6

Essential Control Theory

Control theory is traditionally taught from the point of view of the frequency response, with great emphasis on the manipulation of transfer functions. Instead we will start with the *state space* approach, based on differential equations that you can identify from the "real" system.

6.1 STATE VARIABLES

The relationship between state variables and *initial conditions* has already been mentioned, but let us try to make the concept as clear as possible.

A cup of coffee has just been prepared. It is rather too hot at the moment, at 80°C. If left for some hours, it would cool down to room temperature at 20°C, but just how fast would it cool, and when would it be at 60°C?

The cup remains full for now, so just one variable interests us: T, the temperature of the coffee. It is a reasonable assumption that the rate of fall in temperature is proportional to the temperature above ambient. So we see that

$$\frac{dT}{dt} = -k(T - T_{\text{ambient}})$$

If we can determine the value of the constant k, perhaps by a simple experiment, then the equation can be solved for any particular initial temperature—although we'll look at the form of the solution later.

Essentials of Mechatronics, by John Billingsley

The concept of state variables is so simple, yet it is essential for gaining insight into dynamic systems. As an exercise, consider the following systems, select state variables, and derive state equations for them:

1. The money in a bank account that carries compound interest.
2. The voltage on a capacitor that has a resistor connected across it—assuming that it is originally charged.
3. The distance of the back wheel of a bicycle from a straight line when the front wheel is wheeled along the line—assume that it starts away from the line.
4. The speed of a motor when driven from voltage V.

Think up your own answers before reading on:

1. The state variable in this case is just your credit balance. To find its rate of change, multiply the credit balance at this very instant by the interest rate. If we call the credit c and the interest rate R, then the equation is just

$$\frac{dc}{dt} = Rc$$

   where, of course, time is measured in years.
2. This time the state variable is the voltage on the capacitor, v. The current that will flow through the resistor is v/R. The equation linking voltage and inflowing current i for a capacitor is

$$i = C\frac{dv}{dt}$$

   Since the current in the resistor is flowing *out* of the capacitor, we have

$$i = -\frac{v}{R}$$

so

$$C\frac{dv}{dt} = -\frac{v}{R}$$

or

$$\frac{dv}{dt} = -\frac{v}{RC}$$

3. First let us assume that the bicycle's angle is small, so that its sine can be assumed to be equal to the value of the angle in radians. Now the state variable can be defined as the distance of the back wheel from the line along which the bicycle is being wheeled. If this distance is x and the length between the wheels is L, then a bit of trigonometry shows that for a small angle the component of the velocity of the back wheel perpendicular to the line is Vx/L toward it, where V is the forward speed. We end up with

$$\frac{dx}{dt} = -V\frac{x}{L}$$

4. As the motor speeds up, a backward electromotive force (back-emf) is generated that opposes the applied voltage V. If the angular velocity is ω, the motor current will be proportional to

$$V - k\omega$$

So, we have

$$\frac{d\omega}{dt} = aV - b\omega$$

where $b = ak$. When the motor reaches its top speed, the acceleration will be zero, so

$$\omega_{\max} = aV/b$$

Equations of this sort apply to a vast range of situations. A rainwater barrel has a small leak at the bottom. The rate of leakage is proportional to the depth, H, and so

$$\frac{dH}{dt} = -kH$$

The water will leak out until eventually the barrel is empty. But suppose now that there is a steady flow *into* the barrel, sufficient to raise the level (without leak) at a speed u. Then the equation becomes

$$\frac{dH}{dt} = -kH + u$$

What will the level of the water settle down at now? When it has reached a steady level, however long that takes, the rate of change of depth will have fallen to zero, so

$$\frac{dH}{dt} = 0$$

It is not hard to see that $-kH + u$ must also be zero, and so

$$H = u/k$$

Now, if we really want to know the depth as a function of time, a mathematical formula can be found for the solution. But let us try another approach first: simulation. See http://www.EssMech.com/6/2.

