11 CONTROL FUNDAMENTALS

11.1 Introduction

11.1.1 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 macro-economics 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 plant 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	inflation, M1
Nuclear reactor	cooling, neutron flux	heat, power level	temp., pressure

11.1.2 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 heuristic 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 other equations.

Beyond prediction of plant behavior based on physics, *system identification* generates a plant model from actual 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.

11.1.3 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).

11.2 Partial Fractions

Partial fractions are presented here, in the context of control systems, as the *fundamental* link between pole locations and stability. Solving linear time-invariant systems by the Laplace Transform method will generally create a signal containing the (factored) form

$$Y(s) = \frac{K(s+z_1)(s+z_2)\cdots(s+z_m)}{(s+p_1)(s+p_2)\cdots(s+p_n)}. \tag{1}$$

Although for the moment we are discussing the signal $Y(s)$, later we will see that dynamic systems are described in the same format: in that case we call the impulse response $G(s)$ a transfer function. A system transfer function is identical to its impulse response, since $L(\delta(t)) = 1$.

The constants $-z_i$ are called the zeros of the transfer function or signal, and $-p_i$ are the poles. Viewed in the complex plane, it is clear that the magnitude of $Y(s)$ will go to zero at the zeros, and to infinity at the poles.

Partial fraction expansions alter the form of $Y(s)$ so that the simple first- and second-order transform pairs can be used to find the time-domain output signals. We must have $m < n$ for this procedure; if this is not the case, then we have to strip off extra powers of s to solve the problem, and then add them back on at the end.

11.2.1 Partial Fractions: Unique Poles

Under the condition $m < n$, it is a fact that $Y(s)$ is equivalent to

$$Y(s) = \frac{a_1}{s+p_1} + \frac{a_2}{s+p_2} + \cdots \frac{a_n}{s+p_n}, \tag{2}$$

in the special case that all of the poles are unique and real. The coefficient a_i is termed the *residual* associated with the i'th pole, and once all these are found it is a simple matter to go back to the transform table and look up the time-domain responses.

How to find a_i? A simple rule applies: multiply the right-hand sides of the two equations above by $(s+p_i)$, evaluate them at $s = -p_i$, and solve for a_i, the only one left.

Example: Partial Fractions with Unique Real Poles

$$G(s) = \frac{s(s+6)}{(s+4)(s-1)}e^{-2s}.$$

Since we have a pure delay and $m = n$, we can initially work with $G(s)/se^{-2s}$. We have

$$\frac{s+6}{(s+4)(s-1)} = \frac{a_1}{s+4} + \frac{a_2}{s-1}, \text{ giving}$$

$$\begin{aligned} a_1 &= \left[\tfrac{(s+6)(s+4)}{(s+4)(s-1)}\right]_{s=-4} = -\frac{2}{5} \\ a_2 &= \left[\tfrac{(s+6)(s-1)}{(s+4)(s-1)}\right]_{s=1} = \frac{7}{5} \end{aligned}$$

Thus

$$\begin{aligned} L^{-1}(G(s)/se^{-2s}) &= -\frac{2}{5}e^{-4t} + \frac{7}{5}e^{t} \longrightarrow \\ g(t) &= \delta(t-2) + \frac{8}{5}e^{-4(t-2)} + \frac{7}{5}e^{t-2}. \end{aligned}$$

The impulse response is needed to account for the step change at $t = 2$. Note that in this example, we were able to apply the derivative operator s *after* expanding the partial fractions. For cases where a second derivative must be taken, i.e., $m \geq n + 1$, special care should be used when accounting for the signal *slope* discontinuity at $t = 0$. The more traditional method, exemplified by Ogata, may prove easier to work through.

The case of repeated real roots may be handled elegantly, but this condition rarely occurs in applications.

11.2.2 Partial Fractions: Complex-Conjugate Poles

A complex-conjugate pair of poles should be kept together, with the following procedure: employ the form

$$Y(s) = \frac{b_1 s + b_2}{(s+p_1)(s+p_2)} + \frac{a_3}{s+p_3} + \cdots, \tag{3}$$

where $p_1 = p_2^*$ (complex conjugate). As before, multiply through by $(s + p_1)(s + p_2)$, and then evaluate at $s = -p_1$.

