controldesign.com

control design

FOR MACHINE BUILDERS

HOW TO TUNE PID LOOPS

PutmanMedia®

How to tune PID loops

Servo-motor applications and temperature-control applications often need training after the auto-tune

By Mike Bacidore, editor in chief

A *Control Design* reader writes: I often have difficulty tuning PID loops, especially for temperature control applications and servo-motor motion applications. If I use a temperature controller, the auto-tune built into the device often works well if I follow the manufacturer's recommendations. However, if the temperature control or motion control is through a PLC or other advanced controller, it's often a long training process to find the optimum coefficients for the proportional, integral and derivative of the control loop.

Whether I am controlling the temperature of my medical cleaning process skid or moving my servo-controlled three-axis gantry, if the process variable (PV), such as temperature or position, needs to change slowly, it's easy. As soon as I need better performance, such as a fast change in temperature or position, or if dynamic loads are involved, my PID loops become unstable and oscillate or the process variable overshoots or undershoots the setpoint. I can't afford to burn up the medical device I'm cleaning or have a spongy motion-control system that can't snap to a position.

How do I achieve a quick-responding PID control loop that can handle dynamic changes to the process variable without chasing PID coefficients for days? What are some rules when designing the control system for these closed-loop control applications?

Where do I start and what are some best practices to get stability and optimal control in process-skid and motion-control applications from both a hardware and a PID tuning standpoint. What are some procedures to follow to help reduce PID-loop-tuning time? Any suggestions are appreciated?

ANSWERS

P IS FOR PROPORTIONAL BAND

First, determine what engineering units the tuning parameters use. This allows you to understand if increasing or decreasing a parameter has a positive or negative effect.

The P in PID is for proportional band. It is also known as gain. Increasing the proportional-band setting decreases its effect on the loop. Whereas increasing the gain parameter increases the effect. Proportional band = 100/gain.

I is for integral. This is the time parameter. It is also known as reset. Increasing integral parameter increases the effect; decreasing reset parameter increases the effect. Integral = 1/reset.

D is for the derivative parameter. It is also referred to as rate. These terms act the same: Increasing the value will increase the effect on the loop.

For both integral and derivative, the units of time must be understood, as well (milliseconds or seconds).

A reasonably quick method to create a stable loop requires a process upset through heat loss. This method is used without the process actually being in operation.

Before beginning, adjust the set point (SP) to about 80% of the normal process setting.

1. Start with the proportional band at whatever the default value is, or 50% of setting range.
2. Set both the integral and derivative parameters to zero.

Wait a few minutes for the temperature to stabilize. Note that it will not be at the set point, generally lower.

Upset the loop by introducing loss either by raising the set point 10% or removing heat through whatever method is practical. I have used air guns; opened doors of furnaces; sprayed cold water from a garden hose; turned off main-burner gas-supply valves; tripped circuit breakers. And I've even shoveled snow onto rotary kiln infeed conveyors in an attempt to simulate process heat loss.

Observe the response. If the output slowly increases and the temperature does not oscillate, decrease (increase the effect) the proportional band and repeat the upset. Do this several times until you see a pronounced oscillation of the process variable; it still will not be at set point.

Now, without disturbing the process, increase the integral term one step at a time,

waiting a reasonable length of time after making a change until the temperature is now oscillating around the set point. This oscillation may not be very noticeable, but remember you are in a steady state condition at this time.

Next, upset the loop using whatever method available and observe the response. You can now add in a small amount of derivative term one step at a time after upsetting the process until there is acceptable performance. Decent enough control should now be occurring, allowing you to run the system under real process conditions.

Further fine tuning is now performed with small changes to one parameter at a time and allowing sufficient time for the change to be observed. Try to operate the loop in as many different process conditions as practical. Record all changes along with performance notes to aid in solving future problems.

— Mike Krummey, electrical engineering manager, Matrix Packaging Machinery, www.matrixpm.com

NO DERIVATIVE VALUE

In my experience, using PID-loop function blocks in a PLC the first thing that I do is to remove the derivative value completely. I have yet to find an application where a derivative value was needed. The second thing to consider is the resolution of your control variable. If the resolution is too high, then you will never get stable control, no matter what. I will give a simplified example of what I mean by resolution of the control variable.

If the application is controlling the temperature in a small 5-ft-by-5-ft room, for example, and you try using a 500,000-Btu furnace as your control variable, then your loop will never be stable. You will always overshoot to an extreme oscillation around the set point because you have too coarse of a control variable.

— Dan Michki, senior robotics/controls engineer, East Balt Bakeries, www.eastbalt.com, Chicago

EQUIPMENT, ALGORITHM, SAMPLE RATE

First, know your equipment. Don't try to tune over an issue that is caused by faulty or erratic equipment. Make sure there are no external equipment problems causing issues in your PID control—for example, tuning a methane-powered generator with a dirty methane supply or tuning a generator on top of a governor. Also watch out for lag or dead time in your equipment, which can be caused by actuators or sensors that are slow to respond to dynamic changes. And, lastly, keep an eye out for external noise on your PV signal; use a filter whenever possible to be sure the feedback is pure.

Next, know your algorithm. All controllers are not created equal, and most manufacturers have their own interpretation of the PID equation. Look through the help file or product documentation to determine which

version of the PID equation your controller is using. This will help you to determine the proper values for your coefficients. This also explains why the auto-tune works so well with the temperature controller you mention. Using the manufacturer's recommendations, which are fine-tuned to the manufacturer's algorithm, will undoubtedly produce better results.

Set your sample rate. PID sample rate identifies how often the instruction will calculate a new output value. Make sure the rate at which you are calculating is set appropriately. Setting the rate too low will cause your controller to miss important changes, and sampling too fast could cause instability in your integral and derivative terms. Try to shrink the sample rate as much as possible, but, more importantly, keep that value constant.

As for the tuning itself, I'm not saying you are trying to tune the values all at once, but don't. You'll go insane. Instead, cancel out the integral and derivative, and focus on the proportional term first. Get the proportional value to a point where the system is correcting quickly in the right direction. Then add the integral term in and tune until you get the desired response in the low-error range. For most systems the PI control is good enough, but if you need derivative control, remember that the derivative is based on the rate of change in the error, not the error itself. For example, looking at the derivative part of this PID equation, Kr*(en – e(n-1)), Kr is the calculated coefficient, but the gain is determined based on the error at the current time minus the error at the time before. So, with a calculated coefficient of 1, if the error goes from 2 to 10, then the D term is 8; but, if the error then goes from 10 to 12, the D term is 2 with the error still increasing. The D term dropped to 2 because the error rate of increase slowed even though the error itself increased. On the flip side, with the goal of eliminating error as fast as possible, the derivative term can also become a hindrance. If the D term sees a dramatic swing toward correcting the error, it will mitigate the gain to avoid an overshoot, which could unintentionally cause an undershoot if not set properly. In other words, use the derivative term carefully and only when absolutely necessary.

— Bill Dehner, technical marketer, AutomationDirect, www.automationdirect.com

MULTIPLE PIDs

I mainly work with temperature controls, and each piece of equipment has a unique temperature profile within the temperature zone. The battle of having a PID loop that reacts quickly without overshooting is a common problem. In the aerospace-parts manufacturing industry, this overshoot is tightly controlled via AMS 2750 requirements.

One of the easiest ways that I have found to help resolve this is with multiple PID loops. In the past, we just had a single PID for the controller, but more modern controllers are

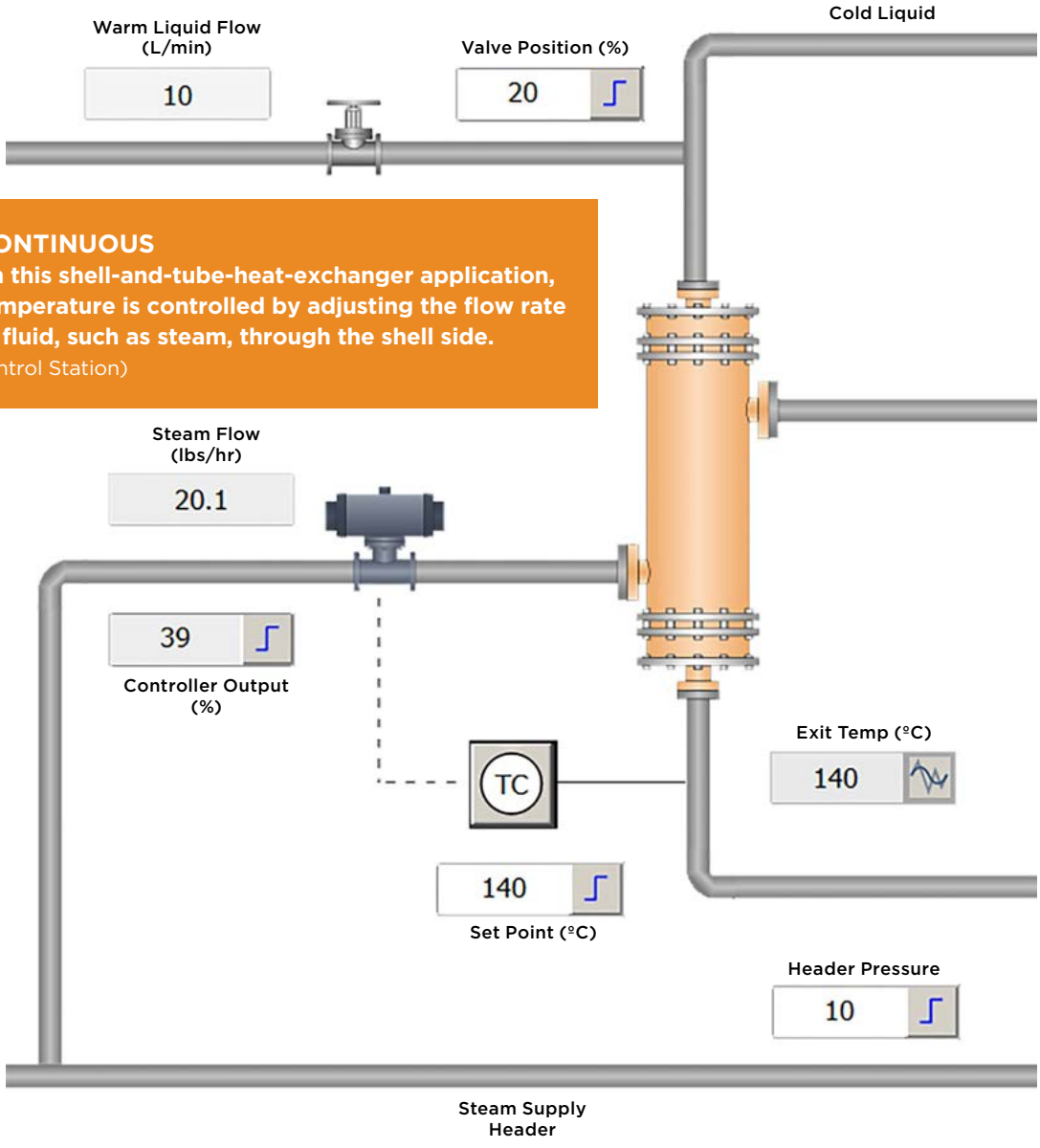
allowing multiple PIDs. This is achieved by setting a PID for a temperature range. This will allow you to have an aggressive PID and not worry about the overshoot. Then, when you get closer to your setting, you can have a PID that is less aggressive and will not overshoot. For example: I'm trying to heat a furnace to 1,200 °F at a ramp rate of 50 °F/ minute. We will use a PID setting for 0-1,100 °F; then there will be a different PID setting for the range from 1,101 °F and up. This will allow us to tune the 1,101 °F-and-up differently than we do the lower range. If the first PID was used, then we would surely overshoot. However, once the temperature reaches 1,101 °F, then the second PID takes over and controls less aggressively so that we don't overshoot.

