PID Based Servo Tuning: Black Art, Rocket Science, or Walk in the Park

This article provides an overview of PID (proportional, integral, derivative) based servo tuning, and introduce two standard manual tuning methods that work well for a large variety of systems. It also shows that 'optimal' parameters vary by application and performance goals, even for the exact same motor and amplifier setup.

Servo This

There are two types of servo motors commonly used for positioning applications; the DC Servo motor, which uses mechanical brushes to commutate the motor, and the brushless DC motor, aka the BLDC motor, aka the PM (permanent magnet) brushless 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.

A servo loop, also called a compensator, has the job of driving the motor to a particular position. It does this by comparing the desired position from the path 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.

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.

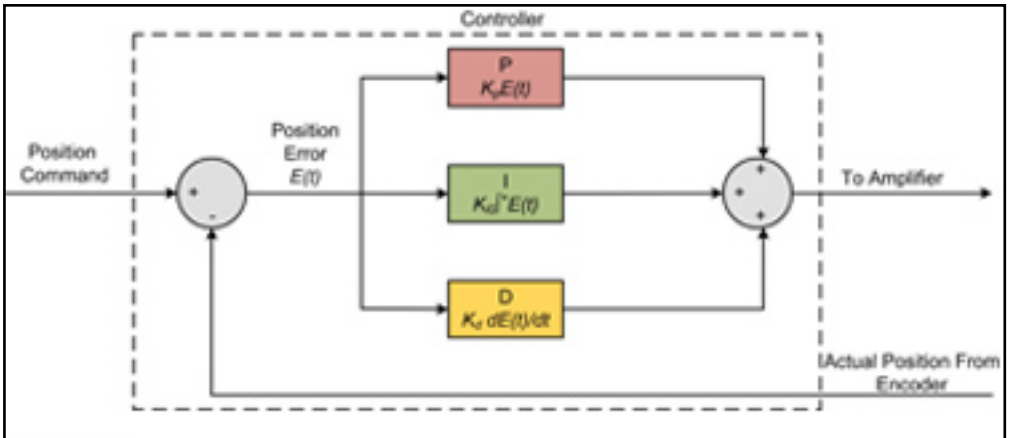

Figure 1.

The PID position loop requires us to determine three values: the position loop 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, and frequency-domain filtering such as notch filters or band-pass filters.

Turning The Dial To '11'

So how best to set these values? 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 desired position and actual position. 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.

