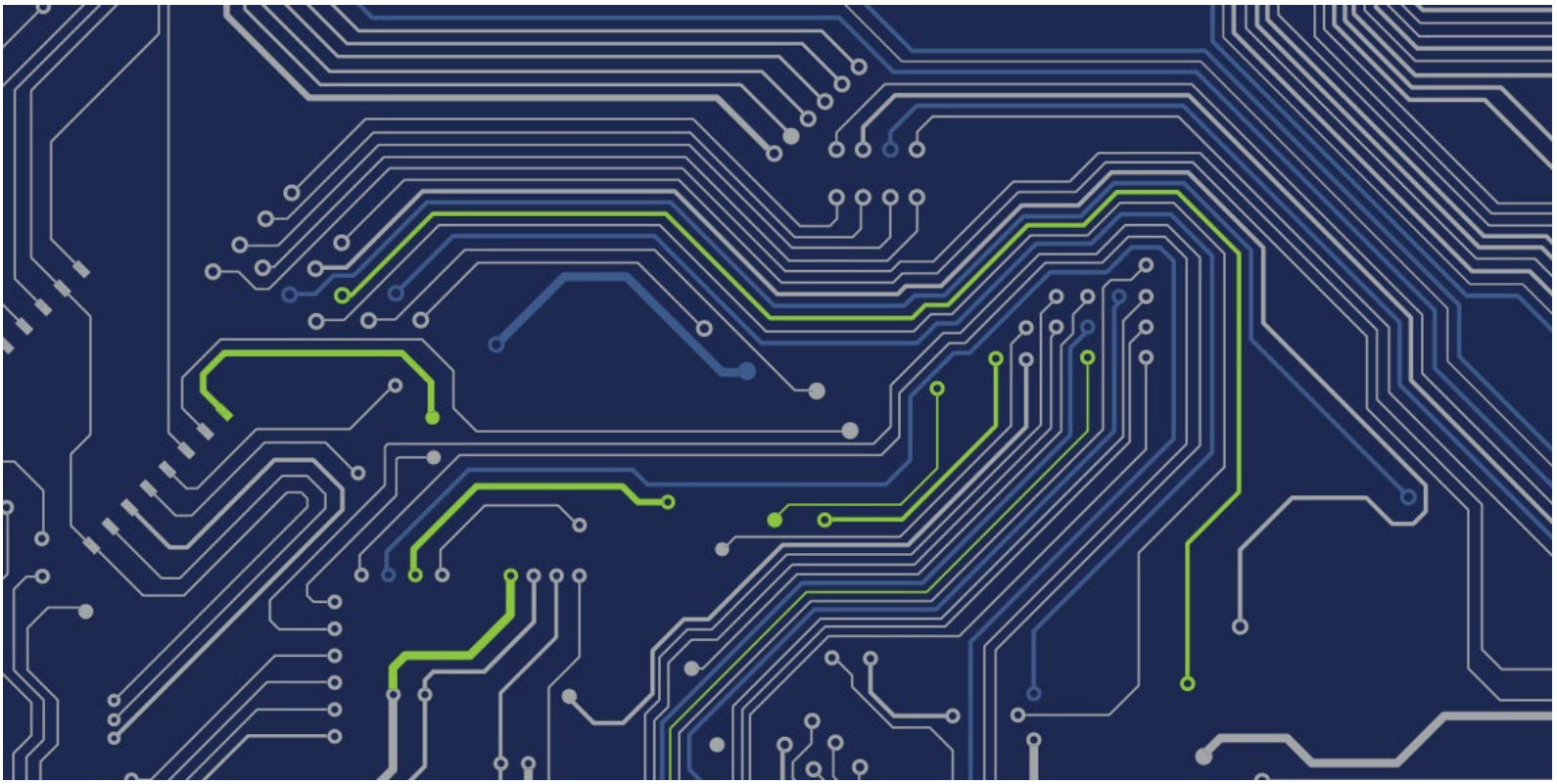

Servo Motor Tuning – Rocket Science or Walk In The Park?

