - E G_c G \quad (10-14)$$

The time delay effect is canceled out, and this equation at the summing point is equivalent to a system without dead time (where the forward path is $C = G_c G E$). With simple block diagram algebra, we can also show that the closed-loop characteristic polynomial with the Smith predictor is simply

$$1 + G_c G = 0 \quad (10-15)$$

The time delay is removed. With the delay compensator included, we can now use a larger proportional gain without going unstable. Going back to the fact that the feedback information is $G_c G R$, we can also interpret the compensator effect as in Fig. 10.9. The Smith predictor is essentially making use of state feedback as opposed to output feedback.

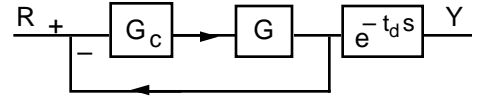


Figure 10.9. An interpretation of the compensator effect.

Just like feedforward control (or any other model-based control), we only have perfect compensation if we know the precise process model. Otherwise, the effectiveness of the compensator (or predictor) is diminished. Assume that we have an imperfect model approximation $H(s)$ and dead time estimation θ ($H \neq G$ and $\theta \neq t_d$), the feedback information is now

$$\begin{aligned} Y + \left[H (1 - e^{-\theta s}) \right] G_c R &= \left[G_c G e^{-t_d s} + G_c H (1 - e^{-\theta s}) \right] R \\ &= G_c \left[G e^{-t_d s} + H (1 - e^{-\theta s}) \right] R \end{aligned}$$

where the right hand side becomes $G_c G R$ if and only if $H = G$ and $\theta = t_d$. Note that the time delay term is an exponential function. Error in the estimation of the dead time is more detrimental than error in the estimation of the process function G .

Since few things are exact in this world, we most likely have errors in the estimation of the process and the dead time. So we only have partial dead time compensation and we must be conservative in picking controller gains based on the characteristic polynomial $1 + G_c G = 0$.

In a chemical plant, time delay is usually a result of transport lag in pipe flow. If the flow rate is fairly constant, the use of the Smith predictor is acceptable. If the flow rate varies for whatever reasons, this compensation method will not be effective.

10.6 Multiple-input Multiple-output Control

In this section, we analyze a multiple input-multiple output (MIMO) system. There are valuable insights that can be gained from using the classical transfer function approach. One decision that we need to appreciate is the proper pairing of manipulated and controlled variables. To do that, we also need to know how strong the interaction is among different variables.

The key points will be illustrated with a blending process. Here, we mix two streams with mass flow rates m_1 and m_2 , and both the total flow rate F and the composition x of a solute A are to be controlled (Fig. 10.10). With simple intuition, we know changes in both m_1 and m_2 will affect F and x . We can describe the relations with the block diagram in Fig. 10.11, where interactions are represented by the two, yet to be derived, transfer functions G_{12} and G_{21} .

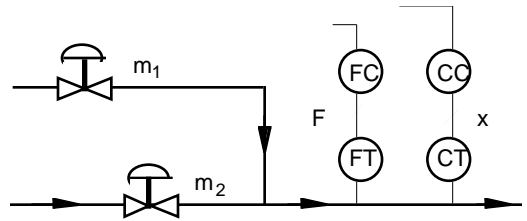


Figure 10.10. A blending system with manipulated and controlled variable pairings yet to be determined.

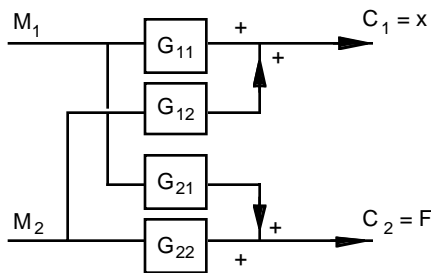


Figure 10.11. Block diagram of an interacting 2 x 2 process, with the output x and F referring to the blending problem.

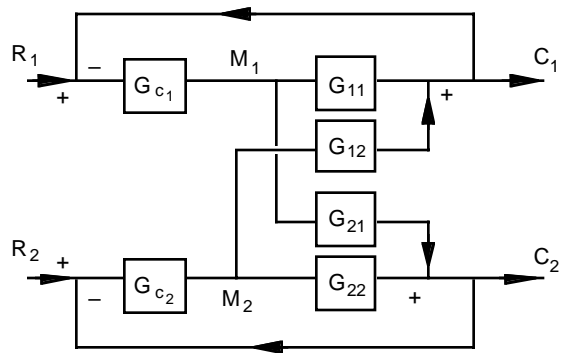


Figure 10.12. Block diagram of a 2 x 2 servo system. The pairing of the manipulated and controlled variables is not necessarily the same as shown in Fig. 10.11.

Given Fig. 10.11 and classical control theory, we can infer the structure of the control system, which is shown in Fig. 10.12. That is, we use two controllers and two feedback loops, where for simplicity, the measurement and actuator functions have been omitted.

Simple reasoning can illustrate now interactions may arise in Fig. 10.12. If a disturbance (not shown in diagram) moves C_1 away from its reference R_1 , the controller G_{c1} will response to the error and alter M_1 accordingly. The change in M_1 , however, will also affect C_2 via G_{21} . Hence C_2 is forced to change and deviate from R_2 . Now the second controller G_{c2} kicks in and adjusts M_2 , which in turn also affects C_1 via G_{12} .

With this scenario, the system may eventually settle, but it is just as likely that the system in Fig. 10.12 will spiral out of control. It is clear that loop interactions can destabilize a control system, and tuning controllers in a MIMO system can be difficult. One logical thing that we can do is to reduce loop interactions by proper pairing of manipulated and controlled variables. This is the focus of the analysis in the following sections.

10.6.1 MIMO Transfer functions

We now derive the transfer functions of the MIMO system. This sets the stage for more detailed analysis that follows. The transfer functions in Fig. 10.11 depend on the process that we have to control, and we'll derive them in the next section for the blending process. Here, we consider a general system as shown in Fig. 10.12.

With the understanding that the errors are $E_1 = R_1 - C_1$, and $E_2 = R_2 - C_2$ in Fig. 10.12, we can write immediately,

$$M_1 = G_{c1} (R_1 - C_1) \quad (10-16)$$

$$M_2 = G_{c2} (R_2 - C_2) \quad (10-17)$$

The process (also in Fig. 10.11) can be written as

$$C_1 = G_{11} M_1 + G_{12} M_2 \quad (10-18)$$

$$C_2 = G_{21} M_1 + G_{22} M_2 \quad (10-19)$$

Substitute for M_1 and M_2 using (10-16, 17), and factor C_1 and C_2 to the left, Eqs. (10-18) and (10-19) becomes

$$(1 + G_{11}G_{c1}) C_1 + G_{12} G_{c2} C_2 = G_{11}G_{c1} R_1 + G_{12} G_{c2} R_2 \quad (10-20)$$

$$G_{21}G_{c1} C_1 + (1 + G_{22}G_{c2}) C_2 = G_{21}G_{c1} R_1 + G_{22} G_{c2} R_2 \quad (10-21)$$

Making use of Kramer's rule, we should identify (derive!) the *system characteristic equation*:

$$p(s) = (1 + G_{11}G_{c1})(1 + G_{22}G_{c2}) + G_{12} G_{21} G_{c1} G_{c2} = 0 \quad (10-22)$$

which, of course, is what governs the dynamics and stability of the system. We may recognize that when either $G_{12} = 0$ or $G_{21} = 0$, the interaction term is zero.⁸ In either case, the system characteristics analysis can be reduced to those of two single loop systems:

$$1 + G_{11}G_{c1} = 0, \quad \text{and} \quad 1 + G_{22}G_{c2} = 0$$

Now back to finding the transfer functions with interaction. To make the algebra appear a bit cleaner, we consider the following two cases. When $R_2 = 0$, we can derive from Eq. (10-20) and (10-21),

$$\frac{C_1}{R_1} = \frac{G_{11}G_{c1} + G_{c1}G_{c2}(G_{11}G_{22} - G_{12}G_{21})}{p(s)} \quad (10-23)$$

And when $R_1 = 0$, we can find

$$\frac{C_1}{R_2} = \frac{G_{12}G_{c2}}{p(s)} \quad (10-24)$$

⁸ When both $G_{12} = G_{21} = 0$, the system is decoupled and behaves identically to two single loops. When either $G_{12} = 0$ or $G_{21} = 0$, the situation is referred to as one-way interaction, which is sufficient to eliminate recursive interactions between the two loops. In such a case, one of the loops is not affected by the second while it becomes a source of disturbance to this second loop.

If both references change simultaneously, we just need to add their effects in (10-23) and (10-24) together. (What about C_2 ? You'll get to try that in the Review Problems.)

It is apparent from Eq. (10-22) that with interaction, the controller design of the MIMO system is different from a SISO system. One logical question is under what circumstances may we make use of SISO designs as an approximation? Or in other words, can we tell if the interaction may be weak? This takes us to the next two sections.

10.6.2 Process gain matrix

We come back to derive the process transfer functions for the blending problem.⁹ The total mass flow balance is

$$F = m_1 + m_2 \quad (10-25)$$

where F is the total flow rate after blending, and m_1 and m_2 are the two inlet flows that we manipulate. The mass balance for a solute A (without using the subscript A explicitly) is

$$xF = x_1 m_1 + x_2 m_2 \quad (10-26)$$

where x is the mass fraction of A after blending, and x_1 and x_2 are the mass fractions of A in the two inlet streams. We want to find the transfer functions as shown in Fig. 10.11:

$$\begin{bmatrix} X(s) \\ F(s) \end{bmatrix} = \begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix} \begin{bmatrix} M_1(s) \\ M_2(s) \end{bmatrix} \quad (10-27)$$

We take stream m_1 to be pure solute A , and stream m_2 to be pure solvent. In this scenario, $x_1 = 1$ and $x_2 = 0$, and Eq. (10-26) is simplified to

$$x = \frac{m_1}{F} = \frac{m_1}{m_1 + m_2} \quad (10-28)$$

Since x_1 and m_1 are functions of time, we need to linearize (10-26). A first order Taylor expansion of x is

$$x \approx \left(\frac{m_1}{F} \right)_s + \left[\frac{m_2}{(m_1 + m_2)^2} \right]_s (m_1 - m_{1,s}) - \left[\frac{m_1}{(m_1 + m_2)^2} \right]_s (m_2 - m_{2,s})$$

where the subscript s of the brackets denotes terms evaluated at steady state. The first term on the right is really the value of x at steady state, x_s , which can be moved to the left hand side to make the deviation variable in x . With that, we take the Laplace transform to obtain the transfer functions of the deviation variables:

$$X(s) = G_{11}(s)M_1(s) + G_{12}(s)M_2(s) \quad (10-29)$$

where

$$G_{11}(s) = \left[\frac{m_2}{(m_1 + m_2)^2} \right]_s = K_{11}, \quad \text{and} \quad G_{12}(s) = - \left[\frac{m_1}{(m_1 + m_2)^2} \right]_s = K_{12} \quad (10-30)$$

⁹ Since the point is to illustrate the analysis of interactions, we are using only steady state balances and it should not be a surprise that the transfer functions end up being only steady state gains in Eq. (10-32). For a general dynamic problem where we have to work with the transfer functions $G_{ij}(s)$, we can still apply the results here by making use of the steady state gains of the transfer functions.

The transfer functions are constants and hence we denote them with the gains K_{11} and K_{12} . If the solvent flow rate m_2 increases, the solute will be diluted. Hence, K_{12} is negative.

The functions G_{21} and G_{22} are much easier. From Eq. (10-25), we can see immediately that

$$G_{21}(s) = K_{21} = 1, \quad \text{and} \quad G_{22}(s) = K_{22} = 1 \quad (10-31)$$

Thus, in this problem, the process transfer function matrix Eq. (10-27) can be written in terms of the steady state gain matrix:

$$\begin{bmatrix} X(s) \\ F(s) \end{bmatrix} = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \begin{bmatrix} M_1(s) \\ M_2(s) \end{bmatrix} \quad (10-32)$$

In more general terms, we replace the LHS of (10-32) with a controlled variable vector:

$$\mathbf{C}(s) = \mathbf{K} \mathbf{M}(s) \quad (10-33)$$

where $\mathbf{C} = [\mathbf{X} \ \mathbf{F}]^T$. If there is a policy such that the manipulated variables can regulate the controlled variables, we must be able to find an inverse of the gain matrix such that

$$\mathbf{M}(s) = \mathbf{K}^{-1} \mathbf{C}(s) \quad (10-34)$$

Example 10.3. If $m_1 = 0.1$ g/s, $m_2 = 10$ g/s, What is the process gain matrix? What are the interpretations?

Making use of (10-30), we can calculate $K_{11} = 9.8 \times 10^{-2}$, and $K_{12} = -9.8 \times 10^{-2}$. With (10-31), the process gain matrix is

$$\mathbf{K} = \begin{bmatrix} 9.8 \times 10^{-2} & -9.8 \times 10^{-2} \\ 1 & 1 \end{bmatrix}$$

Under circumstances of the particular set of numbers given, changing either m_1 or m_2 has a stronger effect on the total flow rate F than x . With respect to the composition x , changing the solute flow m_1 has a much stronger effect than changing the solvent flow. The situation resembles very much a one-way interaction.

We may question other obvious scenarios of the process gain matrix. The sweetest is an identity matrix, meaning no interaction among the manipulated and controlled variables. A quick summary of several simple possibilities:¹⁰

$$\mathbf{K} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{No interaction. Controller design is like single-loop systems.}$$

$$\mathbf{K} = \begin{bmatrix} 1 & \delta \\ \delta & 1 \end{bmatrix} \quad \text{Strong interaction if } \delta \text{ is close to } 1; \text{ weak interaction if } \delta \ll 1.$$

$$\mathbf{K} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \quad \text{One-way interaction}$$

¹⁰ There is more to “looking” at \mathbf{K} . We can, for example, make use of its singular value and condition number, which should be deferred to a second course in control.

10.6.3 Relative gain array

You may not find observing the process gain matrix satisfactory. That takes us to the *relative gain array* (RGA), which can provide for a more quantitative assessment of the effect of changing a manipulated variable on different controlled variables. We start with the blending problem before coming back to the general definition.

For the blending process, the relative gain parameter of the effect of m_1 on x is defined as

$$\lambda_{x, m_1} = \frac{\partial x / \partial m_1 \big|_{m_2}}{\partial x / \partial m_1 \big|_F} \quad (10-35)$$

It is the ratio of the partial derivative evaluated under two different circumstances. On top, we look at the effect of m_1 while holding m_2 constant. The calculation represents an open-loop experiment and the value is referred to as an open-loop gain. In the denominator, the total flow rate, the other controlled variable, is held constant. Since we are varying (in theory) m_1 , F can only be constant if we have a closed-loop with perfect control involving m_2 . The partial derivative in the denominator is referred to as some closed-loop gain.

How do we interpret the relative gain? The idea is that if m_2 does not interfere with m_1 , the derivative in the denominator should not be affected by the closed-loop involving m_2 , and its value should be the same as the open-loop value in the numerator. In other words, if there is no interaction, $\lambda_{x, m_1} = 1$.

Example 10.4. Evaluate the relative gain array matrix for the blending problem.

The complete relative gain array matrix for the 2 x 2 blending problem is defined as

$$\Lambda = \begin{bmatrix} \lambda_{x, m_1} & \lambda_{x, m_2} \\ \lambda_{F, m_1} & \lambda_{F, m_2} \end{bmatrix} \quad (E10-4)$$

For the first element, we use (10-28) to find

$$\frac{\partial x}{\partial m_1} \bigg|_{m_2} = \frac{m_2}{(m_1 + m_2)^2}, \quad \text{and} \quad \frac{\partial x}{\partial m_1} \bigg|_F = \frac{1}{F} = \frac{1}{m_1 + m_2}$$

Hence, with the definition in (10-35),

$$\lambda_{x, m_1} = \frac{m_2}{m_1 + m_2} = 1 - x \quad (E10-5)$$

Proceed to find the other three elements (see Review Problems) and we have the RGA for the blending problem:

$$\Lambda = \begin{bmatrix} 1 - x & x \\ x & 1 - x \end{bmatrix} \quad (E10-6)$$

There are several notable and general points regarding this problem, *i.e.*, without proving them formally here. The sum of all the entries in each row and each column of the relative gain array Λ is 1. Thus in the case of a 2 x 2 problem, all we need is to evaluate one element. Furthermore, the calculation is based on only open-loop information. In Example 10.4, the derivation is based on (10-25) and (10-26).

We can now state the general definition of the **relative gain array**, Λ . For the element relating the i -th controlled variable to the j -th manipulated variable,

$$\lambda_{i,j} = \frac{\partial c_i / \partial m_j \big|_{m_k, k \neq j}}{\partial c_i / \partial m_j \big|_{c_k, k \neq i}} \quad (10-36)$$

where the (open-loop) gain in the numerator is evaluated with all other manipulated variables held constant and all the loops open (no loops!). The (closed-loop) gain in the denominator is evaluated with all the loops—other than the i -th loop—closed. The value of this so-called closed-loop gain reflects the effect from other closed-loops *and* the open-loop between m_j and c_i .

