16 CONTROL FUNDAMENTALS

16.1 Introduction

16.1.1 Plants, Inputs, and Outputs

Controller design is about creating dynamic systems that behave in useful ways. Many target systems are physical; we employ controllers to steer ships, fly jets, position electric motors and hydraulic actuators, and distill alcohol. Controllers are also applied in macroeconomics and many other important, non-physical systems. It is the fundamental concept of controller design that a set of input variables acts through a given "plant" to create an output. Feedback control then uses sensed plant outputs to apply corrective inputs:

Plant	Inputs	Outputs	Sensors
Jet aircraft	elevator, rudder, etc.	altitude, hdg	altimeter, GPS
Marine vessel	rudder angle	heading	gyrocompass
Hydraulic robot	valve position	tip position	joint angle
U.S. economy	fed interest rate, etc.	prosperity	inflation, M1
Nuclear reactor	cooling, neutron flux	power level	temp., pressure

(Continued on next page)

16.1.2 The Need for Modeling

Effective control system design usually benefits from an accurate model of the plant, although it must be noted that many industrial controllers can be tuned up satisfactorily with no knowledge of the plant. Ziegler and Nichols, for example, developed a general recipe which we detail later. In any event, plant models simply do not match real-world systems exactly; we can only hope to capture the basic components in the form of differential or integro-differential equations.
Beyond prediction of plant behavior based on physics, the process of *system identification* generates a plant model from data. The process is often problematic, however, since the measured response could be corrupted by sensor noise or physical disturbances in the system which cause it to behave in unpredictable ways. At some frequency high enough, most systems exhibit effects that are difficult to model or reproduce, and this is a limit to controller performance.

16.1.3 Nonlinear Control

The bulk of this subject is taught using the tools of linear systems analysis. The main reason for this restriction is that nonlinear systems are difficult to model, difficult to design controllers for, and difficult overall! Within the paradigm of linear systems, there are many sets of powerful tools available. The reader interested in nonlinear control is referred to the book by Slotine and Li (1991).

16.2 Representing Linear Systems

Except for the most heuristic methods of tuning up simple systems, control system design depends on a model of the plant. The transfer function description of linear systems has already been described in the discussion of the Laplace transform. The state-space form is an entirely equivalent *time-domain* representation that makes a clean extension to systems with multiple inputs and multiple outputs, and opens the way to standard tools from linear algebra.

16.2.1 Standard State-Space Form

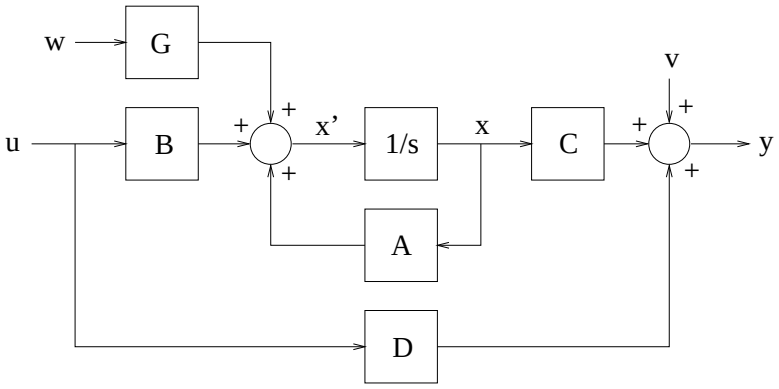
We write a linear system in a state-space form as follows

$$\begin{aligned} \dot{x} &= Ax + Bu + Gw \\ y &= Cx + Du + v \end{aligned} \tag{193}$$

where

- x is a state vector, with as many elements as there are orders in the governing differential equations.
- A is a matrix mapping x to its derivative; A captures the natural dynamics of the system without external inputs.

- B is an input gain matrix for the control input u.
- G is a gain matrix for unknown disturbance w; w drives the state just like the control u.
- y is the observation vector, comprised mainly of a linear combination of states Cx (where C is a matrix).
- Du is a direct map from input to output (usually zero for physical systems).
- v is an unknown sensor noise which corrupts the measurement.



16.2.2 Converting a State-Space Model into a Transfer Function

There are a number of canonical state-space forms available, which can create the same transfer function. In the case of no disturbances or noise, the transfer function (or transfer matrix) can be written as

$$G(s) = \frac{y(s)}{u(s)} = C(sI - A)^{-1}B + D, \tag{194}$$

where I is the identity matrix with the same size as A. A similar equation holds for $y(s)/w(s)$, and clearly $y(s)/v(s) = I$.