— Scott Richardson, continuous improvement manager, special processes

TEMPERATURE CONTROL CHALLENGES

Temperature control applications can be broken into two main categories—fully continuous and semi-continuous (or batch). An example of a fully continuous temperature controller is a shell-and-tube heat exchanger (Figure 1). In this application, the exit temperature is controlled by adjusting the flow rate of heating fluid, such as steam, through the shell side. As the flow of heating fluid is increased, the temperature also increases until it eventually settles at a temperature well below that of the heating fluid. Similarly, by decreasing the flow of steam, the temperature drops. The change in steam flow impacts the final steady state temperature of the exit steam. Processes such as this are relatively easy to tune using the basic tuning rules available. This type of process is also known as a self-regulating process.

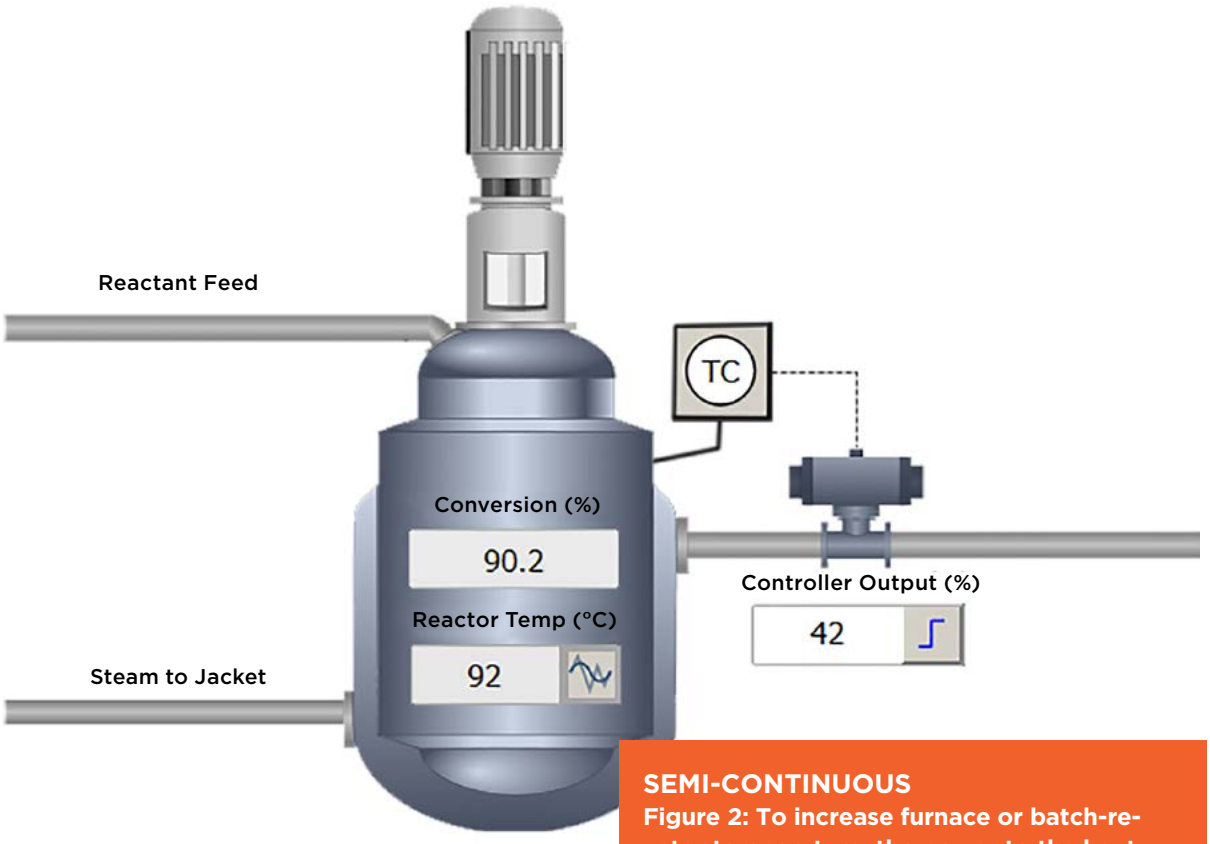
An example of a semi-continuous temperature control application would be temperature control of a furnace or batch reactor (Figure 2). To increase the temperature, the power to the heating element or flow of steam through the reactor jacket is increased. As the amount of heat to the vessel increases, the rate of temperature change increases. By decreasing the heat into the system, the temperature will still rise, albeit at a much slower rate. Note, however, that turning off the heat completely won't cause the vessel to start cooling instantaneously. The dynamics between the heating and cooling can be remarkably different. Additionally, the inertia of the heating processes is challenging. The walls of the vessels, or even the material that is being heated, all absorb energy and generally are slow to release the energy. This presents a challenge. These semi-continuous temperature control applications are highly nonlinear both in terms of their operating range and in terms of their operating direction (heating versus cooling). Temperature control exhibits give-and-take dynamics that practitioners must consider in order to strike the proper balance.



FULLY CONTINUOUS

Figure 1: In this shell-and-tube-heat-exchanger application, the exit temperature is controlled by adjusting the flow rate of heating fluid, such as steam, through the shell side.
(Source: Control Station)

To understand the solution, one must understand how a PID controller reacts to disturbances and set-point changes. Most PID controllers work using three corrective forces: proportional, integral and derivative. The proportional term simply adds a correction term to the output based on the current error (SP-PV). The integral term adjusts the output based upon how much error and for how long the error has persisted, accumulating influence over time. The proportional and integral terms drive the process toward set point in order to eliminate error, and they will not stop until the error is gone. The derivative term, on the other hand, does not account for the value of error (if it's large or small); it only accounts for the rate of change of the error. If the error is changing quickly, the derivative term has a large influence and works to prevent the error



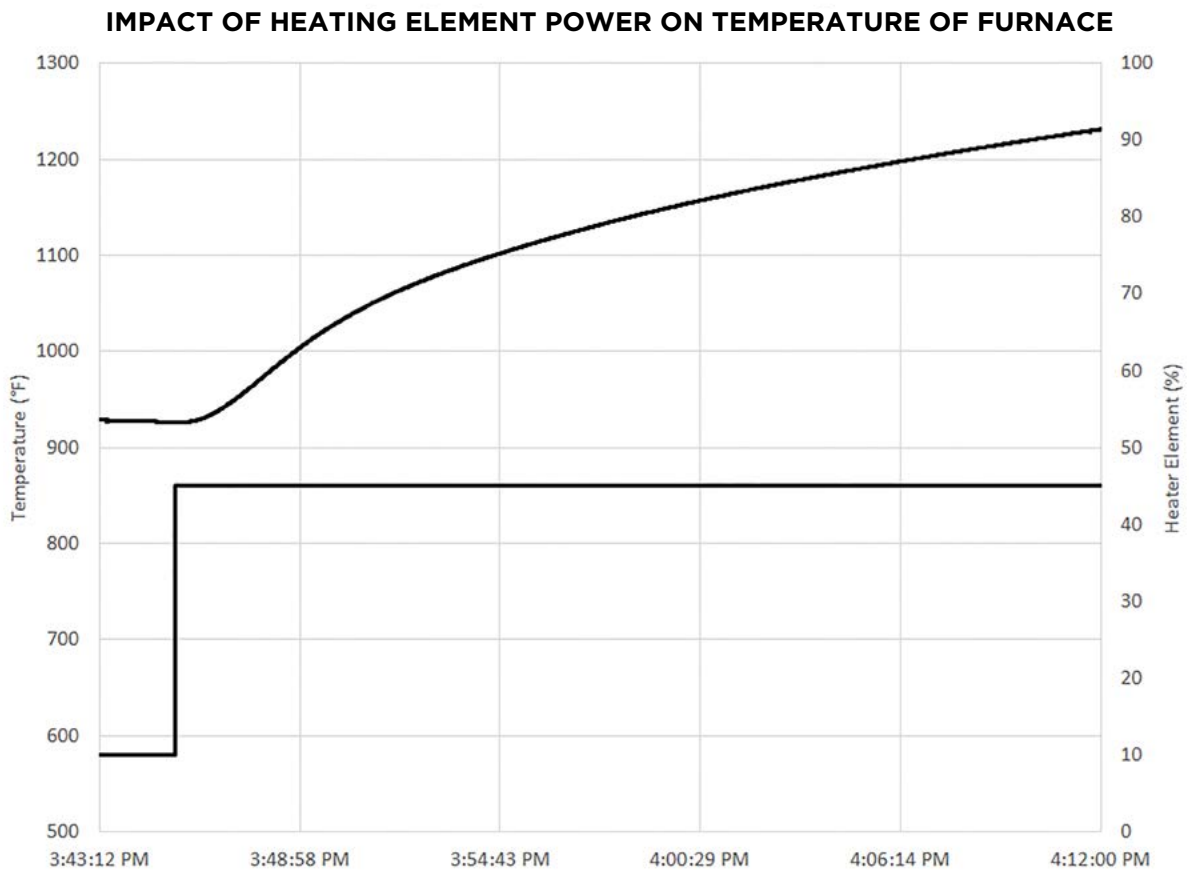
SEMI-CONTINUOUS
Figure 2: To increase furnace or batch-reactor temperature, the power to the heating element or flow of steam through the reactor jacket is increased. As the amount of heat to the vessel increases, the rate of temperature change increases. By decreasing the heat into the system, the temperature will still rise, albeit at a much slower rate.
(Source: Control Station)