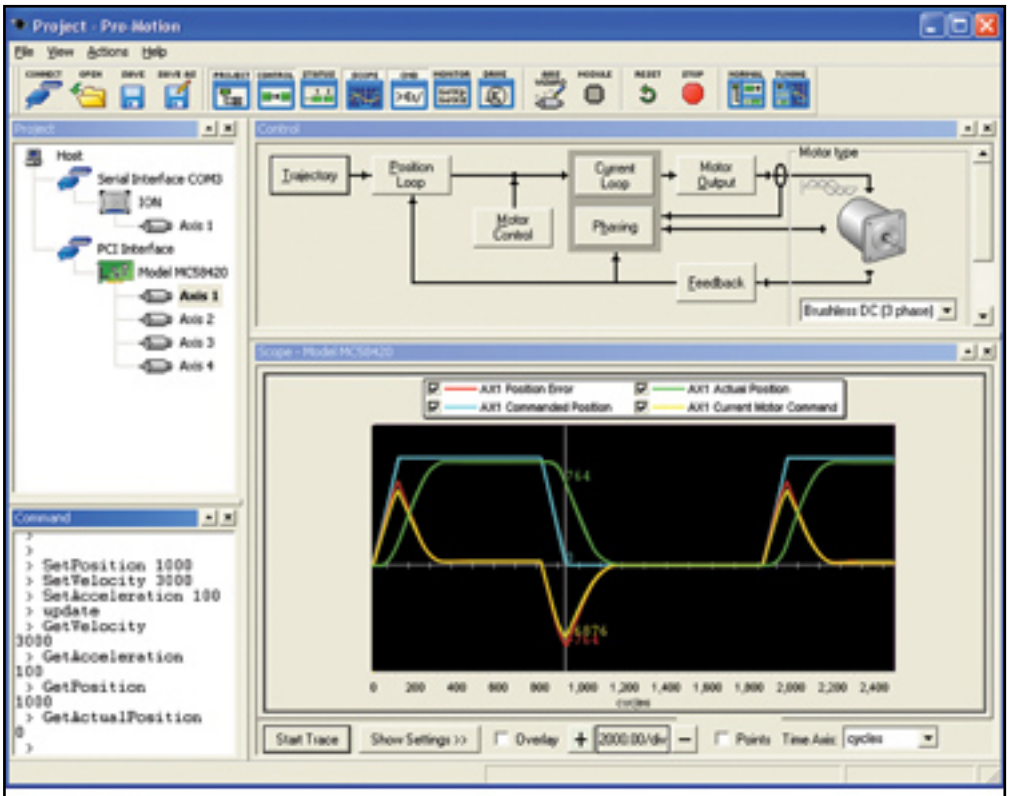

Figure 2.

If possible, display the motor command 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:

1. Initialize the I & P terms to zero, and set the D term to a small non-zero value
2. Increase P from zero until the system overshoots and shows an underdamped response
3. Increase D until the oscillation is 'critically damped'
4. 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.

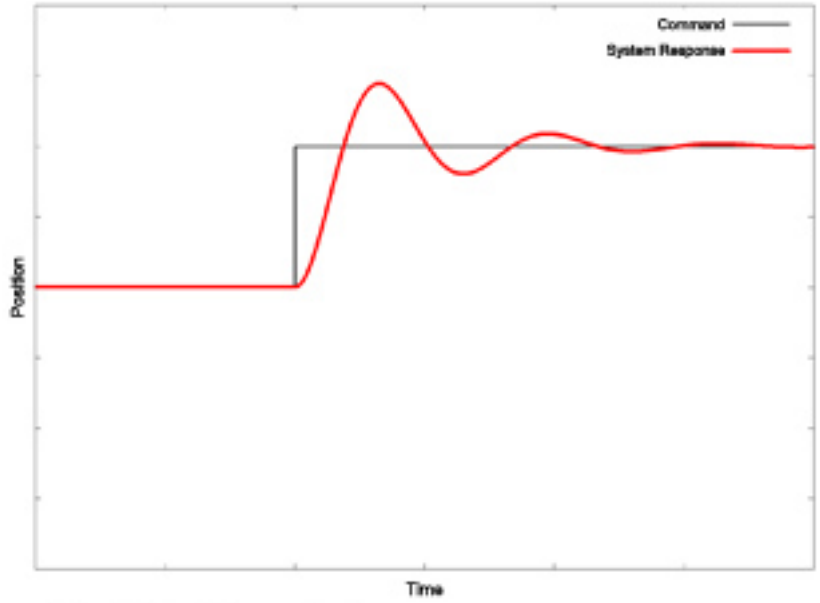

Figure 3a. Underdamped step response.