6.2 SIMULATION

With very little effort, we can construct a computer program that will imitate the behavior of the barrel. The depth right now is H, and we have already described the rate of change of depth dH/dt as $(-kH + u)$. In a short time dt, the depth will have changed by

$$(-kH + u)dt$$

so that in program terms we have

```
H = H + (-k*H + u)*dt
```

This will work as it stands in most computer languages, although some might insist on it ending with a semicolon. Even when wrapped up in input and output statements to make a complete program, the simulation is very simple. In QBasic it is

```
PRINT "Plot of Leaky Barrel
INPUT "Initial level - 0 to 40 (try 0 first) "; h
INPUT "Input U, 0 to 20     (try 20 first) "; u

k = 0.5
dt = 0.01  'Edit to try various values of steplength dt

Screen 12
WINDOW (-.5, 0)-(5.5, 40)
PSET(t, h)      'starting point
DO
   h = h + (-k * h + u) * dt  'This is the simulation
   t = t + dt
   LINE - (t, h)  'This joins the points with lines
LOOP UNTIL t > 5
```

Take care! This simulation will not be exact. The change in H over time dt will be accurate only if dt is very small. For longer timesteps, dH/dt will change during the interval and the simulated change in H will be in error.

Try values such as $dt = 1$ to see the error. See also that for small values, reducing dt makes no perceptible change.

6.3 SOLVING THE FIRST-ORDER EQUATION

At last we must consider the formal solution of the simple first-order example, where we assume that the system is linear. The treatment here may seem overelaborate, but later on we will apply the same methods to more demanding systems.

By using the variable x instead of H or T_{coffee} or such, we can put all these examples into the same form

$$\frac{dx}{dt} = ax + bu \tag{6.1}$$

where a and b are constants that describe the system. u is an input, which can simply be a constant such as T_{ambient} in the first example or else be a signal that we can vary as a control.

Rearranging, we see that

$$\frac{dx}{dt} - ax = bu$$

Since we have a mixture of x and dx/dt in this expression, we cannot simply integrate it. We must somehow find a function of x and of time that will fit in with both terms on the left of the equation.

If we multiply both sides by a mystery function $f(t)$, we get

$$\frac{dx}{dt} f(t) - axf(t) = -buf(t) \tag{6.2}$$

Now consider

$$\frac{d}{dt}(xf(t))$$

When we differentiate by parts, we see that

$$\frac{d}{dt}(xf(t)) = \frac{dx}{dt} f(t) + xf'(t)$$

where $f'(t)$ is the derivative of $f(t)$.

If we can choose $f(t)$ so that

$$f'(t) = -af(t)$$

then this will fit the lefthand side of Equation (6.2) to give

$$\frac{d}{dt}(xf(t)) = buf(t) \tag{6.3}$$

and we can simply integrate both sides to get the solution.

The function that satisfies

$$f'(t) = -af(t)$$

is

$$f(t) = e^{-at}$$

Now we can integrate both sides of Equation (6.3) to obtain

$$\left[xe^{-at}\right]_0^t = \int_0^t bue^{-at}dt$$

that is

$$x(t)e^{-at} - x(0) = \int_0^t bue^{-at}dt$$

so

$$x(t) = x(0)e^{at} + e^{at}\int_0^t bue^{-at}dt$$

Now, if a is positive, the first term will represent a value that will run off to infinity as time increases. If our system is to be stable, a has to be negative. So the coffee cup, the water barrel and the bicycle are stable, but the bank account is not.

If u remains constant throughout the interval 0 to t, we can simplify this still further:

$$x(t) = x(0)e^{at} + ub\left(e^{at} - 1\right)/a \tag{6.4}$$

We will come back to this equation when we look at sampled data control.