Tuning a servo motor manually can be a daunting exercise but in this paper we show you two simple techniques that will get your servo motor up and running quickly and smoothly.

Chuck Lewin
Performance Motion Devices, Inc.

www.pmdcorp.com

PERFORMANCE MOTION DEVICES
MOTION CONTROL AT ITS CORE

In this paper we provide an overview of PID (proportional, integral, derivative) based servo tuning, and introduce two manual tuning methods that work well for a large variety of systems. We also show that 'optimal' parameters vary by application and performance goals, even for the exact same motor and amplifier setup.

A tale of two servos

There are two types of servo motors commonly used for positioning applications; the DC Brush motor, which uses mechanical brushes to commutate the motor, and the Brushless DC motor, aka the BLDC motor, aka the AC Synchronous motor, aka the PM (permanent magnet) motor, which is commutated electronically by external circuitry.

Unlike step motors, which move in discrete position steps, servo motors have no built-in sense of where they are, and thus require a feedback device such as a quadrature encoder to control their position.

The servo loop (technically the position servo loop) has the job of driving the motor to a particular location. It does this by comparing the desired position from the trajectory generator at any given moment with the actual motor position and applying a continuous correcting motor command.

Servo schemes require gain parameters to be set by the user to accommodate different machine loads and operating conditions. The more optimally these parameters are set, the more accurately the motor will track the desired position under a variety of motion profiles.

I PID therefore I am

Theorists and engineers have developed a number of servo compensation schemes over the years, but the overwhelming favorite for motor positioning is the PID loop, which means Proportional, Integral, Derivative when spelled out.

As it turns out, several different implementations of the digital PID loop exist, and these PID loops connect to several different types of amplifiers. To ground the discussion, we will focus on the PID Position loop shown in Figure 1 and connect the output of this loop to a torque mode amplifier, also called a current mode amplifier. This is by far the most common overall configuration of a PID position loop and off-the-shelf amplifier.

