Signal filtering: Why and how

Prevent over-filtering by simultaneously optimizing loop tuning and filter parameters.

By Mark Darby and Greg McMillan

Filtering is the modification of a measured or calculated signal—using an algorithm and/or logic—to remove undesirable aspects of the signal before it is used in a calculation or a controller. Examples in control are the feedback (or controlled) variable to a PID or APC controller, or the input to a feedforward controller. Calculated signal examples include computations based on steady-state material/energy balances or process-oriented relationships, as well as process and control metrics.

The main reason to filter a signal is to reduce and smooth out high-frequency noise associated with a measurement such as flow, pressure, level, pH or temperature. A common example is the noise associated with the differential pressure (DP) across an orifice plate used to infer flow rate. High-frequency noise is normally considered to be random and additive to a measured signal, and is usually uncorrelated in time; i.e., the value of the noise at any time t does not depend on previous values of the noise. Ideally, we want to estimate the underlying signal without noise, introducing as little distortion as possible.

When a noisy signal is used in control, filtering is important for effective derivative action and for avoiding excessive movement in the controller output that causes valve wear or disturbs other control loops. A complicating factor for the control case is that both filtering and controller tuning can be used to reduce movement of the controller output.

The downside of filtering is the lag introduced, especially with heavier filtering, which can have a detrimental effect on timely detection of changes in the underlying signal. When used for feedback, the filtered value can result in control that is sluggish or, in the worst case, becomes oscillatory or even unstable. In the authors' experience, the more typical problem encountered is excessive filtering of a signal, as opposed to under-filtering.

The basic idea behind filtering is illustrated in Figure 1, where an underlying unknown signal s and the noise-contaminated measured signal s_n are shown. The goal of filtering is to estimate the underlying signal s with as little distortion as possible. Two different filtered signals are shown, y_{F1}andy_{F2}, where y_{F2} is more heavily filtered (smoothed), but has more lag than y_{F1}.

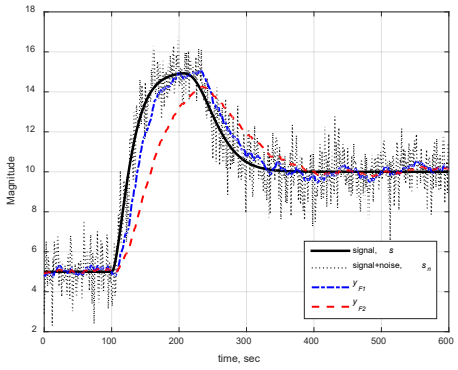

Figure 1. Noise-contaminated signal and filtered values y_{F1} and y_{F2}; y_{F2} is more heavily filtered than y_{F1}.

The source of the noise may be electronic or from the process itself. "Process noise" may originate from mixing, flashing, condensation/vaporization or from improper installation of a measurement device, for example, not enough straight-run pipe length upstream or downstream of a flowmeter. Unfortunately, in many cases, filtering is used in an attempt to mask the effect of an unmeasured disturbance or a problem such as valve resolution or dead band in the control loop. In many cases, the filtering makes problems worse. A filter will often be set and then forgotten until the loop is reexamined due to loop troubleshooting of poor performance, or when a control project is executed (e.g., APC).

The literature contains a wide range of approaches to signal filtering, which is a subcategory under the general topic of signal processing. Broadly speaking, filters can be grouped according to the following types:

- Linear filters (constant coefficients), which can be expressed as a linear difference equation and analyzed as a transfer function;
- Nonlinear filters, which depend on the signal value itself or the amplitude of the noise level. Filters based on logic such as the signal rate of change also fall into this category; or
- Model-based schemes that include an underlying process model, and may also include statistics of the noise itself.