from changing too quickly. When all three forces work together, you get a fast correction using proportional and integral driving the output to get the process variable to set point, and the counteraction of the derivative pulling back on the correction once the process variable starts to change too rapidly.

In most cases, the full PID algorithm is recommended for semi-continuous type temperature controllers.

There are many methods for tuning semi-continuous temperature controllers, and all apply a similar approach: model the process to characterize its dynamics, and then use that model to calculate tuning values based on tuning correlations.

Let's focus on the modeling part first. For the sake of simplicity, we will stick to first order models. Let's look at an open-loop bump test of a semi-continuous furnace, like an annealing furnace (Figure 3), and examples of open-loop step tests for self-regulating and non self-regulating processes (Figure 4). Notice that the temperature-control example from the furnace doesn't really look like either of the models. In fact, these types of semi-continuous temperature controllers are best represented by a hybrid



FURNACE TEST

Figure 3: In an open-loop bump test of a semi-continuous furnace, the temperature-control example doesn't really look like either of the models. In fact, these types of semi-continuous temperature controllers are best represented by a hybrid of the two models.

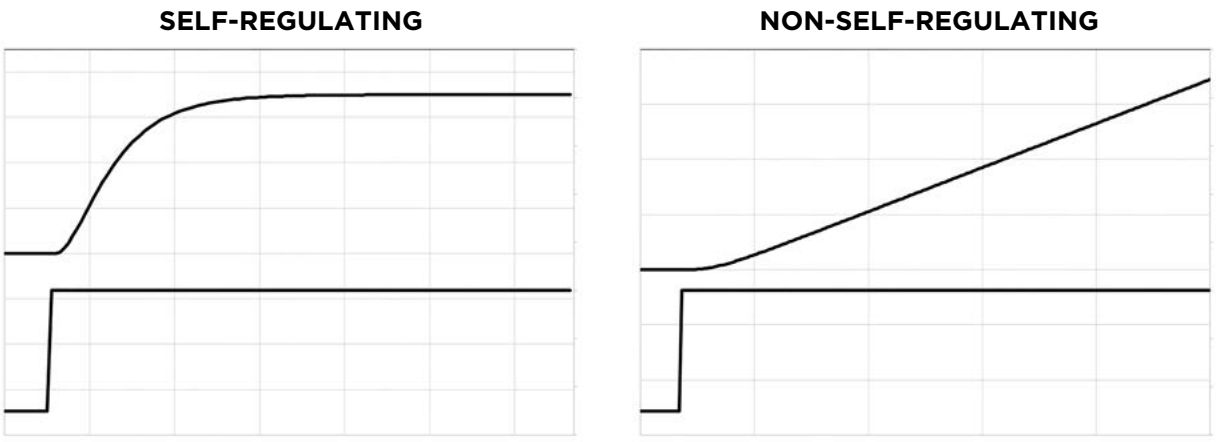
(Source: Control Station)

of the two models. That is a different topic for another time.

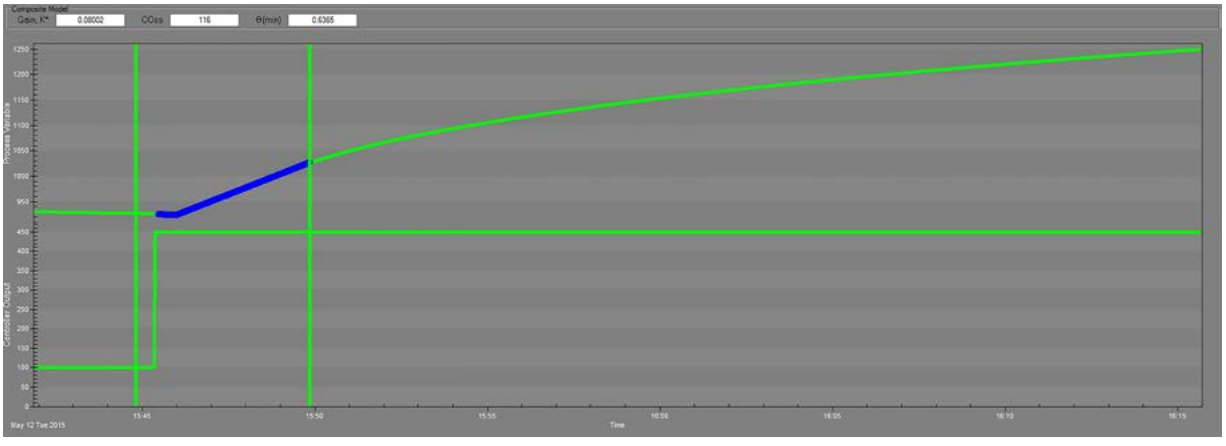
In practice, you have two options. The traditional method to generate a model is to do a step test in manual mode and wait for the process to complete its response. The challenge with this method is that it takes a very long time for these types of processes to steady out, if they ever do. The other option is to treat the initial process dynamics as a non-self-regulating process. Figure 5 shows an integrating process applied to the initial dynamics. Since we only require the initial dynamic response, the time required for testing can be significantly reduced. The integrating model parameters (K^*→Integrator Gain and θ_p→Process Deadtime) can be generated either with software or by hand using the slope method detailed at www.controldesign.com/controlguru.

Once a model has been generated, tuning correlations can be employed to generate a set of tuning parameters without having to trial-and-error tune. For this example, we can use the internal model control (IMC) method to generate full PID Tuning values for an integrating process *(See equation 1 on the next page).*

The biggest challenge with using an IMC approach to tuning temperature controllers is the selection of the closed-loop time constant, τ_c , which is often referred to as lambda. This value represents the time needed for the process to reach 63% of a set-point change. The closed-loop time can be based either off of a particular design objective or



SELF-REGULATING AND NON-SELF-REGULATING
Figure 4: These examples show open-loop step tests for self-regulating and non-self-regulating processes. (Source: Control Station)



INTEGRATING PROCESS
Figure 5: An integrating process is applied to the initial dynamics. Since we only require the initial dynamic response, the time required for testing can be significantly reduced.
(Source: Control Station)

IMC TUNING CORRELATIONS
NON-SELF — REGULATING PROCESSES

	K_C	T_I	T_D
$Ideal - PI$	$\frac{1}{K_P^*}\left(\frac{2\tau_C + \theta_P}{(\tau_C + \theta_P)^2}\right)$	$2\tau_C + \theta_P$	–
$Ideal - PID$	$\frac{1}{K_P^*}\left(\frac{2\tau_C + 1.5\theta_P}{{\tau_C}^2 + 2\tau_C\theta_P + 0.5{\theta_P}^2}\right)$	$2\tau_C + 1.5\theta_P$	$\frac{0.5{\theta_P}^2 + \tau_C\theta_P}{2\tau_C + 1.5\theta_P}$

EQUATION 1

$$\left(\frac{1}{\left|K_p^*\right|} * \frac{PV\ span}{CO\ Span}\right) * 0.63 = \tau_C$$

EQUATION 2

$$\frac{1}{0.08\frac{°F}{amps \cdot min}} * \frac{1800 - 0\ °F}{1000amps - 0amps} * 0.63 \approx 14\ minutes$$

EQUATION 3

on the process' dynamics. An estimate for the closed-loop time constant for a non-self-regulating process can be calculated by determining the amount of time that it takes for the PV to change by 1% when the controller output (CO) is changed by 1%. Once this time is determined, it can be multiplied by 0.63 to determine the amount of time it takes to reach 63% of the 1% PV change. A simple method for estimating τ_c is shown above in *equation 2.*

Applying this method to the temperature loop explored earlier, the closed-loop time constant is estimated above in *equation 3.*

This method is designed to generate tuning parameters quickly, using minimal data and with the goal of providing a quick initial response with minimal overshoot. This method has been used with success on furnaces, ovens, fermenters and batch reactors. In the end, there is always a trade-off between fast initial response and overshoot. Indeed, the PID controller offers a simple means for control that has its limitations, especially as the deadtime of the system gets very large or as nonlinearity increases. In those cases, a simple PID controller may not be adequate, and more advanced forms of control such as adaptive PID, model predicative control or even fuzzy logic may be appropriate.

— Robert Rice, Ph.D., vice president of engineering, Control Station, www.controlstation.com

TAME THAT LOOP

A major aspect of controlling both thermal and position control loops begins with making the loop as tame as possible. This starts at the system design stage.