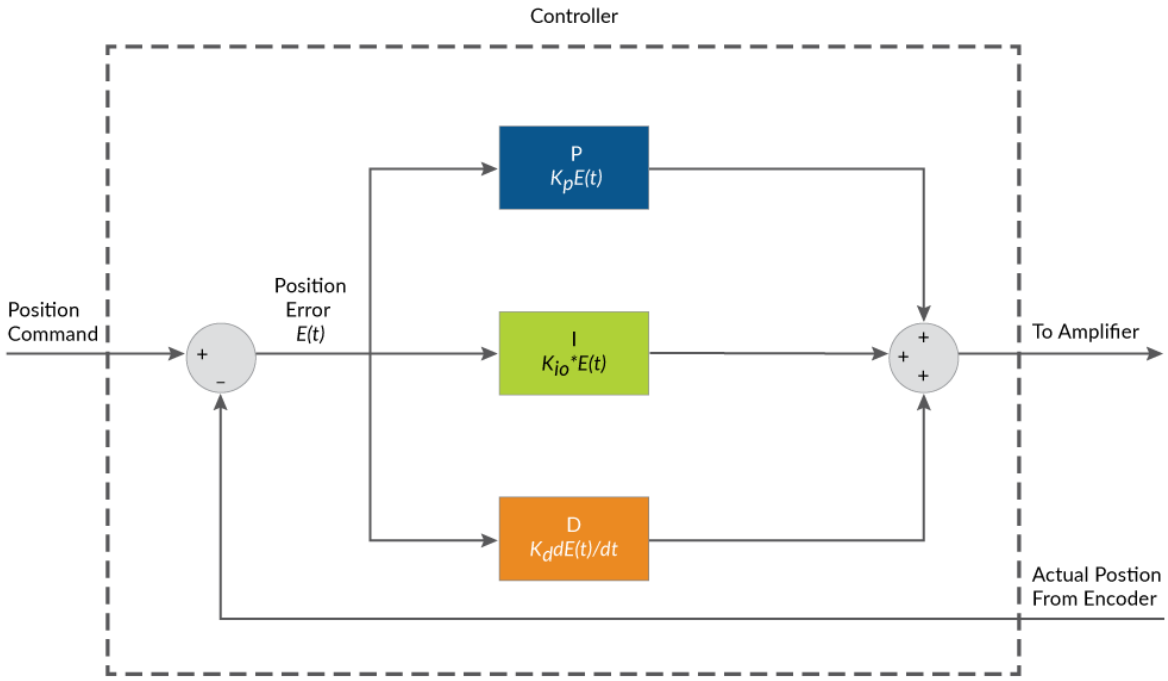

Figure 1: PID Position Loop

The PID position loop requires us to determine three values: the proportional gain, (Kp) the integral gain (Ki) and the derivative gain (Kd). Modern motion vendors provide a bevy of additional options, including an integral limit, programmable derivative time, feed-forward gains, motor bias, deadband filters, and frequency domain filtering such as notch filters or band-pass filters. Several of these concepts will be discussed later.

What is in a PID and what do the terms really mean? The reason that PID loops are so popular is that its constituent pieces, the P, the I, and the D, can be described and understood intuitively.

The P term is called the proportional term because it provides a proportional restoring correction to the amplifier output command. When presented with the servo error, which for a position loop is the difference between the desired (commanded) position and the actual (measured) position, the P term functions like a spring. The larger the servo position error, the larger the corrective restoring motor command.

The I term is the integral (or integration) term because it integrates, over time, the position servo error. *Why would this be useful?* Because if only a P (proportional) term is available it may be difficult to arrive at the exact commanded position due to forces or mechanical issues such as gravity, stiction, motor detents, or other factors. The I term builds up over time and can get the servo 'over the hump' to the final desired position.

The contribution of the D term is calculated by subtracting the previous servo error from the current servo position error. It has two main practical affects; it provides a feed forward boost whenever the profile velocity increases or decreases, and it provides a general purpose drag term, thereby tamping down oscillations.

Turning the dial to ‘11’

So how best to set these values? We start with what has become, hands down, the most common approach for quick tuning of a position loop. Referred to as the step response method, this approach centers around the reaction of the motor to an instantaneous change in commanded position (the step).

To make this method work, or for that matter any manual tuning method, we need a position trace facility to display the results of our moves.

At a minimum, we need to display the desired position (the position commanded by the trajectory generator) and the actual position (the actual measured location of the motor). Figure 2 shows an example screen capture of such a trace facility, in this case the relatively elaborate trace system provided by PMD's Pro-Motion Software. You can use this or a similar third-party product, something that you develop yourself, or in theory you can even just use an oscilloscope.