The relative gain array can be derived in terms of the process steady state gains. Making use of the gain matrix equation (10-32), we can find (not that hard; see Review Problems)

$$\lambda_{x,m_1} = \frac{1}{1 - \frac{K_{12}K_{21}}{K_{11}K_{22}}} \quad (10-37)$$

which can be considered a more general form of (E10-5) and hence (E10-6).¹¹

The next question comes back to the meaning of the RGA, and how that may influence our decision in pairing manipulated with controlled variables. Here is the simple interpretation making use of (10-36) and (10-37):

$\lambda_{i,j} = 1$	Requires $K_{12}K_{21} = 0$. “Open-loop” gain is the same as the “closed-loop” gain. The controlled variable (or loop) i is not subject to interaction from other manipulated variables (or other loops). Of course, we know nothing about whether other manipulated variables may interact and affect other controlled variables. Nevertheless, pairing the i -th controlled variable to the j -th manipulated variable is desirable.
$\lambda_{i,j} = 0$	The open-loop gain is zero. The manipulated variable j has no effect on the controlled variable i . Of course m_j may still influence other controlled variables (via one-way interaction). Either way, it makes no sense to pair m_j with c_i in a control loop.
$0 < \lambda_{i,j} < 1$	No doubt there are interactions from other loops, and from (10-37), some of the process gains must have opposite signs (or act in different directions). When $\lambda_{i,j} = 0.5$, we can interpret that the effect of the interactions is identical to the open-loop gain—recall statement after (10-36). When $\lambda_{i,j} > 0.5$, the interaction is less than the main effect of m_j on c_i . However, when $\lambda_{i,j} < 0.5$, the interactive effects predominate and we want to avoid pairing m_j with c_i .
$\lambda_{i,j} > 1$	There are interactions from other loops as well, but now with all the process gains having the same sign. Avoid pairing m_j with c_i if $\lambda_{i,j}$ is much larger than 1.
$\lambda_{i,j} < 0$	We can infer using (10-36) that the open-loop and closed-loop gains have different signs or opposing effects. The overall influence of the

¹¹ For your information, relative gain array can be computed as the so-called Hadamard product, $\lambda_{ij} = K_{ij}K_{ji}^{-1}$, which is the element-by-element product of the gain matrix \mathbf{K} and the transpose of its inverse. You can confirm this by repeating the examples with MATLAB calculations.

other loops is in opposition to the main effect of m_j on c_i . Moreover, from (10-37), the interactive product $K_{12}K_{21}$ must be larger than the direct terms $K_{11}K_{22}$. Undesirable interaction is strong. The overall multiloop system may become unstable easily if we open up one of its loops. We definitely should avoid pairing m_j with c_i .

To sum up, the key is to pair the manipulated and controlled variables such that the relative gain parameter is positive and as close to one as possible.

Example 10.5. If $m_1 = 0.1$ g/s, $m_2 = 10$ g/s, what is the proper pairing of manipulated and controlled variables? What if $m_1 = 9$ g/s, $m_2 = 1$ g/s?

In the first case where m_1 is very small, it is like a dosing problem. From (10-28), $x = 0.0099$. Since $x \ll 1$, λ_{x,m_1} is very close to 1 by (E10-5). Thus interaction is not significant if we pair x with m_1 , and F with m_2 . Physically, we essentially manipulate the total flow with the large solvent flow m_2 and tune the solute concentration by manipulating m_1 .

In the second case, $x = 0.9$. Now $\lambda_{x,m_1} = 0.1$ by (E10-5). Since $\lambda_{x,m_1} \ll 1$, we do not want to pair x with m_1 . Instead, we pair x with m_2 , and F with m_1 . Now we regulate the total flow with the larger solute flow m_1 and tune the concentration with the solvent m_2 .

10.7 Decoupling of interacting systems

After proper pairing of manipulated and controlled variables, we still have to design and tune the controllers. The simplest approach is to tune each loop individually and conservatively while the other loop is in manual mode. At a more sophisticated level, we may try to decouple the loops mathematically into two non-interacting SISO systems with which we can apply single loop tuning procedures. Several examples applicable to a 2×2 system are offered here.

10.7.1 Alternate definition of manipulated variables

We seek choices of manipulated variables that may decouple the system. A simple possibility is to pick them to be the same as the controlled variables. In the blending problem, the two new manipulated variables can be defined as ¹²

$$\mu_1 = F \quad (10-38)$$

and

$$\mu_2 = x \quad (10-39)$$

Once the controller (a computer) evaluates these two manipulated variables, it also computes on the fly the actual signals necessary for the two mass flow rates m_1 and m_2 . The computation

¹² The blending problem can be reduced to one-way interaction if we use m_1 instead of x as the new manipulated variable μ_2 . We'll do that in the Review Problems.

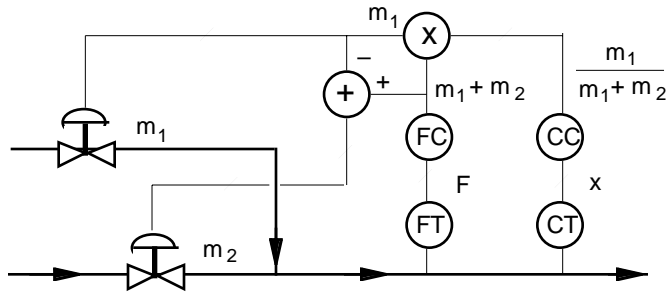


Figure 10.13. A decoupled control scheme. The controller outputs are the manipulated variables in Eqs. (10-38) and (10-39) and they are rewritten based on their definitions in (10-25) and (10-28).

follows directly the balance equations (10-25) and (10-28). Fig. 10.13 is a schematic diagram on how this idea may be implemented.

10.7.2 Decoupler functions

In this section, we add the so-called decoupler functions to a 2×2 system. Our starting point is Fig. 10.12. The closed-loop system equations can be written in matrix form, virtually by visual observation of the block diagram, as

$$\begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} G_{c1} & 0 \\ 0 & G_{c2} \end{bmatrix} \begin{bmatrix} R_1 - C_1 \\ R_2 - C_2 \end{bmatrix} \quad (10-40)$$

In matrix form, this equation looks deceptively simple, but if we expand the algebra, we should arrive at Eqs. (10-20) and (10-21) again.

In a system with interactions, G_{12} and G_{21} are not zero, but we can manipulate the controller signal such that the system appears (mathematically) to be decoupled. So let's try to transform the controller output with a matrix \mathbf{D} , which will contain our decoupling functions. The manipulated variables are now

$$\begin{bmatrix} M_1 \\ M_2 \end{bmatrix} = \begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{bmatrix} \begin{bmatrix} G_{c1} & 0 \\ 0 & G_{c2} \end{bmatrix} \begin{bmatrix} R_1 - C_1 \\ R_2 - C_2 \end{bmatrix}$$

and the system equations become

$$\begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{bmatrix} \begin{bmatrix} G_{c1} & 0 \\ 0 & G_{c2} \end{bmatrix} \begin{bmatrix} R_1 - C_1 \\ R_2 - C_2 \end{bmatrix} = \mathbf{GDG}_c \begin{bmatrix} R_1 - C_1 \\ R_2 - C_2 \end{bmatrix} \quad (10-41)$$

To decouple the system equations, we require that \mathbf{GDG}_c be a diagonal matrix. Define $\mathbf{G}_0 = \mathbf{GDG}_c$, and the previous step can be solved for \mathbf{C} :

$$\begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = [\mathbf{I} + \mathbf{G}_0]^{-1} \mathbf{G}_0 \begin{bmatrix} R_1 \\ R_2 \end{bmatrix} \quad (10-42)$$

Since \mathbf{G}_0 is diagonal, the matrix $[\mathbf{I} + \mathbf{G}_0]^{-1} \mathbf{G}_0$ is also diagonal, and happily, we have two decoupled equations in (10-42).

Now we have to find \mathbf{D} . Since \mathbf{G}_c is already diagonal, we require that \mathbf{GD} be diagonal:

$$\begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{bmatrix} = \begin{bmatrix} H_1 & 0 \\ 0 & H_2 \end{bmatrix} \quad (10-43)$$

A little bit of algebra to match term by term, we should find (see Review Problems)

$$d_{11} = \frac{G_{22} H_1}{G_{11} G_{22} - G_{12} G_{21}}, \quad d_{22} = \frac{G_{11} H_2}{G_{11} G_{22} - G_{12} G_{21}} \quad (10-44)$$

$$d_{21} = \frac{-G_{21}}{G_{22}} d_{11}, \quad d_{12} = \frac{-G_{12}}{G_{11}} d_{22} \quad (10-45)$$

We have six unknowns (four d_{ij} and two H_i) but only four equations. We have to make two (arbitrary) decisions. One possibility is to choose (or define)

$$H_1 = \frac{G_{11} G_{22} - G_{12} G_{21}}{G_{22}}, \quad \text{and} \quad H_2 = \frac{G_{11} G_{22} - G_{12} G_{21}}{G_{11}} \quad (10-46)$$

such that d_{11} and d_{22} become 1. (We can also think in terms of choosing both $d_{11} = d_{22} = 1$ and then derive the relations for H_1 and H_2 .) It follows that

$$d_{21} = \frac{-G_{21}}{G_{22}}, \quad \text{and} \quad d_{12} = \frac{-G_{12}}{G_{11}} \quad (10-47)$$

Now the closed-loop equations are

$$\begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} H_1 & 0 \\ 0 & H_2 \end{bmatrix} \begin{bmatrix} G_{c1} & 0 \\ 0 & G_{c2} \end{bmatrix} \begin{bmatrix} R_1 - C_1 \\ R_2 - C_2 \end{bmatrix} = \begin{bmatrix} H_1 G_{c1} & 0 \\ 0 & H_2 G_{c2} \end{bmatrix} \begin{bmatrix} R_1 - C_1 \\ R_2 - C_2 \end{bmatrix} \quad (10-48)$$

from which we can easily write, for each row of the matrices,

$$\frac{C_1}{R_1} = \frac{G_{c1} H_1}{1 + G_{c1} H_1}, \quad \text{and} \quad \frac{C_2}{R_2} = \frac{G_{c2} H_2}{1 + G_{c2} H_2} \quad (10-49)$$

and the design can be based on the two characteristic equations

$$1 + G_{c1} H_1 = 0, \quad \text{and} \quad 1 + G_{c2} H_2 = 0 \quad (10-50)$$

Recall Eq. (10-46) that H_1 and H_2 are defined entirely by the four plant functions G_{ij} . This is another example of model-based control. With the definitions of H_1 and H_2 given in (10-46), the calculations are best performed with a computer.

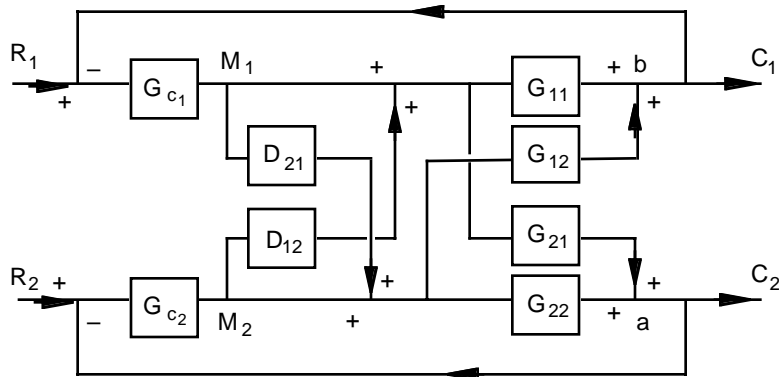


Figure 10.14. A decoupling scheme using two feedforward-like decoupler functions.

10.7.3 Feedforward decoupling functions

A simpler approach is to use only two decoupler functions and implement them as if they were feedforward controllers that may reduce the disturbance arising from loop interaction. As implemented in Fig. 10.14, we use the function D_{12} to “foresee” and reduce the interaction due to G_{12} . Likewise, D_{21} is used to address the interaction due to G_{21} . To find these two decoupling functions, we focus on how to cancel the interaction at the points identified as “a” and “b” in Fig. 10.14.

Let’s pick the point “a” first. If the signal from M_1 through G_{21} can be cancelled by the compensation through D_{21} , we can write

$$G_{21}M_1 + G_{22}D_{21}M_1 = 0$$

Cancel out M_1 and we have

$$D_{21} = -G_{21}/G_{22} \quad (10-51)$$

Similarly, if D_{12} can cancel the effect of G_{12} at the point “b,” we have

$$G_{12}M_2 + G_{11}D_{12}M_2 = 0$$

or

$$D_{12} = -G_{12}/G_{11} \quad (10-52)$$

We may notice that Eqs. (10-51) and (10-52) are the same as d_{21} and d_{12} in (10-47). The strategy of implementing D_{12} and D_{21} is similar to the discussion of feedforward controllers in Section 10.2, and typically we remove the time delay terms and apply a lead-lag compensator as in Eq. (10-8). If the time constant of the first-order lead is similar to time constant of the first-order lag, then we just need a steady state compensator.

Example 10.6: A classic example of an MIMO problem is a distillation column.¹³ From open-loop step tests, the following transfer functions are obtained:

$$\begin{bmatrix} X_D(s) \\ X_B(s) \end{bmatrix} = \begin{bmatrix} \frac{0.07 e^{-3s}}{12s+1} & \frac{-0.05 e^{-s}}{15s+1} \\ \frac{0.1 e^{-4s}}{11s+1} & \frac{-0.15 e^{-2s}}{10s+1} \end{bmatrix} \begin{bmatrix} L(s) \\ V(s) \end{bmatrix}$$

In this model, x_D and x_B are the distillate and bottom compositions, L is the reflux flow rate, and V is the boil-up rate. Design a 2x2 MIMO system with PI controllers and decouplers as in Fig. 10.14.

Before we design the MIMO system, we need to check the pairing of variables. The steady state gain matrix is

¹³ Pardon us if you have not taken a course in separation processes yet, but you do not need to know what a distillation column is to read the example. In a simple-minded way, we can think of making moonshine. We have to boil a dilute alcohol solution at the bottom and we need a condenser at the top to catch the distillate. This is how we have the V and L manipulated variables. Furthermore, the transfer functions are what we obtain from doing an experiment, not from any theoretical derivation.

$$\mathbf{K} = \begin{bmatrix} 0.07 & -0.05 \\ 0.1 & -0.15 \end{bmatrix}$$

With Eq. (10-37) and (E10-6), the relative gain array is

$$\mathbf{\Lambda} = \begin{bmatrix} 1.91 & -0.91 \\ -0.91 & 1.91 \end{bmatrix}$$

The relative gain parameter λ_{x_D-L} is 1.91. It is not 1 but at least it is not negative. Physically, it also makes sense to manipulate the distillate composition with the more neighboring reflux flow. So we will pair x_D-L and x_B-V . Next, with (10-51) and (10-52), the two decoupling functions are

$$D_{12} = K_{d,12} \frac{12s+1}{15s+1}, \quad \text{and} \quad D_{21} = K_{d,21} \frac{10s+1}{11s+1} \approx K_{d,21}$$

To do the tuning, we can use the initial values $K_{d,12} \approx -0.7$ ($-0.05/0.07$), and $K_{d,21} \approx -0.7$ ($-0.1/0.15$).

We will have to skip the details for the remainder of the exercise. You may try to generate a plot similar to Fig. E10.6 in the Review Problems.

This is roughly how we did it. All the simulations are performed with Simulink. First, we use G_{11} and G_{22} as the first order with dead time functions and apply them to the ITAE tuning relations in Table 6.1. With that, we have the PI controller settings of two SISO systems. The single loop response to a unit step change in the set point of x_D is labeled SISO in Fig. E10.6.

We retain the ITAE controller settings and apply them to a Simulink block diagram constructed as in Fig. 10.12. The result is labeled MIMO in the figure. Finally, we use Fig. 10.14 and the two decouplers, and the simulation result with the initial setting is labeled “MIMO with decouplers.”

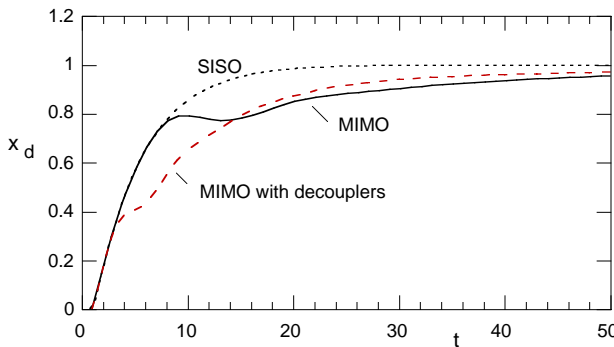


Figure E10.6

In this illustration, we do not have to detune the SISO controller settings. The interaction does not appear to be severely detrimental mainly because we have used the conservative ITAE settings. It would not be the case if we had tried Cohen-Coon relations. The decouplers also do not appear to be particularly effective. They reduce the oscillation, but also slow down the system response. The main reason is that the lead-lag compensators do not factor in the dead times in all the transfer functions.

□ Review Problems

1. Derive (10-3) and (10-3a) with measurement transfer functions G_{m1} and G_{m2} in the primary and secondary loops. Confirm the footnote to (10-3a) that this equation can be reduced to that of a single loop system.
2. Do the root locus plots in Example 10-1(d). Confirm the stability analysis in Example 10-1(e).
3. Draw the block diagram of the system in Example 10-2. Label the diagram with proper variables.
4. Attempt a numerical simulation of a feedforward-feedback system in Fig R10.4. Consider the simplified block diagram with

$$G_v = \frac{0.5}{s+1}, \quad G_p = \frac{0.8}{2s+1}, \quad \text{and} \quad G_L = \frac{-0.4}{2s+1}.$$

- (a) The load function has a negative gain. What does it mean?
- (b) Omit for the moment the feedback loop and controller G_C , and consider only G_{FF} as defined in (10-8). Use MATLAB functions (or simulink) to simulate the response in C when we impose a unit step change to L . Experiment with different values of the gain and time constants in the lead-lag element.

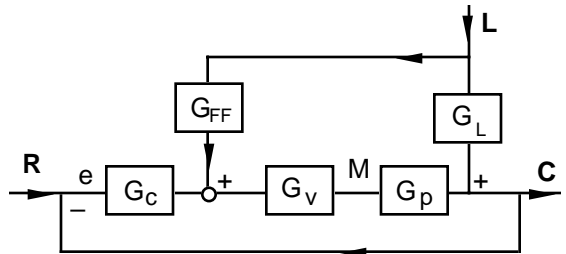


Figure R10.4

- (c) Consider a PI controller for the feedback loop with an integral time of 1.5 s, find the proportional gain such that the system has an underdamped behavior equivalent to a damping ratio of 0.7.
 - (d) With the feedback loop and PI controller in part (c), use MATLAB to simulate the response of C to a unit step change in L . Repeat the different values of the feedforward controller as in part (b).
5. Consider the simpler problem in Fig. R10.5 based on Fig. 10.12. If we only implement one feedback loop and one controller, how is the transfer function C_1/M_1 affected by the interaction?

