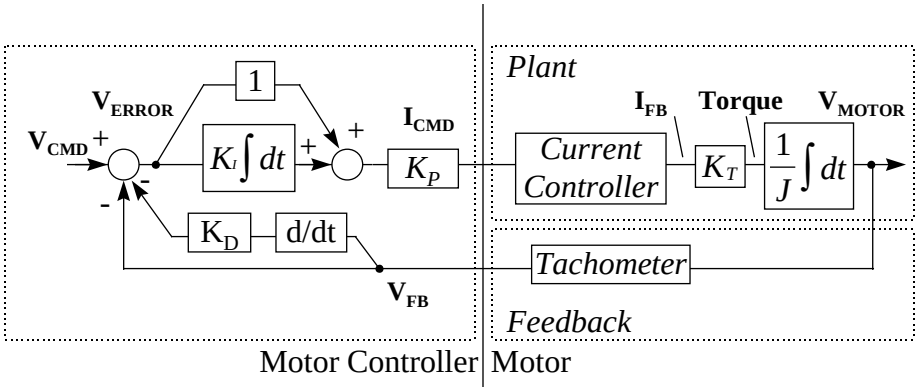
This article was the basis for the February, 1999 edition of "20-minute Tune-Up" printed in PT Design Magazine.

Closing a PID Velocity Loop

February, 1999

Closed-loop motion controllers are top performers. They are used in virtually every industry segment when applications demand the highest precision or the fastest motion. That performance comes at a cost: these controllers must be tuned, a process where various gains are adjusted so the machine achieves optimal performance. Tuning can appear mysterious, even intimidating, to engineers new to motion control. This article reviews the basics of the PID control loop including a tuning procedure. If you want to try it out yourself, you will be able to download a simulated PID controller and practice some of these principles. This article won't make you a tuning expert, but if you are new to motion control, it will give you a good start.

Every closed-loop system has three components: a controller, a "plant," and a feedback device. The plant is the mechanism that is being controlled. It might be a temperature bath, a pressure chamber, or, as in this case, motor and load. A feedback sensor is used so the system will have precise control of the plant. The controller compares the command and feedback signals, and after some processing, sends a control signal to the plant. That "processing" can take many forms, but the most popular is probably the "PID" controller. PID controllers are so named because they combine proportional, integral and derivative signals. Each of those three signals is scaled by a "gain." This is shown in Figure 1.



PID Velocity Control System
Figure 1

In Figure 1, the controller is on the left. The error is the difference between velocity command and feedback. The error is integrated and scaled by the gain K_I. A proportional gain of unity ("1") is summed with the integral term and then both are scaled by the proportional gain, K_P. The derivative is only in the feedback path and is scaled by the gain K_D. One unique characteristics of the PID controller is that the derivative is only in the feedback and not in the forward path with K_I and K_P. This is common in velocity controllers because this holds down overshoot; derivatives in the forward path usually produce excessive overshoot in response to a square wave command. The plant is shown in the upper right of Figure 1. This includes the current controller and the motor. The plant produces velocity from the current command. The feedback device is shown as a tachometer.

Each of the three gains has different functions. K_P, in the correct measure, provides stability allowing all gains to increase. But make K_P too large and the system will become noisy; larger still and the system will become unstable. K_I makes the system stiff—a large K_I will make it difficult for disturbance torques to move the shaft. But a large K_I also generates a lot of overshoot. K_D provides damping—it eliminates much of the overshoot caused by K_I. But too much K_D causes high-frequency oscillations. This behavior is shown in Table 1.

Gain	When you have enough…	When you have too much…
K_P	Stable fast response.	Noisy, even unstable.
K_I	Stiff system.	Overshoot, low-frequency oscillations.
K_D	Reduce overshoot from K_I.	High frequency oscillations.

