Tuning Servomotors

Chuck Lewin, CEO of Performance Motion Devices

Introduction

To paraphrase an adage, there are two types of motion control engineers, those that are comfortable tuning a servo loop, and those that aren't. And if you are one of those engineers that aren't comfortable, you in turn, have two options. The first is to use a non-servo device such as a step motor, and the second is to *get* comfortable!

Whether you are a relative novice, or an experienced hand with servo tuning, this article will help. It provides an overview of PID (proportional, integral, derivative) based servo loops, and introduces two standard manual tuning methods that work well for a large variety of systems. It will also provide an introduction to the increasingly popular technique of auto-tuning, which, despite the name, isn't necessarily as automatic is it may seem. Finally, we will look at advanced servo techniques such as feed-forward and frequency domain bi-quad filtering.

Servoing up an ace

The two servomotors that engineers most commonly use for positioning are the DC servomotor, which uses mechanical brushes to commute the motor, and the brushless DC motor, also known as the PM (permanent magnet) brushless motor, which is commutated electronically by external circuitry.

Unlike step motors, which move in discrete position steps, servomotors have no built-in sense of where they are, and thus require a feedback device such as a quadrature encoder to provide position information. The servo loop, also called a compensator, has the job of keeping the servomotor at the desired position. It does this by comparing the desired position at any given moment with the actual motor position from the feedback device, and applying corrective motor commands. The better the servo loop performs, the more accurately the motor will track the desired position under a variety of loads and motion profiles.

It's all PID

Theorists and engineers have developed a number of servo compensation schemes over the years, but the overwhelming favorite is the PID loop. Several different implementations of the PID loop exist however, and it is not uncommon for different vendors to use different approaches.

Broadly speaking, PID controllers fall into two groups; the first is the "PID position loop," and the second is the "cascaded position/velocity loop." Figures 1A and 1B, on the following page, provide an overview of these two schemes.

The more common PID position loop requires us to determine three values, the position loop gain, (Kp) the Integral gain (Ki) and the derivative gain (Kd). Even for this "basic" servo system however, modern motion vendors provide a bevy of additional options. The most common of these are an integrator limit, feed-forward gains, motor bias, and frequency-domain filtering such as notch filters or band-pass filters. Several of these concepts will be discussed in later sections of this article.

Cascaded position/velocity loops are tuned inside-out, and either four or five parameters are set by the user. The inner velocity loop (usually a PI controller) is tuned first, and then the outer position loop (generally either a PI or PID controller) is tuned. While we will not focus on the cascaded position/velocity loop in this article, it shouldn't be hard to adapt some of the techniques presented to this type of loop. Note also that if you are using an external amplifier that provides velocity control, you already are, in effect, utilizing a cascaded position velocity loop. Generally speaking the type of amplifier you are using has an important effect on position loop tuning, so you should make sure that the complete system controller, including whatever control loops may exist in the amplifier, are taken into account.

Equations 1 & 2 below provide the basic form of the PID filter equation in both the continuous time domain and the discrete time domain. For anyone using a DSP or microprocessor to construct a servo loop, only the discrete time form applies. The continuous form is used in system modeling and analysis, and can be used as a representation of the discrete time form as long as the sampling rate is high compared to the system bandwidth.

