How to control processes with large dead times

This article appeared in Control Engineering

Processes with large dead times present a special challenge for a controller—any controller. The controller must wait until the dead time has passed before it gets any feedback from the process.

The best thing to do for controlling this type of process is to try to reduce the dead time. Simply moving a probe closer to the valve may do it. But sometimes there is nothing that can be done to the process. What then?

Pick your controller

The control algorithm for a dead time process is actually very simple. A PI controller does a fine job.

You can also use a fancy model predictive, or model based, or pole-cancellation, or Internal Model Control, or Dynamic Matrix controller or Dahlin or Direct Synthesis Controller. They all amount to the same thing of using a model inside the controller. Lets just call them all Model based controllers. Model based controllers can eek out slightly better response than PI - but at the sacrifice of robustness.

The robustness penalty using model based control (made to be even the same speed as a PI) is severe—the dead time of the process can go up or down and the loop will go unstable. Increasing dead time with any process will always eventually drive you unstable. But decreasing dead time to make you unstable is weirdly counter-intuitive. If the dead time were to go down with a PI controller, the loop would remain stable. So why not keep it simple and just use a PI controller?

Tune your controller

The optimal tuning of a PI controller for a dead time only process is so simple you can hardly call it tuning:

controller gain = .3/(process gain)
integral time = .42 * (process dead time)

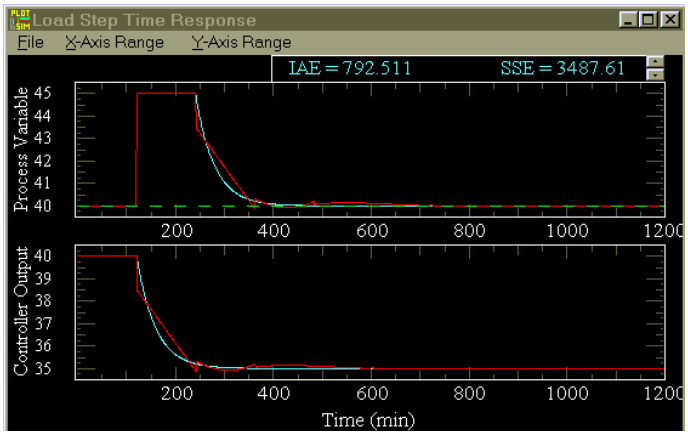
The integral units are in time—either minutes or seconds. This optimal tuning gives minimum error to a step load upset. If you want slower response, simply lower the gain a little. (These equations apply for usual ideal and series PI algorithms. For the parallel you've got to divide the integral time by the controller gain.)

Dead time processes with an additional lag

If the process has a small lag in addition to the dead time, you can "cancel" the lag by setting the derivative time equal to it. This works as long as the lag is about 5 times less than the dead time.

.

Process example—Dead time of 2 hours



Disturbance time plot comparison, red is PI, blue is model predictive

Here is a process with a gain of 1 and dead time of 120 minutes. In the red line, it is controlled with a PI controller having an integral time of 60 minutes, and a gain of 0.3. It shows that 240 minutes or 2 dead times pass before the process reaches setpoint or recovers from an upset.

Adaptive dead time controller

You can make a powerful adaptive dead time controller if you can measure the dead time. Or infer the dead time from a variable like the speed of a machine or flow.

To make your controller adaptive, simply adjust the Integral action in the controller using the formula above.

The key is to be able to measure or infer the dead time.

This kind of adaptive controller is much more robust than using some kind of identification technique that depends on normal process noise. Attempting to identify the dead time using a least squares fit or some other modeling method that looks at normal process noise is subject to gross modeling errors with disturbances. Identification techniques relying on setpoint changes, however, are accurate.

Comparison to model predictive

In a model predictive controller you set a desired speed of response. Its the first order response time to setpoint that you would like. The goal of the model predictive controller is to match the setpoint response to a first order time constant (or lag time) you enter. Response is first delayed by the process dead time. The blue line is the response using a model predictive controller with a speed of response adjusted to be the similar robustness as the PI.