Filters used in process control applications are usually linear filters, which will be our focus. The linear filter is a linear combination of current and previous values of the signal and filtered values. The general filter equation is given by

$$y_F(k) = a_1 y_F(k-1) + \cdots + a_n y_F(k-n) + b_0 y(k) + b_1 y(k-1) + \cdots + b_n y(k-n)$$

where $y(k)$ is the signal value and $y_F(k)$ is the filtered value at time step k, and a_n and $b_n$are the filter parameters (constants) that must be specified. The time difference or separation between subsequent values (e.g., between k and k-1) is denoted as Δt and is, in effect, another parameter to specify. If only measured values appear in the filter equation (i.e., no previous values of y_F), the filter is known as a finite impulse response (FIR) filter:

$$y_F(k) = \sum_{i=0}^{N-1} b_i \, y(k-i)$$

where N is the number of measurement values used. The moving average filter is an example of an FIR filter. The general form of a filter, with both and y_F and y terms, is also known as an infinite impulse response (IIR) filter, since it can be represented as an infinite series of only measured values. The above equations are similar to a discrete process model. The difference is that a filter includes the current $y(k)$ as an input; a change in $y(k)$ therefore has an immediate effect on $y_F(k)$, whereas a process model will usually have at least one Δt of delay, and more if dead time is present. The other notable aspect of a filter is that it has a gain of one, which implies that for a fixed value into the filter, the filtered value y_F will ultimately go to the same value.

Before beginning our discussion of filters, it is useful to look at how a control system produces sampled values of a continuous measurement and where filter(s) can be applied. Figure 1 shows the continuous or analog measurement coming into the analog input (AI) card, where the analog signal is first filtered (the anti-alias filter) before the analog to digital (A/D) converter to remove higher frequencies that would otherwise lead to fictitious frequencies in the sampled values. The A/D converter converts the analog signal to a digital value at a specified scan interval Δt. We assume Δt has been appropriately specified for the given signal. The conventional rule of thumb is that the dead time from discontinuous updates should be less than 10% of the total loop dead time (wireless update rates and PID execution rates less than 20% of dead time, assuming zero latency). This is only really true if you are pursing aggressive control where the implied dead time is near the actual dead time. According to Shinskey's equation for the integrated error for a step load disturbance in terms of PID tuning settings, this corresponds to an execution time that is less than about 5% of the integral time.

Many control systems provide the option to digitally filter the sampled value (embedded filter) signal before subsequent use in such tasks as alarming, control or calculation, but may not retain the raw sampled value. This can be problematic (think of the problem caused by delaying an alarm). If filtering is required for other applications, such as PID control or in APC, another tag and filter should be configured. The other thing to remember is to not introduce excessive lag by applying the filter at a significantly slower Δt than the scan Δt of the A/D converter. For example, a common mistake is to apply a filter to the value coming through the gateway to an APC application like model predictive control (MPC) that may execute at a slower interval, say, once per minute. Again, this is circumvented by creating additional tags for any required filtered values. This also provides the opportunity to apply a more advanced filter.

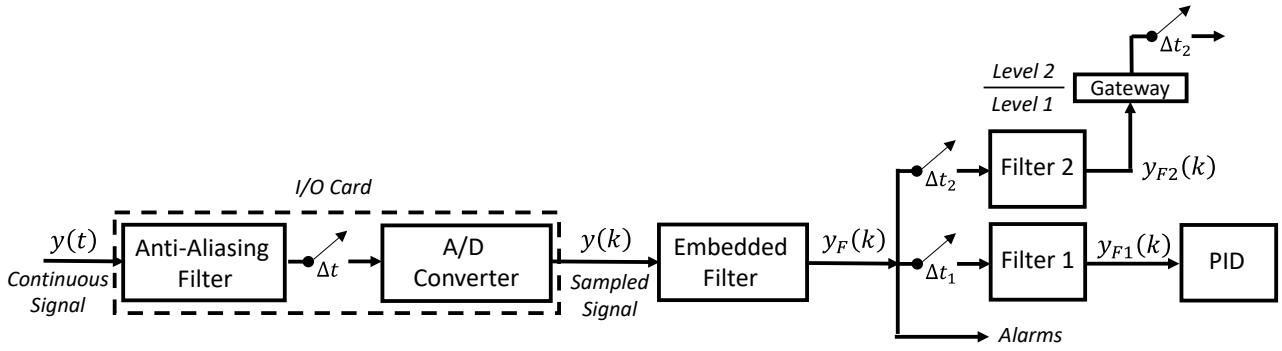

Figure 2. Filtering can be applied at several places and at different Δt values in a controlled system. Filter 1 and Filter 2 illustrate that different filters may be required depending on the application.