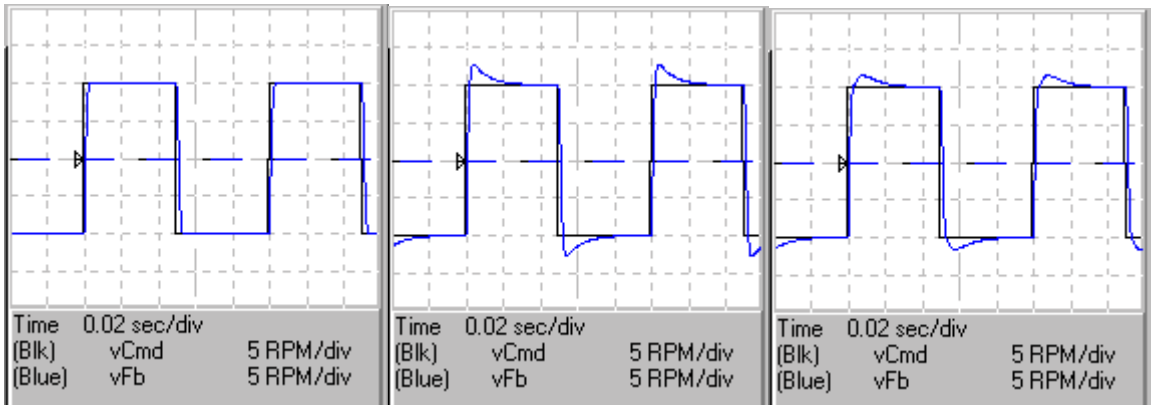
Effects of Tuning Gains
Table 1

A simple process for tuning

The process of setting K_I, K_D, and K_P can be tedious because of the problem posed by simultaneously adjusting three constants. And when the combination is wrong, the system will become unstable, generating a lot of noise and occasionally damaging the machine. The key to adjusting gains efficiently is using a process that allows you do decouple, as much as possible, the effects of one gain term to another. The process we will use is as follows:

1. Set all gains to zero.
2. Apply a square wave command.
3. Raise K_P until the system begins to overshoot.
4. Raise K_I until overshoot is approximately 15%.
5. Raise K_D to remove some of the overshoot.

When applied to the simulated system, Step 3 produced a K_P of 1.3, Step 4 produced a K_I of about 100, and Step 5 produced a K_D of about 0.0005. Figure 2 below shows simulated oscilloscope curves after steps 3, 4, and 5. Note that using K_I produces overshoot which normally cannot be completely removed. However, bear in mind that most velocity controllers do not need to respond to square wave commands; usually the command is from a position controller which is usually much gentler than a square wave. So, all overshoot does not need to be removed. Note that if you don't need integral gain, don't use it. Integral gain has the benefit of removing long-term error, but if that's not important for the application, don't expose the system to the problems it causes, chiefly overshoot.



2a *2b* *2c*

PID Control System with Different Gains
a) KP = 1.3, b) KP = 1.3 and KI = 100, c) KP = 1.3, KI = 100, and KD = 0.0004
Figure 2

Two problems that arise in tuning systems are noise and resonance. Motion controllers both produce and are susceptible to noise. Resonance, a common problem caused by the motor and load connected by a compliant shaft or belt system, causes oscillation which cannot be tolerated in most applications. The level

of K_P and K_D may need to be reduced to help with these problems; K_I does not have much effect on noise or resonance.

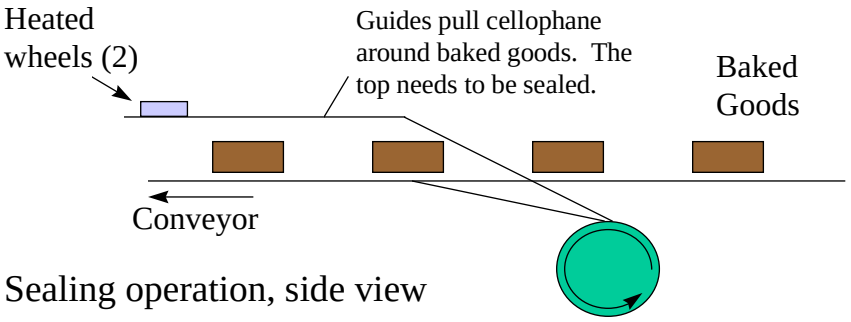
Practice tuning

The best way to learn is by doing. The simulation above was run on a stand-alone simulation package called ModelQ. ModelQ is designed to help teach principles of control systems. It runs on Windows 95, Windows 98, and Windows NT. It is available at no charge at {web site}. After installation, just click on "Run" and at top left, select the "Constants" tab and start adjusting constants. If you have questions about the operation of ModelQ, it has an integrated help manual. Finally, if you are wondering about how useful it is to learn on a simulation tool, you should know that this simulation has been compared to a real motor system and the behavior here is demonstrated in field equipment. If you are interested in practicing your controls skills, run the model and give it a try. Becoming skilled at tuning controllers takes time and experience. Like other crafts, the more you do it, the better you get.