Robustness—the other side of the story

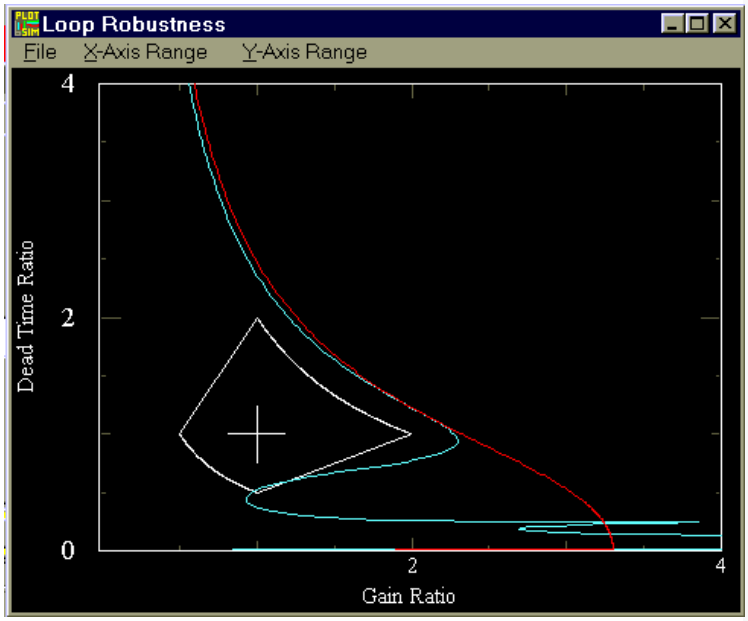
Robustness plot, red is PI, blue is model predictive.
There is always a trade off between fast response and robustness or sensitivity to the process changing. Robustness plots easily show this tradeoff. They show how sensitive (or robust) your loop is to process gain or process dead time changes. The two axis of the plot are:

gain ratio = (process gain)/(process gain the controller was tuned for)

delay ratio = (process dead time)/(process dead time the controller was tuned for)

The plot has a region of stability and a region of instability. The red and blue lines on the robustness plot are the limit of stability for PI and Model Predictive controllers. To the right and above the solid lines (higher gain and delay ratios) the closed loop process is unstable. To the left and below the solid lines, the closed loop system is stable. At the cross, where both ratios are 1, the process gain and dead time are at the process values you tuned for.

Generally, a safety factor or divisor of 2 is “reasonable” for a loop. These points are represented by the vertices of the blue line “box” in the robustness plot. It as a design aid. For practical system stability you want the limit of stability line outside the “box”. The vertices are connected by lines that are straight on a log log plot.



For example, for either PI or model predictive controller, if the process gain were to increase by a factor of 2.3, the loop would be unstable. Start at the white cross and move horizontally to the right. Where you intersect the stability lines the gain ratio is about 2.3.

Another example that works for either controller is if the dead time were to increase by 2.4. Let your eye start at the cross and move straight up until you smash into the stability lines. Here, the dead time ratio is about 2.4. At this point the loop would again be at the verge of stability.

Counter-intuitive robustness of the model predictive

A third example that applies to the Model Predictive Controller only is if the process dead time were to decrease. Again start at the cross and move straight down until you smash into the blue line. At this point the dead time ratio is about 0.5. The loop with the model predictive controller goes unstable with a DECREASE in dead time. Intuitively, like all controllers the Model Predictive gets unstable with a large enough increase in the process dead time. But, counter to your intuition and the behavior of a PI, it also gets unstable when the process dead time goes down.

Conclusions

A PI controller is simple, familiar and does a great job controlling processes with large dead times. Simply set the gain and integral time as a factor of the process gain and dead time. Although a model predictive controller may perform slightly better, it suffers from poor robustness and would require extensive training for operations personnel.