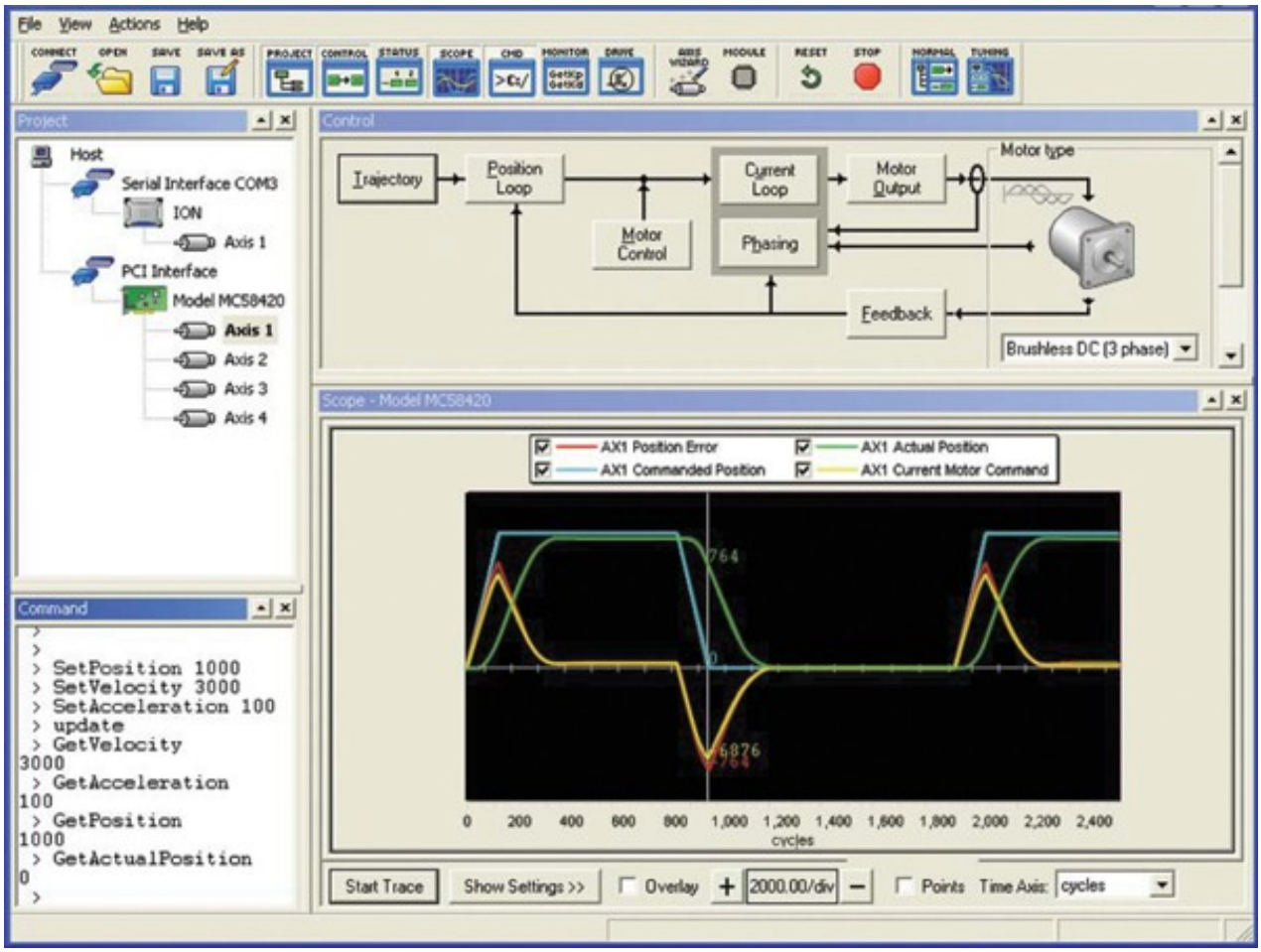

Figure 2: Desired and Actual Position Trace Facility

If possible, you also want to display the motor command (the commanded torque value to the amplifier output by the PID loop) because for all of these procedures we want to avoid running the motor with a saturated command. Saturation causes windup, and therefore distorts the accuracy of our PID values.

Here is the basic approach used with step response tuning:

- Initialize the I & P terms to zero, and set the D term to a small non-zero value
- Increase P from zero until the system overshoots and shows an underdamped response
- Increase D until the oscillation is 'critically damped'

Repeat from step 2 and increase P and D until you find the highest practical values that can still generate motion that is critically damped

Figures 3a, 3b, and 3c show approximate traces of underdamped, overdamped, and critically damped step responses.