6.4 SECOND-ORDER PROBLEMS

I hope that you had no difficulty coming to grips with first-order systems, ones that had a single state variable. The following are second-order systems. You should be able to spot two state variables for each of them. You should also be able to write two differential equations for each example:

1. A mass hanging on a spring, bouncing vertically
2. A pendulum swinging in a plane
3. The distance between the back wheel of a bicycle and a straight line when the handlebar angle is varied (small movements away from the line)
4. The voltage on a capacitor when it, a resistor and an inductance are all connected in parallel
5. The position of a servomechanism where acceleration is the input

Once again, give it a try before reading on. Answers are as follows:

1. For the mass, bouncing vertically on a spring, there are two state variables. The first is the height x of the mass above the rest position, and the second is its upward velocity v. The first differential equation can be seen as subtle or obvious, depending on how you look at it

$$\frac{dx}{dt} = v$$

since the rate of change of position is simply the velocity. The second equation is not quite as easy. The rate of change of velocity is the acceleration. Now the acceleration is proportional to the deflection, the displacement away from the rest position, where the constant is the stiffness of the spring divided by the mass. The second equation is therefore

$$\frac{dv}{dt} = -\frac{Sx}{M}$$

If we add an input to the system, by allowing the top of the spring to be moved up or down a distance u, we have

$$\frac{dv}{dt} = \frac{S(u-x)}{M}$$

2. This is almost exactly the same as the previous example. This time the state variables can be taken as the angle of the pendulum and the angle's rate of change. Instead of the constant S/M, however, the constant for the

second equation is now g/L, the acceleration due to gravity divided by the length of the pendulum.

3. Unlike the previous bicycle example, this time both front and back wheels can move away from the straight line. We could take these two distances as the state variables. If we call them x for the rear wheel and w for the front wheel, then we see that the bicycle is pointing at an angle $(w - x)/L$ to the line. The rate of change of the rear-wheel distance will be this angle times the forward speed:

$$\frac{dx}{dt} = (w - x)\frac{V}{L}$$

The direction in which the front wheel is pointing will be $(w - x)/L + u$, where u is the handlebar angle, so

$$\frac{dw}{dt} = -(w - x)\frac{V}{L} + Vu$$

This looks rather different from the other examples. But the choice of state variables is not unique. Instead of the deflection of the front wheel, we could instead have taken the angle the bicycle is pointing as our second state variable. If we call this angle a, we have

$$\frac{dx}{dt} = Va$$

For the second equation, since $a = (w - x)/L$, we have

$$\frac{da}{dt} = \left(\frac{dw}{dt} - \frac{dx}{dt}\right)\Big/L$$

that is

$$\frac{da}{dt} = u\frac{V}{L}$$

4. The two things that cannot change instantaneously are the voltage v on the capacitor and the current i through the inductor. For the inductor, we have

$$L\frac{di}{dt} = v$$

and for the capacitor we have

$$C\frac{dv}{dt} = \text{current into the capacitor}$$
$$= -i - \frac{v}{R}$$

where R is the resistance. Rearrange these slightly, and you arrive at equations for di/dt and dv/dt.

5. This is almost too easy! State variables are now position x and velocity v, so we have

$$\frac{dx}{dt} = v$$

and

$$\frac{dv}{dt} = bu$$

where u is the drive applied to the servomotor and b is a constant.

6.5 MODELING POSITION CONTROL

A servomotor drives a robot axis to position x. The speed of the axis is v. The acceleration is proportional to the drive current u; for now there is no damping.

Can we model the system to deduce its performance?

We have just found equations for the rate of change of x and v:

$$\frac{dx}{dt} = v$$
$$\frac{dv}{dt} = bu$$

We can carry their values forward over an interval dt by adding dt times these rates of change to their values, just as we did for the water barrel.

With a fixed input, the response will not be very interesting. The real use for such a simulation will be to try out various values of feedback. We can start with a position error of 1, say, and ask the user to input values for f, the position feedback, and d, the damping or velocity feedback.