For the thermal loops, limiting the lag time helps to reduce the overshoot. Time lag is a high (theoretically infinite) order system but may be modeled reasonably by fewer stages to get a reasonable approximation, as long as the pure lag time is not too long. Pure lag time, or transport time, is the time between applying a pulse to the input and seeing any change in the feedback sensor response.

There is a trade-off in selecting the physical position of the temperature sensor with respect to the heating element and with respect to the controlled load zone. Remember, the control loop tries to minimize the error as seen by the sensor. If the sensor is too far from the load and too close to the heater, the response at the load can be slow and the sensor may not see variations at the load; the loop will, however, be very easy to control. Place the sensor too far from the heater, and the lag time can make overshoot very hard to avoid. Place the sensor on the other side of the thermal mass away from the heater, and the problem only gets worse.

The overshoot can be particularly bad if the heater can supply sufficient heat into the thermal mass to overshoot the set point prior to the sensor being able to respond (lots of power, large lag, small thermal mass). Such a configuration needs a very slow control response and gentle gains to keep from overreacting in the lag time. Moving the sensor closer to the heater cuts the lag time but may also blind the sensor to what is happening at the load. Either experimentation or modeling, or both, can help to locate a reasonable sensor placement to minimize lag. Other solutions, such as heat pipes, may also be used to reduce the lag and to make the system closer to equal temperatures, if that is a critical need.

Additional sensors may also be used to limit heating to a reasonable level approximating the desired temperature while the main sensor is slower responding but sensing nearer to the temperature control point (dual loop control). Alternatively, intermediate temperatures may also be estimated, using a model of the system, as well as the power being driven into the system, but these both are well past the realm of a basic PID controller.

On several medical-application temperature-control projects of which I have been involved, PID was not an optimal control strategy; a targeted control system design was used instead to allow significantly better stability and response. The addition of a zero (lead = differentiation) into the integrator, plus an additional zero (differentiator) with gain, gave a pair of zeros that were avail-

able to attract the poles of the system for a stable, fast response. The control point stayed to within ~.01 °C (as measured with the control sensor), with a fraction of a second response time to transients. Air load on the system from waving a piece of paper at it were readily detected and rejected. One project used a heat pipe to equalize the temperature over a fairly wide area that also had variable thermal loads. Any cold spot tended to condense the gas phase, quickly releasing heat into the colder area while not causing measurable changes to those regions already at temperature.

Knowledge about the system being controlled is of great help in determining the proper control approach. In the absence of a good model, a little experimentation as to the placement of your temperature feedback device(s) can significantly make a system easier to control. Apply a short pulse of power to the heater and then watch the response on a digital oscilloscope. You may want a second (temporary) sensor at the load to allow you to visualize the lag time and filtering effects of the thermal circuit, both at the control sensor and at the load. For example, a slight overshoot at the sensor may still produce an underdamped response at the region you are trying to control. Even if you are using a straight PID rather than an optimized controller, making the problem easier by proper sensor location selection will make for a much more stable and robust control system.

When handling larger inertial mismatches, the system may be modeled at a mass-spring-mass (rotary inertia, torsional spring, rotary inertia, to be more exact), which has an anti-resonance (both masses rotating in opposition and a resonance, which is approximately the motor mass and the spring working against the significantly larger load inertia [series and parallel resonances, if you will]).

This is actually the motor inertia, the spring constant of any shaft and coupler, and the load inertia. Most people will recognize a mass connected via a spring to a second mass. The equations are the same, but many did not see the math worked in rotary units in basic physics.

Optimizing the mechanical system for fast response control also starts long before tuning. Such factors as presence or absence of gearheads, the degree of backlash and the style of motor couplers can significantly change the settling time and can limit the response of any best control method. Over the years, I have moved well away from PID. We at QuickSilver Controls use position velocity integration acceleration (PVIA) due to the large range of loads it accommodates. The motor and its load may be modeled as a torque source coupled to the motor inertia, which is in turn, coupled to the load mass via a spring. The gain is the ratio of acceleration at the motor shaft and encoder to the input torque. The spring

represents the shaft and coupling stiffness. At low frequencies, the motor will see the combined inertias ($J_{motor} + J_{load}$). The low frequency motion does not significantly excite the spring. The gain (acceleration per torque) drops off with frequency. At a certain frequency, the lag introduced by the spring causes the load inertia to have opposite phase to the motor inertia; that is, the motions will be in opposite directions. This is the anti-resonance frequency and corresponds to a local low point in gain/frequency point. As the excitation frequency further increases, the load inertia has less and less motion relative to the motor, as the springiness of the coupling from the motor to the load inertia limits the acceleration of the load inertia. The slope of the gain/frequency plot doubles each octave in this frequency range. This gain increases until the spring constant of the shaft and coupler interacting with the motor inertia will cause a maximum gain point called resonance. The peak in gain here may be very strong if there is little damping in the system. Beyond this frequency, the motor inertia will dominate and the gain/frequency plot will again drop off by a factor of 2 for each octave. Note this is a simplified model; things can get much more interesting when more elements are considered and especially if there is any backlash in the system.

Increasing the relative inertia of the load compared to the motor inertia will cause a greater ratio of the resonance frequency to anti-resonance frequency, causing a greater change in gain for which the control system will have to cope. Appropriate filtering, which is not present in a basic PID system, can shape the overall system response to provide a robust and responsive system. From the design side, the use of an adequately stiff shaft and coupler can push the resonance frequencies higher to move them away from the range of frequencies in which the motion is desired. Adding inertia to the motor, or very near to the motor, can reduce the frequency ratio of the resonance and anti-resonance frequencies, reducing excess gain, which may cause trouble in simpler PID systems. Finally, gearing may be used to reduce the load inertia as seen by the motor to effectively reduce the ratio of motor to load inertia. This again limits the gain rise between anti-resonance frequencies and resonant frequencies, which tames the system for simple PID controllers. Gearing reduces the reflected inertia by the square of the gear ratio. The addition of a properly filtered acceleration estimate provides an electronic approximation of a viscous inertial damper, which provides an additional damping and significantly boosts the phase margin over a range of frequencies that can be as large as two or three octaves. This may eliminate the need for tight notch filters, which may have issues as the load changes.

Finally, we use a patented interaction between the motor and the driver stage

to introduce additional damping into the system, by a combination of active and passive methods, which further adds to the gain margin. As shipped, the system typically will operate open shaft to approximately 5x motor inertia with minimal additional tuning. With about 10 minutes of effort, a 100:1 inertial mismatch may be easily obtained. According to varying literature, PID is typically limited to less than a 10:1 inertial mismatch.

Using straight PID, it is important to be aware that there are series and parallel implementations, and various implementations in between. The method of tuning each of these significantly varies. The series implementation tends to have more tuning interactions but is more common of analog controllers.

Remember, the first step is to start with an easily controllable solution. For thermal, pick a sensor location without too much lag, if possible, and (almost) always between the heater and the region to be thermally controlled. Try to minimize lag in the system, which includes both the physical system and any computational lag in the system, such as a slower or overtaxed PLC.

With mechanical systems, for fast responses, minimize the springiness and lost motion in the system—tight and stiff couplers; lower backlash gearing, if needed; shorter, thicker shafts; solid anchors for the motor and for the gearhead, if needed, to handle the reaction forces. This is missed more often than would be expected. Minimizing the load where possible is also helpful.

Yes, there is expertise available to handle more difficult control problems, but in a large majority of the systems, a little effort up front can make the control effort much easier, less expensive and more robust.

Why lag is so bad: control systems are all about timely responses to measured stimulus. Lag increases uncertainty in the system. Think about investing with information that is months or years old, or driving watching a 10-second-delayed video. It would be very difficult to respond to a light changing when you were already through it by the time you saw it change. The control system has the same problem if a heater has already supplied sufficient energy into a system to overshoot the set temperature by the time the sensor starts to respond. In the absence of timely data, a model of the system that gives a predictive response may be needed, or the maximum rate of power must be limited—driving really slowly, in the comparison. Mathematically, lag degrades the phase margin of a system, forcing either lower gains or low stability margins (overshoot), or even oscillations (loss of stability).

— Donald Labriola P.E., president, QuickSilver Controls, www.quicksilvercontrols.com

START WITH THE LOOP

Here are some techniques I have used in the past when a fast temperature ramp followed by minimal overshoot is desired.

Obviously a good understanding of how PID loops work is essential (See **www.innovative-controls.com/blog/basics-tuning-pid-loops**).

Use a ramping set point with an exponential decay at the end to knock off the terminal elbow. That was a technique I came across back in 2006 for exactly this problem—a difficult temperature loop that needed to ramp up quickly to a target set point, but not significantly overshoot it. Ramping the set point at a proper rate keeps the loop under control the whole way up, meaning your output is not pegged. If it is pegged, the PV will lag, and you are out of control, and it will overshoot badly. The solutions are to either slow your ramp rate or add physical heating capacity to achieve the desired rate. But, if you ramp in control at a constant rate until you reach the target, then stop. The loop will have settled controlling at a steady rise in temperature and will have a small but avoidable overshoot when you switch to the steady target set point. Instead, slow the ramp at the end to knock the corner off the ramp-target transition, and you can consistently nail the target.

SP outrun limits: program your ramping set point to never be more than x degrees above the PV. If for some reason the ramp is faster than the system can physically support, the output will be pegged and you are out of control. At that point the SP will pull away from the PV. Having an outrun limit means that when you regain control, your overshoot will be less severe than it would have otherwise been, but it's still better to keep the ramp slow enough to stay in control so that this never happens.

If, rather than controlling on the way up, you want to heat up to the target as fast as the system's capacity will allow, an alternate technique is to set up a learning system. This is something I did last year with an oven system that had a huge latency; it would continue heating up very significantly even after we cut the heating output. When it's time to heat, have your PID loop output track its maximum. Then, when the PV gets within x degrees of the target SP, cut the output track to zero and release the loop to automatic. You may need to hold the output at zero for a few seconds before releasing. Here's the tricky part: Have a timer that runs for about 1.5 times as long as it takes that system to get to its maximum initial overshoot. Capture the highest (smooth/filtered) PV during that entire duration. When the timer expires, automatically adjust the x offset based on how much the PV overshot or undershot the set point so it hits it closer the next time. Have clamps on that x offset so, if something odd happens, it will still be reasonable the next time.