Equation 1. Continuous time form PID

$$Output\ (t) = K_p E(t) + K_i \int_0^+ E(t) + K_d \frac{dE(t)}{dt}$$

where $E(t)$ is the position error at time t

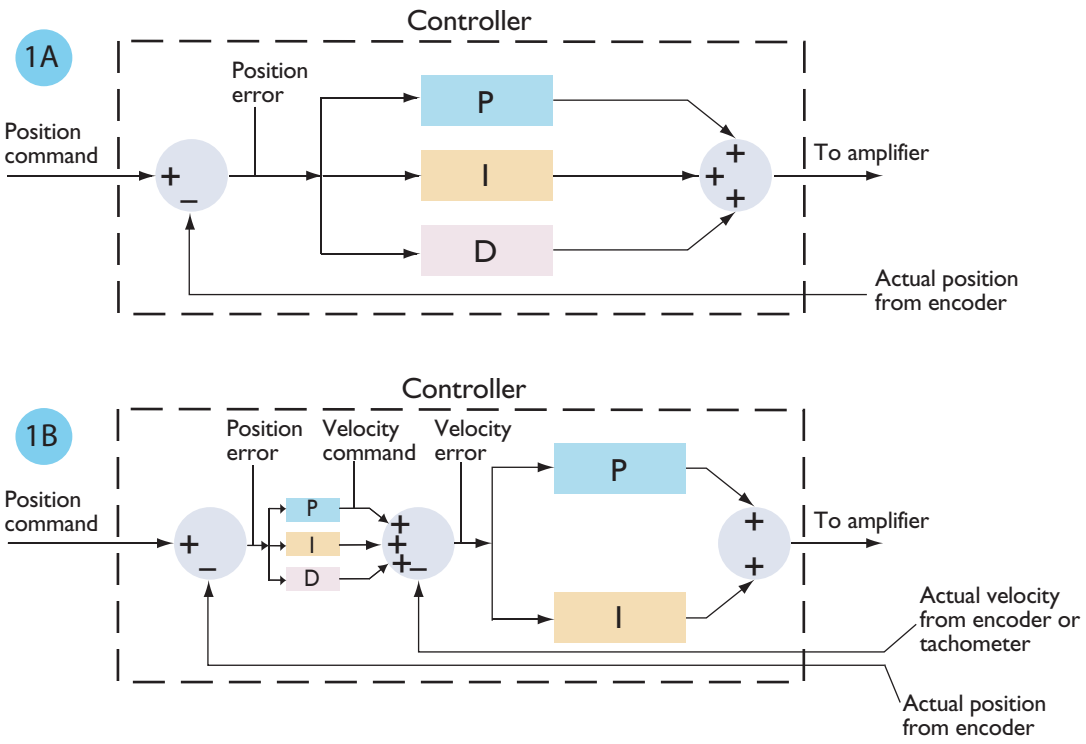
Equation 2. Discrete time form PID

$$E_{sum} = E_{sum} + E_n$$
$$Output\ (n) = K_p E_n + K_i E_{sum} + K_d (E_n - E_{n-1})$$

where E_n is the position at sample time n

Using your in-tune-ition

One of the reasons PID compensators are so popular is that it is easy to conceive of how each term contributes to the overall output. The D (derivative) term introduces resistance or drag, the P (proportional) term introduces a linear restoring force, and the I (integral) introduces a time-dependent windup term.



Position and cascaded position/velocity loops are two common PID controllers. The most significant difference between them is how they are turned: The position loop alters PID terms iteratively, while the cascaded loop is turned inside out.

Figure 1. PID position loop (1A); cascaded position/velocity loop (1B)

The first manual tuning method that we will discuss works directly in conjunction with this "intuitive" notion of the PID loop. Referred to as the step-response method, it measures the response of the servo system to an instantaneous (within one servo cycle) change in position. To make this method work, or for that matter any manual tuning method, we need an accurate performance trace facility to display the results of our moves. See INSET for more information. For step-response tuning we need to display desired position, actual position, and position error (the difference between these two).

Here is the basic approach used with step-response tuning: Initialize the I term to zero, and set the D term to a small non-zero value. Increase P from zero until the system substantially overshoots. Then increase D until the oscillation is "critically damped." Figures 2A, 2B, and 2C show approximate traces of underdamped, overdamped, and critically damped step responses. Continue this process until you find values that have a high P while still being critically damped

Although very easy to use, this method has the problem that increasing D will cause the optimum value of P to change, which in turn changes the optimum value of D, etc. This requires a number of iterations to get to stable values. In general terms this is because the D term of a PID operates at the highest frequency zone, the P term at a middle point, and the I term at the lowest frequency zone. What would be better is if we could first tune the highest frequency component, then move to the middle-range value, and finish with the low frequency part.