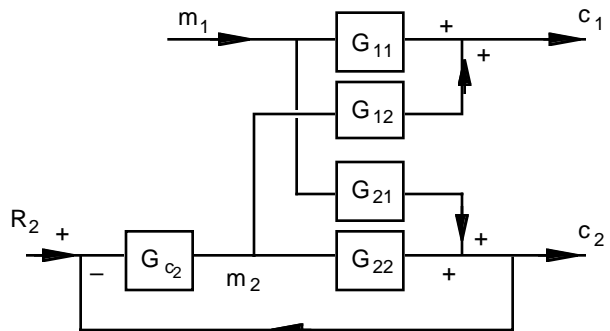


Figure R10.5

7. Fill in the details and derive the RGA (E10-6) in Example 10.4.
8. Derive Eq. (10-37).
9. Show that we also can obtain (E10-6) by applying (10-37) to the blending problem.
- (f) Repeat Section 10.7.1 by replacing the second manipulated variable in (10-39) with

$$\mu_2 = m_1$$

Find the gain matrix and show that the relative gain parameter is 1. Show how this partially decoupling scheme can be implemented as analogous to Fig. 10.13.

11. Derive Eqs. (10-44) and (10-45).
12. Try to do the Simulink simulations in Example 10.6. If you need help the Simulink file is on our *Web Support*.

Hint:

2. The MATLAB statements can be:

```
Part (d)
Gp=tf(0.8,[2 1]);
Gv=tf(0.9,[0.1 1]); %With cascade control
taui=0.05;           %Just one example
Gi=tf([taui 1],[taui 0])
rlocus(Gi*Gv*Gp)
```

```
Part (e)
Gvo=tf(0.5,[1 1]);
rlocus(Gi*Gvo*Gp)
```

4. (a) If L is the inlet process stream flow rate, how would it affect the furnace temperature?
 (b) Use (10-9), and the comments that follow, to select the parameters for the feedforward controller. Compare with the case when we do not have a feedforward controller by setting $K_{FF} = 0$. You should observe that the major compensation to the load change is contributed by the steady-state compensator.
 (c) The proportional gain is about 1.4. The feedforward controller does not affect the system stability and we can design the controller G_c with only G_v , G_p , and the feedback loop. We have to use, for example, the root locus method in Chapter 6 to do this part. Root locus can also help us to determine if $\tau_1 = 1.5$ s is a good choice.
 (d) You should find that the feedback loop takes over much of the burden in load changes. The system response is rather robust even with relatively large errors in the steady-state compensator.

$$5. \quad C_2 = G_{22}G_{c_2}(R_2 - C_2) + G_{21}M_1$$

$$C_1 = G_{11}M_1 + G_{12}G_{c_2}(R_2 - C_2)$$

Setting $R_2 = 0$,

$$C_2 = \frac{G_{21}}{1 + G_{c_2}G_{22}} M_1$$

Substitute C_2 into the C_1 equation, we can find after two algebraic steps,

$$C_1 = \left[G_{11} - \frac{G_{12}G_{21}G_{c_2}}{1 + G_{c_2}G_{22}} \right] M_1$$

The second term in the bracket is due to interaction.

6. We apply Kramer's rule to find C_2 just as we had with C_1 . The solution has the same characteristic polynomial in (10-22). The transfer functions:

With $R_1 = 0$,

$$\frac{C_2}{R_2} = \frac{G_{22}G_{c_2} + G_{c_1}G_{c_2}(G_{11}G_{22} - G_{12}G_{21})}{p(s)}$$

With $R_2 = 0$,

$$\frac{C_2}{R_1} = \frac{G_{21}G_{c_1}}{p(s)}$$

7. We still use (10-28) as in Example 10.4. To find λ_{x,m_2} :

$$\left. \frac{\partial x}{\partial m_2} \right|_{m_1} = \frac{-m_1}{(m_1 + m_2)^2} \quad \left. \frac{\partial x}{\partial m_2} \right|_F = \frac{\partial}{\partial m_2} \left(\frac{F - m_2}{F} \right) = -\frac{1}{F}$$

and

$$\lambda_{x,m_2} = \frac{m_1}{m_1 + m_2} = x$$

To find λ_{F,m_1} :

$$\left. \frac{\partial F}{\partial m_1} \right|_{m_2} = 1, \text{ using Eq. (10-25)} \quad \left. \frac{\partial F}{\partial m_1} \right|_x = \frac{1}{x}, \text{ using } F = m_1/x$$

$$\lambda_{F,m_1} = x$$

To find λ_{F,m_2} :

$$\left. \frac{\partial F}{\partial m_2} \right|_{m_1} = 1 \quad \left. \frac{\partial F}{\partial m_2} \right|_x = \frac{1}{1-x}, \text{ using } xF = F - m_2, F = m_2/(1-x)$$

$$\lambda_{F,m_2} = 1 - x$$

8. We may just as well use Eq. (10-32) in its time-domain form

$$\begin{bmatrix} x \\ F \end{bmatrix} = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \end{bmatrix}$$

where now x , F , m_1 , and m_2 are deviation variables. From the first row, it is immediately obvious that

$$\left. \frac{\partial x}{\partial m_1} \right|_{m_2} = K_{11}$$

We next substitute for m_2 using the second row to get

$$x = K_{11}m_1 + K_{12} \frac{(F - K_{21}m_1)}{K_{22}}$$

Now we can find

$$\left. \frac{\partial x}{\partial m_1} \right|_F = K_{11} - \frac{K_{12}K_{21}}{K_{22}}$$

From here on, getting (10-37) is a simple substitution step.

9. To derive E10-6 using K. This is just a matter of substituting the values of the K_{ij} 's from (10-30) and (10-31) into (10-37). We should find once again $\lambda_{x,m_1} = 1 - x$ as in (E10-5), and (E10-6) follows.
10. We need to find how μ_1 and μ_2 affect F and x . With $\mu_1 = F$ and $\mu_2 = m_1$, we can rewrite the definition of $x = m_1/F$ as $x = \mu_1/\mu_2$. This is the form that we use to take a first order Taylor

expansion as we have done with the step after Eq. (10-28). The result in matrix form of the Laplace transform of the deviation variables is

$$\begin{bmatrix} F(s) \\ x(s) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -\frac{\mu_2}{\mu_1} \Big|_{s,s.} & \frac{1}{\mu_1} \Big|_{s,s.} \end{bmatrix} \begin{bmatrix} \mu_1(s) \\ \mu_2(s) \end{bmatrix}$$

By putting F in the first row, it is clear that we have a one-way interaction system. By (10-37), $\lambda = 1$. And with $F = m_1 + m_2$ and m_1 as the output of the controllers, we can implement this scheme as in Fig. R10.10.

11. We'll find d_{11} and d_{21} as an illustration. The first column of the RHS of (10-43) is rewritten as the two equations:

$$G_{11} d_{11} + G_{12} d_{21} = H_1$$

$$G_{21} d_{11} + G_{22} d_{21} = 0$$

Solving them simultaneously will lead to d_{11} and d_{21} in (10-44) and (10-45). And choosing $d_{11} = 1$, (10-44) can be rewritten as (10-46).

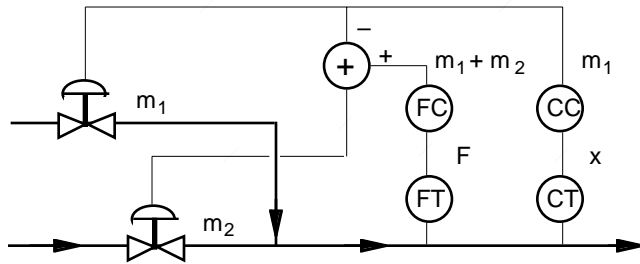


Figure R10.10

❖ 4. State Space Representation

The limitation of transfer function representation becomes plain obvious as we tackle more complex problems. For complex systems with multiple inputs and outputs, transfer function matrices can become very clumsy. In the so-called modern control, the method of choice is state space or state variables in time domain—essentially a matrix representation of the model equations. The formulation allows us to make use of theories in linear algebra and differential equations. It is always a mistake to tackle modern control without a firm background in these mathematical topics. For this reason, we will not overreach by doing both the mathematical background and the control together. Without a formal mathematical framework, we will put the explanation in examples as much as possible. The actual state space control has to be delayed until after tackling classical transfer function feedback systems.

What are we up to?

- How to write the state space representation of a model.
- Understand the how a state space representation is related to the transfer function representation.

4.1 State space models

Just as you are feeling comfortable with transfer functions, we now switch gears totally. Nevertheless, we are still working with *linearized* differential equation models in this chapter. Whether we have a high order differential equation or multiple equations, we can always rearrange them into a set of first order differential equations. Bold statements indeed! We will see that when we go over the examples.

With state space models, a set of differential equations is put in the standard matrix form

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \quad (4-1)$$

and

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \quad (4-2)$$

where \mathbf{x} is the **state variable** vector, \mathbf{u} is the input, and \mathbf{y} is the output. The time derivative is denoted by the overhead dot. In addition, \mathbf{A} is the process (plant) matrix, \mathbf{B} is the input matrix, \mathbf{C} is the output matrix, and \mathbf{D} is the feed-through matrix. Very few processes (and systems) have an input that has a direct influence on the output. Hence \mathbf{D} is usually zero.

When we discuss single-input single-output models, u , y , and D are scalar variables. For convenience, we keep the notation for \mathbf{B} and \mathbf{C} , but keep in mind that in this case, \mathbf{B} is a column vector and \mathbf{C} is a row vector. If \mathbf{x} is of order n , then \mathbf{A} is $(n \times n)$, \mathbf{B} is $(n \times 1)$, and \mathbf{C} is $(1 \times n)$.¹

The idea behind the use of Eqs. (4-1) and (4-2) is that we can make use of linear system theories, and complex systems can be analyzed much more effectively. There is no unique way to define the state variables. What we will show is just one of many possibilities.

¹ If you are working with only single-input single-output (SISO) problems, it would be more appropriate to replace the notation \mathbf{B} by \mathbf{b} and \mathbf{C} by \mathbf{c}^T , and write d for \mathbf{D} .

▮ **Example 4.1:** Derive the state space representation of a **second order differential equation** of a form similar to Eq. (3-16) [on page 3-5](#):

$$\frac{d^2 y}{dt^2} + 2\zeta\omega_n \frac{dy}{dt} + \omega_n^2 y = Ku(t) \quad (\text{E4-1})$$

We can do blindfolded that the transfer function of this equation, with zero initial conditions, is

$$G_p(s) = \frac{Y(s)}{U(s)} = \frac{K}{s^2 + 2\zeta\omega_n s + \omega_n^2} \quad (\text{E4-2})$$

Now let's do something different. First, we rewrite the differential equation as

$$\frac{d^2 y}{dt^2} = -2\zeta\omega_n \frac{dy}{dt} - \omega_n^2 y + Ku(t)$$

and define state variables ¹

$$x_1 = y \quad \text{and} \quad x_2 = \frac{dx_1}{dt} \quad (\text{E4-3})$$

which allow us to redefine the second order equation as a set of two coupled first order equations. The first equation is the definition of the state variable x_2 in (E4-3); the second equation is based on the differential equation,

$$\frac{dx_2}{dt} = -2\zeta\omega_n x_2 - \omega_n^2 x_1 + Ku(t) \quad (\text{E4-4})$$

We now put the result in a matrix equation:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta\omega_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ K \end{bmatrix} u(t) \quad (\text{E4-5})$$

We further write

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (\text{E4-6})$$

as a statement that x_1 is our output variable. Compare the results with Eqs. (4-1) and (4-2), and we see that in this case,

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta\omega_n \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} 0 \\ K \end{bmatrix}; \quad \mathbf{C} = \begin{bmatrix} 1 & 0 \end{bmatrix}; \quad \mathbf{D} = 0$$

To find the eigenvalues of \mathbf{A} , we solve its characteristic equation:

$$|\lambda \mathbf{I} - \mathbf{A}| = \lambda(\lambda + 2\zeta\omega_n) + \omega_n^2 = 0 \quad (\text{E4-7})$$

We can use the MATLAB function `tf2ss()` to convert the transfer function in (E4-2) to state space form:

¹ This exercise is identical to how we handle higher order equations in numerical analysis and would come as no surprise if you have taken a course on numerical methods.

```

z=0.5; wn=1.5; % Pick two sample numbers for ζ and ωn
p=[1 2*z*wn wn*wn];
[a,b,c,d]=tf2ss(wn*wn,p)

```

However, you will find that the MATLAB result is not identical to (E4-5). It has to do with the fact that there is no unique representation of a state space model. To avoid unnecessary confusion, the differences with MATLAB are explained in MATLAB Session 4.

One important observation that we should make immediately: the characteristic polynomial of the matrix **A** (E4-7) is *identical* to that of the transfer function (E4-2). Needless to say that the eigenvalues of **A** are the poles of the transfer function. It is a reassuring thought that different mathematical techniques provide the same information. It should come as no surprise if we remember our linear algebra.

Example 4.2: Draw the block diagram of the state space representation of the **second order differential equation** in the previous example.

The result is in Fig. E4.2. It is quite easy to understand if we take note that the transfer function of an integrator is $1/s$. Thus the second order derivative is located prior to the two integrations. The information at the summing point

also adds up to the terms of the second order differential equation. The resulting transfer function is identical to (E4-2). The reduction to a closed-loop transfer function was done in Example 2.16 (p. 2-33).

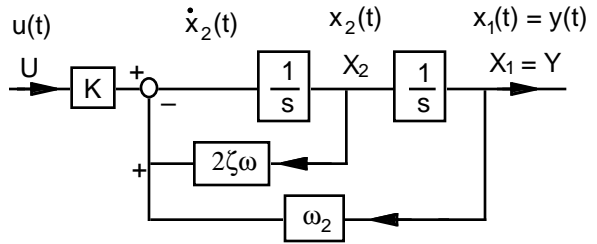


Figure E4.2

Example 4.3: Let's try another model with a slightly more complex input. Derive the state space representation of the differential equation

$$\frac{d^2 y}{dt^2} + 0.4 \frac{dy}{dt} + y = \frac{du}{dt} + 3u, \quad y(0) = dy/dt(0) = 0, \quad u(0) = 0,$$

which has the transfer function $\frac{Y}{U} = \frac{s+3}{s^2+0.4s+1}$.

The method that we will follow is more for illustration than for its generality. Let's introduce a variable X_1 between Y and U :

$$\frac{Y}{U} = \frac{X_1}{U} \frac{Y}{X_1} = \left(\frac{1}{s^2+0.4s+1} \right) (s+3)$$

The first part X_1/U is a simple problem itself.

$$\frac{X_1}{U} = \left(\frac{1}{s^2+0.4s+1} \right) \quad \text{is the Laplace transformed of} \quad \frac{d^2 x_1}{dt^2} + 0.4 \frac{dx_1}{dt} + x_1 = u$$

With Example 4.1 as the hint, we define the state variables $x_1 = x_1$ (i.e., the same), and $x_2 = dx_1/dt$. Using steps similar to Example 4.1, the result as equivalent to Eq. (4-1) is

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & -0.4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \quad (\text{E4-8})$$

As for the second part $Y/X_1 = (s + 3)$, it is the Laplace transformed of $y = \frac{dx_1}{dt} + 3x_1$. We can use the state variables defined above to rewrite as

$$y = x_2 + 3x_1, \text{ or in matrix form } y = \begin{bmatrix} 3 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad (\text{E4-9})$$

which is the form of Eq. (4-2).

With MATLAB, the statements for this example are:

```
q=[1 3];
p=[1 0.4 1];
roots(p)
[a,b,c,d]=tf2ss(q,p)
eig(a)
```

Comments at the end of Example 4.1 also apply here. The result should be correct, and we should find that both the roots of the characteristic polynomial p and the eigenvalues of the matrix a are $-0.2 \pm 0.98j$. We can also check by going backward:

```
[q2,p2]=ss2tf(a,b,c,d,1)
```

and the original transfer function is recovered in $q2$ and $p2$.

✎ **Example 4.4:** Derive the state space representation of the **lead-lag** transfer function

$$\frac{Y}{U} = \frac{s+2}{s+3}.$$

We follow the hint in Example 4.3 and write the transfer function as

$$\frac{Y}{U} = \frac{X}{U} \frac{Y}{X} = \frac{1}{s+3} s + 2$$

From $X/U = 1/(s+3)$, we have $sX = -3X + U$, or in time domain,

$$\frac{dx}{dt} = -3x + u \quad (\text{E4-10})$$

and from $Y/X = s+2$, we have $Y = sX + 2X$ and substitution for sX leads to

$$Y = (-3X + U) + 2X = -X + U$$

The corresponding time domain equation is

$$y = -x + u \quad (\text{E4-11})$$

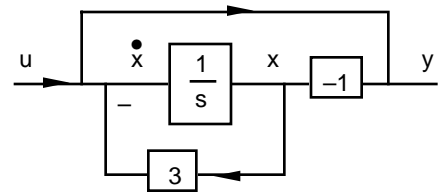


Figure E4.4

Thus all the coefficients in Eqs. (4-1) and (4-2) are scalar, with $A = -3$, $B = 1$, $C = -1$, and $D = 1$. Furthermore, (E4-10) and (E4-11) can be represented by the block diagram in Fig. E4.4.

We may note that the coefficient D is not zero, meaning that with a lead-lag element, an input can have instantaneous effect on the output. Thus while the state variable x has zero initial condition, it is not necessarily so with the output y . This analysis explains the mystery with the inverse transform of this transfer function in Eq. (3-49) on page 3-15.

The MATLAB statement for this example is:

```
[a,b,c,d]=tf2ss([1 2], [1 3])
```

In the next two examples, we illustrate how state space models can handle a multiple-input multiple output (MIMO) problem. We'll show, with a simple example, how to translate information in a block diagram into a state space model. Some texts rely on signal-flow graphs, but we do not need them with simple systems. Moreover, we can handle complex problems easily with MATLAB. Go over MATLAB Session 4 before reading Example 4.7A.

Example 4.5: Derive the state space representation of two continuous flow stirred-tank reactors in series (**CSTR-in-series**). Chemical reaction is first order in both reactors. The reactor volumes are fixed, but the volumetric flow rate and inlet concentration are functions of time.