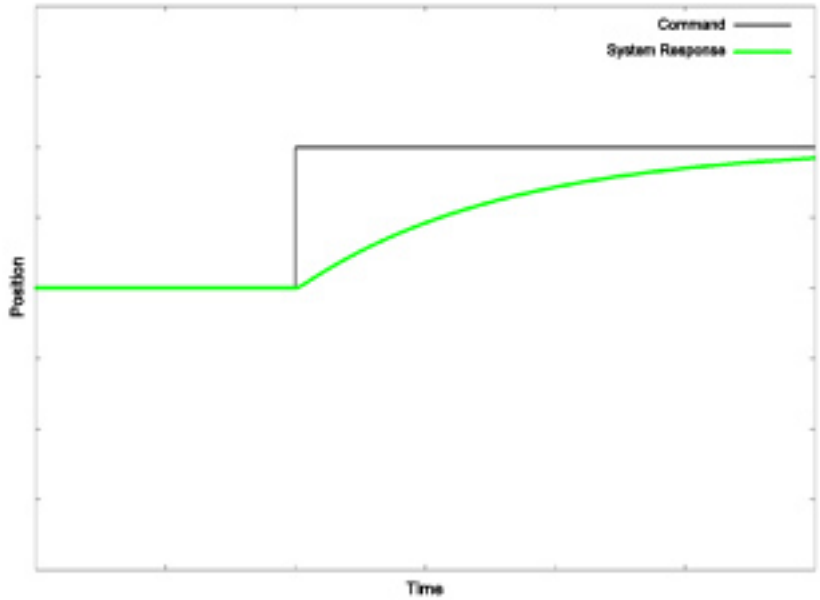

Figure 3b. Overdamped step response.

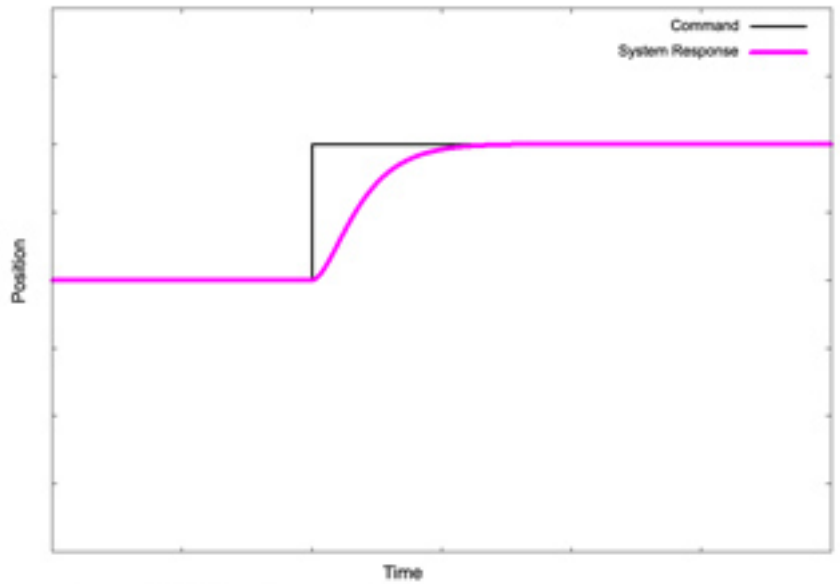

Figure 3c. Critically damped step response.

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 little 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 generally 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 percent 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.

That’s Far Too Easy

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, "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:

1. Set the profile so that it accelerates instantaneously between a velocity of zero and a fixed velocity, and back to zero.

2. 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 4 shows a representative well-tuned velocity-versus-time graph.

3. 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.

4. 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 25 percent 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
.