You're in the zone

This is exactly what "zone-based tuning" does, the second manual method that we will introduce. "Zone-based" refers to the frequency zones of the P, I, and D terms, and is adapted from George Ellis' excellent book, *Control System Design Guide.*

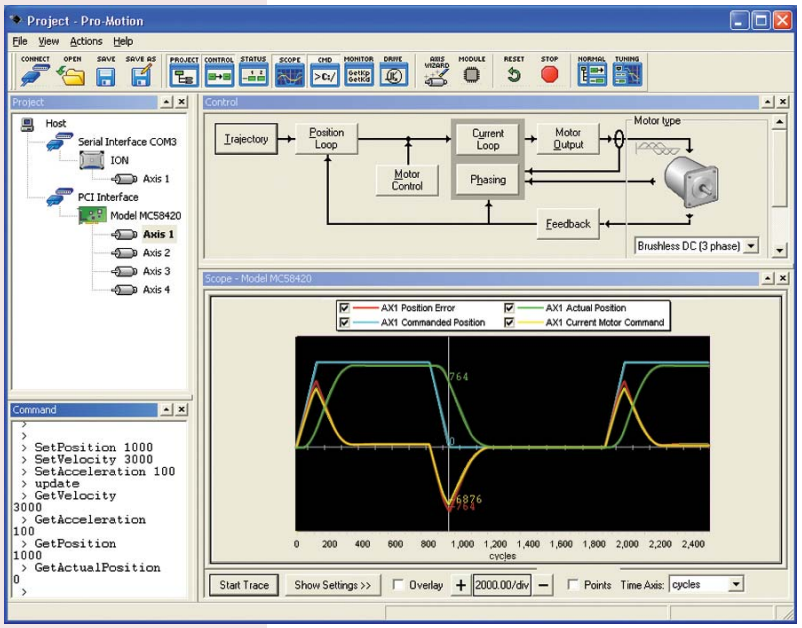
In this method we will plot velocity versus time and the desired profile will be a step function of the velocity (not the position). Set the profile so that it accelerates instantaneously between a velocity of zero and a fixed velocity, and back to zero. Leaving the P and I terms at zero, increase D until the actual velocity profile closely matches the desired velocity profile. Do not worry about whether the destination positions match, you are only examining differences in velocity (velocity error) at this stage. Figure 3 shows a well-tuned D term using this approach.

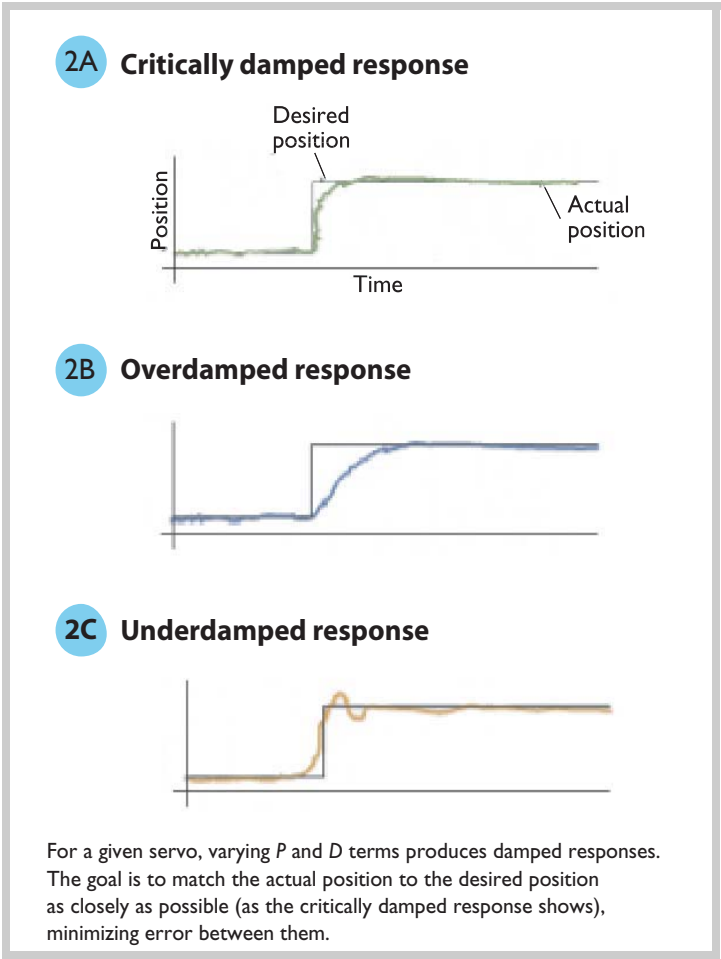
Hardware trace key to high performance motion analysis

Servo tuning has long relied on visual feedback to let engineers determine how well the motion parameters are working. Historically this was done with a standard electronic oscilloscope, but in the past ten years or more, motion analysis has typically occurred on a PC. These PC-based systems have two major elements, the first is a software display and analysis package which runs on the PC, and the second is a hardware motion controller that provides the ability to capture real-time servo data and upload it to the PC.