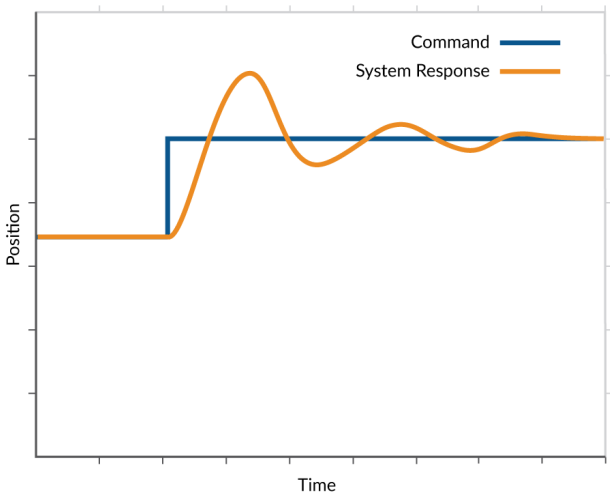

Figure 3a: Underdamped Step Response

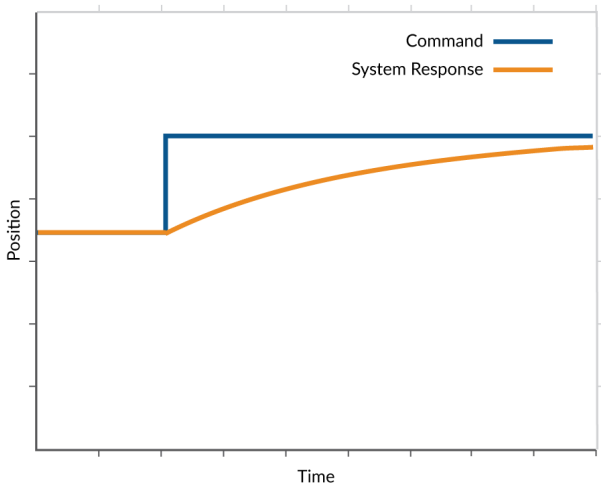

Figure 3b: Overdamped Step Response

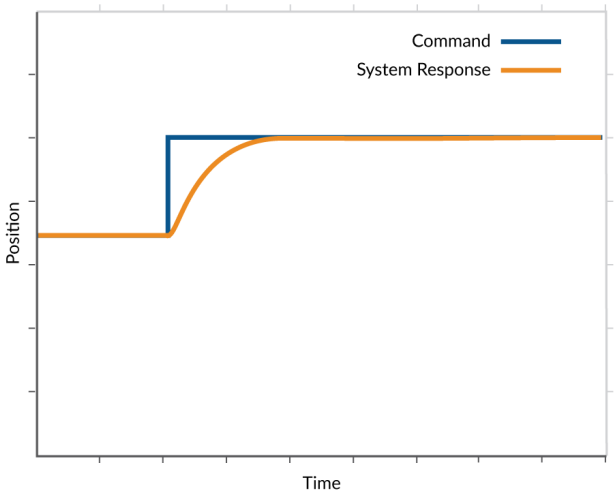

Figure 3c: Critically Damped Step Response

If you increase P to the point where you have stable critically damped motion but the motor is too noisy during motion or at rest, you may want to back off on P and D to quiet things down. A bit later, we will talk about approaches for 'having your cake and eating it' when it comes to giving the motor the most servo 'oomph' while still keeping the motor quiet.

My kingdom for a small settling error

Conspicuously absent from this discussion is Ki, the integral gain. This term is calculated from the sum of the present and all previous servo errors.

How should we set this 'historical' term of the PID output? To begin with, in general we want to keep the integral term as small as possible. This is because historical windup is a direct contributor to servo instability – or as expressed in servo analysis terms, to a loss of phase margin.

On the other hand, use of the integral term can eliminate the last bit of error in the system, often bringing the servo error to +/- 1 count or even zero counts for the final motor position. So some amount of Ki can do a lot of good for our system's accuracy.

In addition to a settable Ki term, commercial controllers provide a settable integral limit term, or Ilim for short. Ilim caps the total contribution of the integral term, effectively reducing its 'memory'. This is a very useful feature for reducing the overshoot that can occur from windup. A good starting approach is to have the maximum contribution of the I term (Ki and Ilim working together) be 10-25% of the maximum contribution from the P (proportional) term. To figure all this out you may need to check your controller's manual, or experiment a bit to see how Kp, Ki, and Ilim contribute to the motor output command.

We have to go around

You may spend some time iterating values of P,D & I, but it doesn't take long to get a feel for the effect of parameters changes – when you should increase, when you should back off, and when you have reached a peak. Unfortunately, there may also be times when you chase your tail. Increasing D causes the optimum value of P to change, which in turn changes the optimum value of D, etc.