16.2.3 Converting a Transfer Function into a State-Space Model

It may be possible to write the corresponding differential equation along one row of the state vector, and then cascade derivatives. For example, consider the following system:

$$\begin{aligned} my''(t) + by'(t) + ky(t) &= u'(t) + u(t) \text{ (mass-spring-dashpot)} \\ G(s) &= \frac{s+1}{ms^2 + bs + k} \end{aligned}$$

Setting $\vec{x} = [y', y]^T$, we obtain the system

$$\begin{aligned} \frac{d\vec{x}}{dt} &= \begin{bmatrix} -b/m & -k/m \\ 1 & 0 \end{bmatrix} \vec{x} + \begin{bmatrix} 1/m \\ 0 \end{bmatrix} u \\ y &= [1 \;\; 1]\, \vec{x} \end{aligned}$$

Note specifically that $dx_2/dt = x_1$, leading to an entry of 1 in the off-diagonal of the second row in A. Entries in the C-matrix are easy to write in this case because of linearity; the system response to u' is the same as the derivative of the system response to u.

16.3 PID Controllers

The most common type of industrial controller is the proportional-integral-derivative (PID) design. If u is the output from the controller, and e is the error signal it receives, this control law has the form

$$\begin{aligned} u(t) &= k_p e(t) + k_i \int_0^t e(\tau) d\tau + k_d e'(t), \\ C(s) = \frac{U(s)}{E(s)} &= k_p + \frac{k_i}{s} + k_d s \\ &= k_p \left[1 + \frac{1}{\tau_i s} + \tau_d s \right], \end{aligned} \tag{195}$$

where the last line is written using the conventions of one overall gain k_p, plus a time characteristic to the integral part (τ_i) and and time characteristic to the derivative part (τ_d). In words, the proportional part of this control law will create a control action that scales linearly with the error – we often think of this as a spring-like action. The integrator is accumulating the error signal over time, and so the control action from this part will continue to grow as long as an error exists. Finally, the derivative action scales with the derivative of the error. The controller will retard motion toward zero error, which helps to reduce overshoot.
The common variations are: P, PD, PI, PID.

16.4 Example: PID Control

Consider the case of a mass (m) sliding on a frictionless table. It has a perfect thruster that generates force $u(t)$, but is also subject to an unknown disturbance $d(t)$. If the linear position of the mass is $y(t)$, and it is perfectly measured, we have the plant

$$my''(t) = u(t) + d(t).$$

Suppose that the desired condition is simply $y(t) = 0$, with initial conditions $y(0) = y_o$ and $y'(0) = 0$.

16.4.1 Proportional Only

A proportional controller alone invokes the control law $u(t) = -k_p y(t)$, so that the closed-loop dynamics follow

$$my''(t) = -k_p y(t) + d(t).$$

In the absence of $d(t)$, we see that $y(t) = y_o \cos\sqrt{\frac{k_p}{m}}t$, a marginally stable response that is undesirable.

16.4.2 Proportional-Derivative Only

Let $u(t) = -k_p y(t) - k_d y'(t)$, and it follows that

$$my''(t) = -k_p y(t) - k_d y'(t) + d(t).$$

The system now resembles a second-order mass-spring-dashpot system where k_p plays the part of the spring, and k_d the part of the dashpot. With an excessively large value for k_d, the system would be overdamped and very slow to respond to any command. In most applications, a small amount of overshoot is employed because the response time is shorter. The k_d value for critical damping in this example is $2\sqrt{mk_p}$, and so the rule is $k_d < 2\sqrt{mk_p}$. The result, easily found using the Laplace transform, is

$$y(t) = y_o e^{\frac{-k_d}{2m}t}\left[\cos\omega_d t + \frac{k_d}{2m\omega_d}\sin\omega_d t\right],$$

where $\omega_d = \sqrt{4mk_p - k_d^2}/2m$. This response is exponentially stable as desired. Note that if the mass had a very large amount of natural damping, a *negative* k_d could be used to cancel some of its effect and speed up the system response.

Now consider what happens if $d(t)$ has a constant bias d_o: it balances exactly the proportional control part, eventually settling out at $y(t = \infty) = d_o/k_p$. To achieve good rejection of d_o with a PD controller, we would need to set k_p very large. However, very large values of k_p will also drive the resonant frequency ω_d up, which is unacceptable.

16.4.3 Proportional-Integral-Derivative

Now let $u(t) = -k_p y(t) - k_i \int_0^t y(\tau)d\tau - k_d y'(t)$: we have

$$my''(t) = -k_p y(t) - k_i \int_0^t y(\tau)d\tau - k_d y'(t) + d(t).$$

The control system has now created a third-order closed-loop response. If $d(t) = d_o$, a time derivative leads to

$$my'''(t) + k_p y'(t) + k_i y(t) + k_d y''(t) = 0,$$

so that $y(t = \infty) = 0$, as desired, provided the roots are stable.