The Pro-Motion package, provided by Performance Motion Devices, is typical of software programs for motion analysis. It provides the ability to view four traced variables simultaneously, and has numerous other features such as one-time or continuous rolling mode, programmable capture interval and storage to file. Pro-Motion works with any motion controller that utilizes a Magellan Motion Processor. The motion processor is programmed before-hand to continuously store "traced" parameters into a hardware memory buffer. Once trace is complete, this buffer is then uploaded into Pro-Motion for display and analysis.

Compared to older "polling" methods, this hardware trace approach has the advantage that captured data elements are guaranteed to be synchronous with the servo loop. This improves accuracy and eliminates aliasing between the servo loop rate and the software polling rate.





For a given servo, varying *P* and *D* terms produces damped responses. The goal is to match the actual position to the desired position as closely as possible (as the critically damped response shows), minimizing error between them.

Figure 2. Driving waveform for intuitive tuning

Now set up your profiler so that you are using moves with accelerations and velocities typical for your application, and change the capture facility so that it plots the desired position, actual position, and position error. Increase P until the servo error is minimized. At some point as you increase P the motion may have high overshoot, or become unstable, at which point you should back off of this value by at least 20% for the final value.

Zone-based tuning has a number of advantages over step-response tuning. For one, it is less iterative, because it tunes the PID terms in order of the frequency response domain. Secondly, it allows you to utilize real motion profiles with ramps, rather than unrealistic position jumps. In all cases, whether using step-response or zone-based manual tuning, check the motion in both the positive and negative direction to make sure the gain parameters work well in both directions.

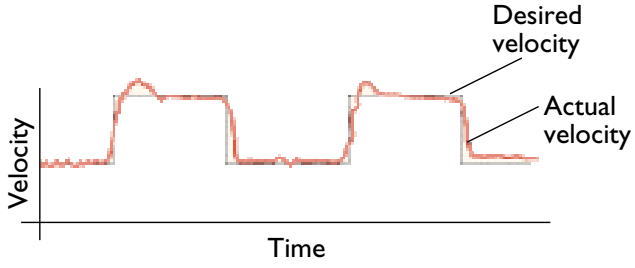


Figure 3. Driving waveform — zone-based tuning

I beg to integrate

Conspicuously absent from this discussion of manual tuning methods is Ki, the integral gain. In general, we want to keep the integral term as small as possible, because it is a direct contributor to servo instability — or as it is expressed in servo analysis terms, to a loss of phase margin. Typically, in manual tuning methods, Ki is the last parameters set, and is used to offset DC biases on the load such as gravity, or to bring final position errors to a very small value, or to reduce position errors at higher velocities.

At this point it is important to discuss the idea of tuning your parameters toward a certain goal. It is a fallacy to believe that one set of PID parameters are optimized for all uses of a motion system. Some systems must have very safe, conservative servo parameters. Others can have aggressive parameters which optimize a specific characteristics such as point-to-point transfer time. Others emphasize having a very small errors during the move, etc.