Why does this happen? The answer has to do with how the frequency domains of the various P, I, and D terms overlap. Higher frequency terms, most notably the derivative term, affect all frequencies from low to high. Low frequency terms such as integral only affect the low frequency. And P is somewhere in the middle.

To tune in such a way that minimizes these interactions, it would be better 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 tuning 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, now in its fourth edition, "Control System Design Guide".

So how does it work? In this method we plot velocity versus time and the desired profile will be a step function of the velocity (not the position). Here is the approach step by step:

- 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 graph 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 4 shows a representative well- tuned velocity versus time.

- 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 this value by at least 25% for the final value.

- As before, finish off by bringing in integral as required to land the motor with the required accuracy, but not so much as to introduce instability.

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. So it should arrive at a set of control parameters faster. Secondly, it allows you to utilize real motion profiles with ramps, rather than unrealistic instantaneous position jumps.

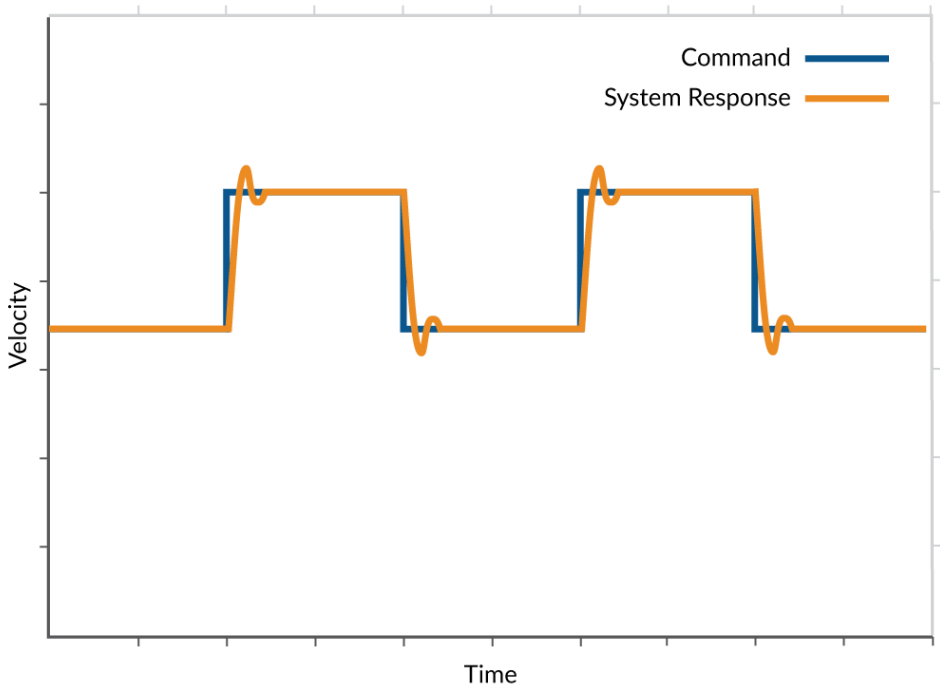

Figure 4: Zone-based Tuning

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.

Keepin' it real

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 characteristic such as point-to-point transfer time. Others emphasize very small errors during the move, and still others must operate without any audible noise.

You may also be interested in: Servo Motor Noise and How to Fix It

Another important influence on your actual final servo settings is the load, and the forces stemming from the motion profiles that accelerate and decelerate the load. Depending on your specific machine's mechanics, these forces can have a large impact on servo operation through the effect known as reflected inertia. As a rule, highly geared motors experience relatively little reflected inertia, and direct drive or minimally geared motors experience large reflected inertias.

Gain scheduling is a blanket term to describe a standard approach for handling such complicating conditions. The general idea is to switch in various sets of gain parameters while the machine is operating in different modes or carrying different loads.

It's easy to get carried away with gain scheduling, but at a minimum you may find it advantageous to develop a more aggressive 'moving' set of servo gains and quieter, less aggressive 'at settle' gains used to hold the axis in place. Many controllers provide a software accessible 'in motion' flag, making it easy to trigger on this condition, or on other conditions such as specific values of position, velocity, acceleration, or time.