The most commonly used filter is the exponential, or first-order filter:

$$y_F(k) = a y_F(k-1) + (1-a) y(k)$$

where a is expressed in terms of Δt and the filter time constant τ_F as $a = \exp(-\Delta t/\tau_F)$. Not all control systems implement exponential filters in the same way. It may be that the user must specify a parameter value instead of a time constant, and the user-specified parameter may be on the previously filtered value or on the signal value. It is helpful to think of the exponential filter in terms of "old" and "new" parameter values, corresponding to the previously filtered value and latest signal, respectively:

$$y_F(k) = f_{old} y_F(k-1) + f_{new} y(k)$$

where $f_{old} + f_{new} = 1$, i.e., each parameter is one minus the other. Note that the time constant can be calculated from f_{old} as $\tau_F = -\Delta t/\ln(f_{old})$. It's always good to know the actual filter time τ_F used in a filter.

As mentioned, there is a trade-off between removing high-frequency noise and the lag introduced into the filtered signal. To show this, consider a spike (of magnitude 1) to represent a single instance of noise, followed by a pure step of magnitude 1 without noise. Ideally, we would like the filter to ignore the spike, but track the step perfectly. In Figure 3, we see that suppression of noise comes at the price of lag in the filter to the new value. We also see that filtering at a slower Δt relative to the scan Δt has a detrimental effect, similar to increasing τ_F. However, the effect of a slower filter Δt relative to the scan Δt is actually worse than shown in the figure. In the figure, the step time was synchronized so that all filters "see" the step as the same instance. In reality, the measured value to the filter (and to subsequent applications) may be delayed by as much as the filter Δt.

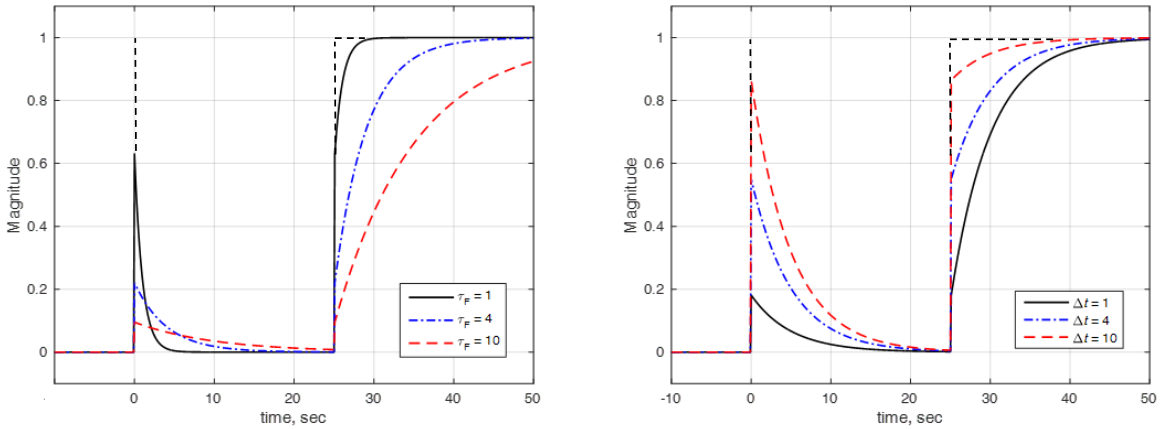

Figure 3. The effect of τ_F and filter Δt on a noise spike and step response with an exponential filter. Left: Increasing τ_F suppresses noise, but adds lag to the step response. Right: Increasing filter Δt relative to scan Δt adds lag and also introduces delay (not shown).

A natural question is whether a higher-order filter offers significant advantage over the first-order or exponential filter. Higher-order filters result in more terms and more constants that must be stored, but this doesn't pose a problem for today's control systems. Here, we consider 2nd- and 3rd-order overdamped filters, with a repeated time constant τ_F. It is derived from the corresponding continuous filter using a technique called Pole- Zero Mapping (Yang, 2009). Other higher-order filters, such as the Butterworth, exhibit an underdamped step response.