The other big technique here is to use adaptive gain control. Use one set of tuning parameters during the ramp, and then switch to a different set for the soak. I find the above techniques usually eliminate the need for this, but in some systems it can be beneficial.

Derivative action is only useful when there is a significant time delay from a step change in output until the result of that change starts to be seen in the PV. Loops controlling variables such as flow, pressure, level speed, and position—where an output change affects the PV everywhere instantly—never need derivative. By contrast, loops controlling variables such as temperature, pH and other chemical concentrations deal with fluids where the change in output must be mixed, convected or transported from the point where the output affects the fluid to the measurement point. Those loops may benefit from derivative action. However, when manually tuning, I usually avoid using derivative, except as a last resort.

These techniques also are covered in my loop tuning blog at www.innovativecontrols.com/blog/advanced-pid-loop-tuning-methods.

As for motion control, most of my experience is in process controls. But in my little experience with servo controllers, I've always just sent the target position from the PLC to the servo controller and let the servo controller tune itself, occasionally adjusting the S curves in the servo controller.

— Chris Hardy, systems integration engineer,Cross Integrated Systems Group, www.crossisg.com, CSIA (www.controlsys.org) member, Birmingham, Alabama

REMEMBER TO RECORD PAST SETTINGS

While PID loops make life easier by automating process control functions, they also present challenges in a variety of applications to achieve optimal stability and control. The following method can be applied across the board to improve loop tuning.

Before making any changes, be sure to record the current PID settings. It is also a good idea to have a record of past settings, stable or otherwise. Finally, check for safety concerns.

The following steps are for a controller set to interacting type. Most controllers offer a choice.

1. a. Turn integral to no integral action.
   b. Turn the derivative time to as small a number as possible. These settings are often called Off.
2. a. Increase the gain in steps until the loop begins to cycle. This gain will be called Kcu.
   b. Record the period of cycling in minutes. Call this Pn.
   c. Decrease gain to stable levels as soon as possible.

These readings are only estimates.

3. Calculate Kc = Kcu/2 for aggressive loop tuning.
4. Calculate Kc = Kcu/4 for conservative tuning.
5. Ti (integral time in minutes) = 1.2Pn.
6. Td (derivative time in minutes) = Pn/8.
7. Tf (Filter time) = Pn/8.

If the process is not too noisy, it may be as well to turn the filter off. To some degree, filter and derivative just offset each other.

For especially difficult cases, an experienced expert consultant may be required on-site.

— Lee Payne, president, Dataforth, www.dataforth.com

FOCUS ON PI

Motion systems usually only use PI controllers and add filters on top of it to filter out machine-specific interferences. If you have a specific frequency, you could use a simple notch filter; for more challenging issues a biquad could get you some really nice results and a better-performing motion system.

We have a built-in system that measures the system response and you can apply filter directly into the resulting bode plot.

As for temperature or other control loops, we have similar functionality based on our mechatronics libraries. We can identify the system the loop needs to control and apply default settings based on the identification. That usually provides some really good results to start with.

In general, I think full PID controllers are rather rarely used, but they have their place. Obviously, the more elements you need to control, the more difficult it becomes to find the optimal solution.

— Marcel Voigt, senior solutions engineer,

B&R Industrial Automation, www.br-automation.com

5 TUNING TIPS

This is a complex question in two parts—process PID tuning for temperature and motion control PID tuning.

The customer doesn't give specifics about the system or mechanics. Thus, his difficulty to be able to tune his system might come from poor mechanics, inappropriate motor-and-gearbox sizing, the low bandwidth of his system, saturated amplifier or a number of other things.

For motion-control PID tuning, here are some general guidelines.

1. Always perform sizings to take in account the load to motor inertial mismatch, which is the most common problem and the source of loop instability. It will be hard to tune if this load inertia is too high.
2. Remove system backlash as much as possible in the driven components. For example, use servo gearboxes with 16 arc-min or less; and use backlash-free couplings, low backlash nuts or belt and pulleys.

3. Use the step response method (command vs. response) to see how the system is reacting. This is commonly performed using an oscilloscope or is sometimes provided with the controller manufacturer's software. You need to see what you are doing; otherwise, you are just shooting in the dark.
4. Start tuning with integral gain at zero; increase proportional gain to get a bit of overshoot response; and then adjust the derivative gain a bit at the same time to damp the oscillations. Add integral gain at the end to remove the static error. Think of it as a pyramid. The base of the pyramid is KD; the middle of the pyramid is KP; and the top of the pyramid is KI. The KD (derivative gain) would have the highest value, followed by KP (proportional gain) with a lower value, followed by KI (integral gain) with the lowest value. Manual tuning is a trial-and-error process. For systems with underdamped conditions, increase KP and decrease KD in tandem. For systems with poor accuracy, gradually increase KI. For systems with high frequency ringing, increase KD.
5. At all time, make sure the current output of the amplifier is not saturated; otherwise, this procedure is invalid, and tuning is impossible. Saturated current can mean you are asking too much torque from the motor and amplifier or you are asking for a speed higher than what your system can achieve.

A deeper analysis of the complete system might still be needed to help diagnose flaws or weaknesses.

— Warren Osak, founder and CEO, Electromate, www.electromate.com

MOTION PROFILE

Even though the control of temperatures and servo positioning both use PID, they are different animals. Both systems have to deal with inertia. In a temperature control system there is thermal mass (thermal inertia). This mass is normally quite high and causes a long lag between an output change and actually seeing a resultant. Normally this is many minutes. This is the time constant of the system. In the case of heating a large metal object such as an extruder barrel, we will have a much higher time constant than an air heating system where we are controlling a gas valve for a furnace. With a servo system, the time constants are typically in the millisecond range. When we ask for motion, we get a response very quickly.

Most servo systems these days have the ability to generate their own gains. This is internally done by the system making several velocity changes, and as a result of those changes, it calculates the time constant of the system. From that information, it will generate the gains.

In a servo control system, we normally generate a motion profile for the velocity part of the control. This can typically be done in the motion command and is normally trapezoidal or triangular. When we use a profile, we don't just command the system to jump

immediately to the new position, but to move to the destination with a velocity profile. In the case of a trapezoidal profile, we cause the servo to ramp up to a constant velocity, remain at that velocity for a period and then decelerate to zero. When we reach zero, we should be at our requested position. A trapezoidal profile accomplishes the requested motion. This keeps the servo from seeing a huge position error and thus causing a large velocity command to the servo. This typically results in a quite large overshoot and most likely ringing badly around the desired target point depending on how aggressive the gains are.

When using a profile move internally the servo sees a constantly increasing position set point throughout the whole position move until it reaches the target. We can think of this as a virtual set point with a profile. When this is utilized, the servo doesn't see the huge position error but follows the profile and just having to make minor velocity/position corrections to stay on the profile. This allows the system to have higher gains for tight position control without causing the excursions that would normally result from those higher gains. This is like making thousands of very small moves continuously until the target is reached. The profile's acceleration and deceleration slopes will be dependent on the mass and inertia of the system. Just because it's a servo doesn't mean it can snap to a commanded position, as physics just won't allow it, depending on the amount of power available and the mass of the system. Using a profile doesn't mean the system can't be snappy; it just means that we are anticipating the work needed to accelerate the mass ahead of time, which keeps the positioning system from having to make huge corrections during the move.

An example might be if we wanted to move our car exactly 50 feet ahead and stop. We wouldn't tromp on the gas pedal to get there. We would slightly accelerate, then hold and then decelerate. It's a very basic example, but relevant.

— Doug Moldenhauer, senior controls engineer, Optimation, www.optimation.us

LIMITED PID

PID is limited for these problems. On the servo side, it is well documented that PID cannot handle dynamic loads without changing gains. Most servo manufacturers provide S-curve to reduce startup and shutdown disturbance, gain adjustment for systems that have a load buildup or build down (such as a winder/unwinder), and adaptive controls based on speed or operating variables. Some use alternate control loop strategies, such as entering time constants and inertia instead of proportional and integral values. Or fuzzy logic and neural networks are good options for very dynamic loads.

On ovens, temperature gets challenging if there is a large thermal mass or signifi-

cant multi-zone influence. The ways out of this one are using derivative, which I could never figure out; using cascade control if there are other temperature zones; or using model predictive or neural networks as is done on glass furnaces or high-end building HVAC systems.

The problem is that the question is how to tune PID, and these answers suggest not to use PID or to put a layer above it.

— Mike Triassi, business development manager, Optimation, www.optimation.us

LAMBDA TUNING

For the temperature applications, I can answer, having done this in 12 pharmaceutical or biotech plants. I believe a similar approach will help with the motion control applications. The PID tuning is part of a specialty that we call "control performance" in Emerson. It's part because you can only tune the controller after you make everything else right in the loop—the control valve or slave flow loop, the process sensor/transmitter and the controller configuration including choice of PID algorithm.

Instead of chasing PID coefficients for days, we use a synthesis method called lambda tuning. Our overall model of the loop has a model of the process and a model of the controller, wired as a feedback system. Synthesis means we know how we want the loop to perform; we can measure how the process responds to the controller output; and we have a model of the controller from the manufacturer's instructions. Then we only need a little math to back-calculate what parameters to use in the controller.

If the process is self-regulating (like some temperature processes), then lambda tuning means the loop will not oscillate and the temperature will make a first-order approach to the set point, with a time constant (lambda) that you choose. This aligns perfectly with your goal.

If the process is integrating (like some other temperature processes), then any standard PI or PID controller will give a temperature overshoot on set-point steps. Lambda tuning still means the loop will not oscillate, and you choose a response time (lambda). But to prevent the overshoot, you need a little more math to choose an alternate PID algorithm, a set-point filter or a helpful nonlinearity in the process or controller.