Sidebar: Sealing cellophane in food packaging

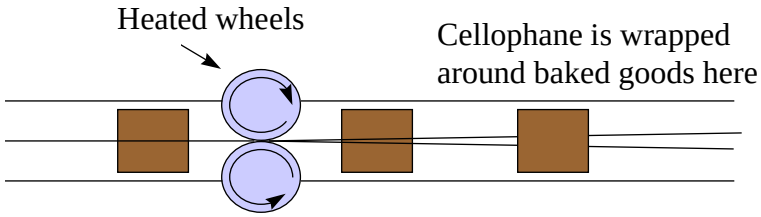
Tuning controllers requires an in-depth knowledge of the application—what the machine can do and what the customer expects. Every application is different and determining the needs is as important as setting the gains to meet those needs. One application is a sealer for a food packaging machine. On this machine, baked goods travel on a conveyor and are wrapped in a continuous roll of cellophane. The cellophane is joined across the top of the baked goods and must be sealed. This is done by two heated wheels which are turning in opposite directions at the same speed, and are synchronized to the conveyor. When the cellophane reaches the wheels, it is drawn between them and the heat on the wheel pinches the two sides together. The process can be seen in Figure A.

For this application, the most important behavior is that the wheels maintain synchronization with the belt. Since no long-term error could be tolerated, an integral gain was required. When the cellophane is drawn into the wheels, it frequently kinks and small folds are introduced. This makes the cellophane thicken and that increases the load on the motors which would otherwise maintain constant speed. However, the thicker cellophane slows the wheels. The tuning parameters K_P and K_I need to be as large as possible to keep the wheels turning at a constant speed. Because the command is near constant, overshoot to a square wave is acceptable and no derivative term is required.



Sealing operation, side view

Cellophane Roll



Sealing operation, top view

This article was the basis for the March, 1999 edition of "20-minute Tune-Up" printed in PT Design Magazine.

Using Bode Plots

March, 1999

Two fundamental characteristics of a well-tuned control systems are stability and responsiveness. Responsiveness is a measure of how quickly a control system moves to follow a command; stability measures the quality of that response. Demanding applications require fast, stable motion. When engineers measure responsiveness and stability, the instrument of choice is usually the oscilloscope. In most cases, the response to a step command is evaluated, as shown in Figure 1. Settling time, how long it takes the system to come within range (say, 3%) of the final value, often characterizes responsiveness; overshoot, the ratio of the peak of the response curve to the final value, characterizes stability.