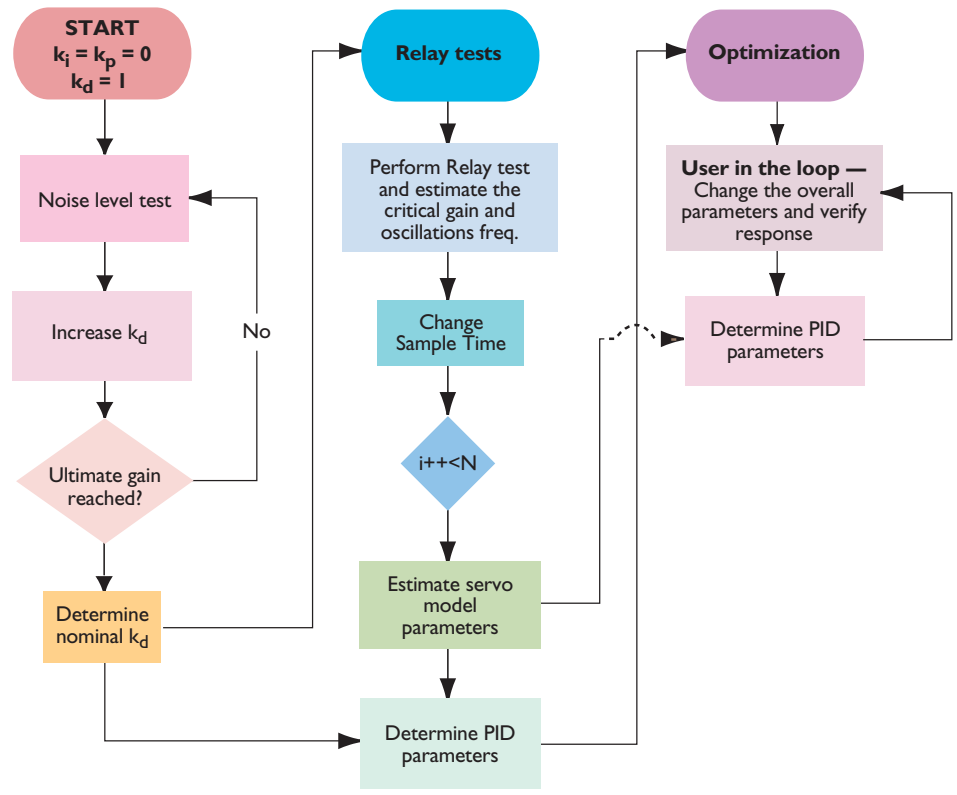
Remember that *what* you are optimizing is an important factor in determining the best servo parameters. This fact, although obvious, is often overlooked both by motion control end-users, and vendors that provide auto-tuning programs for determining the "best" values.

Automatic for the people

In general, manual tuning methods rely on subjective assessments such as "over damped" or "under damped." Automatic tuning, generally referred to as "auto-tuning," holds out the promise of making this process more scientific and repeatable. Better still, automatic tuning places much (but not all) of the burden of tuning onto an algorithm.

Auto-tuning methods tend to use academically researched tuning methods. Of these, Zeigler-Nichols (ZN) is the best known. Unlike the manual methods described above, this method assumes a certain mathematical model to describe the process to be controlled, and then performs tests which are translated through a series of rules into the PID parameters.

The first proposed version of Zeigler-Nichols was not optimized for automatic tuning however, and thus modified ZN methods were developed. One such approach is known as the "frequency response method." This method replaces the PID with a relay (all on positive, and then all on negative) controller during tuning. Using this approach the servo loop becomes oscillatory, and the measured frequency and gain of this oscillation are used to determine the PID parameters.



This flowchart describes an autotuning procedure used in conjunction with PID controllers and uses three phases. The first derives a value for *D* by applying a method like zone-based tuning. The second phase derives values for *P* and *I* using the relay-test approach. And the third lets users apply system information to hand-optimize results.

Figure 4. Sample auto-tuning process

Despite all this, there is relatively little that has been published regarding implementation of auto-tuning procedures for servo loop tuning. This may be due to the fact that most of this work has been done by vendors which consider the algorithms proprietary, or it may be due to the fact that interest in auto-tuning has increased relatively recently, as inexpensive computing power has become available.

Figure 4 provides a flow chart for a specific auto-tuning procedure used in conjunction with PID controllers. This method uses three phases. The first phase is used to derive a value for D, using a method similar to that used in zone-based tuning. The second phase derives values for P and I using the relay test approach described above. And the third phase uses information acquired about the system to allow the user to hand-optimize the results, but using inputs such as "quieter versus noisier" and "aggressive versus less aggressive" to provide meaningful yet easy-to-understand control inputs.

The care and feeding of your servo...forward

In a perfect world, every force that your motor experiences could be predicted in advance. In the real world however, some forces are predictable, while others, such as loads that are larger or smaller than expected, or motor characteristics that change over time, are not. In fact, one of the most valuable aspects of servo control is that even if we know nothing at all about the motor or the load, by using techniques such as those described above, we can still develop passable PID parameters.