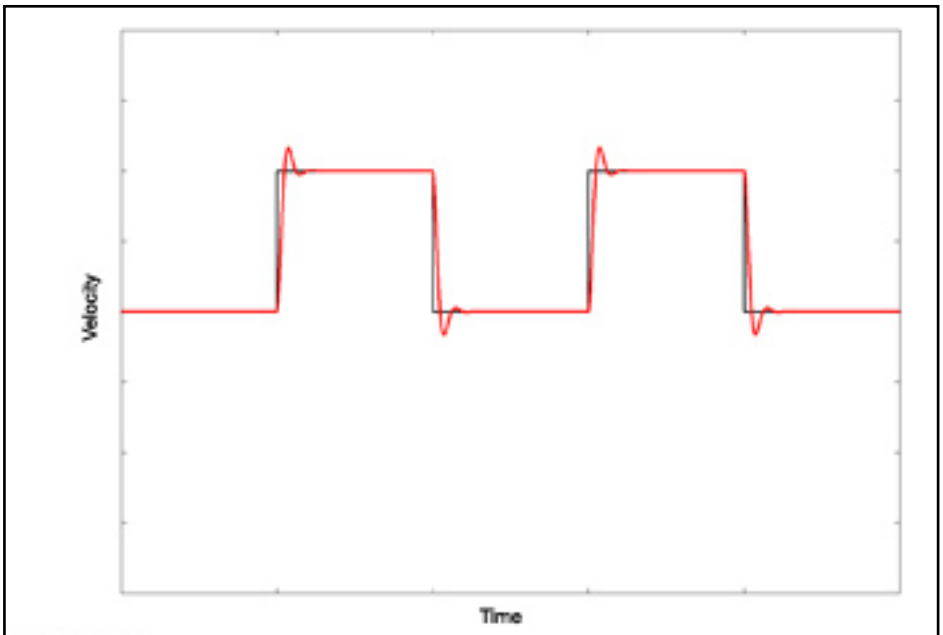

Figure 4.

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. Secondly, it allows you to use real motion profiles with ramps, rather than unrealistic instantaneous 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.

Everything Was Great Until It Started Working

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.

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 very 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. 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.

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. Better still, automatic tuning places much, if not all, of the tuning burden on 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.

As we have learned from the sections above though, auto tuner values generally don't provide values under

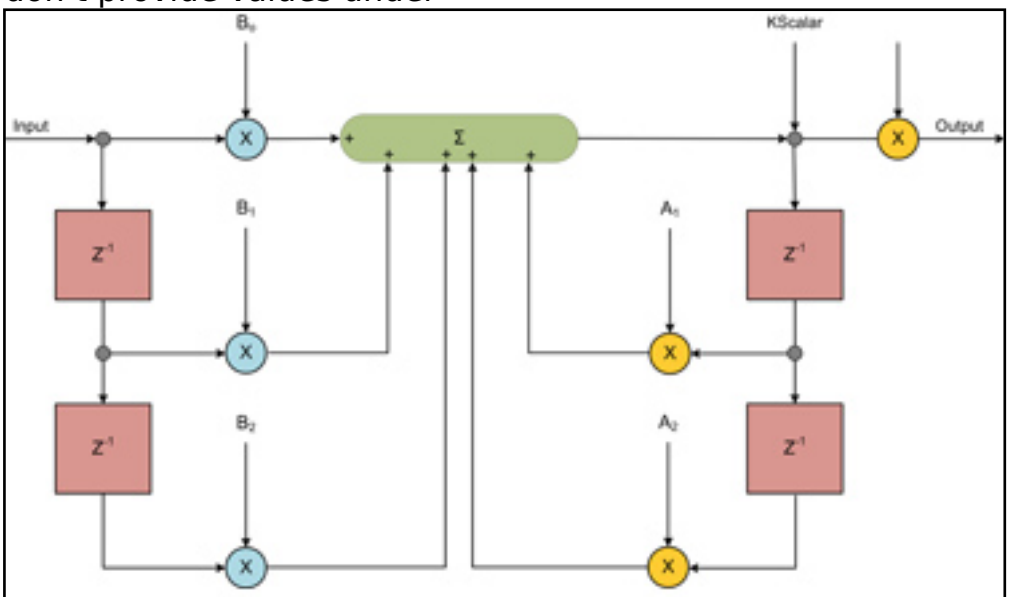

Figure 5.

actual machine operating conditions. So treat auto-tuning parameters as an initial suggestion, and plan to hand-optimize from there.

Frequen-cy 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 bi-quad filter, shown in Figure 5. By choosing the right values for A1, A2, B0, B1, and B2, this filter can work as a variety of filtering functions including a notch filter, a band-pass filter, and a high or low-pass filter.

If you are not familiar with use of a bi-quad filter, there are a number of resources that provide information including the website www.octave.org [1]. 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.

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.

Source URL (retrieved on *04/21/2014 - 3:49am*):
http://www.ecnmag.com/articles/2011/07/pid-based-servo-tuning-black-art-rocket-science-or-walk-park?qt-most_popular=0&qt-video_of_the_day=0

Links:
[1] http://www.octave.org