16.5 Heuristic Tuning

For many practical systems, tuning of a PID controller may proceed without any system model. This is especially pertinent for plants which are open-loop stable, and can be safely tested with varying controllers. One useful approach is due to Ziegler and Nichols (e.g., Bélanger,1995), which transforms the basic characteristics of a step response (e.g., the input is $1(t)$) into a reasonable PID design. The idea is to approximate the response curve by a first-order lag (gain k and time constant τ) and a pure delay T:

$$G(s) \simeq \frac{ke^{-Ts}}{\tau s + 1} \tag{196}$$

The following rules apply *only* if the plant contains no dominating, lightly-damped complex poles, and has no poles at the origin:

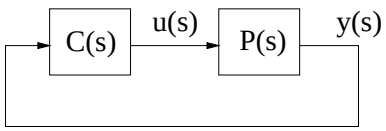
P	$k_p = 1.0\tau/T$		
PI	$k_p = 0.9\tau/T$	$k_i = 0.27\tau/T^2$	
PID	$k_p = 1.2\tau/T$	$k_i = 0.60\tau/T^2$	$k_d = 0.60\tau$

Note that if no pure time delay exists ($T = 0$), this recipe suggests the proportional gain can become arbitrarily high! Any characteristic other than a true first-order lag would therefore be expected to cause a measurable delay.

16.6 Block Diagrams of Systems

16.6.1 Fundamental Feedback Loop

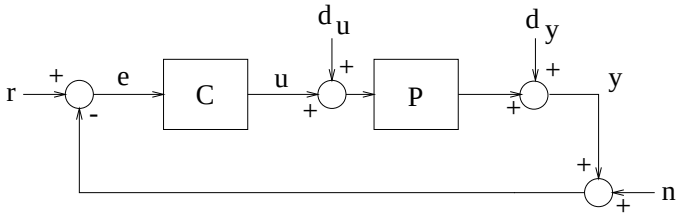
The topology of a feedback system can be represented graphically by considering each dynamical system element to reside within a box, having an input line and an output line. For example, the plant used above (a simple mass) has transfer function $P(s) = 1/ms^2$, which relates the input, force $u(s)$, into the output, position $y(s)$. In turn, the PD-controller has transfer function $C(s) = k_p + k_d s$; its input is the error signal $E(s) = -y(s)$, and its output is force $u(s)$. The feedback loop in block diagram form is shown below.



16.6.2 Block Diagrams: General Case

The simple feedback system above is augmented in practice by three external inputs. The first is a process disturbance, which can be taken to act at the input of the physical plant, or at the output. In the former case, it is additive with the control action, and so has

some physical meaning. In the second case, the disturbance has the same units as the plant output.
Another external input is the *reference command* or *setpoint*, used to create a more general error signal $e(s) = r(s) - y(s)$. Note that the feedback loop, in trying to force $e(s)$ to zero, will necessarily make $y(s)$ approximate $r(s)$.
The final input is sensor noise, which usually corrupts the feedback signal $y(s)$, causing some error in the evaluation of $e(s)$, and so on. Sensors with very poor noise properties can ruin the performance of a control system, no matter how perfectly understood are the other components.



16.6.3 Primary Transfer Functions

Some algebra shows that

$$\begin{aligned}
\frac{e}{r} &= \frac{1}{1+PC} = S \\
\frac{y}{r} &= \frac{PC}{1+PC} = T \\
\frac{u}{r} &= \frac{C}{1+CP} = U.
\end{aligned}$$

$e/r = S$ relates the reference input and noise to the error, and is known as the *sensitivity function*. We would generally like S to be small at low frequencies, so that the tracking error there is small. $y/r = T$ is called the *complementary sensitivity function*. Note that $S + T = 1$, implying that these two functions must always trade off; they cannot both be small or large at the same time. Other systems we encounter again later are the *(forward) loop transfer function* PC, the loop transfer function broken between C and P: CP, and

$$\begin{aligned}
\frac{e}{d_u} &= \frac{-P}{1+PC} \\
\frac{y}{d_u} &= \frac{P}{1+PC} \\
\frac{u}{d_u} &= \frac{-CP}{1+CP} \\
\frac{e}{d_y} &= \frac{-1}{1+PC} = -S
\end{aligned}$$

$$\begin{aligned}
\frac{y}{d_y} &= \frac{1}{1+PC} = S \\
\frac{u}{d_y} &= \frac{-C}{1+CP} = -U \\
\frac{e}{n} &= \frac{-1}{1+PC} = -S \\
\frac{y}{n} &= \frac{-PC}{1+PC} = -T \\
\frac{u}{n} &= \frac{-C}{1+CP} = -U.
\end{aligned}$$

If the disturbance is taken at the plant output, then the three functions S, T, and U (control action) completely describe the system. This will in fact be the procedure when we address loopshaping.