Tuning a PID Controller

Guillermo J. Costa

Nomenclature

D_{out}	=	Derivative contribution parameter
$e\ (t)$	=	Error term with respect to time
I_{out}	=	Integral contribution parameter
K_d	=	Derivative gain
K_i	=	Integral gain
K_p	=	Proportional gain
K_u	=	Ultimate gain
L	=	Delay time, Ziegler-Nichols reaction curve method
P_{out}	=	Gain contribution parameter
T	=	Time constant, Ziegler-Nichols reaction curve method
T_u	=	Ultimate period
$V_m\ (t)$	=	Measured variable value with respect to time

Management Summary

This paper introduces the basic fundamentals of proportional-integral-derivative (PID) control theory, and provides a brief overview of control theory and the characteristics of each of the PID control loops. Because the reader is not expected to have a background in control theory, only the basic fundamentals are covered. Several methods for tuning a PID controller are given, along with some disadvantages and limitations of this type of control.

Introduction

The PID controller is a feedback mechanism widely used in a variety of applications. The controller calculates an "error" that is the difference between a measured process variable and the desired set-point value needed by the application. PID controllers will attempt to minimize the process error by continually adjusting the inputs. Although this is a powerful tool, the controllers must be correctly tuned if they are to be effective. Additionally, the limitations of a PID controller should be recognized in order to ensure that they are not used in applications that cannot make use of their unique advantages. This article covers the basics of PID controllers, as well as several methods for tuning them.

The most common question asked about the topic of PID controllers is, "Why learn to tune them?" The answer is simple. PID controllers are literally everywhere in industrial applications. For many applications, PID controllers are the optimum choice and will simply outperform almost any other control option. This is why they are currently used in over 95% of closed-loop processes worldwide (Ref. 1), governing everything from temperatures, flow rates, mixing rates, chemical compositions and pressures in a limitless number of applications. PID controllers can also be tuned by operators who do not possess a strong background in differential equations, electrical engineering or modern control theory; this grants PID controllers a very powerful ability to drastically change a given process (called a "plant model") with a system that is very simple and robust.

Basics of Control Theory

A common example of a control system is a person adjusting the temperature of water coming from a faucet. This involves the mixing of two process streams—hot and cold water—which is followed by a person touching the water stream, measuring the process variable to gauge its temperature. Based upon this feedback, the person adjusts the amount of hot or cold water fed into the faucet until a desired temperature—the set-point value—is reached. However, this set-point value isn't reached immediately; there is usually an error value (*e*) between the measurement of the process variable and its set-point value. By measuring the process variable and calculating the error, the person will decide to change the positions of the hot and cold valves—the measured variables—by a certain amount until the water temperature resolves to its set-point value.

If the person only adjusts the position of the hot water valve, this is an example of proportional control. If the hot water does not arrive quickly enough, the person may open the hot water valve by an increasing amount as time goes by; this is an example of an integral control. By only using the proportional and integral methods (a PI controller), the water is likely to oscillate wildly between too hot and too cold because the valves are being adjusted too quickly and the process is overshooting the set-point. In order to dampen future oscillations, the person may wish to adjust the positions of the water valves more gradually, leading to a derivative control method.

This simple example is a wonderful demonstration of how a PID works. A PID controller involves three separate system parameters:

- ***Proportional (sometimes called the "gain")***: determines the reaction to the current error
- ***Integral:*** calculates the system reaction based on the sum of recent errors
- ***Derivative:*** calculates the rate at which the system error has been changing

The weighted sum of these three values is used to adjust a process by adjusting a control element, which could literally be nearly anything within the process. For instance, flow rates into or out of a mixing tank could be controlled through the position of a valve (as with the tap water example), or the output of a heating element could be controlled via its power supply. These three summed terms constitute the measured variable, i.e.—the aspect of the application that one is trying to manipulate:

$$V_m(t) = P_{out} + I_{out} + D_{out} \tag{1}$$

Where:

P_{out}, I_{out}, and D_{out} are the output contributions of each of the three PID parameters. These three outputs are given by their respective parameter loops, which are:

$$P_{out} = K_p e(t) \tag{2a}$$

$$I_{out} = K_i \int_0^t e(t)dt \tag{2b}$$

$$D_{out} = K_d \frac{d}{dt} e(t) \tag{2c}$$

Thus, the PID algorithm from Equation 1 can be rewritten in its final form as:

$$V_m(t) = K_p\, e(t) + K_i \int_0^t e(t)dt + Kd \frac{d}{dt} e(t) \tag{3}$$

As may be seen, there are quite a few options here for tuning the controller. Each of the characteristics of the three loops is discussed below.