We use this example to illustrate how state space representation can handle complex models. First, we make use of the solution to Review Problem 2 in Chapter 3 (p. 3-18) and write the mass balances of reactant A in chemical reactors 1 and 2:

$$V_1 \frac{dc_1}{dt} = q(c_o - c_1) - V_1 k_1 c_1 \quad (\text{E4-12})$$

and
$$V_2 \frac{dc_2}{dt} = q(c_1 - c_2) - V_2 k_2 c_2 \quad (\text{E4-13})$$

Since q and c_o are input functions, the linearized equations in deviation variables and with zero initial conditions are (with all apostrophes omitted in the notations):

$$V_1 \frac{dc_1}{dt} = q_s c_o + (c_{os} - c_{1s}) q - (q_s + V_1 k_1) c_1 \quad (\text{E4-14})$$

and
$$V_2 \frac{dc_2}{dt} = q_s c_1 + (c_{1s} - c_{2s}) q - (q_s + V_2 k_2) c_2 \quad (\text{E4-15})$$

The missing steps are very similar to how we did Example 2.11 (p. 2-28). Divide the equations by the respective reactor volumes and define space times $\tau_1 = V_1/q_s$ and $\tau_2 = V_2/q_s$, we obtain

$$\frac{dc_1}{dt} = \frac{1}{\tau_1} c_o + \left(\frac{c_{os} - c_{1s}}{V_1} \right) q - \left(\frac{1}{\tau_1} + k_1 \right) c_1 \quad (\text{E4-16})$$

and
$$\frac{dc_2}{dt} = \frac{1}{\tau_2} c_1 + \left(\frac{c_{1s} - c_{2s}}{V_2} \right) q - \left(\frac{1}{\tau_2} + k_2 \right) c_2 \quad (\text{E4-17})$$

Up to this point, the exercise is identical to what we learned in Chapter 2. In fact, we can

now take the Laplace transform of these two equations to derive the transfer functions. In state space models, however, we would put the two linearized equations in matrix form. As analogous to Eq. (4-1), we now have

$$\frac{d}{dt} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} -(\frac{1}{\tau_1} + k_1) & 0 \\ \frac{1}{\tau_2} & -(\frac{1}{\tau_2} + k_2) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + \begin{bmatrix} \frac{1}{\tau_1} & \frac{c_{os} - c_{1s}}{V_1} \\ 0 & \frac{c_{1s} - c_{2s}}{V_2} \end{bmatrix} \begin{bmatrix} c_o \\ q \end{bmatrix}, \quad (\text{E4-18})$$

The output y in Eq. (4-2) can be defined as

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \quad (\text{E4-19})$$

if we are to use two outputs. In SISO problems, we likely would only measure and control c_2 , and hence we would define instead

$$y = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \quad (\text{E4-20})$$

with c_2 as the only output variable.

✎ **Example 4.6.** Derive the transfer function Y/U and the corresponding state space model of the block diagram in Fig. E4.6.

From Chapter 2 block diagram reduction, we can easily spot that

$$\frac{Y}{U} = \frac{\frac{2}{s(s+1)}}{1 + \frac{2}{s(s+1)(s+10)}},$$

which is reduced to

$$\frac{Y}{U} = \frac{2(s+10)}{s^3 + 11s^2 + 10s + 2} \quad (\text{E4-21})$$

This transfer function has closed-loop poles at -0.29 , -0.69 , and -10.02 . (Of course, we computed them using MATLAB.)

To derive the state space representation, one visual approach is to identify locations in the block diagram where we can assign state variables and write out the individual transfer functions. In this example, we have chosen to use (Fig. E4.6)

$$\frac{X_1}{X_2} = \frac{1}{s}; \quad \frac{X_2}{U - X_3} = \frac{2}{s+1}; \quad \frac{X_3}{X_1} = \frac{1}{s+10}; \quad \text{and the output equation } Y = X_1$$

We can now rearrange each of the three transfer functions from which we write their time domain equivalent:

$$sX_1 = X_2 \quad \frac{dx_1}{dt} = x_2 \quad (\text{E4-22a})$$

$$sX_2 = -X_2 - 2X_3 + 2U \quad \frac{dx_2}{dt} = -x_2 - 2x_3 + 2u \quad (\text{E4-22b})$$

$$sX_3 = -10X_3 + X_1 \quad \frac{dx_3}{dt} = x_1 - 10x_3 \quad (\text{E4-22c})$$

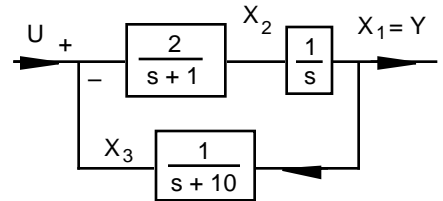


Figure E4.6

The rest is trivial. We rewrite the differential equations in matrix form as

$$\frac{d}{dt} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -1 & -2 \\ 1 & 0 & -10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix} u, \text{ and } y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad (\text{E4-23, 24})$$

We can check with MATLAB that the model matrix \mathbf{A} has eigenvalues -0.29 , -0.69 , and -10.02 . They are identical with the closed-loop poles. Given a block diagram, MATLAB can put the state space model together for us easily. To do that, we need to learn some closed-loop MATLAB functions, and we will defer this illustration to MATLAB Session 5.

An important reminder: Eq. (E4-23) has zero initial conditions $\mathbf{x}(0) = 0$. This is a direct consequence of deriving the state space representation from transfer functions. Otherwise, Eq. (4-1) is not subjected to this restriction.

4.2 Relation with transfer function models

From the last example, we may see why the primary mathematical tools in modern control are based on linear system theories and time domain analysis. Part of the confusion in learning these more advanced techniques is that the umbilical cord to Laplace transform is not entirely severed, and we need to appreciate the link between the two approaches. On the bright side, if we can convert a state space model to transfer function form, we can still make use of classical control techniques. A couple of examples in Chapter 9 will illustrate how classical and state space techniques can work together.

We can take the Laplace transform of the matrix equation in Eq. (4-1) to give

$$s\mathbf{X}(s) = \mathbf{A}\mathbf{X}(s) + \mathbf{B}U(s) \quad (4-3)$$

where the capital \mathbf{X} does not mean that it is a matrix, but rather it is used in keeping with our notation of Laplace variables. From (4-3), we can extract \mathbf{X} explicitly as

$$\mathbf{X}(s) = (s\mathbf{I} - \mathbf{A})^{-1} \mathbf{B}U(s) = \Phi(s)\mathbf{B}U(s) \quad (4-4)$$

where

$$\Phi(s) = (s\mathbf{I} - \mathbf{A})^{-1} \quad (4-5)$$

is the resolvent matrix. More commonly, we refer to the **state transition matrix** (also called the fundamental matrix) which is its inverse transform

$$\Phi(t) = \mathcal{L}^{-1}[(s\mathbf{I} - \mathbf{A})^{-1}] \quad (4-6)$$

We will use the same notation Φ for the time function and its Laplace transform, and only add the t or s dependence when it is not clear in which domain the notation is used.

Setting $\mathbf{D} = 0$ and $\mathbf{X}(s)$ derived in (4-4), the output $Y(s) = \mathbf{C}\mathbf{X}(s)$ becomes

$$Y(s) = \mathbf{C}\Phi(s)\mathbf{B}U(s) \quad (4-7)$$

In this case where U and Y are scalar quantities, $\mathbf{C}\Phi\mathbf{B}$ must also be scalar.¹ In fact, if we make an association between Eq. (4-7) and what we have learned in Chapter 2, $\mathbf{C}\Phi\mathbf{B}$ is our ubiquitous transfer function. We can rewrite Eq. (4-7) as

$$Y(s) = G_p(s)U(s), \quad (4-7a)$$

where

$$G_p(s) = \mathbf{C}\Phi(s)\mathbf{B} \quad (4-8)$$

Hence, we can view the transfer function as how the Laplace transform of the state transition matrix Φ mediates the input \mathbf{B} and the output \mathbf{C} matrices. We may wonder how this output equation is tied to the matrix \mathbf{A} . With linear algebra, we can rewrite the definition of Φ in Eq. (4-5) as

$$\Phi(s) = (\mathbf{sI} - \mathbf{A})^{-1} = \frac{\text{adj}(\mathbf{sI} - \mathbf{A})}{\det(\mathbf{sI} - \mathbf{A})} \quad (4-5a)$$

Substitution of this form in (4-8) provides a more informative view of the transfer function:

$$G_p(s) = \frac{\mathbf{C} [\text{adj}(\mathbf{sI} - \mathbf{A})] \mathbf{B}}{\det(\mathbf{sI} - \mathbf{A})} \quad (4-8a)$$

The characteristic polynomial clearly is

$$\det(\mathbf{sI} - \mathbf{A}) = 0 \quad (4-9)$$

This is the result that we have arrived at, albeit less formally, in Example 4.1. Again, the poles of G_p are identical to the eigenvalues of the model matrix \mathbf{A} .

✎ **Example 4.7:** We'll illustrate the results in this section with a numerical version of Example 4.5. Consider again **two CSTR-in-series**, with $V_1 = 1 \text{ m}^3$, $V_2 = 2 \text{ m}^3$, $k_1 = 1 \text{ min}^{-1}$, $k_2 = 2 \text{ min}^{-1}$, and initially at steady state, $\tau_1 = 0.25 \text{ min}$, $\tau_2 = 0.5 \text{ min}$, and inlet concentration $c_{os} = 1 \text{ kmol/m}^3$. Derive the transfer functions and state transition matrix where both c_o and q are input functions.

With the steady state form of (E4-12) and (E4-13), we can calculate

$$c_{1s} = \frac{c_{os}}{1 + k_1 \tau_1} = \frac{1}{1 + 0.25} = 0.8, \text{ and } c_{2s} = \frac{c_{1s}}{1 + k_2 \tau_2} = \frac{0.8}{1 + 2(0.5)} = 0.4$$

In addition, we find $1/\tau_1 = 4 \text{ min}^{-1}$, $1/\tau_2 = 2 \text{ min}^{-1}$, $(1/\tau_1 + k_1) = 5 \text{ min}^{-1}$, $(1/\tau_2 + k_2) = 4 \text{ min}^{-1}$, $(c_{os} - c_{1s})/V_1 = 0.2 \text{ kmol/m}^6$, and $(c_{1s} - c_{2s})/V_2 = 0.2 \text{ kmol/m}^6$. We substitute these numerical values in (E4-16) and (E4-17), and take the Laplace transform of these equations to obtain (for more general algebraic result, we should take the transform first)

$$C_1(s) = \frac{4}{s + 5} C_o(s) + \frac{0.2}{s + 5} Q(s) \quad (E4-25)$$

and

$$C_2(s) = \frac{2}{s + 4} C_1(s) + \frac{0.2}{s + 4} Q(s)$$

¹ From Eq. (4-5), we see that Φ is a $(n \times n)$ matrix. Since \mathbf{B} is $(n \times 1)$, and \mathbf{C} is $(1 \times n)$, $\mathbf{C}\Phi\mathbf{B}$ must be (1×1) .

Further substitution for $C_1(s)$ with (E4-25) in $C_2(s)$ gives

$$C_2(s) = \frac{8}{(s+4)(s+5)} C_o(s) + \frac{0.2(s+7)}{(s+4)(s+5)} Q(s) \quad (\text{E4-26})$$

Equations (E4-25) and (E4-26) provide the transfer functions relating changes in flow rate Q and inlet concentration C_o to changes in the two tank concentrations.

With the state space model, substitution of numerical values in (E4-18) leads to the dynamic equations

$$\frac{d}{dt} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} -5 & 0 \\ 2 & -4 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + \begin{bmatrix} 4 & 0.2 \\ 0 & 0.2 \end{bmatrix} \begin{bmatrix} c_o \\ q \end{bmatrix} \quad (\text{E4-27})$$

With the model matrix \mathbf{A} , we can derive

$$(s\mathbf{I} - \mathbf{A}) = \begin{bmatrix} s+5 & 0 \\ -2 & s+4 \end{bmatrix},$$

and
$$\Phi(s) = (s\mathbf{I} - \mathbf{A})^{-1} = \frac{1}{(s+5)(s+4)} \begin{bmatrix} s+4 & 0 \\ 2 & s+5 \end{bmatrix} \quad (\text{E4-28})$$

We will consider (E4-19) where both concentrations c_1 and c_2 are outputs of the model. The transfer function in (4-7) is now a matrix

$$\mathbf{G}_p(s) = \mathbf{C}\Phi(s)\mathbf{B} = \frac{1}{(s+5)(s+4)} \begin{bmatrix} s+4 & 0 \\ 2 & s+5 \end{bmatrix} \begin{bmatrix} 4 & 0.2 \\ 0 & 0.2 \end{bmatrix} \quad (\text{E4-29})$$

where \mathbf{C} is omitted as it is just the identity matrix (E4-19).¹ With input $\mathbf{u}(s) = [C_o(s) \quad Q(s)]^T$, we can write the output equation (4-6) as

$$\begin{bmatrix} C_1(s) \\ C_2(s) \end{bmatrix} = \mathbf{C}\Phi(s)\mathbf{B}\mathbf{u}(s) = \frac{1}{(s+5)(s+4)} \begin{bmatrix} 4(s+4) & 0.2(s+4) \\ 8 & 0.2(s+7) \end{bmatrix} \begin{bmatrix} C_o(s) \\ Q(s) \end{bmatrix} \quad (\text{E4-30})$$

This is how MATLAB returns the results. We of course can clean up the algebra to give

$$\begin{bmatrix} C_1(s) \\ C_2(s) \end{bmatrix} = \begin{bmatrix} \frac{4}{(s+5)} & \frac{0.2}{(s+5)} \\ \frac{8}{(s+5)(s+4)} & \frac{0.2(s+7)}{(s+5)(s+4)} \end{bmatrix} \begin{bmatrix} C_o(s) \\ Q(s) \end{bmatrix} \quad (\text{E4-30a})$$

which is identical to what we have obtained earlier in (E4-25) and (E4-26). The case of only one output as in (E4-20) is easy and we'll cover that in Example 4.7A.

To wrap things up, we can take the inverse transform of (E4-30a) to get the time domain solutions:

$$\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 4e^{-5t} & 0.2e^{-5t} \\ 8(e^{-4t} - e^{-5t}) & 0.2(3e^{-4t} - 2e^{-5t}) \end{bmatrix} \begin{bmatrix} c_o \\ q \end{bmatrix} \quad (\text{E4-31})$$

¹ Be careful with the notation. Upper case C is for concentration in the Laplace domain. The boldface upper case \mathbf{C} is the output matrix.

▮ **Example 4.7A:** Repeat Example 4.7 using MATLAB.

If you understand the equations in Example 4.7, we are ready to tackle the same problem with MATLAB.

```

t1=0.25; t2=0.5;           % Define the variables
k1=1; k2=2;
V1=1; V2=2;
cos=1;

% Calculate the steady state values. MATLAB should return
c1s=cos/(1+k1*t1);         % 0.8
c2s=c1s/(1+k2*t2);         % 0.4

                                % Coefficients of A and B using (E4-18)
a11=-(1/t1+k1);            % -5
a12=0;
a21=1/t2;
a22=-(1/t2+k2);            % -4
b11=1/t1;
b12=(cos-c1s)/V1;          % 0.2
b21=0;
b22=(c1s-c2s)/V2;          % 0.2

                                % Finally build A and B in (E4-27)
a=[a11 a12; a21 a22];      % [-5 0; 2 4]
b=[b11 b12; b21 b22];      % [4 0.2; 0 0.2]
eig(a)                      % Check that they are -4, -5

c=[1 0; 0 1];              % Define C such that both  $C_1$  and  $C_2$  are outputs
d=[0 0; 0 0];

```

With all the coefficient matrices defined, we can do the conversion to transfer function. The function `ss2tf()` works with only one designated input variable. Thus, for the *first* input variable C_0 , we use

```

                                % MATLAB returns, for input no. 1
[q1,p]=ss2tf(a,b,c,d,1)      % q1=[0 4 16; 0 0 8]
                                % p=[1 9 20] = (s+4)(s+5)

```

The returned vector p is obviously the characteristic polynomial. The matrix $q1$ is really the first *column* of the transfer function matrix in Eq. (E4-30), denoting the two terms describing the effects of changes in C_0 on C_1 and C_2 . Similarly, the second column of the transfer function matrix in (E4-30) is associated with changes in the *second* input Q , and can be obtained with:

```

[q2,p]=ss2tf(a,b,c,d,2)      % q2=[0 .2 .8; 0 .2 1.4]
                                % The first term is 0.2(s+4) because
                                % MATLAB retains p=(s+4)(s+5)

```

If C_2 is the only output variable, we define C according to the output equation (E4-20). Matrices A and B remain the same. The respective effects of changes of C_0 and Q on C_2 can be obtained with

```

c=[0 1]; d=[0 0];           %  $C_2$  is the only output
[q21,p]=ss2tf(a,b,c,d,1)    %  $C_0$  as input; q21=[0 0 8]
[q22,p]=ss2tf(a,b,c,d,2)    %  $Q$  as input; q22=[0 .2 1.4]

```

We should find that the result is the same as the second *row* of (E4-30), denoting the two terms describing the effects of changes in C_0 and Q on C_2 .

Similarly, if C_1 is the only output variable, we use instead:

```
c=[1 0]; d=[0 0]; % C1 is the only output
[q11,p]=ss2tf(a,b,c,d,1) % q11=[0 4 16]
[q12,p]=ss2tf(a,b,c,d,2) % q12=[0 .2 .8]
```

and the result is the first row of (E4-30).

Example 4.8: Develop a fermentor model which consists of two mass balances, one for the cell mass (or yeast), C_1 , and the other for glucose (or substrate), C_2 . We have to forget about the alcohol for now. The cell mass balance (E4-32) has two terms on the right. The first one describes cell growth using the specific growth rate $\mu = \mu(C_2)$. The second term accounts for the loss of cells due to outlet flow Q , which in turn is buried inside the notation D , the dilution rate.

$$\frac{dC_1}{dt} = \mu C_1 - DC_1 \quad (\text{E4-32})$$

The specific growth rate and dilution rate are defined as:

$$\mu = \mu(C_2) = \mu_m \frac{C_2}{K_m + C_2}, \quad \text{and} \quad D = \frac{Q}{V}$$

The glucose balance has three terms on the right. The first accounts for the consumption by the cells. The last two terms account for the flow of glucose into and out of the fermentor.

$$\frac{dC_2}{dt} = -\frac{\mu C_1}{Y} + D(C_{20} - C_2) \quad (\text{E4-33})$$

The maximum specific growth rate μ_m , Monod constant K_m , and cell yield coefficient Y are constants. In (E4-33), C_{20} is the inlet glucose concentration.

The dilution rate D is dependent on the volumetric flow rate Q and the volume V , and really is the reciprocal of the space time of the fermentor. In this problem, the fermentor volume, V , is fixed, and we vary the flow rate, Q . Hence, it is more logical to use D (and easier to think) as it is proportional to Q .