Application of feedback means, "Giving u a value depending on the state." So, before we update the variables, we must make

```
u = -f * x - d * v
```

The following code will perform the simulation; if we define input u in terms of acceleration, rather than motor drive, we can make $b = 1$:

```
SCREEN 12
WINDOW (-.05, -1.1)-(2.05, 1.1)
INPUT "Feedback, damping (suggest 2,2 to start) "; f, d
LINE (0, 0)-(2, 0), 9   'Axis, blue
dt = .0001     'Make dt smaller to slow display
x = 1        'Initial values
v = 0
t = 0
PSET (0, x)     'move to the first point
DO
   u = - f * x - d * v   'u is determined by feedback
   x = x + v * dt        'This is the simulation
   v = v + u * dt
   t = t + dt

   LINE -(t, x)          'This displays the result
LOOP UNTIL t > 2
```

Start with *f*, *d* values of 2, 2.
Next try 10, 5. What do you notice?
How about 1000, 50?
Now try 10000, 200.

It seems that we can speed up the response indefinitely by giving bigger and bigger values of feedback. Is control really so simple?

Of course, the answer is "No." Our simulation has assumed that the system is linear, that doubling the input doubles everything else. But if we have a real motor, there is a limit at which it can accelerate. Let us suppose that the units in our simulation are meters and that it runs for 2 s.

Let us also make the maximum acceleration 10 meters per second. After the u = line, insert two lines

```
IF u > 10 THEN u = 10
IF u < -10 THEN u = -10
```

to impose a limit on u. Now run the program and try all the pairs of values again.

Values 2,2 and 10,5 give the same sluggish responses as before. But 1000,50 overshoots wildly, and 10000,200 is even worse. The limit on the motor drive has had a dramatic effect on performance.

But now try 10000, 1600. There is no overshoot and the response has settled in considerably less than one second. It seems that if we know how, we can design good controllers for nonlinear systems. There will be more on that in Chapter 10.

You can borrow a couple of lines from the code in Chapter 3 to construct a real-time "target," changing as you tap the keys, then feed back `(target - x)` in the equation for `u`.

Now you can look at some examples on the book's Website, at http://www.essmech.com/6/5/, to see similar simulations written in JavaScript. Do not just run them; experiment with them and modify their code.

6.6 MATRIX STATE EQUATIONS

We have succeeded in finding a way to simulate the second-order problem, and there seems no reason why the same approach should not work for third, fourth, and more. How can the approach be formalized?

First we must find a set of variables that describe the present state of the system—in this case x and v.

They must all have derivatives that can be expressed as combinations of just the state variables themselves and the inputs, together with constant parameters that are properties of the system.

We have a set of equations that express the change of each variable from instant to instant. If there should happen to be some unknown term, then we have clearly left out one of the state variables; we must hunt for its derivative to work it in as an extra equation.

In the present position control example, the equations can be laid out as follows:

$$\frac{dx}{dt} = v$$

$$\frac{dv}{dt} = bu$$

As soon as a mathematician sees a pair of equations, there is an irresistible urge to put them in the form of a single matrix equation

$$\frac{d}{dt}\begin{bmatrix} x \\ v \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ b \end{bmatrix} u$$

and then to push the shorthand even further. The vector, for that is what x and v have become, is represented by a single symbol $\mathbf{x}$. The 2×2 matrix is given the symbol A, and the matrix that multiplies u is given the symbol B. Just in case u might "fatten up" and have two components, it is also made a vector, $\mathbf{u}$. So we have

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

Well, it does look a lot neater. There is one more change. There is a convention to represent the time derivative by a dot over the variable, so we end up with

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} \qquad (6.5)$$

which looks very much like the form we used for the first-order systems.

But this still describes the open-loop system, the one that we would like to change with some feedback. How can we deal with feedback in matrix terms? The secret is in the `u =` line. The mixture of variables that we feed back can be expressed in matrix terms as

$$\mathbf{u} = F\mathbf{x} + G\mathbf{d}$$

where **d** is some external demand such as the target value for position. Now we can substitute into the state equation to get

$$\dot{\mathbf{x}} = A\mathbf{x} + B(F\mathbf{x} + G\mathbf{d})$$

which simplifies to

$$\dot{\mathbf{x}} = (A + BF)\mathbf{x} + BG\mathbf{d}$$

Apart from A having become $A + BF$ and B having become BG, this has exactly the same form as Equation (6.5). The effect of our feedback has been to change the A and B matrices to values that we like better. But how can we decide what we will like?