Proportional (gain) loop. The purpose of the proportional gain is to create a change to the system's output that is directly proportional to the system's current error value. Stated another way, a gain can be thought of as an amplifier to the controller, as it only serves to multiply the current error value by a given gain value. A large gain value will yield a large change in a system's output for a given error, and thus gain can be used to amplify the speed with which a controller reacts to a certain state condition. However, if the gain is too large, the system can become unstable very quickly; conversely, if the gain value is too small, the controller will have a subsequently small response to an error value. This latter condition will result in a less-sensitive controller, which may not respond correctly to errors or disturbances.

In an ideal state—i.e., free of any disturbances—a purely proportional control system will not settle at the set-point value, but will retain a steady error that is a function of the proportional and process gain. However, despite the presence of the steady-state offset, it is common practice to design control systems wherein the greatest amount of control response is provided by the controller's proportional gain. An example of this steady-state error is shown in Figure 1.

Integral (reset) loop. The value contributed from the integral loop is proportional to both the magnitude and duration of the error. Summing the recent error values over time (integrating the error) gives the offset value that should have been previously corrected. This accumulated-error value is then multiplied by the integral gain (which defines the magnitude of the contribution of the integral loop) and added to the

continued

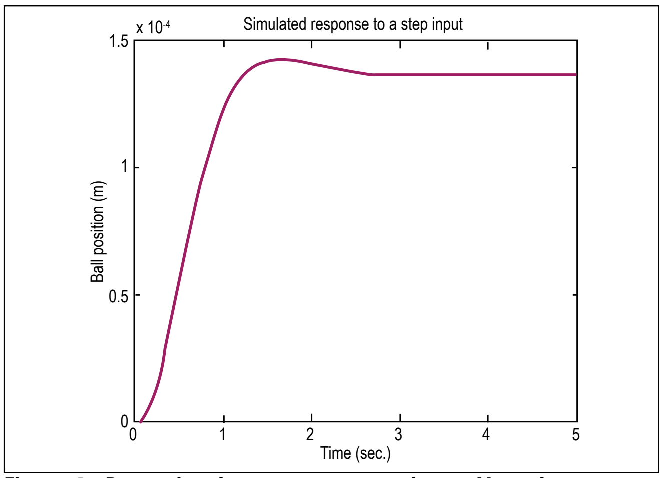

Figure 1—Proportional response to step input. Note the presence of a steady-state error value. (Image copyright Carnegie Mellon University)

controller output. When added to the proportional term, the integral loop accelerates the response of the process towards the set-point value and eliminates the residual steady-state error of a proportional-only controller. The integral loop is only responding to the summation of recent errors, however, which will cause the response to overshoot the set-point value and thus create an error in the opposite direction. Left alone, this PI controller may eventually settle on the set-point value over time, but there are many applications—such as stability control systems in aircraft—where rapidly settling upon the set-point value without oscillation is both desirable and necessary. Figure 2 shows the effects of adding an integral loop to a proportional controller. Note how changing the value of the integral gain affects the response of the system. Although a PI controller will not resolve to a steady-state error (as a proportional-only will), the amount of overshoot is directly related to the value of the integral gain. Notice in Figure 2 that the highest value of integral gain gave the fastest response to the step input (as evidenced by the steep slope of K_i = 2, relative to the other values), but also required the most amount of oscillations and the longest amount of time to resolve to the set-point value. By contrast, the red line of K_i = 0.5 has the slowest response time of the three options, but notice that it resolves to the set-point value with no noticeable overshoot.

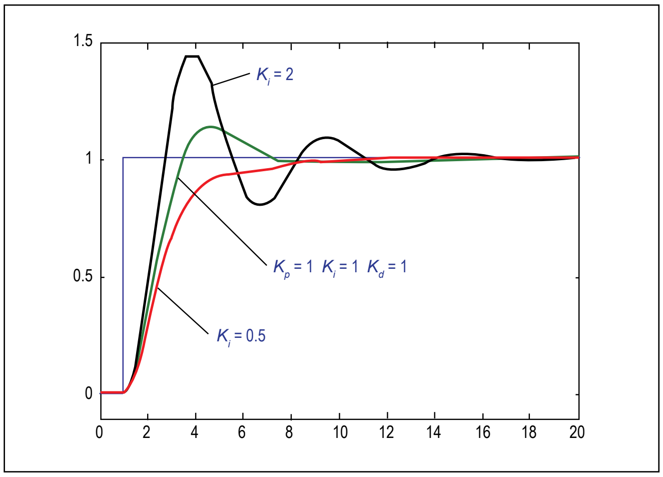

Figure 2—Controller response to step input with proportional and derivative values held constant. (Image copyright Wikipedia)