Our question is to formulate this model under two circumstances: (1) when we only vary the dilution rate, and (2) when we vary both the dilution rate and the amount of glucose input. Derive also the transfer function model in the second case. In both cases, C_1 and C_2 are the two outputs.

To solve this problem, we obviously have to linearize the equations. In vector form, the nonlinear model is

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, D) \quad (\text{E4-34})$$

where $\mathbf{x} = [C_1 \ C_2]^T$, and

$$\mathbf{f}(\mathbf{x}, D) = \begin{bmatrix} f_1(\mathbf{x}, D) \\ f_2(\mathbf{x}, D) \end{bmatrix} = \begin{bmatrix} (\mu(C_2) - D) C_1 \\ -\frac{\mu(C_2) C_1}{Y} + D(C_{20} - C_2) \end{bmatrix} \quad (\text{E4-35})$$

We first take the inlet glucose, C_{20} , to be a constant (*i.e.*, no disturbance) and vary only the dilution rate, D . From the steady state form of (E4-32) and (E4-33), we can derive (without special notations for the steady state values):

$$D(C_{20} - C_2) = \frac{\mu C_1}{Y}, \text{ and } C_1 = Y(C_{20} - C_2) \quad (\text{E4-36})$$

Now we linearize the two equations about the steady state. We expect to arrive at (with the apostrophes dropped from all the deviation variables and partial derivatives evaluated at the steady state):

$$\frac{d}{dt} \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial C_1} & \frac{\partial f_1}{\partial C_2} \\ \frac{\partial f_2}{\partial C_1} & \frac{\partial f_2}{\partial C_2} \end{bmatrix}_{s.s} \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} + \begin{bmatrix} \frac{\partial f_1}{\partial D} \\ \frac{\partial f_2}{\partial D} \end{bmatrix}_{s.s} D \quad (\text{E4-37})$$

Using (E4-35) to evaluate the partial derivatives, we should find

$$\frac{d}{dt} \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 0 & C_1 \mu' \\ -\frac{\mu}{Y} & -\frac{C_1}{Y} \mu' - \mu \end{bmatrix}_{s.s} \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} + \begin{bmatrix} -C_1 \\ \frac{C_1}{Y} \end{bmatrix}_{s.s} D = \mathbf{Ax} + \mathbf{BD} \quad (\text{E4-38})$$

where μ' is the derivative with respect to the substrate C_2 :

$$\mu' = \frac{d\mu}{dC_2} = \mu_m \frac{K_m}{(K_m + C_2)^2} \quad (\text{E4-39})$$

All the coefficients in \mathbf{A} and \mathbf{B} are evaluated at steady state conditions. From Eq. (E4-32), $D = \mu$ at steady state. Hence the coefficient a_{11} in \mathbf{A} is zero.

To complete the state space model, the output equation is

$$\begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} \quad (\text{E4-40})$$

where \mathbf{C} is the identity matrix.

Now, we'll see what happens with **two inputs**. In practice, we most likely would make a highly concentrated glucose stock and dose it into a main feed stream that contains the other ingredients. What we manipulate is the dosage rate. Consider that the glucose feed stock has a fixed concentration C_{2f} and adjustable feed rate q_f , and the other nutrients are being fed at a rate of q_o . The effective glucose feed concentration is

$$C_{20} = \frac{q_f C_{2f}}{q_f + q_o} = \frac{q_f C_{2f}}{Q} \quad (\text{E4-41})$$

where $Q = q_f + q_o$ is the total inlet flow rate, and the dilution rate is

$$D = \frac{Q}{V} = \frac{q_f + q_o}{V} \quad (\text{E4-42})$$

The general fermentor model equation as equivalent to (E4-34) is

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \quad (\text{E4-43})$$

where the state space remains $\mathbf{x} = [C_1 \ C_2]^T$, but the input is the vector $\mathbf{u} = [D_o \ D_f]^T$. Here, $D_o = q_o/V$ and $D_f = q_f/V$ are the respective dilution rates associated with the two inlet streams. That is, we vary the main nutrient feed and glucose dosage flow rates to manipulate this system. The function, \mathbf{f} , is

$$f(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} f_1(\mathbf{x}, \mathbf{u}) \\ f_2(\mathbf{x}, \mathbf{u}) \end{bmatrix} = \begin{bmatrix} \mu(C_2)C_1 - (D_o + D_f) C_1 \\ -\frac{\mu(C_2) C_1}{Y} + D_f C_{2f} - (D_o + D_f) C_2 \end{bmatrix} \quad (\text{E4-44})$$

At steady state,

$$\mu = (D_o + D_f) = D \quad (\text{E4-45})$$

and

$$C_1 = Y(C_{2o}^* - C_2), \text{ where } C_{2o}^* = \frac{D_f C_{2f}}{D_o + D_f} \quad (\text{E4-46})$$

The equations linearized about the steady state (with the apostrophes dropped from the deviation variables as in E4-38) are

$$\frac{d}{dt} \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 0 & C_1 \mu' \\ -\frac{\mu}{Y} - \frac{C_1}{Y} \mu' - \mu \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} + \begin{bmatrix} -C_1 & -C_1 \\ -C_2 & (C_{2f} - C_2) \end{bmatrix}_{s.s.} \begin{bmatrix} D_o \\ D_f \end{bmatrix} = \mathbf{Ax} + \mathbf{Bu} \quad (\text{E4-47})$$

The output equation remains the same as in (E4-40). Laplace transform of the model equations and rearrangement lead us to

$$\begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} D_o \\ D_f \end{bmatrix} \quad (\text{E4-48})$$

where the four open-loop plant transfer functions are:

$$G_{11} = \frac{-\left(\frac{C_1}{Y}\right)_{s.s.} s - \left[C_1 \left(\frac{C_1 \mu'}{Y} + \mu\right) + C_1 \mu' C_2\right]_{s.s.}}{p(s)} \quad (\text{E4-49})$$

$$G_{12} = \frac{-\left(\frac{C_1}{Y}\right)_{s.s.} s + \left[C_1 \mu' (C_{2f} - C_2) - C_1 \left(\frac{C_1 \mu'}{Y} + \mu\right)\right]_{s.s.}}{p(s)} \quad (\text{E4-50})$$

$$G_{21} = \frac{-(C_2)_{s.s.} s + \left[\frac{C_1 \mu}{Y}\right]_{s.s.}}{p(s)} \quad (\text{E4-51})$$

$$G_{22} = \frac{(C_{2f} - C_2)_{s.s.} s + \left[\frac{C_1 \mu}{Y}\right]_{s.s.}}{p(s)} \quad (\text{E4-52})$$

and the characteristic polynomial

$$p(s) = s^2 + \left(\frac{C_1 \mu'}{Y} + \mu\right)_{s.s.} s + \left(\frac{C_1 \mu \mu'}{Y}\right)_{s.s.} \quad (\text{E4-53})$$

Until we can substitute numerical values and turn the problem over to a computer, we have to admit that the state space form in (E4-47) is much cleaner to work with.

This completes our "feel good" examples. It may not be too obvious, but the hint is that linear system theory can help us analysis complex problems. We should recognize that state space representation can do everything in classical control and more, and feel at ease with the language of

state space representation.

4.3 Properties of state space models

This section contains brief remarks on some transformations and the state transition matrix. We limit the scope to materials that one may draw on introductory linear algebra.

4.3.1 Time-domain solution

We can find the solution to Eq. (4-1), which is simply a set of first order differential equations. As analogous to how Eq. (2-3) on page 2-2 was obtained, we now use the matrix exponential function as the integration factor, and the result is (hints in the Review Problems)

$$\mathbf{x}(t) = e^{\mathbf{A}t} \mathbf{x}(0) + \int_0^t e^{-\mathbf{A}(t-\tau)} \mathbf{B}u(\tau) d\tau \quad (4-10)$$

where the first term on the right evaluates the effect of the initial condition, and the second term is the so-called convolution integral that computes the effect of the input $u(t)$.

The point is that state space representation is general and is not restricted to problems with zero initial conditions. When Eq. (4-1) is homogeneous (*i.e.*, $\mathbf{B}u = 0$), the solution is simply

$$\mathbf{x}(t) = e^{\mathbf{A}t} \mathbf{x}(0) \quad (4-11)$$

We can also solve the equation using Laplace transform. Starting again from (4-1), we can find (see Review Problems)

$$\mathbf{x}(t) = \Phi(t) \mathbf{x}(0) + \int_0^t \Phi(t-\tau) \mathbf{B}u(\tau) d\tau \quad (4-12)$$

where $\Phi(t)$ is the state transition matrix as defined in (4-6). Compare (4-10) with (4-12), and we can see that

$$\Phi(t) = e^{\mathbf{A}t} \quad (4-13)$$

We have shown how the state transition matrix can be derived in a relatively simple problem in Example 4.7. For complex problems, there are numerical techniques that we can use to compute $\Phi(t)$, or even the Laplace transform $\Phi(s)$, but which of course, we shall skip.

One idea (not that we really do that) is to apply the Taylor series expansion on the exponential function of \mathbf{A} , and evaluate the state transition matrix with

$$\Phi(t) = e^{\mathbf{A}t} = \mathbf{I} + \mathbf{A}t + \frac{1}{2!} \mathbf{A}^2 t^2 + \frac{1}{3!} \mathbf{A}^3 t^3 + \dots \quad (4-14)$$

Instead of an infinite series, we can derive a closed form expression for the exponential function. For an $n \times n$ matrix \mathbf{A} , we have

$$e^{\mathbf{A}t} = \alpha_0(t) \mathbf{I} + \alpha_1(t) \mathbf{A} + \alpha_2(t) \mathbf{A}^2 + \dots + \alpha_{n-1}(t) \mathbf{A}^{n-1} \quad (4-15)$$

The challenge is to find those coefficients $\alpha_i(t)$, which we shall skip.¹

¹ We only need the general form of (4-15) later in Chapter 9. There are other properties of the state transition matrix that we have skipped, but we have structured our writing such that they are not needed here.

4.3.2 Controllable canonical form

While there is no unique state space representation of a system, there are “standard” ones that control techniques make use of. Given any state equations (and if some conditions are met), it is possible to convert them to these standard forms. We cover in this subsection a couple of important canonical forms.

A tool that we should be familiar with from introductory linear algebra is similarity transform, which allows us to transform a matrix into another one but which retains the same eigenvalues. If a state \mathbf{x} and another $\bar{\mathbf{x}}$ are related via a so-called similarity transformation, the state space representations constructed with \mathbf{x} and $\bar{\mathbf{x}}$ are considered to be equivalent.¹

For the n -th order differential equation:²

$$y^{(n)} + a_{n-1}y^{(n-1)} + \dots + a_1y^{(1)} + a_0y = u(t) \quad (4-16)$$

we define

$$x_1 = y, \quad x_2 = y^{(1)}, \quad x_3 = y^{(2)}, \quad \dots, \quad \text{and} \quad x_n = y^{(n-1)} \quad (4-17)$$

The original differential equation can now be reformulated as a set of first order equations:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ &\vdots \\ \dot{x}_{n-1} &= x_n \end{aligned} \quad (4-18)$$

and finally

$$\dot{x}_n = -a_0x_1 - a_1x_2 - \dots - a_{n-1}x_n + u(t)$$

This set of equation, of course, can be put in matrix form as in Eq. (4-1):

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ -a_0 & -a_1 & -a_2 & \dots & -a_{n-1} \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} u = \mathbf{Ax} + \mathbf{Bu} \quad (4-19)$$

The output equation equivalent to Eq. (4-2) is

$$y = [1 \ 0 \ 0 \ \dots \ 0] \mathbf{x} = \mathbf{Cx} \quad (4-20)$$

The system of equations in (4-19) and (4-20) is called the **controllable canonical form**

¹ That includes transforming a given system to the controllable canonical form. We can say that state space representations are unique up to a similarity transform. As for transfer functions, we can say that they are unique up to scaling in the coefficients in the numerator and denominator. However, the derivation of canonical transforms requires material from Chapter 9 and is not crucial for the discussion here. These details are provided on our *Web Support*.

² Be careful when you read the MATLAB manual; it inverts the index of coefficients as in $y^{(n)} + a_1y^{(n-1)} + \dots + a_{n-1}y^{(1)} + a_ny$. Furthermore, we use a simple RHS in the ODE. You'd find more general, and thus messier, derivations in other texts.

(also phase variable canonical form). As the name implies, this form is useful in doing controllability analysis and in doing pole placement system design—topics that we will cover in Chapter 9.

With all the zeros along the leading diagonal, we can find relatively easily that the characteristic equation of \mathbf{A} , $|s\mathbf{I} - \mathbf{A}| = 0$, is

$$s^n + a_{n-1}s^{n-1} + \dots + a_1s + a_0 = 0 \quad (4-21)$$

which is immediately obvious from Eq. (4-16) itself. We may note that the coefficients of the characteristic polynomial are contained in the matrix \mathbf{A} in (4-19). Matrices with this property are called the *companion* form. When we use MATLAB, its `canon()` function returns a companion matrix which is the transpose of \mathbf{A} in (4-19); this form is called the *observable canonical form*. We shall see that in MATLAB Session 4.

4.3.3 Diagonal canonical form

Here, we want to transform a system matrix \mathbf{A} into a diagonal matrix Λ that is made up of the eigenvalues of \mathbf{A} . In other words, all the differential equations are decoupled after the transformation.

For a given system of equations in (4-1) in which \mathbf{A} has *distinct eigenvalues*, we should find a transformation with a matrix \mathbf{P} :

$$\bar{\mathbf{x}} = \mathbf{P}^{-1}\mathbf{x}, \text{ or } \mathbf{x} = \mathbf{P}\bar{\mathbf{x}} \quad (4-22)$$

such that


$$\dot{\bar{\mathbf{x}}} = \Lambda \bar{\mathbf{x}} + \bar{\mathbf{B}} \mathbf{u} \quad (4-23)$$

where now $\bar{\mathbf{B}} = \mathbf{P}^{-1}\mathbf{B}$, and $\Lambda = \mathbf{P}^{-1}\mathbf{A}\mathbf{P}$ is a diagonal matrix made up of the eigenvalues of \mathbf{A} . The transformation matrix (also called the modal matrix) \mathbf{P} is made up of the eigenvectors of \mathbf{A} . In control, (4-23) is called the *diagonal canonical form*.

If \mathbf{A} has repeated eigenvalues (multiple roots of the characteristic polynomial), the result, again from introductory linear algebra, is the *Jordan canonical form*. Briefly, the transformation matrix \mathbf{P} now needs a set of generalized eigenvectors, and the transformed matrix $\mathbf{J} = \mathbf{P}^{-1}\mathbf{A}\mathbf{P}$ is made of Jordan blocks for each of the repeated eigenvalues. For example, if matrix \mathbf{A} has three repeated eigenvalues λ_1 , the transformed matrix should appear as

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_{11} & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_{22} \end{bmatrix} \quad \text{where} \quad \mathbf{J}_{11} = \begin{bmatrix} \lambda_1 & 1 & 0 \\ 0 & \lambda_1 & 1 \\ 0 & 0 & \lambda_1 \end{bmatrix} \quad (4-24)$$

and \mathbf{J}_{22} is a diagonal matrix made up of eigenvalues $\lambda_4, \dots, \lambda_n$. Since Chapter 9 later will not make use of such cases, we shall leave the details to a second course in modern control.

 **Example 4.9:** For a model with the following transfer function

$$\frac{Y}{U} = \frac{1}{(s+3)(s+2)(s+1)},$$

find the diagonal and observable canonical forms with MATLAB.

The statements to use are:


```
G=zpk([],[-1 -2 -3],1);
S=ss(G); % S is the state space system
canon(S) % Default is the diagonal form
canon(S,'companion') % This is the observable companion
```

There is no messy algebra. We can be spoiled! Further details are in MATLAB Session 4.

□ Review Problems

1. Fill in the gaps in the derivation of (E4-25) and (E4-26) in Example 4.7
2. Write down the dimensions of all the matrixes in (4-6) for the general case of multiple-input and multiple-output models. Take \mathbf{x} to be $(n \times 1)$, \mathbf{y} $(m \times 1)$, and \mathbf{u} $(k \times 1)$. And when y and u are scalar, $\mathbf{C}\Phi\mathbf{B}$ is a scalar quantity too.
3. Fill in the gaps in the derivation of (4-9) from (4-3a).
4. For the SISO system shown in Fig. R4.4, derive the state space representation. Show that the characteristic equation of the model matrix is identical to the closed-loop characteristic polynomial as derived from the transfer functions.
5. Derive Eq. (4-10).
6. Derive Eq. (4-12).
7. Derive Eq. (4-23).

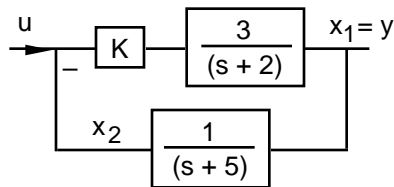


Figure R4.4

Hints:

2. \mathbf{A} is $(n \times n)$, \mathbf{B} $(n \times k)$, \mathbf{C} $(m \times n)$, Φ $(n \times n)$, and $\mathbf{C}\Phi\mathbf{B}$ $(m \times k)$.
4. Multiply the K to the transfer function to give a gain of $3K$. Then the rest is easier than Example 4.6.
5. We multiply Eq. (4-1) by $\exp(-\mathbf{A}t)$ to give $\frac{d}{dt}[e^{-\mathbf{A}t}\dot{\mathbf{x}} - \mathbf{A}\mathbf{x}] = e^{-\mathbf{A}t}\mathbf{B}\mathbf{u}$, which is

$$\frac{d}{dt}[e^{-\mathbf{A}t}\dot{\mathbf{x}}] = e^{-\mathbf{A}t}\mathbf{B}\mathbf{u}$$

Integration with the initial condition gives

$$e^{-\mathbf{A}t}\mathbf{x}(t) - \mathbf{x}(0) = \int_0^t e^{-\mathbf{A}\tau}\mathbf{B}\mathbf{u}(\tau) d\tau$$

which is one step away from Eq. (4-10).