Another important technique for improving real world performance is feedforward. Feedforward has no dynamical effect on system stability and is therefore a sort of 'free lunch' for improving axis performance. As our previous Torque Feedforward Deep Dive indicated, knowledge of the system weight or motion kinematics can be used to feed-forward a motor torque command that lessens the burden on the PID, thereby allowing less aggressive servo values to be used.

You may also be interested in: Use Torque FeedForward to Get More From Your Motion Controller

An auto-tuner in every pot

Most manual tuning methods rely on subjective assessments such as 'over damped' or 'under damped'. Automatic machine tuning, referred to as 'auto-tuning', holds out the promise of making this process more scientific, and certainly less time consuming.

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.

As we have learned from the sections above though, auto tuner values generally do not provide values under actual machine operating conditions. So treat auto-tuning parameters as an initial suggestion, and plan to hand optimize from there.

Frequency asked questions

One final technique for improving your machine's performance is frequency-based filtering. To the extent that a servo loop is a dynamic response system, we can place various kinds of filters on the input signals or the output signals of the servo loop to make them less prone to oscillation.

The most common implementation of such a filter is known as a biquad filter, shown in Figure 5. By choosing the right values for A1, A2, B0, B1, and B2 this filter can function as a variety of filtering functions including a notch filter, a band-pass filter, and a high or low-pass filter.

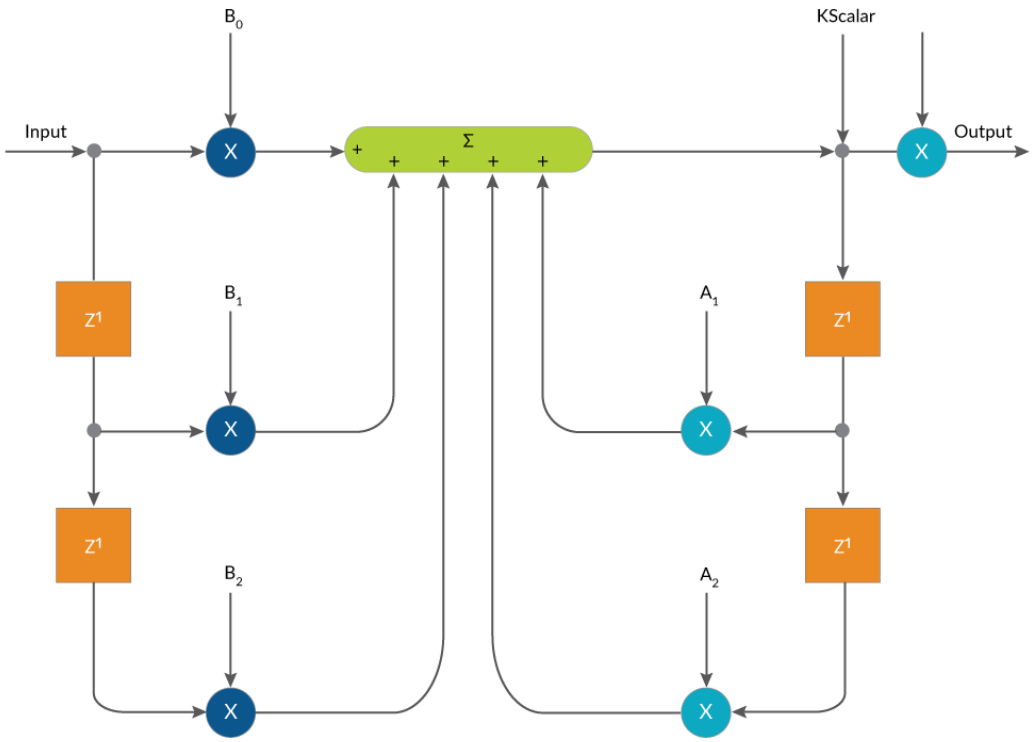

Figure 5: Biquad Filtering

If you are not familiar with use of a biquad filter, there are a number of resources 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.