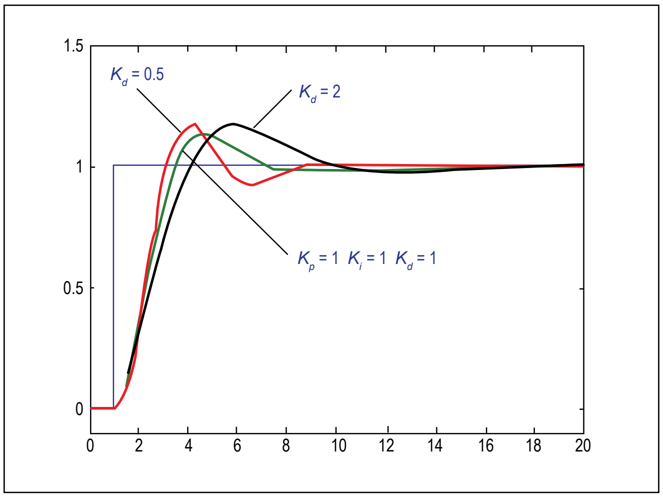

Figure 3— Controller response to step input with proportional and integral values held constant. (Image copyright Wikipedia)

Which response is "best" for a given application will of course depend on the application in question, but it is common practice to limit the number of response oscillations while still maintaining an acceptable response time. This is also done via the derivative gain, as discussed below.

Derivative (rate) loop. With a PI control, the system is able to settle to its set-point value through the use of a steady-state proportional response and the summation of past errors. But how fast have those previous errors been changing with respect to time? In Figure 2, the rate at which the errors change is relatively constant—especially with K_i equal to 2. To increase response time and minimize errors, a term is needed to calculate the rate at which the error term is changing. This is done through a derivative loop, sometimes called a "rate loop."

The derivative loop calculates the rate at which the error is changing by calculating the slope of the error. In essence, this is done by calculating the change in error (rise) over time (run)—the first derivative of the error function. This value is multiplied by a derivative gain K_d to obtain the derivative contribution to the system. As with the proportional and integral loops, the derivative gain can have a great impact on the system's response (Fig. 3). The derivative loop controls the rate at which the controller's response overshoots a given input value—produced by the proportional and integral loops—and is most noticeable when the process variable is close to the set-point. However, derivative loops amplify noise and are thus very sensitive to noise in the error term. For this reason, it is best to use attenuation filters with derivative loops, lest the presence of noise combined with a high value of derivative gain drive the system to instability. Note in Figure 3 that the behavior of the derivative term relative to its gain is the direct opposite of the integral term's response to an identical gain value.

Loop Tuning

Tuning a PID controller involves the control of four variables:

- ***Rise time***: the amount of time necessary for the system's initial output to rise past 90% of its desired value
- ***Overshoot***: the amount by which the initial response exceeds the set-point value
- ***Resolving time***: the amount of time required by the system to converge to the set-point value.
- ***Steady-state error***: the measured difference between the system output and the set-point value

The goal of a PID controller is to take an input value and maintain it at a given set-point over time. But if the values for the three loops of a PID controller are chosen incorrectly, the system will become unstable through any one of a number of failure modes. Typically, these involve an output that diverges—with or without oscillation—and is limited by the physical characteristics of the control mechanisms, including actuators breaking, sensors and encoders burning out, etc. The process of tuning a controller involves adjusting its control parameters—proportional band, integral gain and

Table 1—Three Typical Methods for Tuning a PID Controller.

Method	Advantages	Disadvantages
Manual	No math required; online options available	Requires experience in controll tuning
Ziegler-Nichols	Proven method; online options available	Some process upsetting involved; can be a very aggressive tuning method
Software	Consistent tuning options available; multiple valve and sensor inputs can be simulated and tested before applying to application	Acquisition costs of software (such as *MATLAB*) can be prohibitive for some organizations; software training required

Table 2—Effects of PID Tuning on System.