6. The Laplace transform of Eq. (4-1) with nonzero initial conditions is

$$s\mathbf{X} - \mathbf{x}(0) = \mathbf{A}\mathbf{X} + \mathbf{B}\mathbf{U}$$

or

$$\mathbf{X} = (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{x}(0) + (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\mathbf{U}$$

Substituting in the definition $\Phi(s) = (s\mathbf{I} - \mathbf{A})^{-1}$, we have

$$\mathbf{X} = \Phi(s)\mathbf{x}(0) + \Phi(s)\mathbf{B}\mathbf{U}$$

The time domain solution vector is the inverse transform

$$\mathbf{x}(t) = \mathcal{L}^{-1}[\Phi(s)]\mathbf{x}(0) + \mathcal{L}^{-1}[\Phi(s)\mathbf{B}U]$$

and if we invoke the definition of convolution integral (from calculus), we have Eq. (4-12).

7. We first substitute $\mathbf{x} = \mathbf{P}\bar{\mathbf{x}}$ in Eq. (4-1) to give

$$\mathbf{P}\frac{d}{dt}\bar{\mathbf{x}} = \mathbf{A}\mathbf{P}\bar{\mathbf{x}} + \mathbf{B}u$$

Then we multiply the equation by the inverse \mathbf{P}^{-1}

$$\frac{d}{dt}\bar{\mathbf{x}} = \mathbf{P}^{-1}\mathbf{A}\mathbf{P}\bar{\mathbf{x}} + \mathbf{P}^{-1}\mathbf{B}u$$

which is (4-23).

❖ 9. Design of State Space Systems

We now return to the use of state space representation that was introduced in Chapter 4. As you may have guessed, we want to design control systems based on state space analysis. State feedback controller is very different from the classical PID controller. Our treatment remains introductory, and we will stay with linear or linearized SISO systems. Nevertheless, the topics here should enlighten(!) us as to what modern control is all about.

What are we up to?

- Evaluate the controllability and observability of a system.
- Pole placement design of state feedback systems. Application of the Ackermann's formula.
- Design with full-state and reduced-order observers (estimators).

9.1 Controllability and Observability

Before we formulate a state space system, we need to raise two important questions. One is whether the choice of inputs (the manipulated variables) may lead to changes in the states, and the second is whether we can evaluate all the states based on the observed output. These are what we call the controllability and observability problems.

9.1.1 Controllability.

A system is said to be completely state controllable if there exists an input $u(t)$ which can drive the system from any given initial state $\mathbf{x}_0(t_0=0)$ to any other desired state $\mathbf{x}(t)$. To derive the controllability criterion, let us restate the linear system and its solution from Eqs. (4-1), (4-2), and (4-10):

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u \quad (9-1)$$

$$y = \mathbf{C}\mathbf{x} \quad (9-2)$$

and

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0) + \int_0^t e^{-\mathbf{A}(t-\tau)}\mathbf{B}u(\tau) d\tau \quad (9-3)$$

With our definition of controllability, there is no loss of generality if we choose to have $\mathbf{x}(t) = \mathbf{0}$, *i.e.*, moving the system to the origin. Thus Eq. (9-3) becomes

$$\mathbf{x}(0) = -\int_0^t e^{-\mathbf{A}\tau}\mathbf{B}u(\tau) d\tau \quad (9-4)$$

We next make use of Eq. (4-15) on page 4-14, *i.e.*, the fact that we can expand the matrix exponential function as a closed-form series:

$$e^{\mathbf{A}t} = \alpha_0(t)\mathbf{I} + \alpha_1(t)\mathbf{A} + \alpha_2(t)\mathbf{A}^2 + \dots + \alpha_{n-1}(t)\mathbf{A}^{n-1} \quad (9-5)$$

Substitution of Eq. (9-5) into (9-4) gives

$$\mathbf{x}(0) = -\sum_{k=0}^{n-1} \mathbf{A}^k \mathbf{B} \int_0^t \alpha_k(\tau) u(\tau) d\tau$$

We now hide the ugly mess by defining the $(n \times 1)$ vector β with elements

$$\beta_k(\tau) = \int_0^t \alpha_k(\tau) u(\tau) d\tau$$

and Eq. (9-4) appears as

$$\mathbf{x}(0) = - \sum_{k=0}^{n-1} \mathbf{A}^k \mathbf{B} \beta_k = - \begin{bmatrix} \mathbf{B} & \mathbf{AB} & \mathbf{A}^2\mathbf{B} & \dots & \mathbf{A}^{n-1}\mathbf{B} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{n-1} \end{bmatrix} \quad (9-6)$$

If Eq. (9-6) is to be satisfied, the $(n \times n)$ matrix $[\mathbf{B} \ \mathbf{AB} \ \dots \ \mathbf{A}^{n-1}\mathbf{B}]$ must be of rank n . This is a necessary and sufficient condition for controllability. Hence, we can state that a system is completely controllable if and only if the **controllability matrix**

$$\mathbf{C}_o = [\mathbf{B} \ \mathbf{AB} \ \mathbf{A}^2\mathbf{B} \ \dots \ \mathbf{A}^{n-1}\mathbf{B}] \quad (9-7)$$

is of rank n .

The controllability condition is the same even when we have multiple inputs, \mathbf{u} . If we have r inputs, then \mathbf{u} is $(r \times 1)$, \mathbf{B} is $(n \times r)$, each of the β_k is $(r \times 1)$, β is $(nr \times 1)$, and \mathbf{C}_o is $(n \times nr)$.

When we have multiple outputs \mathbf{y} , we want to control the output rather than the states. Complete state controllability is neither necessary nor sufficient for actual output controllability. With the output $\mathbf{y} = \mathbf{C}\mathbf{x}$ and the result in (9-6), we can *infer* that the **output controllability matrix** is

$$\mathbf{C}_o = [\mathbf{CB} \ \mathbf{CAB} \ \mathbf{CA}^2\mathbf{B} \ \dots \ \mathbf{CA}^{n-1}\mathbf{B}] \quad (9-8)$$

If we have m outputs, \mathbf{y} is $(m \times 1)$ and \mathbf{C} is $(m \times n)$. If we also have r inputs, then the output controllability matrix is $(m \times nr)$. Based on our interpretation of Eq. (9-6), we can also infer that to have complete output controllability, the matrix in (9-8) must have rank m .

9.1.2 Observability

The linear time invariant system in Eqs. (9-1) and (9-2) is completely observable if every initial state $\mathbf{x}(0)$ can be determined from the output $\mathbf{y}(t)$ over a finite time interval. The concept of observability is useful because in a given system, all not of the state variables are accessible for direct measurement. We will need to estimate the unmeasurable state variables from the output in order to construct the control signal.

Since our focus is to establish the link between \mathbf{y} and \mathbf{x} , or observability, it suffices to consider only the unforced problem:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} \quad (9-9)$$

and

$$\mathbf{y} = \mathbf{C}\mathbf{x} \quad (9-10)$$

Substitution of the solution of (9-9) in (9-10) gives

$$\mathbf{y}(t) = \mathbf{C}e^{\mathbf{A}t}\mathbf{x}(0)$$

We again take that we can expand the exponential function as in Eq. (9-5). Thus we have

$$\mathbf{y}(t) = \sum_{k=0}^{n-1} \alpha_k(t) \mathbf{C} \mathbf{A}^k \mathbf{x}(0) = \begin{bmatrix} \alpha_0 & \alpha_1 & \dots & \alpha_{n-1} \end{bmatrix} \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{n-1} \end{bmatrix} \mathbf{x}(0) \quad (9-11)$$

With the same reasoning that we applied to Eq. (9-6), we can infer that to have complete observability, the **observability matrix**¹

$$\mathbf{O}_b = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{n-1} \end{bmatrix} \quad (9-12)$$

must be of rank n . When we have m outputs, \mathbf{y} is $(m \times 1)$, \mathbf{C} is $(m \times n)$, and \mathbf{O}_b is $(mn \times n)$.

🔗 **Example 9.1:** Consider a third order model:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -6 & -11 & -6 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad \mathbf{C} = [1 \ 0 \ 0]$$

which is the controllable canonical form of the problem in Example 4.9 (p. 4-16). Construct the controllability and observability matrices.

To compute the controllability matrix, we can use the MATLAB function `ctrb()`:

```
A=[0 1 0; 0 0 1; -6 -11 -6];
B=[0; 0; 1];
Co=ctrb(A,B)
```

Or we can use the definition itself:

$$\mathbf{C}_o = [\mathbf{B} \ \mathbf{A}^* \mathbf{B} \ \mathbf{A}^{*2} \mathbf{B}]$$

Either way, we should obtain

$$\mathbf{C}_o = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & -6 \\ 1 & -6 & 25 \end{bmatrix}$$

which has a rank of 3 and the model is completely state controllable.

Similarly, we can use the MATLAB function `obsv()` for the observability matrix:

```
C=[1 0 0];
Ob=obsv(A,C)
```

Or we can use the definition:

$$\mathbf{O}_b = [\mathbf{C}; \mathbf{C}^* \mathbf{A}; \mathbf{C}^* \mathbf{A}^{*2}]$$

We should find that \mathbf{O}_b is the identity matrix, which of course, is of rank 3.

¹ Controllability and observability are dual concepts. With $\mathbf{C} = \mathbf{B}^T$ and $\mathbf{A} = \mathbf{A}^T$, we can see that $\mathbf{O}_b = \mathbf{C}_o^T$.

🔗 **Example 4.8A:** We now revisit the fermentor example 4.8 (p. 4-11). Our question is whether we can control the cell mass and glucose concentration by adjusting only D.

From Eq. (E4-38) in Example 4.8, we have

$$\mathbf{A} = \begin{bmatrix} 0 & C_1 \mu' \\ -\frac{\mu}{Y} & -\frac{C_1}{Y} \mu' - \mu \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} -C_1 \\ \frac{C_1}{Y} \end{bmatrix}$$

First, we evaluate

$$\mathbf{AB} = \begin{bmatrix} 0 & C_1 \mu' \\ -\frac{\mu}{Y} & -\frac{C_1}{Y} \mu' - \mu \end{bmatrix} \begin{bmatrix} -C_1 \\ \frac{C_1}{Y} \end{bmatrix} = \begin{bmatrix} \frac{C_1^2 \mu'}{Y} \\ -\frac{C_1^2 \mu}{Y^2} \end{bmatrix}$$

The controllability matrix is

$$\mathbf{C}_0 = [\mathbf{B} \quad \mathbf{AB}] = \begin{bmatrix} -C_1 & \frac{C_1^2 \mu'}{Y} \\ \frac{C_1}{Y} & -\frac{C_1^2 \mu}{Y^2} \end{bmatrix}$$

Since the determinant of \mathbf{C}_0 is 0, the rank of \mathbf{C}_0 is 1, both cell mass and substrate cannot be controlled simultaneously by just varying D. The answer is quite obvious with just a bit of intuition. If we insist on using D as the only input, we can control either C_1 or C_2 , but not both quantities. To effectively regulate both C_1 and C_2 , we must implement a system with two inputs. An obvious solution is to adjust the glucose feed concentration (C_{20}) as well as the total flow rate (dilution rate D).

Now, we'll see what happens with two inputs. Compared with Eq. (E4-38), \mathbf{A} remains the same, while \mathbf{B} in Eq. (E4-47) is now a (2 x 2) matrix with a rank of 2. Hence the controllability matrix $\mathbf{C}_0 = [\mathbf{B} \quad \mathbf{AB}]$ is a (2 x 4) matrix and it must have a rank of 2 (since at least \mathbf{B} is), and both C_1 and C_2 are controllable.

9.2 Pole Placement Design

9.2.1 Pole placement and Ackermann's formula.

When we used root locus for controller design in Chapter 7, we chose a dominant pole (or a conjugate pair if complex). With state space representation, we have the mathematical tool to choose all the closed-loop poles. To begin, we restate the state space model in Eqs. (4-1) and (4-2):

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u \quad (9-13)$$

$$y = \mathbf{C}\mathbf{x} \quad (9-14)$$

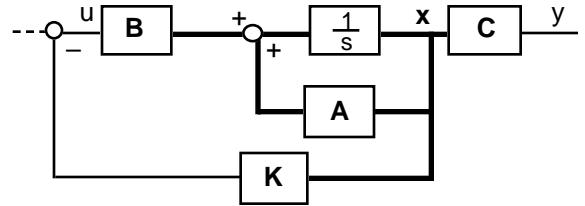


Figure 9.1. Closed-loop system with state feedback

With a control system, the input u is now the manipulated variable that is driven by the control signal (Fig. 9.1). For the moment, we consider only the regulator problem and omit changes in the set point. We state the simple control law which depends on full state feedback as

$$u(t) = -\mathbf{K}\mathbf{x} = -K_1x_1(t) - K_2x_2(t) \dots - K_nx_n(t) \quad (9-15)$$

where \mathbf{K} is the **state feedback gain** ($1 \times n$) vector. In this formulation, the feedback information requires $\mathbf{x}(t)$, meaning that we must be able to measure all the state variables.

We now substitute Eq. (9-15) in (9-13) to arrive at the system equation

$$\dot{\mathbf{x}} = (\mathbf{A} - \mathbf{BK})\mathbf{x} \quad (9-16)$$

The eigenvalues of the system matrix $(\mathbf{A} - \mathbf{BK})$ are called the regulator poles. What we want is to find \mathbf{K} such that it satisfies how we select all the eigenvalues (or where we put all the closed-loop poles).

To do that easily, we first need to put our model (9-13) in the controllable canonical form as in Eq. (4-19) on page 4-15:

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ -a_0 & -a_1 & -a_2 & \dots & -a_{n-1} \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} u \quad (9-17)$$

After substituting for u with Eq. (9-15), the system matrix in (9-16) is

$$\mathbf{A} - \mathbf{BK} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ -a_0 & -a_1 & -a_2 & \dots & -a_{n-1} \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} K_1 & K_2 & \dots & K_n \end{bmatrix}$$

or

$$\mathbf{A} - \mathbf{BK} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_0 - K_1 & -a_1 - K_2 & -a_2 - K_3 & \cdots & -a_{n-1} - K_n \end{bmatrix} \quad (9-18)$$

As in Eq. (4-21) on page 4-16, the closed-loop characteristic equation $|s\mathbf{I} - \mathbf{A} + \mathbf{BK}| = 0$ will appear as

$$s^n + (a_{n-1} + K_n)s^{n-1} + \dots + (a_1 + K_2)s + (a_0 + K_1) = 0 \quad (9-19)$$

We next return to our assertion that we can choose all our closed-loop poles, or in terms of eigenvalues, $\lambda_1, \lambda_2, \dots, \lambda_n$. This desired closed-loop characteristic equation is

$$(s - \lambda_1)(s - \lambda_2) \dots (s - \lambda_n) = s^n + \alpha_{n-1}s^{n-1} + \dots + \alpha_1s + \alpha_0 = 0 \quad (9-20)$$

where the coefficients α_i are computed by expanding the terms in the LHS. By matching the coefficients of like power of s in Eqs. (9-19) and (9-20), we obtain

$$\begin{aligned} a_0 + K_1 &= \alpha_0 \\ a_1 + K_2 &= \alpha_1 \\ &\dots \\ a_{n-1} + K_n &= \alpha_{n-1} \end{aligned}$$

Thus in general, we can calculate all the state feedback gains in \mathbf{K} by

$$K_i = \alpha_{i-1} - a_{i-1}, \quad i = 1, 2, \dots, n \quad (9-21)$$

This is the result of full state feedback pole-placement design. If the system is completely state controllable, we can compute the state gain vector \mathbf{K} to meet our selection of all the closed-loop poles (eigenvalues) through the coefficients α_i .

There are other methods in pole-placement design. One of them is the **Ackermann's formula**. The derivation of Eq. (9-21) predicates that we have put (9-13) in the controllable canonical form. Ackermann's formula only requires that the system (9-13) be completely state controllable. If so, we can evaluate the state feedback gain as ¹

$$\mathbf{K} = [0 \ 0 \ \dots \ 1] [\mathbf{B} \ \mathbf{AB} \ \dots \ \mathbf{A}^{n-1}\mathbf{B}]^{-1} \alpha_c(\mathbf{A}) \quad (9-22)$$

where

$$\alpha_c(\mathbf{A}) = \mathbf{A}^n + \alpha_{n-1}\mathbf{A}^{n-1} + \dots + \alpha_1\mathbf{A} + \alpha_0\mathbf{I} \quad (9-23)$$

is the polynomial derived from the desired eigenvalues as in (9-20), except now $\alpha_c(\mathbf{A})$ is an $(n \times n)$ matrix.

9.2.2 Servo systems.

We now re-introduce the change in reference, $r(t)$. We will stay with analyzing a single-input single-output system. By a proper choice in the indexing of the state variables, we select $x_1 = y$. In a feedback loop, the input to the process model may take the form

$$u(t) = K_r r(t) - \mathbf{K}\mathbf{x}(t)$$

¹ Roughly, the Ackermann's formula arises from the application of the Cayley-Hamilton theorem to (9-20). The details of the derivation are in our *Web Support*.

where K_r is some gain associated with the change in the reference, and \mathbf{K} is the state feedback gain as defined in (9-15). One of the approaches that we can take is to choose $K_r = K_1$, such that $u(t)$ is

$$u(t) = K_1[r(t) - x_1(t)] - K_2x_2(t) - \dots - K_nx_n(t) \quad (9-24)$$

where we may recognize that $r(t) - x_1(t)$ is the error $e(t)$.

The system equation is now

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}[\mathbf{K}_1r - \mathbf{K}\mathbf{x}]$$

or

$$\dot{\mathbf{x}} = (\mathbf{A} - \mathbf{BK})\mathbf{x} + \mathbf{BK}_1r \quad (9-25)$$

The system matrix and thus design procedures remain the same as in the regulator problem in Eq. (9-16).¹

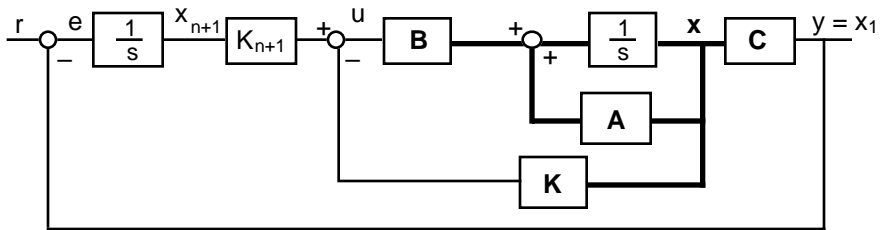


Figure 9.2. State feedback with integral control.