You will have to wait until Chapter 8, when you will become skilled in the art of eigenvalues.

6.7 ANALOG SIMULATION

It should be obvious by now that our state equations tell us which inputs to apply to integrators that will have outputs corresponding to the state variables' values.

It is ironic that analog simulation "went out of fashion" just as the solid-state operational amplifier was perfected. Previously the integrators had involved a variety of mechanical, hydraulic, and pneumatic contraptions, followed by an assortment of electronics based on magnetic amplifiers or thermionic valves. Valve amplifiers were common even in the late 1960s, and required elaborate stabilization to overcome their drift. Power consumption was high and air-conditioning essential.

Soon an operational amplifier was available on a single chip, then four to a chip at a price of a few cents. But by then it was deemed easier and more accurate to simulate a system on a digital computer. The costly part of analog computing had become that of achieving tolerances of 0.01% for resistors and capacitors, and of constructing and maintaining the large precision patchboards on which each problem was set up.

In the laboratory, the analog computer still has its uses. Leave out the patchboard, and solder up a simple problem directly. Forget the 0.1% components—the parameters of the system being modeled are probably not known to better than a percent or two, anyway. Add a potentiometer or two to set up feedback gains, and a lot of valuable experience can be acquired. Take the problem of the previous section, for example.

We have seen in Chapter 5 that an analog integrator can be made from an operational amplifier, but that the signal integrates in the negative sense when a positive signal is applied.

To produce an output that will change in the positive sense, we must follow this integrator with an inverter. That will mean, too, that with both signs of the signal available, we can attach a potentiometer between them to try both negative and positive feedback.

The circuit shown in Figure 6.1 can give some sort of simulation of the position control problem, although as it stands, the range of gains you can try will be very limited.

One feature it does represent is limits. The amplifiers cannot give voltages outside their supply rails of +12 and −12V. You will see that the feedback signals have been mixed in an inverter, connected to the first integrator with a 10kΩ resistor. This gives an effective gain of 10, and the effect on the amplifier limit is equivalent to saying that the motor is capable of accelerating at a rate of 10m/s. This gain of 10 will apply to the feedback coefficients, but they will still be much smaller than the values you used in the "professional" position control experiment.

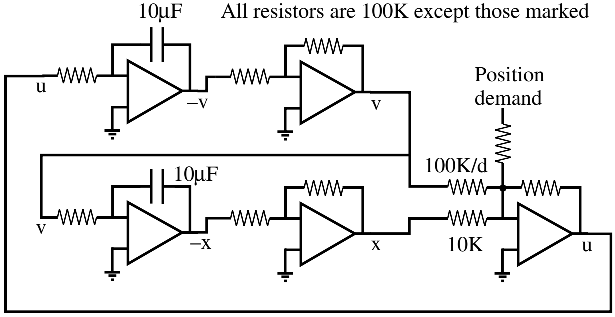

Figure 6.1 *Simulation circuit, gain of 10 from mixer.*

6.8 MORE FORMAL COMPUTER SIMULATION

The simulations we have seen so far are "run up on the spot" as simply and concisely as possible. For more general use, however, we need a more formal methodology.

Software can be written in a host of languages, including QBasic, Visual Basic, JavaScript, or even a package such as Matlab, but the simulation will have a common structure:

1. Define constants and variables.
2. Set variables to their initial conditions, and define the timestep.
3. Begin the loop.
4. Calculate the drive input(s)
5. Calculate the rates of change of the state variables, using the state equations.
6. Update the state variables, by adding rate of change **dt*.
7. Update the current time, by adding *dt* to it.
8. Plot the variables, or capture them for plotting later
9. Repeat the loop until the end of the simulation.

The state equations do not have to be linear. They can include limits or geometric functions as necessary, depending on the detail that we are trying to achieve. We can simulate continuous control, with timesteps that are small in order to preserve accuracy. We can simulate discrete-time control where the drive is allowed to change only at intervals of many steps of the continuous system's update.

The essential requirement is that the state equations used must be an accurate representation of the system.