FEATURE SERIES: PRACTICAL PROCESS CONTROL

6: Derivative Action

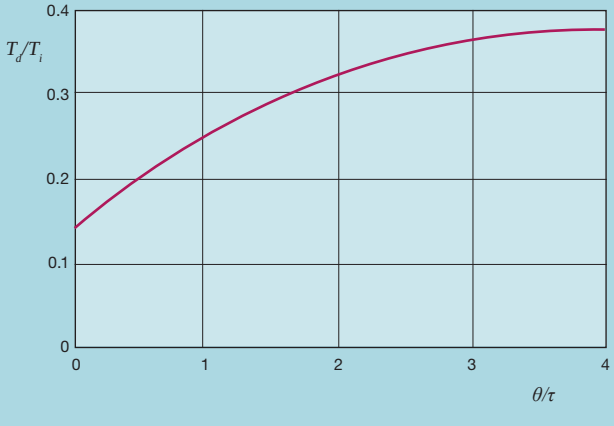
Myke King continues his detailed series on process control, seeking to inspire chemical engineers to exploit untapped opportunities for improvement

THE majority of controllers in use in the process industry use the PI rather than the PID algorithm. There are several reasons for this. The first is probably that the addition of derivative action to a controller is undervalued by the engineer. Certainly, if simply added to a working PI controller, its advantage will not be immediately obvious. This is not helped by some of the published tuning methods. For example, the Internal Model Control (IMC) method we covered in Issue 983, will give the same response to a setpoint (SP) change for both the PI and the PID algorithm. Other methods, like Cohen-Coon, suggest that derivative action is of no benefit if the deadtime (θ) is small.

To see the benefit of adding derivative action to a working PI controller, the existing tuning constants must be redetermined. If trial-and-error is the tuning method of choice, this is impractical. When performed manually, a three-dimensional search is considerably more difficult than a two-dimensional one.

The other main reason is that derivative action is known to amplify any measurement noise, transmitting it to the control valve and potentially causing its failure. This article aims to address both issues.

Figure 1: Need for derivative action



USE THE IDEAL PID ALGORITHM

Figure 1 shows the requirement for derivative action, particularly as the deadtime-to-lag ratio (θ/τ) increases. Presented as a fraction of the integral time, it influences our choice of control algorithm. We previously presented (*Issue 983*) the formulae for converting tuning for the interactive PID algorithm to that for the ideal version. Inverting these formulae gives

$$K_c = \frac{(K_c)_{ideal}}{2}\left[1+\sqrt{1-4\left(\frac{T_d}{T_i}\right)_{ideal}}\right]$$

$$T_i = \frac{(T_i)_{ideal}}{2}\left[1+\sqrt{1-4\left(\frac{T_d}{T_i}\right)_{ideal}}\right]$$

$$T_d = \frac{2T_d}{\left[1+\sqrt{1-4\left(\frac{T_d}{T_i}\right)_{ideal}}\right]}$$

Unlike the conversion in the former direction, there is now the condition that the ratio T_d/T_i in the ideal version cannot exceed 0.25. Figure 1 shows that the required ratio substantially exceeds this value – particularly when θ exceeds τ. Under these circumstances an optimally tuned ideal PID cannot be replaced with an equivalent interactive PID. It gives us a reason to choose the ideal version as our standard.

To see the benefit of adding derivative action to a working PI controller, the existing tuning constants must be redetermined. If trial-and-error is the tuning method of choice, this is impractical

Figure 2: Benefit of derivative control

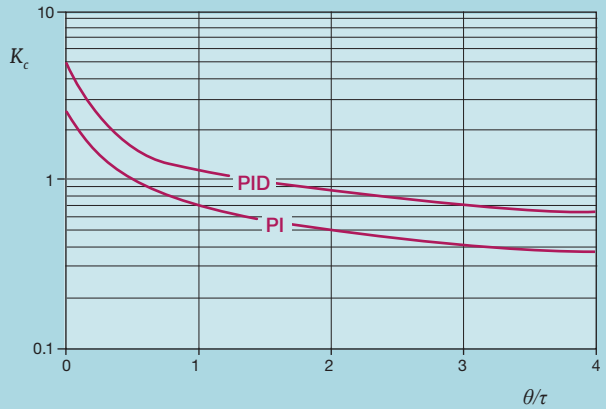
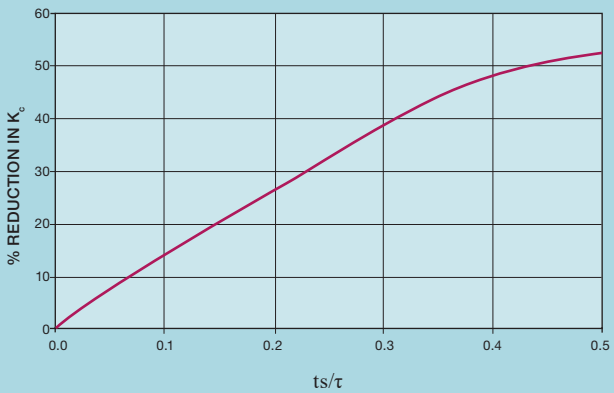