9.2.3 Servo systems with integral control.

You may notice that nothing that we have covered so far does integral control as in a PID controller. To implement integral action, we need to add one state variable as in Fig. 9.2. Here, we integrate the error $[r(t) - x_1(t)]$ to generate the new variable x_{n+1} . This quantity is multiplied by the additional feedback gain K_{n+1} before being added to the rest of the feedback data.

The input to the process model now takes the form

$$u(t) = K_{n+1}x_{n+1}(t) - \mathbf{K}\mathbf{x}(t) \quad (9-26)$$

¹ The system must be asymptotically stable. At the new steady state (as $t \rightarrow \infty$), we have

$$0 = (\mathbf{A} - \mathbf{BK})\mathbf{x}(\infty) + \mathbf{BK}_1r(\infty)$$

and subtracting this equation from (9-25), we have

$$\dot{\mathbf{x}} = (\mathbf{A} - \mathbf{BK})(\mathbf{x} - \mathbf{x}(\infty)) + \mathbf{BK}_1(r - r(\infty))$$

If we define $\mathbf{e} = \mathbf{x} - \mathbf{x}(\infty)$, and also $r(t)$ as a step function such that r is really a constant for $t > 0$, the equation is simply

$$\dot{\mathbf{e}} = (\mathbf{A} - \mathbf{BK})\mathbf{e}$$

Not only is this equation identical to the form in Eq. (9-16), but we also can interpret the analysis as equivalent to a problem where we want to find \mathbf{K} such that the steady state error $\mathbf{e}(t)$ approaches zero as quickly as possible.

The differential equation for \mathbf{x}_{n+1} is

$$\dot{\mathbf{x}}_{n+1} = \mathbf{r}(t) - \mathbf{C}\mathbf{x} \quad (9-27)$$

We have written $\mathbf{x}_1 = \mathbf{y} = \mathbf{C}\mathbf{x}$ just so that we can package this equation in matrix form in the next step. Substitution of Eq. (9-26) in the state model (9-13) and together with (9-27), we can write this $(n + 1)$ system as

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{x}}_{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{A} - \mathbf{BK} & \mathbf{BK}_{n+1} \\ -\mathbf{C} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{x}_{n+1} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} \mathbf{r} \quad (9-28)$$

In terms of dimensions, $(\mathbf{A} - \mathbf{BK})$, \mathbf{B} and \mathbf{C} remain, respectively, $(n \times n)$, $(n \times 1)$, and $(1 \times n)$. We can interpret the system matrix as

$$\begin{bmatrix} \mathbf{A} - \mathbf{BK} & \mathbf{BK}_{n+1} \\ -\mathbf{C} & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ -\mathbf{C} & 0 \end{bmatrix} - \begin{bmatrix} \mathbf{B} \\ 0 \end{bmatrix} [\mathbf{K} \quad -\mathbf{K}_{n+1}] = \hat{\mathbf{A}} - \hat{\mathbf{B}} \hat{\mathbf{K}} \quad (9-29)$$

where now our task is to find the $(n + 1)$ state feedback gains

$$\hat{\mathbf{K}} = [\mathbf{K} \quad -\mathbf{K}_{n+1}] \quad (9-30)$$

With Eq. (9-29), we can view the characteristic equation of the system as

$$|s\mathbf{I} - \hat{\mathbf{A}} + \hat{\mathbf{B}} \hat{\mathbf{K}}| = 0 \quad (9-31)$$

which is in the familiar form of the problem in (9-16). Thus, we can make use of the pole-placement techniques in Section 9.2.1.

Example 9.2: Consider the second order model in Example 9.1. What are the state feedback gains if we specify that the closed-loop poles are to be at $-3 \pm 3j$ and -6 ?

With the given model in the controllable canonical form, we can use Eq. (9-21). The MATLAB statements are:

```
A=[0 1 0; 0 0 1; -6 -11 -6]; % Should find
p1=poly(A) % [1 6 11 6], coefficients a_i in (9-19)
P=[-3+3j -3-3j -6];
p2=poly(P) % [1 12 54 108], coefficients alpha_i in (9-20)
p2-p1 % [0 6 43 102], K_i as in Eq.(9-21)
```

To obtain the state feedback gains with Eq. (9-21), we should subtract the coefficients of the polynomial p_1 from p_2 , starting with the last constant coefficient. The result is, indeed,

$$\mathbf{K} = (K_1, K_2, K_3) = (108-6, 54-11, 12-6) = (102, 43, 6)$$

Check 1. The same result can be obtained with the MATLAB function `acker()` which uses the Ackermann's formula. The statements are:

```
B=[0; 0; 1];
acker(A,B,P) %Should return [102 43 6]
```

Check 2. We can do the Ackermann's formula step by step. The statements are:

```
M=[B A*B A^2*B]; %controllability matrix
ac=polyvalm(p2,A); %Eq. (9-23)
[0 0 1]*inv(M)*ac %Eq. (9-22)
```

To evaluate the matrix polynomial in Eq. (9-23), we use the MATLAB function `polyvalm()` which applies the coefficients in `p2` to the matrix `A`.

✎ **Example 4.7B:** Let us revisit the two CSTR-in-series problem in Example 4.7 (p. 4-5). Use the inlet concentration as the input variable and check that the system is controllable and observable. Find the state feedback gain such that the reactor system is very slightly underdamped with a damping ratio of 0.8, which is equivalent to about a 1.5% overshoot.

From (E4-27) of Example 4.7, the model is

$$\frac{d}{dt} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} -5 & 0 \\ 2 & -4 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + \begin{bmatrix} 4 \\ 0 \end{bmatrix} c_o$$

and C_2 is the only output. We can construct the model and check the controllability and observability with

```
A=[-5 0; 2 -4];
B=[4; 0];
C=[0 1];
D=0;
rank(ctrb(A,B))    %should find rank = 2
rank(observ(A,C))  % for both matrices
```

Both the controllability and observability matrices are of rank two. Hence the system is controllable and observable.

To achieve a damping ratio of 0.8, we can find that the closed-loop poles must be at $-4.5 \pm 3.38j$ (using a combination of what we learned in Example 7.5 and Fig. 2.5), but we can cheat with MATLAB and use root locus plots!

```
[q,p]=ss2tf(A,B,C,D); %converts state space to transfer function1
Gp=tf(q,p);
rlocus(Gp)
sgrid(0.8,1)
[kc,P]=rlocfind(Gp) %should find kc = 1.46
```

We now apply the closed-loop poles `P` directly to the Ackermann's formula:

```
K=acker(A,B,P) %should find K = [0 1.46]
```

The state space state feedback gain (K_2) related to the output variable C_2 is the same as the proportional gain obtained with root locus. Given any set of closed-loop poles, we can find the state feedback gain of a controllable system using state-space pole placement methods. The use of root locus is not necessary, but it is a handy tool that we can take advantage of.

¹ Another way here is to make use of the analytical result in Example 4.7:

```
Gp=zpk([], [-5 -4], 8); %transfer function C2/Co taken from (E4-30a)
```

▮ **Example 4.7C:** Add integral action to the system in Example 4.7B so we can eliminate the steady state error.

To find the new state feedback gain is a matter of applying Eq. (9-29) and the Ackermann's formula. The hard part is to make an intelligent decision on the choice of closed-loop poles. Following the lead of Example 4.7B, we use root locus plots to help us. With the understanding that we have two open-loop poles at -4 and -5 , a reasonable choice of the integral time constant is $1/3$ min. With the open-loop zero at -3 , the reactor system is always stable, and the dominant closed-loop pole is real and the reactor system will not suffer from excessive oscillation.

Hence, our first step is to use root locus to find the closed-loop poles of a PI control system with a damping ratio of 0.8. The MATLAB statements to continue with Example 4.7B are:

```
kc=1; tau_i=1/3;
Gc=tf(kc*[tau_i 1],[tau_i 0]);
rlocus(Gc*Gp); %Gp is from Example 4.7B
sgrid(0.8,1)
[kc,P]=rlocfind(Gc*Gp) %should find proportional gain kc=1.66
```

The closed-loop poles P are roughly at -2.15 and $-3.43 \pm 2.62j$, which we apply immediately to the Ackermann's formula using \hat{A} and \hat{B} in Eq. (9-29):

```
Ah=[A zeros(2,1); -C 0]; %Eq. (9-29)
Bh=[B; 0];
Kh=acker(Ah,Bh,P) %should find Kh = [0 1.66 -4.99]
```

The state feedback gain including integral control \hat{K} is $[0 \ 1.66 \ -4.99]$. Unlike the simple proportional gain, we cannot expect that $K_{n+1} = 4.99$ would resemble the integral time constant in classical PI control. To do the time domain simulation, the task is similar to the hints that we provide for Example 7.5B in the Review Problems. The actual statements will also be provided on our Web Support.

▮ **Example 7.5B:** Consider the second order system in Example 7.5 (p. 7-9). What are the state feedback gains if we specify that the closed-loop poles are to be $-0.375 \pm 0.382j$ as determined in Example 7.5A (p. 7-15)?

The problem posed in Examples 7.5 and 7.5A is not in the controllable canonical form (unless we do the transform ourselves). Thus we will make use of the Ackermann's formula. The MATLAB statements are:

```
G=tf(1,conv([2 1],[4 1])); %Make the state space object from
S=ss(G); % the transfer function
scale=S.c(2); %Rescale MATLAB model matrices
S.c=S.c/scale; S.b=S.b*scale;
P=[-0.375+0.382j -0.375-0.382j]; %Define the closed-loop poles
k=acker(S.a,S.b,P) %Calculate the feedback gains
```

MATLAB will return the vector $[0 \ 1.29]$, meaning that $K_1 = 0$, and $K_2 = 1.29$, which was the proportional gain obtained in Example 7.5A. Since $K_1 = 0$, we only feedback the controlled variable as analogous to proportional control. In this very simple example, the state space system is virtually the classical system with a proportional controller.

A note of caution is necessary when we let MATLAB generate the state space model from a transfer function. The vector \mathbf{C} (from $S.c$) is $[0 \ 0.5]$, which means that the indexing is reversed such that x_2 is the output variable, and x_1 is the derivative of x_2 . Secondly, \mathbf{C} is not $[0 \ 1]$, and hence we have to rescale the matrices \mathbf{B} and \mathbf{C} . These two points are further covered in MALTA Session 4.

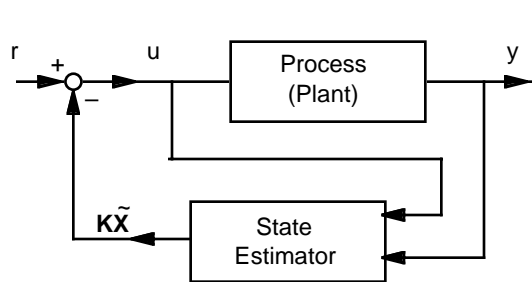


Figure 9.3. Concept of using a state estimator to generate an estimated state feedback signal.

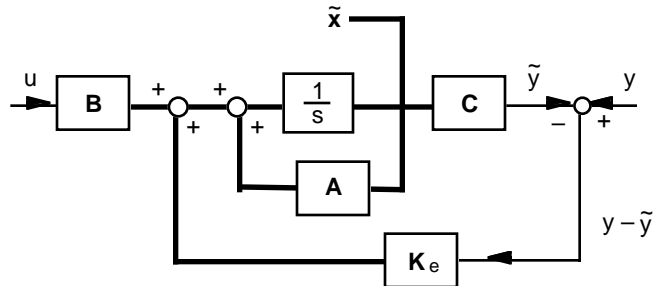


Figure 9.4. A probable model for a state estimator.

9.3 State Estimation Design

9.3.1. State estimator.

The pole placement design predicated on the feedback of *all* the state variables \mathbf{x} (Fig. 9.1). Under many circumstances, this may not be true. We have to estimate unmeasurable state variables or signals that are too noisy to be measured accurately. One approach to work around this problem is to estimate the state vector with a model. The algorithm that performs this estimation is called the **state observer** or the **state estimator**. The estimated state $\tilde{\mathbf{x}}$ is then used as the feedback signal in a control system (Fig. 9.3). A full-order state observer estimates all the states even when some of them are measured. A reduced-order observer does the smart thing and skip these measurable states.

The next task is to seek a model for the observer. We stay with a single-input single-output system, but the concept can be extended to multiple outputs. The estimate should embody the dynamics of the plant (process). Thus one probable model, as shown in Fig. 9.4, is to assume that the state estimator has the same structure as the plant model, as in Eqs. (9-13) and (9-14), or Fig. 9.1. The estimator also has the *identical* plant matrices \mathbf{A} and \mathbf{B} . However, one major difference is the addition of the estimation error, $y - \tilde{y}$, in the computation of the estimated state $\tilde{\mathbf{x}}$.

The estimated state variables based on Fig. 9.4 can be described by (details in Review Problems)

$$\begin{aligned}\dot{\tilde{\mathbf{x}}} &= \mathbf{A}\tilde{\mathbf{x}} + \mathbf{B}u + \mathbf{K}_e(y - \mathbf{C}\tilde{\mathbf{x}}) \\ &= (\mathbf{A} - \mathbf{K}_e\mathbf{C})\tilde{\mathbf{x}} + \mathbf{B}u + \mathbf{K}_ey\end{aligned}\quad (9-32)$$

Here, $\tilde{y} = \mathbf{C}\tilde{\mathbf{x}}$ has been used in writing the error in the estimation of the output, $(y - \tilde{y})$. The $(n \times 1)$ observer gain vector \mathbf{K}_e does a weighting on how the error affects each estimate. In the next two sections, we will apply the state estimator in (9-32) to a state feedback system, and see how we can formulate the problem such that the error $(y - \tilde{y})$ can become zero.

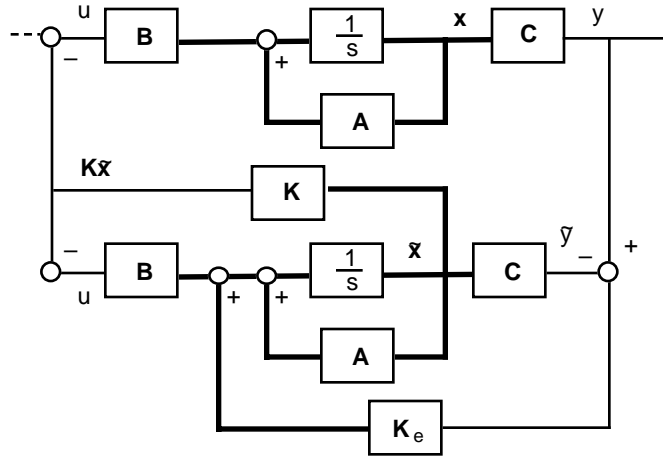


Figure 9.5. A regulator system with controller-estimator

9.3.2. Full-order state estimator system

A system making use of the state estimator is shown in Fig. 9.5, where for the moment, changes in the reference is omitted. What we need is the set of equations that describe this regulator system with state estimation.

By itself, the estimator in Eq. (9-32) has the characteristic equation:

$$|s\mathbf{I} - \mathbf{A} + \mathbf{K}_e\mathbf{C}| = 0 \quad (9-33)$$

Our intention is to use the estimated states to provide feedback information:

$$\mathbf{u} = -\mathbf{K}\tilde{\mathbf{x}} \quad (9-34)$$

The state space model Eq. (9-13) now appears as

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} = \mathbf{A}\mathbf{x} - \mathbf{B}\mathbf{K}\tilde{\mathbf{x}} \quad (9-35)$$

If we substitute $\mathbf{y} = \mathbf{C}\mathbf{x}$ in (9-32), we can integrate Eqs. (9-32) and (9-35) simultaneously to compute $\mathbf{x}(t)$ and $\tilde{\mathbf{x}}(t)$. In matrix form, this set of $2n$ equations can be written as

$$\frac{d}{dt} \begin{bmatrix} \mathbf{x} \\ \tilde{\mathbf{x}} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & -\mathbf{B}\mathbf{K} \\ \mathbf{K}_e\mathbf{C} & \mathbf{A} - \mathbf{K}_e\mathbf{C} - \mathbf{B}\mathbf{K} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \tilde{\mathbf{x}} \end{bmatrix} \quad (9-36)$$

9.3.3. Estimator design

With Eq. (9-36), it is not obvious how \mathbf{K}_e affects the choices of \mathbf{K} . We now derive a form of (9-36) that is based on the error of the estimation and is easier for us to make a statement on its properties. We define the state error vector as

$$\mathbf{e}(t) = \mathbf{x}(t) - \tilde{\mathbf{x}}(t) \quad (9-37)$$

Subtract Eq. (9-32) from (9-35), and use $\mathbf{y} = \mathbf{C}\mathbf{x}$, we should find

$$(\dot{\mathbf{x}} - \dot{\tilde{\mathbf{x}}}) = (\mathbf{A} - \mathbf{K}_e\mathbf{C})(\mathbf{x} - \tilde{\mathbf{x}}) \quad \text{or} \quad \dot{\mathbf{e}} = (\mathbf{A} - \mathbf{K}_e\mathbf{C})\mathbf{e} \quad (9-38)$$

This error equation has the same characteristic equation as the estimator in Eq. (9-33). The goal is to choose eigenvalues of the estimator such that the error decays away quickly. We may note that the form of (9-38) is the same as that of the regulator problem. Thus we should be able to use the

tools of pole-placement for the estimator design. In fact, we can apply, without derivation, a modified form of Ackermann's formula to evaluate

$$\mathbf{K}_e = \alpha_e(\mathbf{A}) \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{n-1} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \quad (9-39)$$

where as analogous to Eq. (9-20),

$$\alpha_e(s) = s^n + \alpha_{n-1}s^{n-1} + \dots + \alpha_1s + \alpha_0 \quad (9-40)$$

is the polynomial derived from our own chosen estimator eigenvalues. Eq. (9-39) is different from Eq. (9-22) because we are now solving the *dual* problem for the $(n \times 1)$ vector \mathbf{K}_e .

