Delays in Control Systems

Anyone who has ever tried to stay comfortable while showering in a crowded building with old plumbing understands how delays in a system can make the control problem much more difficult. It’s not that the delays can’t be incorporated into the system model; that’s easy in sampled-data systems and certainly possible in continuous systems. If you are able design a controller that stabilizes a system containing significant delays, it will likely result in a disappointingly slow response.

In general, you will find that the effects of delays in closed-loop feedback systems resemble the effects of lowering the sampling frequency. Physically, this makes sense: in both cases, the controller is forced to make use of “old” information (information about the output at some time in the past, rather than in the present) in determining the output it supplies to the plant.

Modeling Delays

1. Continuous systems

The Laplace transform for a pure delay is just

$$f(t - \tau) \Leftrightarrow e^{-s\tau} F(s).$$

where \(\tau\) is the delay time in seconds. Thus, it’s easy to derive transfer functions for systems containing delays. For example, a system with a cascade controller and unity feedback, but using an output sensor that is \(\tau\) seconds late in reporting the output (a sensor delay) would have a transfer function calculated from a diagram like this:

The transfer function would have the form,

$$H(s) = \frac{C(s)G(s)}{1 + C(s)G(s)\exp[-s\tau]}$$

If the delay occurred in transmitting the output of the controller, \(C(s)\), to the plant (an actuator delay), the block diagram and the transfer function would look like this:
with $H(s)$ given by

$$H(s) = \exp[-s\tau]\frac{C(s)G(s)}{1 + C(s)G(s)\exp[-s\tau]}.$$ 

Note that the second transfer function is just a delayed version of the first. Specifically, in the frequency domain, the gain vs. frequency curves will be the same for both.

The biggest obstacle to handling delay in design is not in the modeling, which you can see is quite simple: the problem lies in the fact that the delay cannot be expressed as a rational polynomial. There are some approximations that have rational polynomial forms:

\[
\exp\{-s\tau\} \approx 1 - s\tau + \frac{(s\tau)^2}{2} - \frac{(s\tau)^3}{6} \ldots, \text{ or}
\]

\[
\exp\{-s\tau\} \approx \frac{1}{1 + s\tau + \frac{(s\tau)^2}{2} + \frac{(s\tau)^3}{6} \ldots}, \text{ or}
\]

\[
\exp\{-s\tau\} \approx \frac{1 - s\tau/2}{1 + s\tau/2}, \text{ or}
\]

\[
\exp\{-s\tau\} \approx \frac{1}{\left[1 + \frac{s\tau}{n}\right]^n}.
\]

The last of these can be made as accurate as needed by choosing $n$ large enough.

A little experimentation with your design software will illustrate the strengths and weaknesses of these approximations. To be “valid” for a particular design, the approximation you use should accurately track (in both magnitude and phase) the Bode plot of $\exp\{j\omega\tau\}$ over the range of frequencies corresponding to the bandwidth of the designed system.

2. Sampled-data systems

Suppose we consider the sampled-data system with actuator delay below:

![Sampled-data system diagram]

It has a $z$-domain transfer function,

$$H(z) = \frac{Q(z)}{1 + Q(z)}, \text{ where}$$
The importance of the above result is that even with delay, the rational polynomial nature of the transfer function is preserved in the z-domain. This means we can use sampled-data design techniques on continuous systems with delays, provided we set the sampling frequency, $f_s$, high enough to force the sampled-data system to approach the continuous one.

A more general sampled-data system with both actuator and sensor delays is shown below.

\[
Q(z) = z^{-1} \left\{ \frac{C(s)G(s) \exp[-s\tau]}{s} \right\} \approx z^{-N} \left\{ \frac{C(s)G(s)}{s} \right\}, \text{ and}
\]

$N = f_s\tau$ is the (necessarily integer) number of sample periods in the delay, $\tau$.

In this system, $C(s)$ and its digital equivalent, $D(z)$, are the only parts of the system subject to alteration by the designer. When working in the z-domain, we can only express delays in terms of an integral number of sample periods, i.e.:

- $N_A = f_s\tau_A$ is the (integer) number of sample periods in the actuator delay time, $\tau_A$, and
- $N_S = f_s\tau_S$, is the (integer) number of sample periods in the sensor delay time, $\tau_S$.

Applying the general procedure for finding the transfer function of closed-loop systems yields the z-domain transfer function as

\[
H(z) = \frac{O(z)}{R(z)} = \frac{z^{-N_A} \left( \frac{z-1}{z} \right) \left\{ \frac{C(s)G(s)}{s} \right\}}{1 + z^{-(N_A+N_S)} \left( \frac{z-1}{z} \right) \left\{ \frac{C(s)G(s)}{s} \right\}} , \text{ where}
\]
System design with delays

1. Continuous systems

You can incorporate any rational polynomial approximation for a delay into the transfer functions for the plant or the feedback loop, and then proceed with the design process as you would have done in the absence of any delays. When you are finished with the design, you should check the approximation for $G(s)$ you used in the frequency domain to see that its Bode plot closely follows one for $G(s) \exp\{-j\omega \tau\}$ over all frequencies within the bandwidth of the design.

Alternatively, and perhaps more safely, you can design your continuous system in the $z$-domain, where delays can be modeled exactly, provided you use a high sampling frequency. In the limit of very large sampling frequency, the sampled-data system will perform like the continuous one.

If you use the $z$-domain design approach, you can start your design process neglecting delays and introduce them gradually in order to see how delay degrades the performance of the closed-loop system.

2. Sampled-data design

In addition to being useful in the design of continuous systems containing delays, as discussed above, sampled-data design techniques are of course necessary for systems where the controller will be implemented digitally. Because the $z$-domain parameters of controllers vary as sampling frequency is changed, it will be more convenient to design the controller in the $s$-domain and then transform it to the equivalent $z$-domain transfer function, $D(z)$. 