But if we *do* know something about the motor or the load, it is possible to improve system performance further, even for a well-tuned system, by "feeding-forward" offsets directly into the output of the servo loop. In the context of motion control, this technique is generally used with the profile generator to provide velocity feed-forward and acceleration feed-forward controls.

Velocity feed-forward is useful to compensate for any viscous friction or velocity-proportional lagging force. This includes some types of friction forces on the motor or load. It is also common if a voltage-mode amplifier (one without a torque loop) is used, because in these types of amplifiers back-EMF introduces a velocity-proportional lag.

Acceleration feed-forward is useful to compensate for any acceleration-proportional lagging force. This includes, in theory, all hardware with non-zero inertia, because basic physics dictates that if we change velocity, the object will resist this change, and this resistance will show up as an acceleration-proportional lag.

Figure 5 shows a plot of position error versus time for a system that exhibits velocity and acceleration-dependant characteristics, along with the position error after application of velocity and acceleration feed-forward gain values.

Practically speaking, feed-forward only works if inertia, friction, and other system forces can be predicted. Many motion systems have a variable load, or friction forces that change dramatically

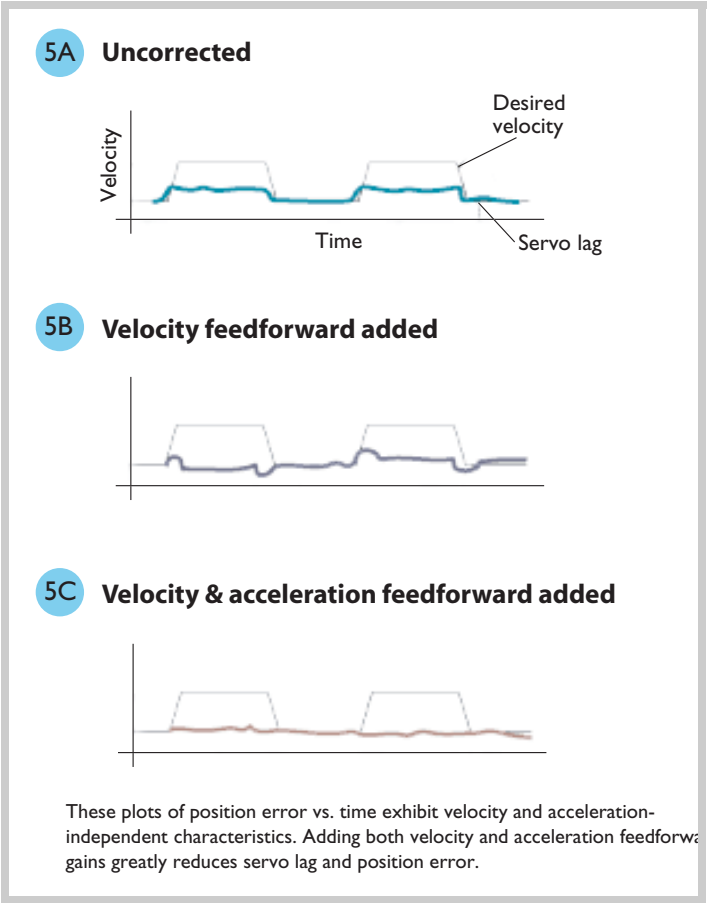


Figure 5. Effects of adding feedforward

over the expected operating temperature range or product lifetime. Be sure that feed-forward gains that are helping you under one set of conditions are not hurting you for a different set of conditions.