For most engineers who seriously dive into frequency-based filtering schemes such as biquad filtering, the control settings for this filtering will be built on the back of a detailed characterization of the mechanical system using frequency-based analysis tools such as Bode plots. Although not for the faint of heart, these tools are readily available from most motion vendors and add numerical analysis to what can otherwise be an intuitive process. Bode plots and understanding how to use and interpret them is a complex subject beyond the scope of this article, but to learn more you may be interested paper below, Mechanical Resonant Frequency.

You may also be interested in: Mechanical Resonant Frequency

By way of caution, be careful not to have unrealistic expectations for what a frequency response filter can do. Mechanics age and change over time, and machines vary somewhat even directly from the factory. If you find an improvement from a notch filter or some other kind of filter, make sure that this improvement applies in the field under real world conditions, and over time.

To the laboratory!

What would a good servo tuning article be without some spinning motors and graphs?

To illustrate various aspects of this paper, we have connected a Prodigy/CME board to a Brushless DC motor, and displayed the resulting captured data using PMD's Pro-Motion software package.

Below are links to a couple of videos that show simple underdamped, overdamped, and critically damped motion for a no-load single motion axis responding to a step response command.

- Underdamped Single Motion Axis
  https://www.youtube.com/watch?v=Q1E26NjCsjQ

- Overdamped Single Motion Axis
  https://www.youtube.com/watch?v=uvpQyemWlqc

- Critically Damped Single Motion Axis
  https://www.youtube.com/watch?v=sDj3zKBqLMI

PMD Products That Support Servo Motors

PMD has been producing ICs that provide advanced motion control of DC Brush and Brushless DC motors for more than twenty-five years. Since that time, we have also embedded these ICs into plug and play modules and motion control boards. While different in packaging, all of these products are controlled by C-Motion, PMD's easy to use motion control language and are ideal for use in medical, laboratory, semiconductor, robotic, and industrial motion control applications.

MC58113 Series ICs

The **MC58113** series of ICs are part of PMD's popular **Magellan Motion Control IC Family** and provide advanced position control for step motors, BLDC, and DC Brush motors alike. Standard features include auto-tuning, s-curve profiling, FOC (Field Oriented Control), high/low switch signal control, direct encoder & pulse & direction input, and much more. Whether used for laboratory automation, pump control, pointing systems, or general-purpose automation, the MC58113 family of ICs are the ideal solution for your next machine design.

Learn more >>

ION Digital Drives

ION Digital Drives combine a single axis Magellan IC and an ultra-efficient digital amplifier into a compact rugged package. In addition to advanced servo motor control, IONs provide S-curve point to point moves, i2T power management, downloadable user code, and a range of safety functions including over current, over voltage, and over temperature detect. IONs are easy to use plug and play devices that will get your application up and running in a snap.

Learn more >>

Prodigy/CME Machine-Controller

Prodigy®/CME Machine-Controller boards provide high-performance motion control for medical, scientific, automation, industrial, and robotic applications. Available in 1, 2, 3, and 4-axis configurations, these boards support DC brush, Brushless DC, and step motors and allow user-written C-language code to be downloaded and run directly on the board. The Prodigy/CME Machine-Controller has on-board Atlas amplifiers that eliminate the need for external amplifiers. To build a fully functioning system only a single HV power supply, motors, and cabling are needed. Host interface options include Ethernet UDP and TCP, CANbus, RS-232, and RS-485.

Learn more >>

Pro-Motion Analysis Software

Pro-Motion is PMD's easy-to-use Windows-based exerciser and motion analysis program. It offers ready-to-go capabilities your entire development team will be able to share. A step-by-step axis wizard allows designers to quickly and easily tune position loop, current loop, and field-oriented control motor parameters. Advanced users can access a complete motion analysis package with Bode plot generation and auto-tuning.

Learn more >>

Visit us at www.pmdcorp.com or call +1-978-266-1210 to learn more.

PERFORMANCE MOTION DEVICES
MOTION CONTROL AT ITS CORE