Table 1. Digital Filter Equations

1st-order filter

$$y_F(k) = a y_F(k-1) + (1-a) y(k)$$
$$a = \exp(-\Delta t/\tau_F)$$

(a used in all filters)

2nd-order Filter

$$y_F(k) = a_1 y_F(k-1) + a_2 y_F(k-2) + (1-a_1-a_2) y(k)$$
$$a_1 = 2a$$
$$a_2 = -a^2$$

3rd-order filter

$$y_F(k) = a_1 y_F(k-1) + a_2 y_F(k-2) + + a_3 y_F(k-3) + (1-a_1-a_2-a_3) y(k)$$
$$a_1 = 3a$$
$$a_2 = -3a^2$$
$$a_3 = a^3$$

Note: It's important to carry enough decimal places in the a coefficients as the $y(k)$ coefficient (1 – sum of a coefficients) can be small, especially for larger τ_F.

We will first evaluate filters based on 1) reduction in standard deviation of random noise and 2) lag in the step response. The desire is to obtain the lowest lag for the same amount of noise reduction. The reduction in standard deviation of random noise can be directly calculated using the Yule-Walker equations (Box, Jenkins and Reinsel, 1994). The noise attenuation for the 1st-order filter is given by $\sqrt{(1-a)/(1+a)}$; higher-order filters require a simultaneous solution. The lag is evaluated by direct calculation of the filter to a step input. Figure 4 shows the comparisons of the filters. We see that the higher-order filters provide less lag for a given time constant τ_F. A better way to compare the filters is the time to achieve a given response time to a step input, and here we select the 98% response time. For example, at levels of 0.30 and 0.20 attenuation of noise, the corresponding 98% response times are:

	98% response time, sec		
Attenuation	**1st-order**	**2nd-order**	**3rd-order**
0.30	20.68	15.07	14.08
0.20	47.90	35.13	33.42

We see noticeable improvement of the higher-order filters over the 1st-order filter. The 2nd-order filter achieves a 27% faster and the 3rd-order filter 32% faster time response compared to the first-order. At 0.3 attenuation, the relative improvement is similar, but with slightly less percentage improvement. The relative improvement of the 3rd-order filter over the 2nd-order is less than the improvement of the 2nd-order over the 1st-order, suggesting a dimensioning improvement of higher-ordered filters of this type. It should be noted that at smaller percentage response times—as the percentage response moves away from near steady-state (e.g., at 80%)—the advantage of the higher-order filters continue to hold true although with less relative improvement.

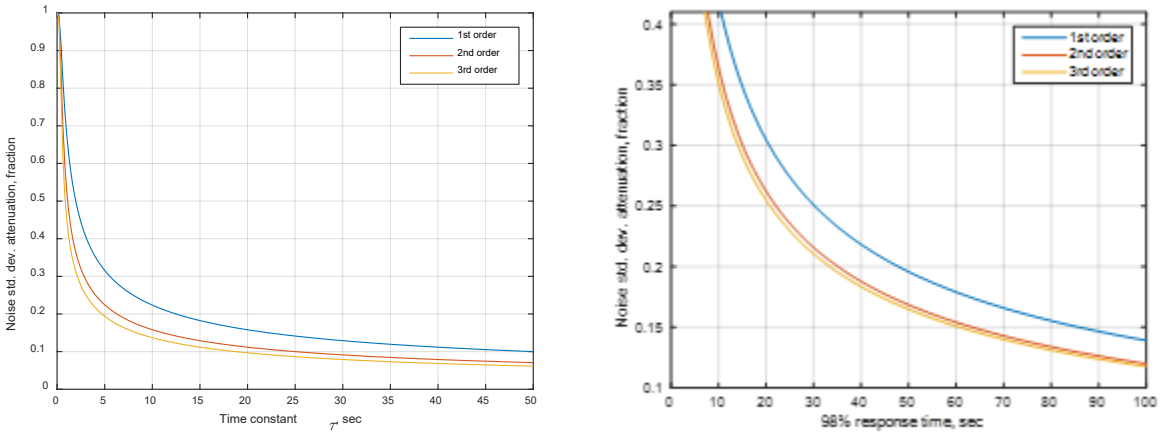

Figure 4. Comparison of 1st-, 2nd- and 3rd-order filters. Left: noise attenuation as a function of τ_F. Right: noise attenuation expressed in terms of 98% response time.