Figure 3: Effect of scan interval on controller gain



INCREASE CONTROLLER GAIN

Figure 2 shows the impact that the addition of derivative action has on controller gain. (Note the logarithmic scale.) It typically allows K_c to be increased by about 80%. This is consistent across the range of process dynamics. So why we might think its predictive nature is of little value when the deadtime is small, it still permits us to substantially increase controller gain. In the same way that choosing the I-PD algorithm permits such an increase, it will substantially reduce the impact of process disturbances.

NOISE PROBLEM

The reluctance to apply derivative action is that it is known to amplify noise. Let's imagine that the noise in our process value (*PV*) can be described as a sine wave with frequency *f*.

$$PV = Asin2\pi ft$$

Derivative action is based on the derivative of *PV*

$$\frac{dPV}{dt} = A.2\pi f cos2\pi ft$$

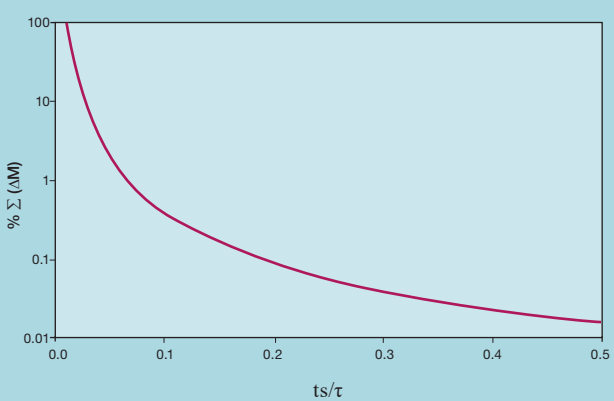
Differentiation increases the amplitude of the noise by a factor of $2\pi f$. The higher the frequency, the greater the amplification. It would not be unusual for noise, say of amplitude 1% of instrument range, to be amplified to the point where the control valve is moved rapidly over its whole range.

INCREASE THE SCAN INTERVAL

Let us consider the effect of the controller scan interval *(ts)*. If we double this, say from 1 to 2 seconds, the amplitude of the noise remains the same but its frequency (as seen by the controller) is halved. This then halves the amplitude produced by derivative action. One of the myths of process control is that the scan interval needs to be very short. Making it so can be counter-productive, requiring filtering (which we'll show in another article, increases the process lag) and restricting the use of derivative action.

Figure 3 shows the impact of scan interval on controller gain, clearly applying a tuning method that takes account of *ts*. We're not suggesting that it ever be increased so much, partly because of the large reduction in K_c. We're merely demonstrating that it's possible. The limit is not controllability but how long we are prepared to wait for the controller to scan and so respond to a process disturbance or a SP change. Figure 4 demonstrates the benefit of even small increases. For example, on a process which has a lag of one minute, increasing the scan interval from 1 second to 5 seconds reduces by 97% the effect that noise and derivative action has on total valve travel. This would require

Figure 4: Reduction in valve travel



a reduction in controller gain of only 2%. Further, if the DCS is operating close to its CPU capacity, increasing scan interval by a few seconds can significantly reduce the system load.

APPLY FILTERING

One of the reasons that process control has the reputation of being highly mathematical is the use of Laplace Transforms. We have so far avoided their use and, where practical, will continue to do so. But they can be an effective means of representing a process or a controller. Indeed, the control algorithms we'll be covering here would occupy half a page if they were described in discrete form. So we promise only to use Laplace notation where the alternative uses far worse mathematics. Further we'll not expect the reader to be able to derive the transforms or manipulate them in any way.