Example: Partial Fractions with Complex Poles

$$G(s) = \frac{s+1}{s(s+j)(s-j)} = \frac{b_1 s + b_2}{(s+j)(s-j)} + \frac{a_3}{s} :$$

$$\begin{aligned}\left[\frac{s+1}{s}\right]_{s=-j} &= [b_1 s + b_2]_{s=-j} \longrightarrow \\ 1 + j &= -b_1 j + b_2 \longrightarrow \\ b_1 &= -1 \\ b_2 &= 1; \text{ also} \\ \left[\frac{s+1}{(s+j)(s-j)}\right]_{s=0} &= a_3 = 1.\end{aligned}$$

Working out the inverse transforms from the table of pairs, we have simply (noting that $\zeta = 0$)

$$g(t) = -\cos t + \sin t + 1(t).$$

11.3 Stability in Linear Systems

In linear systems, *exponential stability* occurs when all the real exponents of e are strictly negative. The signals decay within an exponential envelope. If one exponent is 0, the response never decays or grows in amplitude; this is called *marginal stability*. If at least one real exponent is positive, then one element of the response grows without bound, and the system is *unstable*.

11.4 Stability $\Longleftrightarrow$ Poles in LHP

In the context of partial fraction expansions, the relationship between stability and pole locations is especially clear. The unit step function $1(t)$ has a pole at zero, the exponential e^{-at} has a pole at $-a$, and so on. All of the other pairs exhibit the same property: *A system is stable if and only if all of the poles occur in the left half of the complex plane.* Marginally stable parts correlate with a zero real part, and unstable parts to a positive real part.

11.5 General Stability

There are two definitions, which apply to systems with input $u(t)$ and output $y(t)$.

1. **Exponential**. If $u(t) = 0$ and $y(0) = y_o$, then $|y(t)| < \alpha e^{-\gamma t}$, for some finite α and $\gamma > 0$. The output asymptotically approaches zero, within a decaying exponential envelope.

2. **Bounded-Input Bounded-Output (BIBO)**. If $y(0) = 0$, and $|u(t)| < \gamma, \gamma > 0$ and finite, then $|y(t)| < \alpha, \alpha > 0$ and finite.

In linear time-invariant systems, the two definitions are identical. Exponential stability is easy to check for linear systems, but for nonlinear systems, BIBO stability is usually easier to achieve.

11.6 Representing Linear Systems

The transfer function description of linear systems has already been described in the presentation 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 many standard tools from linear algebra.

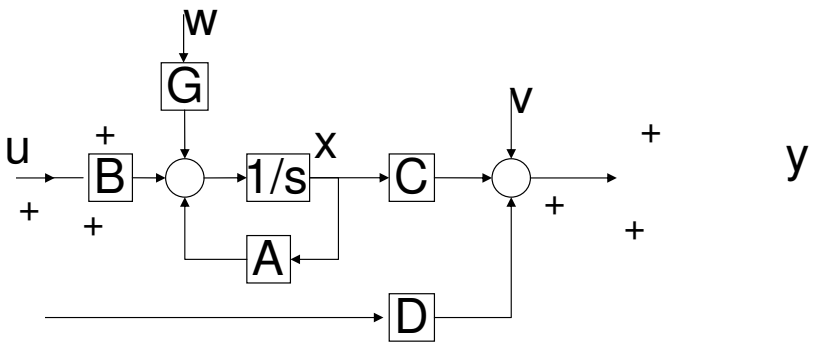
11.6.1 Standard State-Space Form

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

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

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.