The other filter we consider is the moving average filter. It is an FIR filter with all coefficients equal—b_i = 1/*N*. This filter is simply the arithmetic average based on the most recent *N* measured values. Notice that the calculation can be simplified to $y_F(k) = (y(k) + Ny_F(k-1) - y(k-N))/N$. The noise attenuation

for the moving average filter is directly calculated as $1/\sqrt{N}$. Figure 5 shows the attenuation of noise of the moving average filter as a function of the number of measurement values *N*, and compares the moving average filter with the 1st- and 2nd-order filters. We see that the moving average filter provides better noise attenuation at the cost of significantly more filter coefficients and storage of previous signal values. Again, this should not pose a problem for today's control systems, particularly if large *N* moving average filters are not applied widely (i.e., only to selected applications).

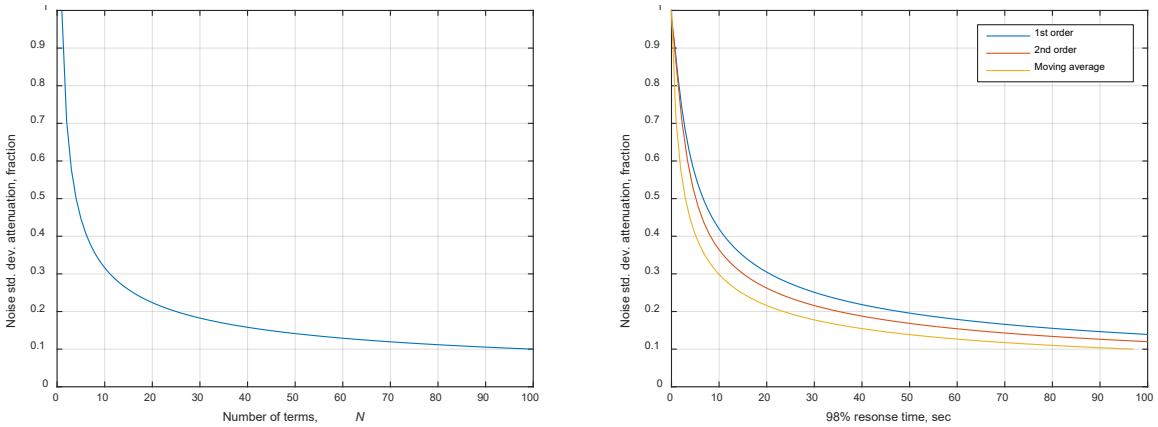

Figure 5. Moving average filter. Left: noise attenuation as a function of N. Right: noise attenuation expressed in terms of 98% response time, showing the improvement compared to 1st- and 2nd-order filters.

The step response of a moving average filter is a ramp starting at value 1/*N* and achieving steady state at $N-1$ time steps, corresponding to a time equal to *N*-1 times the filter Δt. This is illustrated in Figure 6 for a moving average filter with *N* = 10. A benefit of the moving average filter is that it can completely eliminate a constant, periodic disturbance imposed on a signal if the filter values *N* and Δt are set such that their product $N\Delta t$ equals the period of the periodic disturbance. Figure 7 illustrates the case of a periodic disturbance with period equal to 30 sec.

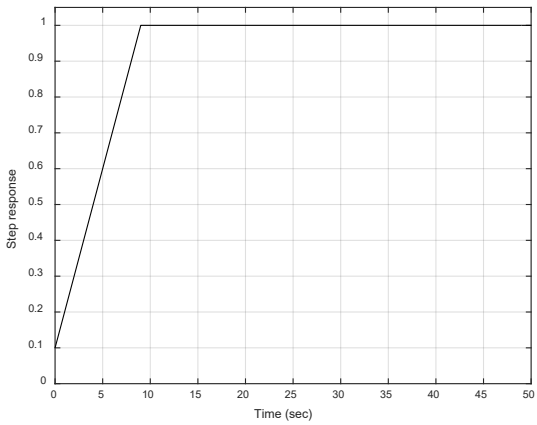

Figure 6. Moving average filter with *N* = 10 applied to a step at time 0.