Variable Change	Rise Time	Overshoot	Resolving Time	Steady-State Error Change	System Stability
Increase K_p	Decrease	Increase	Small Decrease	Decrease	Decrease
Increase K_i	Small Decrease	Increase	Increase	Large Increase	Decrease
Increase K_d	Small Decrease	Decrease	Decrease	Minor effect	Increase for small values of K_d
Decrease K_p	Increase	Decrease	Small Increase	Increase	Increase
Decrease K_i	Small Decrease	Decrease	Decrease	Large Decrease	Increase

derivative gain—in response to a given input until the desired response is attained. This desired response is almost entirely application-driven. For instance, a controller must not allow any overshoot or oscillation if such things would create a hazardous condition within the application (and would yield response graphs similar to the red line in Figure 2). Other applications are inherently non-linear, rendering parameters that are ideal at full-load and maximum-RPM conditions undesirable when starting from zero-load conditions.

There are, generally speaking, three main methods of tuning a PID controller (Table 1).

The most important aspect to remember about control tuning is that it is a bit of an art form, requiring training and practice. Some knowledge of control theory is required—which is why it was introduced earlier in this paper's Basics of Control Theory section—as well as a systems-level understanding of the process in question. For instance, a large change in response to a small error results in a high-gain controller and leads to overshoot. Combining this with the oscillations introduced by an integral loop would result in the system oscillating about the set-point—rather than reaching it—with the system responding as a decaying, constant or increasing sinusoid. These determine the stability of the system, i.e.—stable, marginally stable or unstable, respectively. Initially, this concept may be difficult to grasp, although we humans "tune" our own control processes automatically. Recalling the tap water example, the person is able to learn from past actions, and so does not have to "oscillate" around the desired temperature of the water because a human being is a form of adaptive controller. A simple PID controller, however, does not have this ability to learn from process history and thus must be tuned correctly.

Before deciding on a tuning strategy, it is essential to understand how changing the gain, integral and derivative loops will affect the system as a whole. Table 2 shows the effects (Ref. 2) that tuning these loops independently have on the behavior of the system.

It should be noted that the philosophy of increasing derivative gain to increase system stability is a common belief, but real-world applications may behave in a fashion contrary to this assumption if there is a transport delay present (Ref. 3). This may lead some users to exclude the derivative term entirely from their control system, thus denying themselves a powerful tool in the design of their control system.

Manual tuning. Manual tuning is best used when a system must remain online during the tuning process. The four-step process is as follows:

- Set K_i and K_d to zero
- Increase K_p until the loop output begins to oscillate
- Reduce K_p to one-half of this value to obtain a quarter-wave decay
- Increase K_i to adjust the behavior of the offset so that the system will resolve in an acceptable amount of time (how much resolving time is acceptable will be governed by the process in question)

Note that increasing the integral gain by too great an amount will cause system instability (Table 2). The derivative gain should then be adjusted until the system resolves to its set-point value with acceptable alacrity after experiencing

continued

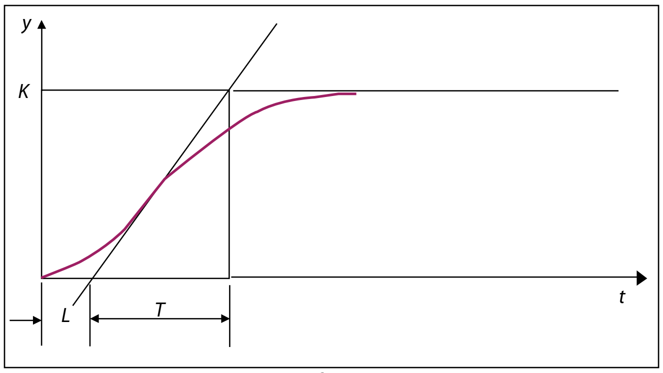

Figure 4—Reaction curve used for Ziegler-Nichols tuning. (Image

Table 3—Ziegler-Nichols Turning Values.

Control Type	K_p	K_i	K_d
P	$0.5\ K_u$	-	-
PI	$0.45\ K_u$	$\frac{1.2K_p}{T_u}$	-
PID	$0.6\ K_u$	$\frac{2K_p}{T_u}$	$\frac{K_p T_u}{8}$

Table 4—Ziegler-Nichols Turning Values: Reaction Curve Method.

Controller Type	K_p	K_i	K_d
P	$\frac{T}{L}$	-	-
PI	$0.9\ \frac{T}{L}$	$0.27\ \frac{T}{L^2}$	-
PID	$1.2\ \frac{T}{L}$	$0.6\ \frac{T}{L^2}$	$0.6T$

a load disturbance. This is simulated with a step doublet or "stick rap"—a step input from 0 to one, followed by a step input from one to 0—or with the sinusoidal or ramp input equivalents. Note that a fast PID loop will usually require a slight overshoot to resolve to the set-point more quickly. But if the system cannot accept an overshoot, an over-damped system will be required. In these instances the K_p value will be less than half of the value causing oscillation.