11.6.2 Converting a State-Space Model into a Transfer Function

Many different state-space descriptions can create the same transfer function - they are not unique. In the case of no disturbances or noise, the transfer function can be written as

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

where I is the identity matrix with the same size as A. To see that this is true, simply transform the differential equation into frequency space:

$$\begin{aligned} sx(s) &= Ax(s) + Bu(s) \longrightarrow \\ x(s)(sI - A) &= Bu(s) \longrightarrow \\ x(s) &= (sI - A)^{-1}Bu(s) \longrightarrow \\ y(s) &= Cx(s) + Du(s) = C(sI - A)^{-1}Bu(s) + Du(s). \end{aligned}$$

A similar equation holds for $y(s)/w(s)$, and clearly $y(s)/v(s) = 1$.

11.6.3 Converting a Transfer Function into a State-Space Model

Because state-space models are not unique, there are many different ways to create them from a transfer function. In the simplest case, it may be possible to write the corresponding differential equation along one row of the state vector, and then cascade derivatives. example, consider the following system:

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

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

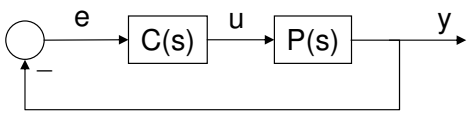
$$\begin{aligned} \frac{d\vec{x}}{dt} &= \begin{bmatrix} -b/m & -k/m \\ 1 & 0 \end{bmatrix} \vec{x} + \begin{bmatrix} 1/m \\ 0 \end{bmatrix} u \\ y &= [1 \;\; 1]\,\vec{x} \end{aligned}$$

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.

11.7 Block Diagrams and Transfer Functions of Feedback Systems

11.7.1 Block Diagrams: Fundamental Form

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, a simple mass driven by a controlled force 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 (see below) 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 $u(s) = -(k_p + k_d s)y(s)$. This feedback loop in block diagram form is shown below.



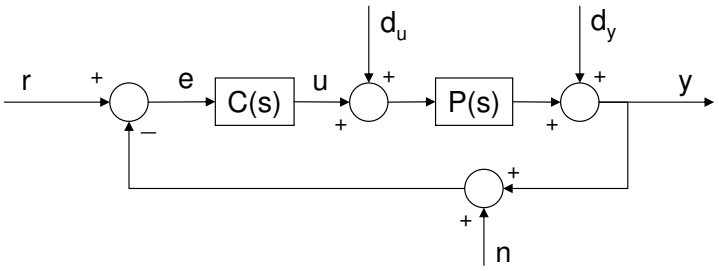
11.7.2 Block Diagrams: General Case

The simple feedback system above is augmented in practice by three external inputs. The first is a process disturbance we call d, 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 n, 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.

Note that the disturbances d_u and d_y, and the noise n are generalizations on the unknown disturbance and sensor noise we discussed at the beginning of this section.



11.7.3 Transfer Functions

Some algebra applied to the above figure (and neglecting the Laplace variable s) shows that

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

Let us derive the first of these. Working directly from the figure, we have

$$\begin{aligned}e(s) &= r(s) - y(s) \\ e(s) &= r(s) - P(s)u(s) \\ e(s) &= r(s) - P(s)C(s)e(s) \\ (1 + P(s)C(s))e(s) &= r(s) \\ \frac{e(s)}{r(s)} &= \frac{1}{1+P(s)C(s)}.\end{aligned}$$

The fact that we are able to make this kind of construction is a direct consequence of the frequency-domain representation of the system, and namely that we can freely multiply and divide system impulse responses and signals, so as to represent convolutions in the time-domain.

Now $e/r = S$ 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 certain frequencies, so that the non-dimensional tracking error e/r 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 some others:

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

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

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 be the procedure when we address loopshaping.

11.8 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

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

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. This will retard motion toward zero error, which helps to reduce overshoot.

The common variations are: P, PD, PI, PID.

11.9 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 $y'(0) = 0$.

11.9.1 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.

11.9.2 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 tolerated 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.

11.9.3 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. Note that for the case of the PD control, it was enough to select k_p positive and k_d positive because these terms represent spring and dashpot-type forces. The use of k_i complicates the stability however, and it is not enough in general to set all three gains positive - stability should be checked explicitly.

11.10 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:

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

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.

2.017J Design of Electromechanical Robotic Systems
Fall 2009