You can read my white paper on lambda tuning at www.controldesign.com/lambdatuning.

— Mark Coughran, senior process control consultant, Emerson Process Management, www.emersonprocess.com

AGGRESSIVE TUNING

Unfortunately, I do not have 3D (or three-axis, robotic) servo-motion-control experience and cannot provide guidance on that application. However, some of the issues posed are fairly common among customers.

There is no magic bullet for quick response over a wide operating envelop. End users have to consider the critical parameters. Is it OK to overshoot a process limit, or does that create too much risk? If overshooting is OK, then you can use more aggressive tuning; if there is too much safety or cost risk, then the loop should be de-tuned or have an over-damped response, characterized as a "conservative" response. If a loop is tuned aggressively, with an under-damped response, then it can easily exhibit the behavior described by the reader's remark, "if dynamic loads are involved, my PID loops become unstable and oscillate." I modified a basic tuning presentation I created with Nathan Lichti of Apache to provide simple visual guidance in tuning loops or in trial-and-error adjustments once standard methods are used for tuning. I believe this presentation (www.controldesign.com/pid-tuningbasics) will provide a good guide to understanding PID tuning basics.

— Gene Chen, DCS/safety systems product manager, North America Yokogawa of America, www.yokogawa.com/us

KNOW YOUR SYSTEM

You can't control what you don't understand unless one uses a lot of trial and error. Therefore, the first thing to do is identify the system by determining the general system type and estimating the parameters.

In this exercise, the engineer records how the system responds to stimulus. It is helpful to log the data in a .txt or .csv file with the time, SP (set point: the target or goal value of the process), PV (the process variable) and MV (the manipulated variable: the control output) listed in columns. This data can also be displayed and graphed using a spreadsheet.

Next, determine if you have a non-integrating (velocity or temperature) system or an integrating system (position or level). Some systems have one, two or more poles. Some of these poles are real, but if there are two or more there can be complex poles that will cause oscillations even when driven open-loop (with no feedback). Try to estimate the open-loop parameters by the response to some form of open-loop excitation. Each different type of open-loop system has a set of equations that can be used to calculate the closed-loop gains. Ackermann's formula is a good way to calculate closed-loop gains for motion systems. The internal model control (IMC) method is good at calculating gains for temperature control, if the plant parameters are known.

Ackermann's formula is basically a pole-placement algorithm. If all the poles are placed on the real axis in the z or s domain, there will be no overshoot or oscillations. Ackermann's formula calculates 1 gain for every pole in the open-loop system, plus one more if an integrator gain is used. For example, a simple motor-control system has a gain and a time constant. The time constant determines the location of the

open-loop system's single pole. However, if the motor is going to be used for position control, then the system has another pole for integrating the velocity to position. Finally the integrator in the PID adds another pole so a simple closed-loop position control system has three poles. Therefore, three gains (P, I and D gains) are required to place the three poles in the s plane.

There are limitations. The controller can only output a maximum of ± 100%. Low-resolution feedback and sample jitter can severely limit the usefulness of the derivative gain. There are also limitations on power. There may not be enough power to accelerate or decelerate as quickly as desired. This is easy to determine by looking at a trend plot. If the MV saturates during acceleration or deceleration, the acceleration or deceleration ramps must be made longer. Tuning will not help this. In fact while the control output is saturated (at its maximum value), the system will behave like an open loop system. As pointed out, using ramps (as opposed to simple on/off controls) makes tuning easier. Motion control systems should always use ramps to keep the control output from saturating. Another limiting factor is deadtime, which is the time between the control output changes and when the PV starts to respond. It is unrealistic to expect a temperature system to reach an SP within the longer of five times the dead time or five times the plant time constant without oscillating using just a simple PID.

In short, know your system's basic transfer function. Use graphs or trend plots to estimate the open loop parameters. Use the appropriate algorithm for calculating the closed loop gains so the closed loop poles are on or close to the real axis, and have realistic expectations on how fast the response can be.

— Peter Nachtwey, president, Delta Computer Systems, www.deltamotion.com

STEADY STATE

The nature of the control loop must be understood before tuning. Is it self-regulating or integrating? In a self-regulating process, the process variable will typically reach a steady state value for a given controller output, such as flow control. In an integrating process, the rate of change of the process variable will typically remain constant for a given controller output—for example, manipulating an inflow or outflow valve to a tank when the process variable is tank level. The two types of processes have different tuning rules.

The tuning goal must also be clear. Is it to reach set point as quickly as possible or to ensure there is no overshoot? Ensure that the tuning parameters selected align with the desired outcome. There is also tuning software available to assist in identifying processes and tuning PID controllers.

Whether tuning manually or with software, the process should be in a steady state prior to bumping the process. Also, the tuning procedure should be performed at or near

the expected process operating point. If the process operates over a wide range and is not linear, tuning parameters may need to be calculated for other operating ranges and the tuning parameters can be scheduled based on the current operating range.

— Paul Wentzell, global process technical consultant, Rockwell Automation, www.rockwellautomation.com

MODEL-FREE ADAPTIVE CONTROL

Since you are providing mission-critical systems, I suggest that you solve the PID tuning problems with a systematic approach. There are mainly three control methods that have achieved commercial success on the market: PID, model-based control (MBC) and model-free adaptive (MFA) control.

1. PID is a "one algorithm fits all" method. Inevitably, its capability is limited. In other words, PID is more like a general-purpose medicine, except there is no way for one medicine to effectively cure all illnesses. So, PID tuning difficulties can be fundamental, especially for those loops where PID won't work well, no matter how you tune it.
2. Model-based control is more like a "one algorithm fits one system" method. If the process dynamics do not change frequently and the process model can be well-maintained, model-based control can provide excellent performance. However, since it is so specialized and has a potentially high installation and maintenance cost, large deployment is difficult.
3. MFA control takes a "one algorithm solves one problem" approach. Each MFA controller is designed to solve a difficult control problem. For instance, the nonlinear MFA controller can control various types of extremely nonlinear processes in different industries and applications. Each MFA controller is like a specifically developed medicine to cure one specific ailment. For the user, it is an effective yet easy-to-use controller. Therefore, large deployment is more likely. For instance, the Siemens Apogee building automation system has embedded MFA controllers to control temperature, flow, pressure and humidity, with no controller tuning required. Nabors Industries use MFA controllers for its DrillSmart auto-drilling systems.

PID is used for a large variety of processes where no detailed process info is required (black box). Model-based control is well-suited for controlling a process where detailed knowledge is available (white box). MFA is suitable for controlling processes with qualitative process knowledge but no detailed process models are available (gray box). In addition, the process dynamics can have significant variations. Many industrial processes are gray boxes that have frequent load, fuel, and dynamic changes.

For your medical-device and motion-control applications, embedded MFA control software may be required, depending on your control system platform.

— Dr. George S. Cheng, CEO, CyboSoft and CyboEnergy, www.cybosoft.com.

INERTIA RATIO

When optimizing PID control loops, or even starting the process of creating a motion system that requires precise PID loop control, the most important starting point is to size the application appropriately, aiming for a 5:1 inertia ratio. As a result, tuning time can be drastically reduced, particularly when using a PC-based control system. Additionally, choosing a robust drive system and servo motors goes a long way toward simplifying the tuning process.

Other helpful features to look for in a control platform, in order to streamline PID tuning, are integrated analytics and monitoring tools. In conclusion, users should seek out system-integrated tools that provide more features and functionality in one place. One must also seriously consider the efficiencies and savings that can be generated from drive configuration and tuning via a central PC-based controller rather than via the individual drives.

— Matt Prellwitz, drives technology application specialist, Beckhoff Automation, www.beckhoff.com

CONTROLLER, PLANT AND PERFORMANCE

One topic mentioned was performing feedback control with a PLC. Be forewarned that there are a number of PLCs out there that have not been designed with real-time control/processing in mind. Some thought needs to be applied to how inputs and outputs are sampled and released and exchanged with the digital control algorithm. I/O processing needs to occur at well-timed intervals, or else you'll experience jitter effects that end up appearing like noise on your I/O. The timing is also important in terms of programming difference equations (filters, PID) whose coefficients can be calculated off-line and applied with predictable performance. Reconstruction of signals sampled on an irregular basis has been researched and studied, but best avoided when trying to do simple and more practical applications like what I think you are trying to do.

I recently saw a PLC that didn't bother to try to execute programs or process I/O at regular time intervals. It just ran programs as fast as it possibly could. Timing changed depending on paths taken in the program and also changed as the program grew or shrunk. Imagine what the result of integrals and differentials would look like if your time delta is constantly changing. I watched a colleague maintain the PID parameters as the program evolved. A lot of the "tuning" unfortunately had nothing to do with the dynamics of the plant or feedback controller; it was just dealing with fundamental processing issues.

I'm not saying all PLCs behave like this. Some do a good job with real-time processing. I only bring this up because this alone could be a big contributor to some of your struggles. Read up on these topics

and arm yourself with tough questions for your PLC supplier.

There were some examples provided for different types of systems to which controllers can be applied, which in control theory is commonly referred to as a "plant." A plant is a generic term for a dynamic system whose states (or process variables) you want to control. It has causal relationships between the inputs you drive and its resultant states that can be directly or indirectly measured and hopefully controlled. It can be a heating or cooling element applied to air. It could be an electric servo motor driving a robot or gantry axis. It could be an hydraulic valve that's driving position, speed or force output of an hydraulic cylinder. It's my opinion that not enough time and attention is spent understanding a plant. Too often, I've seen a PID controller arbitrarily applied to a plant, followed by lengthy trial-and-error tuning sessions. A little bit of attention to the plant model can make this process simpler and less time-consuming. I'll go through a few of the more important properties I think someone should understand about a plant.