So, for example, the Laplace notation for our first order process is

$$PV = \frac{K_p e^{-\theta s}}{1+\tau s} MV$$

We can see that $e^{-\theta s}$ describes deadtime and $1/(1+\tau s)$ represents lag. Similarly our ideal PID algorithm can be described as

$$M = K_c \left[1 + \frac{1}{T_i s} + T_d s\right] E$$

The interactive PID can be factorised

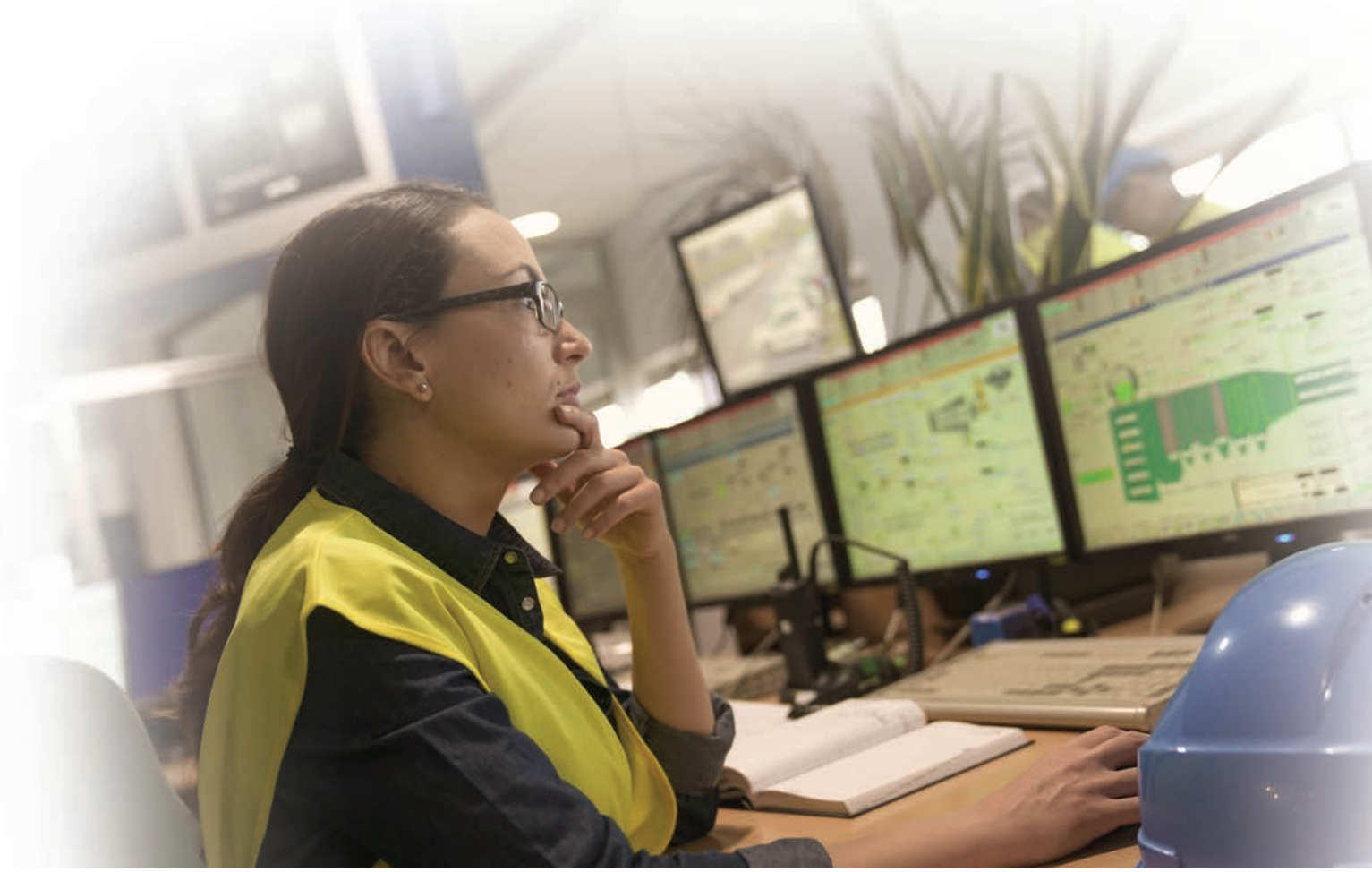
$$M = K_p \left[1 + \frac{T_d}{T_i} + \frac{1}{T_i s} + T_d s\right] E = K_p \left[1 + \frac{1}{T_i s}\right] [1 + T_d s] E$$

So why is this relevant to derivative action?