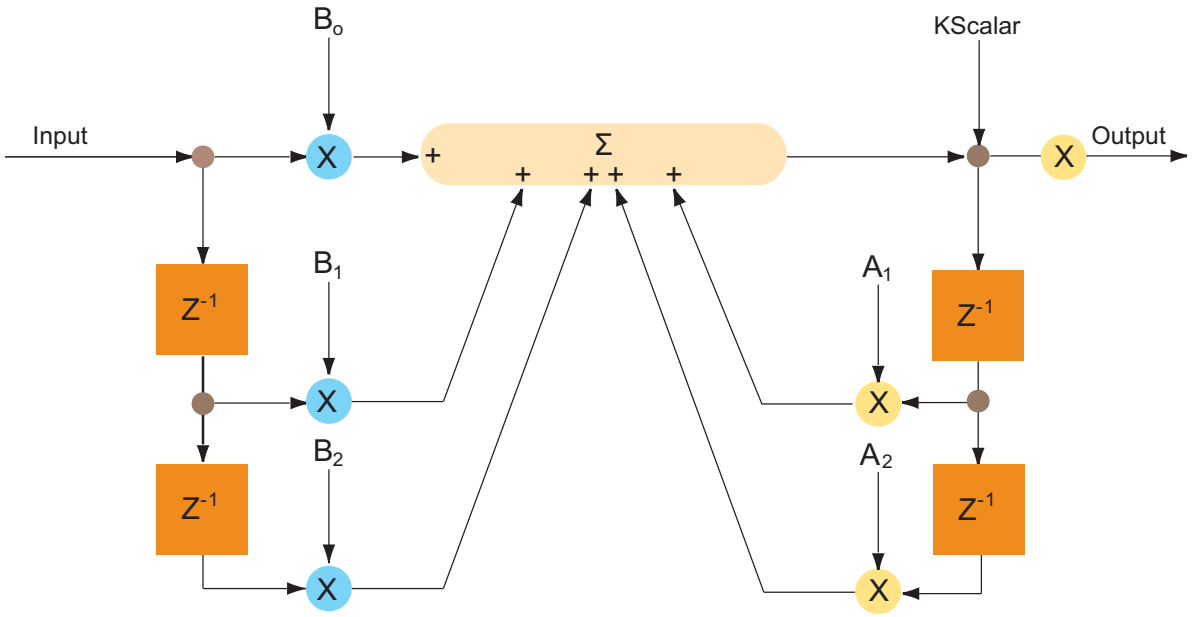
Equation 3. Velocity feedforward

$$Output_n = Output_n + K_{vff} V_n + K_{aff} A_n$$

Frequen-cy asked questions

Many modern servo filters provide some facility for frequency-dependant filtering. This is useful for compensating for mechanical systems that have a resonance at a certain frequency or speed, or to reduce high frequency noise.

The most common implementation of such a filter is known as a bi-quad filter, shown in figure 6. By choosing the right values for A1, A2, B0, B1, and B2 this filter can function as a notch filter, band-pass filter, high or low-pass filter. If you are not familiar with use of a bi-quad filter, there are a number of facilities that provide information including the website *www.octave.org*. This website includes a tool that lets you calculate values for A1, A2, etc. based on the frequency filtering characteristics you want for your system.



To calculate a biquad filter's output, one must combine user-programmed coefficients A1, A2, B0, B2, and K with current and previous input and output values.

Figure 6. Biquad algorithm flow

Equation 4. Bi-quad filter

$$Y_n = K(B0X_n + B1X_{n-1} + B2X_{n-2} + A1Y_{n-1} + A2Y_{n-2})$$

where: Y_n is the filter's output at time n
X_n is the filter's input at time n
K is a positive scalar
$B0$, $B1$, $B2$, $A0$, $A1$ are programmable biquad coefficients.

Conclusion

Servo tuning need not be more difficult than other typical motion tasks such as sizing a motor. There are a number of standard manual methods available, two of which, step-response tuning and zone-based tuning, are discussed in this article. Auto-tuning holds out the promise of eliminating human involvement in the process of servo tuning, but at present, most auto-tuning packages are designed to provide workable initial values, which are then further hand-optimized for a specific application.

Once your basic PID parameters have been determined, techniques such as feed-forward, and bi-quad filtering can be used to further improve performance, or increase smoothness.

Regardless of the process by which you arrive at your tuning parameters, make sure that you exercise your system over the complete load range expected for your application, and if possible, on both new and older hardware to insure that your system will run correctly under real-world conditions.

Performance Motion Devices, Inc

55 Old Bedford Road
Lincoln, MA 01773
e-mail: info@pmdcorp.com

www.pmdcorp.com

About Performance Motion Devices

Performance Motion Devices (PMD) is the recognized world leader in motion control ICs, cards, and modules. Dedicated to providing cost-effective, high performance motion systems to OEM customers, PMD utilizes extensive in-house expertise to minimize time-to-market and maximize customer satisfaction.