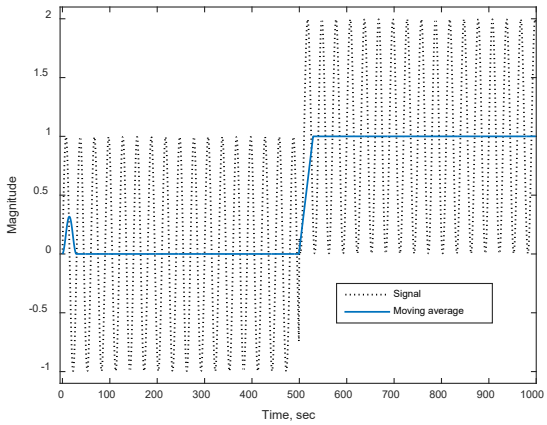

Figure 7. Moving average filter applied to a constant signal followed by a unit step. Notice that the cycle is eliminated after the initial transient or start-up of the filter.

We'll now consider the impact of filtering on PID loop performance. An important reason to filter the feedback variable (PV) is to ensure effective derivative action. Derivative action is important when aggressive control is required, particularly for integrating and near-integrating (lag-dominant) systems with additional lag(s) and significant dead time. If not properly filtered, noisy signals will cause derivative action to be ineffective due to erratic movement of the PID controller output. Fortunately, most industrial PID controllers include a filter on just the derivative term, expressed as a fraction, or divisor, of the specified derivative time. The fraction is usually in the range of 0.10 to 0.20 of the derivative time.

The other reason to filter the PV is to temper the controller output (CO) so the noise does not lead to excessive movement (and wear) of the valve, or cause disturbances to other control loops. In the past, another reason to filter the PV was due to resolution limits caused by the A/D converter, which would show up as noticeable steps in the measurement value; however, with 16 bit I/O cards, this is no longer an issue.

To evaluate the above filters in a closed loop, we simulate an integrating process with a noisy disturbance signal and an aggressively tuned PID controller. The loop and simulation details are shown in Figure 8. The tuning formulas used are the Lambda tuning rules modified by McMillan (2004) to provide better load disturbance rejection for a Standard Form PID, shown in Table 2 for an integrating process, where seemingly insignificant noise is amplified by a high PID gain. A value of $\lambda = 2$ sec. (twice the loop dead time) was used as the arrest time (time to stop excursion) for the tuning as it produced a non-oscillatory response in both the PV and CO provided the system is linear and the dynamics are fixed and well known. The following loop criteria are evaluated for each of the above filters.

Peak error: Maximum error following an unmeasured step in load disturbance on process input. Peak error is important to prevent relief, alarm and SIS activation, and environmental violation.

IAE: Integrated absolute error over time is a common criterion for measuring loop performance. It directly relates to economics as it provides a measure of the amount of process material that is off-setpoint.

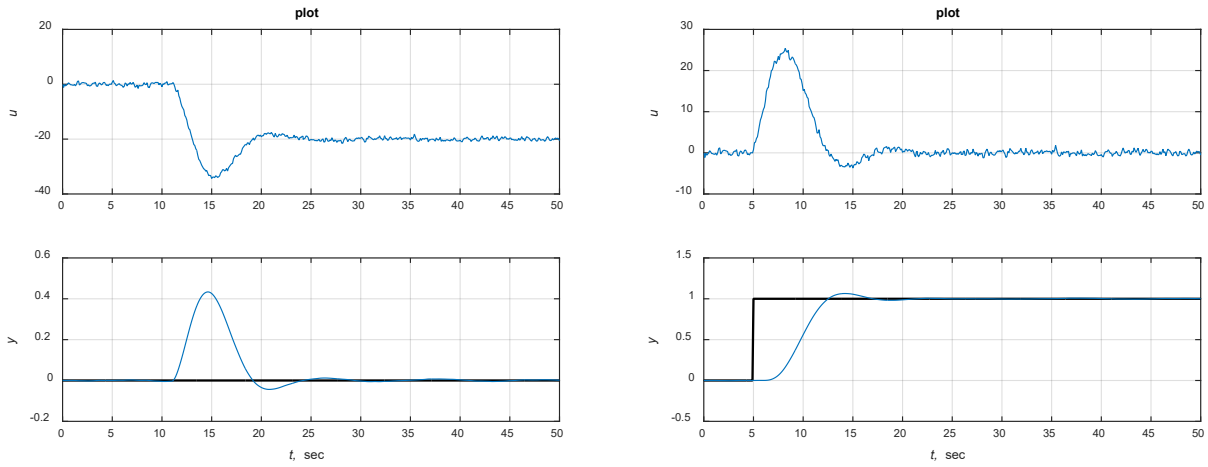

Figure 11. PID closed-loop responses with 2nd order filter. Left: load response. Right: setpoint response.