Control system vendors are well aware of the problem of noise and modify the PID algorithm to lessen its impact on derivative action. A common change to the above interactive algorithm is to include a lag term, in this case $1/(1 + aT_d s)$.

$$M = \left[1 + \frac{1}{T_i s}\right] \left[\frac{1 + T_d s}{1 + aT_d s}\right] E$$

In a future article on filtering, we'll see that the standard filter in most control systems is a lag. Usually applied to the measurement as part of signal conditioning, it can also be included



in the PID algorithm. In the example above, we can see that setting a to 0 removes the filter. Setting it to 1 removes the derivative action. In most control systems a is fixed (usually at 0.1). Some, like Foxboro, use its reciprocal, known as the *derivative gain limit*. Given the notation *KD* it has a default value of 10 but can be modified by the engineer over the range 0.1 to 100. ABB replaces aT_d with the parameter T_{fil} which is similarly adjustable by the engineer.

While traditionally applied to the interactive PID, Honeywell have, in their latest DCS, also modified the ideal version.

$$M = K_c\left[1 + \frac{1}{T_i s} + \frac{T_d s}{1 + aT_d s}\right]E$$

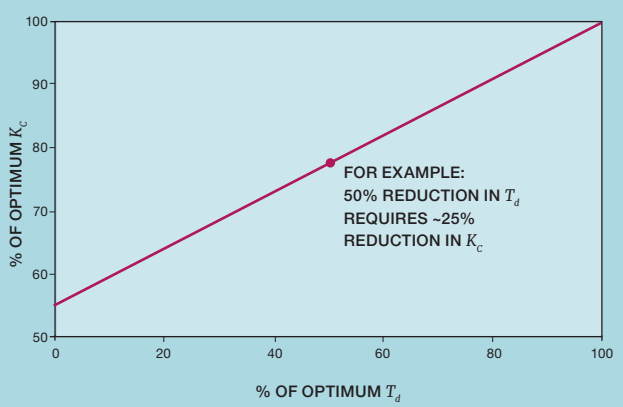
While setting a to 0, as in the interactive PID, removes the filter, it has to be set to a value much greater than 1 to completely eliminate derivative action. Honeywell, however, does not allow it be changed – fixing it at 0.0625.

These changes to the algorithm, although beneficial to noise reduction, must be taken into account when tuning. Another reason why the formulaic approach would fail, it emphasises the need for tuning by trial-and-error using computer simulation. However, the noise reduction achieved by these modifications is modest and may still preclude the use of derivative action. We'll describe, in a future article, a range of additional filtering techniques – one of which is particularly suitable as a means of permitting the use of derivative action.

COMPROMISE

If, after considering all of the above, the inclusion of derivative action remains a problem there may be a compromise. Rather than eliminate it completely, it may be feasible to reduce it. The downloadable tuning software (*Issue 983*) permits constraints to be placed on tuning constants. Doing so on T_d means that K_c will, as expected, need to be significantly reduced *(see Figure 5)*. For example, choosing half the ideal value for T_d will require that K_c be reduced by about 25%. A small reduction in T_i, of about 10%, will also be required.

Figure 5: Compromising on derivative action



If the inclusion of derivative action remains a problem there may be a compromise...

TAKE CARE

- The interactive and ideal versions of the PID controller are identical if T_d is set to zero. Remember the proportional action will be different when derivative action is added.
- If derivative action is based on error then its addition will cause derivative spikes whenever the set-point is changed (*Issue 982*). Despite the claim made by some control system vendors, the inclusion of a does not solve this problem. If the controller is the secondary of a cascade, then SP changes will be frequent and the spikes can easily be mistaken for noise. Make sure that the algorithm in use has derivative action based on PV.
- Retuning regulatory controls to exploit the addition of derivative action will change the overall process dynamics. So, if we need to add one, will a filter. As always, if there is a multivariable predictive controller (MPC) in place, its performance will degrade – potentially becoming unstable. ■

NEXT ISSUE

The next article will first address the control of liquid level in a vessel. Rather than a fast return to SP we often want to take advantage of the surge capacity. We'll show when and how such control should be implemented. We'll also cover the non-linear algorithms common to most DCS, showing their advantages and how to tune them. Finally we'll show how any integrating process can be modelled, using the dynamics to then tune the controller.

Myke King CEng FIChemE is director of Whitehouse Consulting, an independent advisor covering all aspects of process control. The topics featured in this series are covered in greater detail in his book Process Control – A Practical Approach, published by Wiley in 2016.