Next, we can replace $\tilde{\mathbf{x}}$ in Eq. (9-35) by the definition of the error vector, and the equation becomes

$$\dot{\mathbf{x}} = \mathbf{Ax} - \mathbf{BK}(\mathbf{x} - \mathbf{e}) \quad (9-41)$$

Eqs. (9-38) and (9-41) can be put in matrix form as

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{e}} \end{bmatrix} = \begin{bmatrix} \mathbf{A} - \mathbf{BK} & \mathbf{BK} \\ \mathbf{0} & \mathbf{A} - \mathbf{K}_e\mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{e} \end{bmatrix} \quad (9-42)$$

Now, it is clear that the characteristic equation of the controller-estimator system is

$$|s\mathbf{I} - \mathbf{A} + \mathbf{BK}| |s\mathbf{I} - \mathbf{A} + \mathbf{K}_e\mathbf{C}| = 0 \quad (9-43)$$

We have the very important result that choices for the eigenvalues for the pole-placement design and the observer design can be made independently. Generally, we want the observer response to be two to five times faster than the system response. We should not have to worry about saturation since the entire observer is software-based, but we do have to consider noise and sensitivity problems.

Example 9.3: Consider the second order model in Example 9.1, which we have calculated the state feedback gains in Example 9.2. What is the observer gain vector \mathbf{K}_e if we specify that the estimator error should have eigenvalues -9 repeated thrice?

With eigenvalues selected at -9 , we have chosen the estimator to be faster than the state feedback, and all the errors are to decay exponentially. We'll make use of the Ackermann's formula in Eq. (9-39) for observer gains. The MATLAB statements are:

```
A=[0 1 0; 0 0 1; -6 -11 -6]; %Define the model
B=[0; 0; 1];
C=[1 0 0];

pe=poly([-9 -9 -9]); %Make estimator polynomial (9-40)
ae=polyvalm(pe,A);
Ob=[C; C*A; C*A^2];
Ke=ae*inv(Ob)*[0; 0; 1] %Eq. (9-39)
```

We should find that $\mathbf{K}_e = (21, 106, -144)$. The estimator calculations are purely mathematical, and the values of the observer gains can be negative. Furthermore, we can check that the system of equations in Eq. (9-42) has the correct eigenvalues as suggested by Eq. (4-43).

```

K=[102 43 6]; %Feedback gains calculated from Example 9.2
A11=A-B*K; %Submatrices in Eq. (9-42)
A12=B*K;
A21=zeros(3,3);
A22=A-Ke*C;
BIGA=[A11 A12; A21 A22];
eig(BIGA)

```

Indeed, we should find that the big matrix BIGA has eigenvalues $-3 \pm 3j$, -6 , and -9 repeated three times.

9.3.4. Reduced-order estimator

We should not have to estimate variables that we can measure. It is logical to design a reduced-order estimator which estimates only the states that cannot be measured or are too noisy to be measured accurately. Following our introductory practice, we will consider only one measured output. The following development assumes that we have selected x_1 to be the measured variable. Hence, the output is

$$y = \mathbf{C}\mathbf{x} = [1 \ 0 \ \dots \ 0] \mathbf{x} \quad (9-44)$$

Next, we partition the state vector as

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \mathbf{x}_e \end{bmatrix} \quad (9-45)$$

where $\mathbf{x}_e = [x_2 \ \dots \ x_n]$ contains the $(n-1)$ states that have to be estimated. The state model equation (9-13) is partitioned accordingly as

$$\begin{bmatrix} \dot{x}_1 \\ \dot{\mathbf{x}}_e \end{bmatrix} = \begin{bmatrix} a_{11} & \mathbf{A}_{1e} \\ \mathbf{A}_{e1} & \mathbf{A}_{ee} \end{bmatrix} \begin{bmatrix} x_1 \\ \mathbf{x}_e \end{bmatrix} + \begin{bmatrix} b_1 \\ \mathbf{B}_e \end{bmatrix} u \quad (9-46)$$

where the dimensions of \mathbf{A}_{1e} , \mathbf{A}_{e1} , \mathbf{A}_{ee} are, respectively, $(1 \times n-1)$, $(n-1 \times 1)$, and $(n-1 \times n-1)$, and that of \mathbf{B}_e is $(n-1 \times 1)$.

The next task is to make use of the full state estimator equations. Before that, we have to remold Eq. (9-46) as if it were a full state problem. This exercise requires some careful bookkeeping of notations. Let's take the first row in Eq. (9-46) and make it to constitute the output equation. Thus we make a slight rearrangement:

$$\dot{x}_1 - a_{11}x_1 - b_1u = \mathbf{A}_{1e}\mathbf{x}_e$$

such that it takes the form of $y = \mathbf{C}\mathbf{x}$. We repeat with the second row of (9-46) and put it as

$$\dot{\mathbf{x}}_e = \mathbf{A}_{ee}\mathbf{x}_e + (\mathbf{A}_{e1}x_1 + \mathbf{B}_eu)$$

such that it can be compared with $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u$.

The next step is to take the full state estimator in Eq. (9-32),

$$\dot{\hat{\mathbf{x}}} = (\mathbf{A} - \mathbf{K}_e\mathbf{C})\hat{\mathbf{x}} + \mathbf{B}u + \mathbf{K}_ey$$

and substitute term by term using the reduced-order model equations.¹ The result is, finally,

¹ The matching of terms for reduced-order substitution in Eq. (9-31) to derive (9-47) to (9-49):

$$\dot{\tilde{\mathbf{x}}}_e = (\mathbf{A}_{ee} - \mathbf{K}_{er}\mathbf{A}_{1e})\tilde{\mathbf{x}}_e + (\mathbf{A}_{e1}\mathbf{x}_1 + \mathbf{B}_e\mathbf{u}) + \mathbf{K}_{er}(\dot{\mathbf{x}}_1 - \mathbf{a}_{11}\mathbf{x}_1 - \mathbf{b}_1\mathbf{u}) \quad (9-47)$$

which is the reduced-order equivalent to (9-32). Note that in this equation, $\mathbf{x}_1 = \mathbf{y}$.

The computation of the $(n - 1)$ weighting factors in \mathbf{K}_{er} can be based on the equivalent form of Eq. (9-38). Again, doing the substitution for the notations, the error estimate becomes

$$\dot{\mathbf{e}} = (\mathbf{A}_{ee} - \mathbf{K}_{er}\mathbf{A}_{1e})\mathbf{e} \quad (9-48)$$

which means that the Ackermann's formula in Eq. (9-39) now takes the form

$$\mathbf{K}_{er} = \alpha_e(\mathbf{A}_{ee}) \begin{bmatrix} \mathbf{A}_{1e} \\ \mathbf{A}_{1e}\mathbf{A}_{ee} \\ \vdots \\ \mathbf{A}_{1e}\mathbf{A}_{ee}^{n-1} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \quad (9-49)$$

We are not quite done yet. If we use Eq. (9-47) to compute $\tilde{\mathbf{x}}_e$, it requires taking the derivative of \mathbf{x}_1 , an exercise that can easily amplify noise. So we want a modified form that allows us to replace this derivative. To begin, we define a new variable

$$\tilde{\mathbf{x}}_{e1} = \tilde{\mathbf{x}}_e - \mathbf{K}_{er}\mathbf{x}_1 \quad (9-50)$$

This variable is substituted into (9-47) to give

$$(\dot{\tilde{\mathbf{x}}}_{e1} + \mathbf{K}_{er}\dot{\mathbf{x}}_1) = (\mathbf{A}_{ee} - \mathbf{K}_{er}\mathbf{A}_{1e})(\tilde{\mathbf{x}}_{e1} + \mathbf{K}_{er}\mathbf{x}_1) + (\mathbf{A}_{e1}\mathbf{x}_1 + \mathbf{B}_e\mathbf{u}) + \mathbf{K}_{er}(\dot{\mathbf{x}}_1 - \mathbf{a}_{11}\mathbf{x}_1 - \mathbf{b}_1\mathbf{u})$$

After cancellation of the derivative term, we have

$$\begin{aligned} \dot{\tilde{\mathbf{x}}}_{e1} &= (\mathbf{A}_{ee} - \mathbf{K}_{er}\mathbf{A}_{1e})\tilde{\mathbf{x}}_{e1} \\ &\quad + (\mathbf{A}_{ee}\mathbf{K}_{er} - \mathbf{K}_{er}\mathbf{A}_{1e}\mathbf{K}_{er} + \mathbf{A}_{e1} - \mathbf{K}_{er}\mathbf{a}_{11})\mathbf{x}_1 + (\mathbf{B}_e - \mathbf{K}_{er}\mathbf{b}_1)\mathbf{u} \end{aligned} \quad (9-51)$$

This differential equation is used to compute $\tilde{\mathbf{x}}_{e1}$, which then is used to calculate $\tilde{\mathbf{x}}_e$ with (9-50).

With the estimated states, we can compute the feedback to the state space model as

$$\mathbf{u} = - \begin{bmatrix} \mathbf{K}_1 & \mathbf{K}_{1e}^T \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \tilde{\mathbf{x}}_e \end{bmatrix} \quad (9-52)$$

The application of Eqs. (9-50) to (9-52) is a bit involved and best illustrated as shown in Fig. 9.6.

<u>Full-order state estimator</u>	<u>Reduced-order state estimator</u>
-----------------------------------	--------------------------------------

$\tilde{\mathbf{x}}$	$\tilde{\mathbf{x}}_e$
\mathbf{y}	$\dot{\mathbf{x}}_1 - \mathbf{a}_{11}\mathbf{x}_1 - \mathbf{b}_1\mathbf{u}$
\mathbf{C}	\mathbf{A}_{1e}
\mathbf{A}	\mathbf{A}_{ee}
$\mathbf{K}_e, (n \times 1)$	$\mathbf{K}_{er}, (n-1 \times 1)$
$\mathbf{B}\mathbf{u}$	$\mathbf{A}_{e1}\mathbf{x}_1 + \mathbf{B}_e\mathbf{u}$

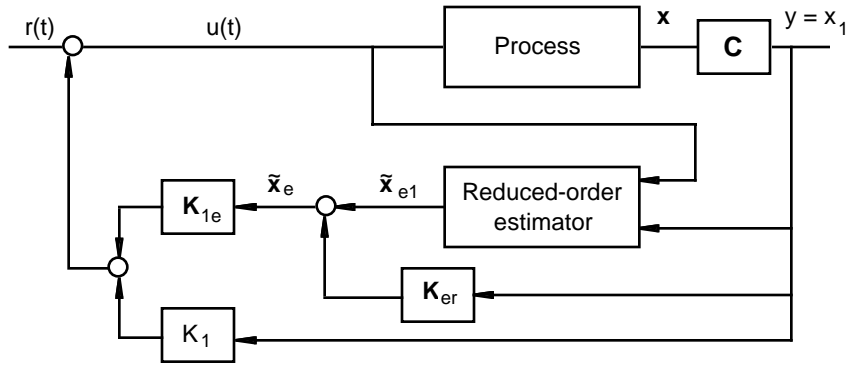


Figure 9.6. State feedback with reduced-order estimator

🔗 **Example 9.4:** Consider the estimator in Example 9.3, what is the reduced-order observer gain vector \mathbf{K}_{er} if we specify that the estimator error should have eigenvalues -9 repeated twice?

We can use Eq. (9-49), and the MATLAB statements are:

```
A=[0 1 0; 0 0 1; -6 -11 -6];
N=size(A,1);
a11=A(1,1); %Extract matrix partitions as in Eq. (9-46)
Ale=A(1,2:N);
Ae1=A(2:N,1);
Aee=A(2:N,2:N);

pe=poly([-9 -9]); %Make estimator polynomial
ae=polyvalm(pe,Aee);
Ob=[Ale; Ale*Aee];
Ker=ae*inv(Ob)*[0; 1] %Eq. (9-49) for n=2
```

We should find that $\mathbf{K}_{er} = (12 \ -2)$.

After all this fancy mathematics, we need a word of caution. It is extremely dangerous to apply the state estimate as presented in this chapter. Why? The first hint is in Eq. (9-32). We have assumed perfect knowledge of the plant matrices. Of course, we rarely do. Furthermore, we have omitted actual terms for disturbances, noises, and errors in measurements. Despite these drawbacks, material in this chapter provides the groundwork to attack serious problems in modern control.

Review Problems

1. For the second order transfer function

$$\frac{Y}{U} = \frac{1}{s^2 + 2\zeta\omega_n s + \omega_n^2},$$

derive the controllable canonical form. If the desired poles of a closed-loop system are to be placed at λ_1 and λ_2 , what should be the state feedback gains?

2. Presume we do not know what the estimator should be other than that it has the form

$$\dot{\tilde{\mathbf{x}}} = \mathbf{F}\tilde{\mathbf{x}} + \mathbf{G}u + \mathbf{H}y$$

Find Eq. (9-32).

- Do the time response simulation in Example 7.5B. We found that the state space system has a steady state error. Implement integral control and find the new state feedback gain vector. Perform a time response simulation to confirm the result.
- With respect to Fig. R9.4, what is the transfer function equivalent to the controller-estimator system in Eq. (9-32)?

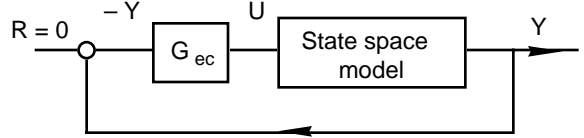


Figure R9.4

Hints:

- The controllable canonical form was derived in Example 4.1. The characteristic polynomial of $(s\mathbf{I} - \mathbf{A} + \mathbf{BK})$ should be

$$s^2 + (2\zeta\omega_n + K_2)s + (\omega_n^2 + K_1) = 0$$

The characteristic polynomial of desired poles is

$$s^2 + (\lambda_1 + \lambda_2)s + \lambda_1\lambda_2 = 0$$

Thus

$$K_1 = \lambda_1\lambda_2 - \omega_n^2 \quad \text{and} \quad K_2 = (\lambda_1 + \lambda_2) - 2\zeta\omega_n$$

- The Laplace transform of the given equation is

$$s\tilde{\mathbf{X}} = \mathbf{F}\tilde{\mathbf{X}} + \mathbf{G}U + \mathbf{H}Y$$

Substituting $Y = \mathbf{C}\mathbf{X}$, we have

$$\tilde{\mathbf{X}} = (s\mathbf{I} - \mathbf{F})^{-1}[\mathbf{G}U + \mathbf{H}\mathbf{C}\mathbf{X}]$$

We further substitute for $\mathbf{X} = (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}U$ with the simple state space model to give

$$\tilde{\mathbf{X}} = (s\mathbf{I} - \mathbf{F})^{-1}[\mathbf{G} + \mathbf{H}\mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}]U$$

What we want is to dictate that the transfer function of this estimator is the same as that of the state space model:

$$(s\mathbf{I} - \mathbf{F})^{-1}[\mathbf{G} + \mathbf{H}\mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}] = (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}$$

Move the second term to the RHS and factor out the $(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}$ gives

$$(s\mathbf{I} - \mathbf{F})^{-1}\mathbf{G} = [\mathbf{I} - (s\mathbf{I} - \mathbf{F})^{-1}\mathbf{H}\mathbf{C}](s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}$$

Thus we can multiply $(s\mathbf{I} - \mathbf{F})$ to both sides to have

$$\mathbf{G} = [(s\mathbf{I} - \mathbf{F}) - \mathbf{H}\mathbf{C}](s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}$$

And finally,

$$[(s\mathbf{I} - \mathbf{F}) - \mathbf{H}\mathbf{C}]^{-1}\mathbf{G} = (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}$$

Compare term by term, we have

$$\mathbf{F} + \mathbf{H}\mathbf{C} = \mathbf{A}, \quad \text{or} \quad \mathbf{F} = \mathbf{A} - \mathbf{H}\mathbf{C}$$

and

$$\mathbf{G} = \mathbf{B}$$

This result is what we need in (9-32) if we also set $\mathbf{H} = \mathbf{K}_e$.

3. For the time response simulation, we also duplicate the classical control design for comparison. Both classical and state space results have the same damping ratio, but not system steady state gain. The statements are:

```
G=tf(1,conv([2 1],[4 1]));
S=ss(G); %MATLAB uses reverse indexing
scale=S.c(2); %And need to rescale B and C too
S.c=S.c/scale;
S.b=S.b*scale;
P=[-0.375+0.382j -0.375-0.382j]; %Define the closed-loop poles
K=acker(S.a,S.b,P)
%Compute the system matrices for plotting
A = S.a - S.b*K %Makes system matrix, Eq. (9-25)
B = S.b*K(2)
C = S.c
D=0;
step(A,B,C,D)
hold %to add the classical design result
Gcl=feedback(1.29*G,1);
%Kc=1.29 was the proportional gain obtained in Example 7.5A
step(Gcl)
```

To eliminate offset, we need Section 9.2.3. With an added state due to integration, we have to add one more closed-loop poles. We choose it to be -1 , sufficiently faster than the real part of the complex poles. The statements are:

```
G=tf(1,conv([2 1],[4 1]));
S=ss(G); %Generates the matrices S.a, S.b, S.c, S.d
Ah=[S.a zeros(2,1); -S.c 0] %  $\hat{\mathbf{A}}$  in (9-29)
Bh=[S.b; 0] %  $\hat{\mathbf{B}}$ 
P=[-0.375+0.382j -0.375-0.382j -1]; %Add a faster pole at -1
kh=acker(Ah,Bh,P) %K-head in (9-29) %  $\hat{\mathbf{K}}$ 
```

We should find $\hat{\mathbf{K}} = [2 \ 3.6 \ -2.3]$. To do the time response simulation, we can use:

```
Asys=Ah-Bh*kh; %System matrix (9-29)
Bsys=[0; 0; 1]; %Follows (9-28)
Csys=[S.c 0];
step(Asys, Bsys,Csys,0)
```

4. For the estimator, y is the input and u the output. With $u = -\mathbf{K}\tilde{\mathbf{x}}$, the Laplace transform of Eq. (9-32) is

$$[s\mathbf{I} - \mathbf{A} + \mathbf{K}_e\mathbf{C} + \mathbf{BK}]\tilde{\mathbf{X}}(s) = \mathbf{K}_e\mathbf{Y}(s)$$

or
$$\tilde{\mathbf{X}}(s) = [s\mathbf{I} - \mathbf{A} + \mathbf{K}_e\mathbf{C} + \mathbf{BK}]^{-1}\mathbf{K}_e\mathbf{Y}(s)$$

We now substitute $\tilde{\mathbf{X}}$ back in the Laplace transform of $u = -\mathbf{K}\tilde{\mathbf{x}}$ to obtain

$$U(s) = -\mathbf{K}[s\mathbf{I} - \mathbf{A} + \mathbf{K}_e\mathbf{C} + \mathbf{BK}]^{-1}\mathbf{K}_e\mathbf{Y}(s) = -\mathbf{G}_{ec}(s)\mathbf{Y}(s)$$