Ziegler-Nichols tuning. The Ziegler-Nichols tuning method is a very powerful way to resolve a system to its set-point value while circumventing a great deal of the mathematical calculations required to find an initial estimation of the PID values. This is especially useful when the system is unknown or when creating state matrices for the system is impractical or impossible. As with manual tuning, with Ziegler-Nichols tuning the integral and derivative gain values are first set to zero. The proportional gain is then increased from zero until the system reaches an oscillatory state, as above. This proportional gain value should be marked K_u, or ultimate gain. The system's oscillatory period at this gain value should also be marked T_u, or ultimate period. These two ultimate values are then used to set the proportional, integral and derivative gain values (Table 3; Ref. 4).

There are, however, limitations to Ziegler-Nichols tuning. It will permit some fluctuation in the controller response as long as each successive oscillation peak is no more than one-fourth the amplitude of the previous peak (Ref. 5)—or, the so-called, "quarter-wave decay." Applications requiring less fluctuation or a faster resolving time will require further tuning.

A second Ziegler-Nichols tuning method is used for plant models with step responses resembling an S-shaped curve (or "reaction curve"), with no overshoot. This is ideally suited for processes that cannot tolerate overshoot or oscillations. A typical reaction curve is shown in Figure 4. The delay time L and constant time T are found by drawing a tangent line to the reaction curve through its inflection point $\left(\frac{d^2y}{dx^2} = 0\right)$ and finding the intersection points with the time axis and the set-point line. Once these intercepts are determined, the values from Table 3 are recalculated (Table 4; Ref. 6).

The parameters in Table 4 will give a system response with an overshoot of approximately 25%, and the system will resolve to the set-point value within polynomial time (Ref. 7).

Software tuning. As it has with most other aspects of life, technology has rendered a great many number of control tuning methods irrelevant. A very large number of modern facilities forego tuning their controllers using the manual calculation methods mentioned previously. Rather, tuning and optimization software are used to ensure that optimum results are obtained in short order. Of course, for some systems—such as those with response times measured in minutes or hours—mathematical tuning is still recommended, as tuning by pure trial-and-error can literally take hours or days. *MATLAB* and *SimuLink* are the most common tools used to design and tune control systems, and they have found widespread use in a variety of industries. Other software packages such as *PIDeasy*, *AdvaControl Tuner*, *IMCTune* and others can often produce optimal responses from either online or offline inputs, and are plug-and-play ready—often with no need of subsequent controller refinements. Many of the features of PID tuning software are also designed directly into the hardware of the controller, most often from the "Big Four" of control vendors—ABB, Honeywell, Foxboro and Yokogawa. Because of the number of variables involved in software tuning, it is recommended that it be done on a case-by-case basis.

Limitations of PID Control

Although a PID controller provides an optimum solution to many processes, it is not a panacea for all control problems that may be encountered. This is especially true for processes with ramp-style changes in set-point values or slow disturbances (Ref. 8). PID controllers can also perform poorly when the gain values must be greatly reduced in order to prevent a constant oscillation—or "hunting"—about the set-point value. Furthermore, PID controllers are linear and so care must be taken when using them with inherently nonlinear systems—i.e., systems that do not satisfy the superposition principle or systems with an output that is not proportional to its input, such as air handling and mixing applications.

For nonlinear systems, gain scheduling—where utilizing a family of linear controllers that are independently activated based upon the values of scheduling variables determines the current operating region of the system—is most often used. Which scheduling variables are used will depend on the system in question. For example, a flight control system on an aircraft might use altitude and true airspeed as its scheduling variables, whereas an air handling application might use mass flow rate and impeller RPM. Nonlinear systems might be controllable with linear control systems if enough data and a sufficiently high sampling rate are known. Oftentimes, however, the use of gain scheduling may be more cost-effective.

Feed-forward control is found in a number of applications, including perceptron (*Ed.'s note: a binary classifier that maps its input x—a real-valued vector—to an output value f (x)—a single binary value—across the matrix) and long distance telephony (L-carrier transmission system of the 1970s*). Feed-forward control can also be used to improve the performance of a PID controller if certain qualities about the system are known beforehand and can be fed forward into the PID controller. This feed-forward value can greatly impact the performance of the controller; best of all, because feed-forward input is not affected by the feedback of the system, the feed-forward value can never cause the control system to oscillate, thus improving controller response and overall system stability.