Based on the above results, the closed-loop simulations show that significant improvement over the 1st-order filter is possible when significant noise is present, and aggressive control and significant reduction in IAD is needed. These results also show that a 2nd-order filter provides the best metrics, although not always by a large margin.

Conclusions: What have we learned, what are the lessons? Be careful to not overly filter. What's good to the eye is usually too much filtering, especially for control. Excessive suppression of noise and the consequential instability or reduced response to disturbances can be costly. The more attenuation, the slower the filtered signal approaches a new average value—i.e., the more time lag. For the same amount of noise attenuation, higher-order filters (2nd- and 3rd-order) and the moving average filter approach a new average value quicker than the often-used exponential or 1st-order filter. The moving average filter can achieve higher levels of noise attenuation than the 2nd- and 3rd-order filters at the added cost of more coefficients and storage of previous signal values, but this should not pose a problem with today's control systems. An advantage of the moving average filter is that it can eliminate a periodic cycle if the filter Δt and *N* are selected appropriately.

When a noisy PV is used in control, the deciding issue to filter is unacceptable movement in the manipulated variable (controller output) caused by the noise. In this case, if aggressive control is required (higher gain with derivative action), higher-order and moving average filters allow tighter control with improved metrics (peak error and IAE) compared to the exponential or 1st-order filter.

Before retuning a loop, make sure to note what the filter parameters are: time constant or filter factor and Δt. Only use filtering to temper movement in the manipulated variable (controller output) caused by the noise. If aggressive control is required (higher gain and derivative action), higher-order and moving average filters allow tighter control with improved metrics (peak error and IAE) compared to the exponential or 1st-order filter. One should be careful to not cause loop instability with filtering. Although not extensive, these simulation results, plus others we have performed, generally show that of the filters considered, the 2nd-order filter provides the best metrics when targeting reduced levels of IAD.

We took the simple approach of keeping the tuning the same to show the improvement possible with more sophisticated filters. Ideally, the tuning *and* the selection of filter parameters should be jointly optimized, e.g., using a loop tuning optimizer.

We have focused on linear filters. However, the following additional suggestions are offered:

- When a process variable (PV) cannot respond faster than by an amount X/sec., a PV rate limit with this setting can be used to screen out noise without adding a lag. Alternatively, a setpoint rate limit to reduce CO movement can be put on the analog output block or secondary loop manipulated by the PID and external-reset feedback (e.g., dynamic reset limit) turned on to prevent the PID output from changing faster than the imposed setpoint rate limit. However, upon download or during tests for checkout and maintenance, the rate limit must be turned off.
- For pH measurements, the use of middle signal selection of three electrodes can reduce noise and eliminate spikes that commonly occur due to imperfect mixing and ground potentials without adding a lag, in addition to ignoring slower responding electrodes due to aging or fouling.
- For electrodes and temperature sensors, increasing back mixing and minimizing phase changes, and for pipeline installations, ensuring the tip is in the middle of the pipe at least 25 pipe diameters downstream of a pump discharge or equipment outlet can greatly reduce the source of process noise for these critical primary loops.

References

1. Yang, W.Y. (2009). *Signals and Systems with MATLAB.* Elsevier, Inc., Oxford.
2. Box, G.E.P., Jenkins, G.M., and Reinsel, G.C. (1994). *Time Series Analysis*, 3rd Ed. Prentice-Hall., Englewood, NJ.
3. McMillan, G.K. (2015). *Tuning and Control Loop Performance*, 4th Ed. Momentum Press, New York.

About the authors

Mark Darby, principal consultant, CMiD Solutions, Houston, can be reached at darbymark@sbcglobal.net.
Greg McMillan, senior principal software engineer, Emerson Process Solutions; Control columnist and Process Automation Hall of Fame member, can be reached at Greg.McMillan@emerson.com.