First property is order. For a first order system, the control input influences the first derivative, or rate of change or the plant output. In a second order system, the control input influences the second derivative of the plant output. A temperature controller is a good example of first order (power applied to a heating element causes thermal energy to increase at some rate) and an electric servo is a good example of second order (current, or torque input to the motor accelerates the axis position). This is good to know because it can be mathematically shown that PID rarely makes sense for a first order system (PI is much more applicable; leave the D set to zero). In the second order case, PID is essential for stability purposes.

The next property is linearity. It's not always the case that 10% control input results in 10% rate of change of plant output and that 90% input equates to exactly 90% output rate of change. If this isn't the case, it's sometimes useful to understand this and compensate your PID with gain schedules, or just scale the output of the PID with lookup tables, so your PID output has as much of a linear relationship with plant output as possible. This will make your controller respond much more consistently with respect to different inputs.

Another property is dynamic response. If you apply 10% control input, and it takes a few seconds for the plant to start to change its outputs, then you are dealing with a plant that has low open-loop bandwidth with no potential for fast (but still stable) closed-loop response. A good rule of thumb is a plant needs to respond four to five times faster than the desired response of the closed-loop system. If the plant can't

meet these criteria, then you will have a lot of trouble achieving your control goals.

The last pointer I have on this is paying attention to units. If you can scale control inputs so they are using the same units as the plant outputs, then you'll find PID tuning to be fairly consistent from one application to another. Let's say the process variable is in x units. If I scale the control input to be in units of x units/second, then I know right away that a Kp of 2*PI radians (1Hz) will get me very close to a 1-second response.

There was mention of tracking performance, stability and robustness to changing loads. These are all important properties of a good controller and very much in line with different measures we use to specify and validate controllers. I'll mention right away that trying to score high marks on all of these topics is not easy to do with just a PID controller.

In terms of tracking, I'd suggest reading a little bit about feed-forward control. If you have some a *priori* knowledge of how to drive control inputs to achieve desired outputs, put that knowledge in your control algorithm and let the PID do the rest of the control for you. If you know x amount of current applied to a heater will eventually get you to a steady state fluid temperature, then just add that current to the output of the PID. The PID has less work to do and can now focus on compensating for disturbance inputs or load changes.

For stability, this needs to be proven by thorough testing. Step response testing is a very common method for testing and is no doubt useful. I like testing with dynamic inputs. Drive your PID input with a sine wave and vary the frequency once. If you see the output moving at a much larger amplitude than the input, you've found a resonance and potential area of instability. Another thing to watch for is when the output is moving at full amplitude of the input, but is lagging by a half wavelength or more; then your controller tuning lacks margin and could potentially go unstable.

In terms of robustness to part variation or load changes, this again can only be proven through testing. Pick extreme operating points to test your controller. If the controller is too sluggish at one operating point and marginally stable at the other extreme, this is a clear indicator that you will need more than a PID to control the system to desired specifications.

One last comment on PID coefficients: They work together to define the poles of your system defined by your closed-loop transfer function. The location of these poles tells a lot about how responsive, stable or unstable a system can behave. If you tune a Kp that gets you close to the responsiveness you need, but then start tuning a Ki to improve steady state error and then find the system starts to resonate a little bit, it might be worth your while to increase Kp rather than

back off on Ki. When you change one coefficient, it's likely you will need to scale the others, as well.

— Jason Kraft, engineering specialist, controls, Eaton, www.eaton.com

DEDICATED TO CONTROL

First, if the controller has an auto-configure and/or -tune feature, by all means use it. Today's tuning technology is greatly enhanced over just a few years ago and the auto-tune feature can automatically adapt to, say, a given motor-drive mechanism, without a decrease in performance in many applications; and, in other applications, it can get you to a working baseline much more quickly than starting from scratch. There are many different versions of PID control and PID topologies found in temperature controllers that are not normally the same as used for motion control. For the purposes of this answer, motion control examples are utilized.

It has been my experience that control loops needing fast dynamic response, such as motion control, require their own dedicated controller, whether a PLC is utilized or not; and, even with this, there are times that signal/feedback latency can have significant detrimental effects, depending on the components selected and/or how they were chosen to work together. Additionally if you can only view the outside loop (position-loop) response, you are basically in the dark as to what is happening within the internal control loops (current and velocity).

That being said, what can we do when an auto-tune feature is not good enough or not available?

Once the auto-tune feature is complete, methodically write down the values of the PID and feed-forward parameters, plus any auto-filtering parameters/values established by the auto-tuner.

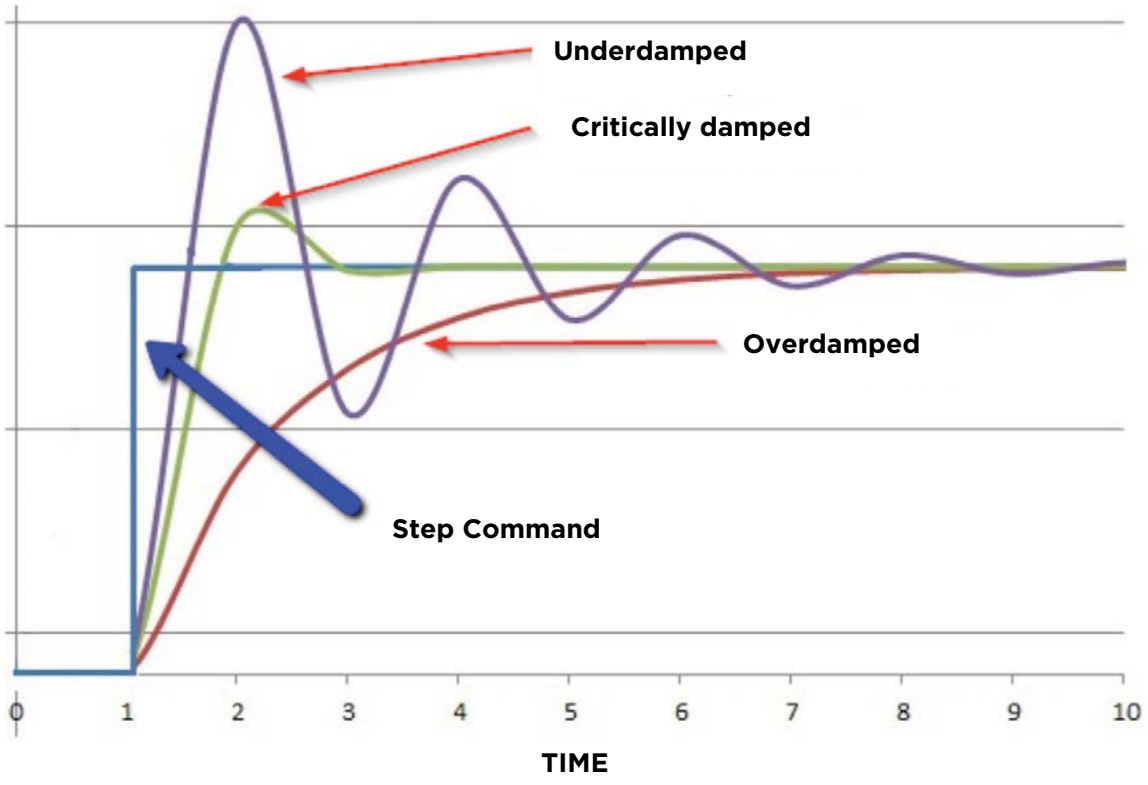
From this point it's important to remember what these specific parameters do.

Proportional (P) gain is the amount of control variable (CV) response action proportionally presented against the error signal determined by the difference of the measured process variable (PV) and the command/set point (SP). For unity gain, the PV range is set directly proportional to the CV range (Figure 6).

Integral (I) gain is the amount of control variable (CV) response action presented against the accumulated error over time (error/Δ_time); thus presenting a proportional CV output signal as a function of its gain setting depending on the amount of error and length of time the error is present. Integral gain presents phase lag into the system, which is important to remember when instability is an issue.

Derivative (D) gain is the amount of control variable (CV) response action presented against the rate of change of the error

PROCESS VARIABLE (PV) RESPONSE WITH DIFFERENT PROPORTIONAL GAINS



HOW MUCH DAMPING?

Figure 6: Proportional (P) gain is the amount of control variable (CV) response action proportionally presented against the error signal determined by the difference of the measured process variable (PV) and the command/set point (SP).

(Source: Kollmorgen)

(Δ_error/Δ_time); thus presenting a proportional CV output signal as a function of its gain setting depending on how fast the error is changing.

Feed forward gain is a non-error-based signal that jumps ahead with a predefined signal, based on a measured disturbance and/or command change. When utilized with a closed-loop system, the signal typically jumps over several hardware/algorithm control blocks, such as PID blocks, filters, amplifier or summing circuit/algorithm, and goes directly into the pre-amp stage driving the power devices, presenting a phase lead into the system. The feed forward signal presents power to the motor to counter the damping effect of the acceleration of mass and thus provides a pre-determined phase lead control response.

Signal filtering, whether utilized in the forward control path or feedback signal, is generally used to smooth out the over-

all system response by reducing electrical feedback noise, audible noise, high-frequency noise and system resonances/harmonics.

Even with a good understanding of how these control blocks work, for any specific controller it can become daunting to explain what adjustment needs to be made for a specific topology's desired response, as compared to some measured step response, due to the different interactions and interdependencies of the responding PID algorithms with a fixed or varying load. P, I and D algorithms can be designed in series, in parallel or in some combination of series and parallel, with each control-design technique presenting a different style CV response against a fixed or varying load, disturbance or command. Your tuning method can also vary as a function of the specifications of the process/work being accomplished, all presenting multiple ways and priorities to optimally tune a servo controlled load.

With the basic definitions in mind with your specific process, it may be reasonable to tweak some of your gains up and down slightly to get a feel for how the system response changes. If applicable, it is best to examine the vel-loop step input response first and separate, from the position loop, if there is a position loop. Methodically write down the values, record what happens and accordingly make slight changes in the gain settings.

If your auto-tuner has established filter settings within the control loops, let them be; establishing filter types and frequencies by manual examination is beyond the scope of the writ; and generally a good auto-tuner will be able to determine the needed filter settings.