Because the derivative loop is susceptible to process noise, it is also important to employ low-pass filters, if needed. However, the use of low-pass filters with derivative control can result in one filter negating the effect of another. For this reason, proper instrumentation or the use of a median filter may be a better option for improving both filter efficiency and overall performance of the controller (Ref. 9). Additionally, the differential loop can be turned off completely—$K_d = 0$—thereby using the PID controller as a PI controller. Note that this may require retuning the proportional and integral loops by utilizing one of the methods discussed in the previous Loop Tuning section.

Conclusion

PID controllers are a widespread control solution due to their simple architecture, generally acceptable control performance and ease of use. Unlike other control options, PID controllers do not require the user to have an extensive background in mathematics, control theory or electrical engineering to understand them. They are found in a wide variety of applications, and if properly tuned will outperform almost any other control option. It is in tuning the controllers that the greatest gains in performance may be found. A wide variety of tuning methods exist, although of the three discussed in this article, the Ziegler-Nichols method provides the most effective "quick- and-dirty" approach to tuning a controller. Software tuning has several advantages over the Ziegler-Nichols method, including the ability to run multiple iterations of tuning variables through process simulations to ensure optimum performance before the control logic of the process is updated. However, this type of tuning requires knowledge of the system's state properties and extensive knowledge of the software in use, with the latter necessitating acquisition and training costs that the organization might not be able to justify. In contrast, the Ziegler-Nichols method requires very little training or specialized knowledge beyond basic algebra, and offers results that are acceptable for the majority of applications. The Ziegler-Nichols method is also advantageous to use when the state properties of the system are unknown or in situations when the determination of these state properties is impossible or impractical.

As mentioned, the Ziegler-Nichols method is not without its disadvantages. Systems that require a very fast rise time and/or zero overshoot require a response other than quarter-wave decay, and as such cannot be tuned with the Ziegler-Nichols methods. Ultimately, it is imperative that the user have a clear understanding of the requirements of the system, and to select the appropriate tuning method as it applies to their own, unique needs.

References

1. Astrom, K.J. and T.H. Hagglund. "New Tuning Methods for PID Controllers," *Proceedings from the 3rd European Control Conference*, 1995.
2. Ang, K.H., G.C.Y. Chong and Y. Li. "PID Control System Analysis, Design and Technology," *IEEE Transactions on Control Systems Technology* 13 (4), 2005, pp. 559–576.
3. Li, Feng et al. "PIDeasy and Automated Generation of Optimal PID Controllers," *Proceedings from the 3rd Asia-Pacific Conference of Control and Measurement,* Dunhuang, P.R. China, 1998, pp. 29–33.
4. Co, Tomas B. "Ziegler Nichols Method," Michigan Technological University Department of Chemical Engineering Website, URL: *http://www.chem.mtu.edu/~tbco/cm416/zn.html* (cited February 3, 2010).
5. Van Doren, Vance J. "Loop Tuning Fundamentals," *Control Engineering* Website, URL: *http://www.controleng.com/article/268148-Loop_Tuning_Fundamentals.php* (cited February 3, 2010).
6. Zhong, J. "PID Controller Tuning: A Short Tutorial" (class lesson), Purdue University, 2006.
7. Ibid.
8. Sung, S. W. and In-Beum Lee. "Limitations and Countermeasures of PID Controllers," Department of Chemical Engineering, Pohang University of Science and Technology, Pohang, Korea, 1996.
9. Ang, K.H., G.C.Y. Chong and Y. Li. "PID Control System Analysis, Design and Technology," *IEEE Trans Control Systems Tech*, 2005, 13 (4), URL: *http://eprints.gla.ac.uk/3817/1/IEEE3.pdf* [cited 2007].

Guillermo "Willie" Costa *has been with L.A.-based Mechanical Drives & Belting for the past eight years, leading most of the company's lean, Six-Sigma and marketing initiatives. He has designed a number of the fabrication tools and equipment used at the company—mostly job-related—including cutting templates and tables; he also ran the company's returns and repair operations for two years. He presently attends Cal Poly Pomona, studying aerospace engineering, and has completed three internships for NASA and some contract design work of his own. In February of 2011, Costa left Mechanical Drives & Belting to found Trinity Aeromotive, an aerospace and power transmission engineering design and consulting organization.*