If slight methodic gain tweaking does not get you within acceptable parameters, one way to proceed is to basically start from scratch except for the auto-tuner filter settings; let them be, at least for the first attempts.

A fresh start: For servo-motor systems we often talk in terms of three different control loops: I-loop (current loop), vel-loop (velocity loop) and position loop.

The I-loop is the innermost loop and fastest of the three. The I-loop is active when the controller is in a current/torque mode, velocity mode or position mode. The current gain setting is normally pre-calculated from such parameters as motor inductance and resistance. It will likely also have integrators to help to compensate for PWM dead time among other things. In most cases you will not need to adjust these settings, and, if one does, it typically should be done in conjunction with the manufacturer's post applications department / instructions. For example, when a servo-motor controller's I-loop is set too high, it will often display itself with a load noise coming from the motor that a

change in vel-loop gain cannot reconcile; and this instability may only manifest itself at specific currents and/or speeds. For this type issue, the I-loop gain setting can typically be reduced 10-15% without affecting the system performance.

The vel-loop is the second fastest of the three and for many the most troubling to compensate or fine-tune. Even if the system is going to be run in position-loop mode, the vel-loop needs to be properly compensated first for the system to act at optimum levels.

The vel-loop is active when the controller is in a velocity mode and position mode.

In order to compensate/tune the vel-loop, provide a step input to determine its step response; then from the CV velocity response, adjust and fine tune the vel-loop's proportional, integral and derivative gains. Too much velocity overshoot and audible noise—objectionable oscillations creating humming, buzzing—are signs that a gain limit has been reached; back it off slightly.

With the controller in its velocity mode, set all vel-loop gains—proportional, integral and derivative—to zero. Utilize filters as set by the auto-tuner; if there is no auto-tuner set filters to zero.

Repeatedly provide a velocity step input command to determine the velocity (CV) response, and increase the vel-loop's proportional gain until you have approximately 10-15% (no more than 15%) with an undershoot of approximately 3-5% (no more than 5%).

The step response may identify oscillations, resonance and/or harmonics that will need filtering or source identification to achieve desired response goal. This is beyond the scope of this brief write-up.

Repeatedly provide a velocity step input command to determine the velocity (CV) response, and increase the vel-loop's integral gain until any steady-state error has been reduced to zero.

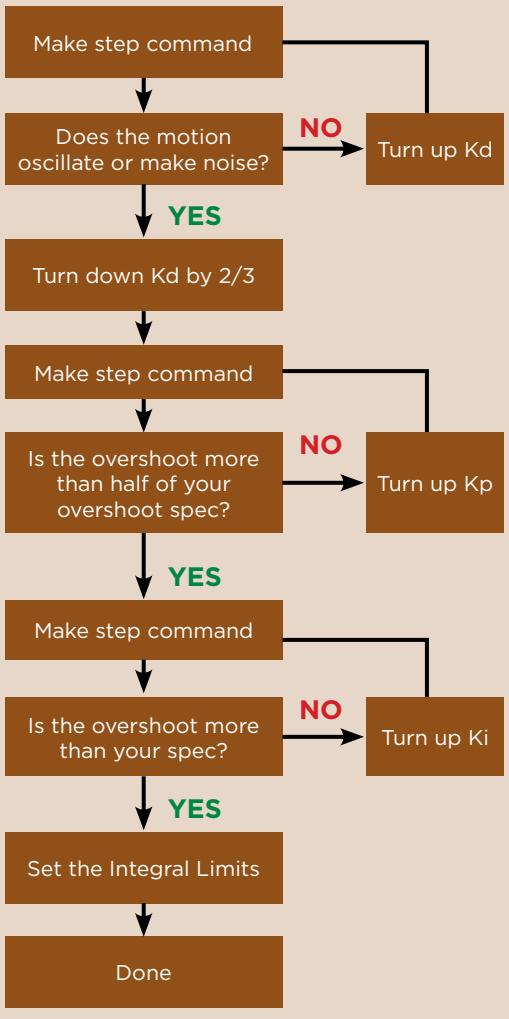
The Integral function presents phase lag to your system, which increases risk of instability; you do not want any more vel-loop integral gain than it takes to reduce the steady-state error within acceptable levels.

When the position-loop is to be closed, many engineers prefer to zero or minimize vel-loop integral gain and allow the position-loop integral gain to dominate, creating a stiffer system. My recommendation is to find the preferred setting for the vel-loop integral gain, even if the controller is to be used in position mode, for possible reference.

Repeatedly provide a velocity step input command to determine the velocity (CV) response, and increase the vel-loop's derivative gain until the desired high speed PV response is achieved.

If the response looks good, do not add in any derivative gain; if the response appears to be not fast enough, add as little derivative gain as possible to achieve your fast-response goal.

Though velocity feed forward is not a function of the closed-loop system, it is based on the measured amplitude of change (disturbance and/or command) to present a pre-defined (open-loop-type) response, similar to a derivative function; thus if your system has velocity feed forward capability, it may be important to consider its gain setting as a function of the load type. Even if only for experimentation, it may be desirable to reduce your velocity feed forward gain to possibly one-third or less of its perceived ideal gain setting—direct drive, fixed load—when there is compliance and/or backlash within the system. Remember that a positive phase shift is presented by the velocity feed forward gain; typically you don't want to add any more positive feed forward than it takes to offset phase lag from the integrator, and filters if required, over the speed range of the axis.



TUNING METHOD
Figure 7: Kd = Derivative Gain; Kp = Proportional Gain, Ki = Integral Gain.
(Source: Kollmorgen)

It is most desirable to have your feed forward parameter and derivative gain set before tuning proportional and integral gains based on a step response. However, unless the user has significant experience with a particular controller and subject PV load, it is unlikely the values will be set close enough on the first try. When either the feed forward parameter and/or derivative gain are changed/updated, the proportional and integral gains need to be retuned for their optimum performance (Figure 7).

When using this tuning method, be sure that the command output is not clipping the step command. If the command output does clip the step command, you have a nonlinear system. If the system is nonlinear,

the generalizations made on this page may not apply. If you are determined to tune despite the nonlinearities, make sure that there is a repeatable way to generate the nonlinearities, as well as the ability to try many different scenarios.

The position-loop when utilized is the slowest of the three. The position-loop is only active when the controller is in its position Mode. The position-loop's capability is a function of the vel-loop compensation. A bad vel-loop = a non-optimized position-loop.

In order to tune the position-loop, present repeating positon commands under a defined move and adjust for position-loop proportional gain for the desired response—quick response with some overshoot before settling, no-overshoot (overdamped), minimum settling time—based on the axis' criteria/specifications. If the axis' criteria demands minimal position error (PE) while holding position or during a specific motion profile segment, one may wish to try introducing integral gain to the position-loop. This will often mean the removal or significant reduction of the vel-loop integral gain. Position-loop derivative gain is not typically utilized.

After tuning, if your PE looks like noise, you know you have a good compensation for most applications; but this is not always possible for many reasons.

— Hurley Gill, senior systems and applications engineer, Kollmorgen, www.kollmorgen.com

VALIDATION

Tuning a PID control can be quite challenging, especially when you take into account load changes and nonlinearities of your system. Basically, the optimal tuning of a PID control requires not only a deeper knowledge of the system in question, but also a more in-depth knowledge of existent techniques for PID tuning. However, there are some heuristics that you can use to better define PID tuning.

Know your system: The first thing you have to do before controlling a system is know how it should behave. This can start by verifying the existent literature that can provide you with mathematical models of your system. This process can be very time-consuming, but it will give you the insight of different nonlinearities that could influence the response, especially when trying to achieve optimal control. You don't need to model every single detail. For control process, we are really interested around the frequency that could invert the feedback loop signal (also known as phase margin crossover frequency), and there is where you really have to be accurate. So, simple approximations for first-order model with delay could be enough to model your system. Sometimes, your model is a series of rules or an input-output mapping that was obtained empirically, but you know how it behaves.

Validate the model of your system: After you have an understanding of how your

system works and hopefully you have qualified parameters, you must have the properties quantified. Then you must do experiments in real life to be sure that you know the parameters' variation. Also, you can design experiments to quantify missing parameters to better calibrate your model. The important aspect of it is that you know how the system behaves and it is based on reality. Also, it is important to test in different operating points since a furnace could have different behavior from near ambient temperature than in high temperature.

Apply some tuning technique for PID: After you define your system, you should try to tune your PID. You have a dynamic model of your system. This consists of using simulation tools integrated with optimization to obtain the best PID parameters based on certain criteria. If your model really reflects reality, the output of this operation could give a very good approximation of the optimum parameters, and, since it is off-line, it can be pretty fast. However, this requires a high-fidelity model, which can very costly, too. Another option is to use empirical techniques that were developed for some specific situations. The most famous is Ziegler-Nichols method, which is used as a benchmark for any other type of techniques. However, this was heavily tailored to process control applications where the s-shaped response was predominant. However, if you are interested in servo control for motion, then another option is to use the Chien-Hrones-Reswick technique, which allows you to optimize for servo operations.

Validate the closed-loop control system for all ranges: This is the most important task of the process. This encompasses validating that the parameters obtained by an auto-tuning feature or by design will work on all conditions of operation of your system. This also requires that you have a supervisory system that will verify if any special dangerous condition could be reached and have a sequence to shut down the system. A good controls developer will define this operation first, before you start to tune a controller. It might even need to be done manually. Then you can see if your system would work in all conditions that were defined. Notice that this validation can also be developed in a hardware-in-the-loop scenario, where your high-fidelity model can perform the extreme conditions while an experiment can validate if your controller is running correctly for some operating points.

Now, this is an iterative process where you need to revisit previous steps until you converge into a proper result. Some vendors have solved that problem for you for specific applications like process control using PLC applications or motor or motion control for motor drivers.

— Alex Barp, senior software engineer, LabView R&D—dynamic control and simulation, National Instruments, www.ni.com