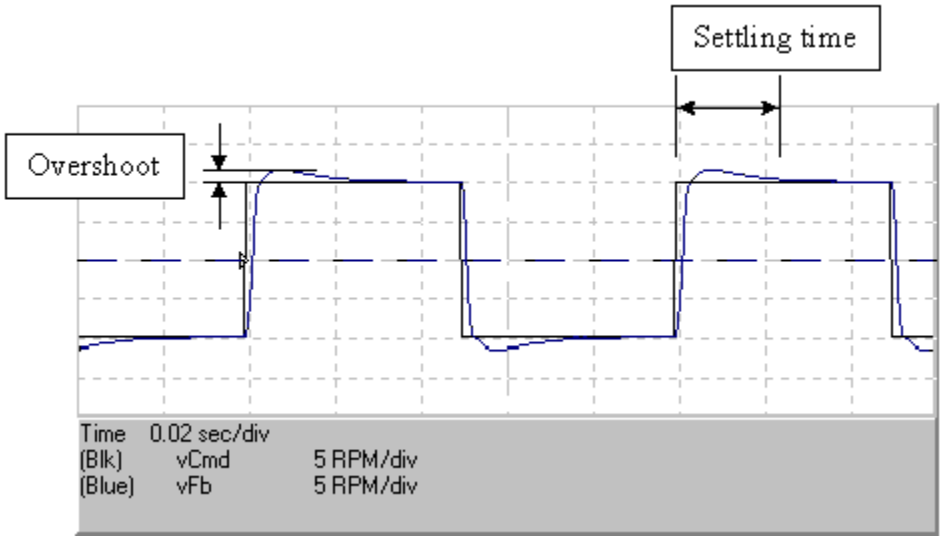


Figure 1: Step response with overshoot and settling time

An oscilloscope measures control system variables as a function of time. That is, it measures in the *time domain*. While the time domain tells a lot about a control system, it doesn't tell everything. To see the complete picture, we also need the *frequency domain*. The frequency domain is based on the principle that when a sinusoid is applied to the input of a control system, the response is another sinusoid at the same frequency. No other repeating waveform does that. For example, the square wave input in Figure 1 produced a response that, although like a square wave, was not square. For most control systems, the rule is "sine in, sine out." Given that, the difference between input and output at any given frequency can be characterized with just two parameters: gain and phase shift.

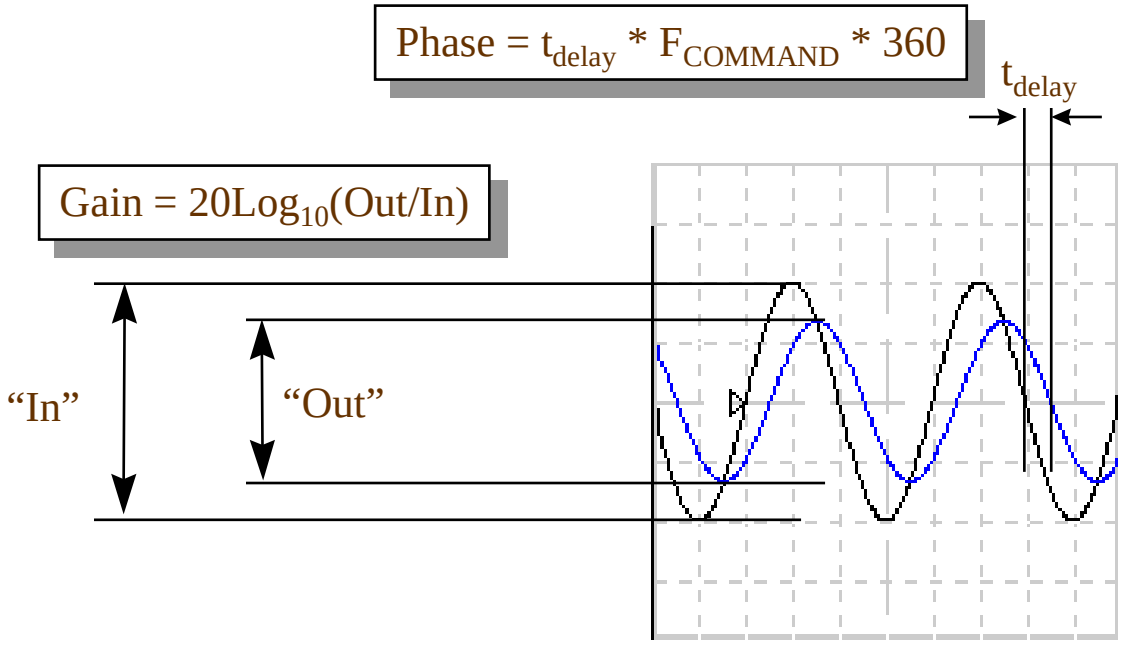


Figure 2: Sine in, sine ou.

A Bode plots measures in the frequency domain much like an oscilloscope measures in the time domain. A Bode plot has two parts: the gain curve and the phase curve. The gain curve displays the ratio of the output to the input in *decibels* or *dB*. Decibels are defined by the equation:

$$\text{Gain} = 20 * \log_{10}(\text{Output/Input})$$

For example, if at, say 20 Hz, the control system were commanded with a 2 volt sine wave and the response was 1.4 volts, the gain would 0.7 or about -3 dB.

Phase displays the time shift of the output in degrees according to:

$$\text{Phase} = t_{DELAY} * F_{COMMAND} * 360$$

So, 20 Hz sine wave command generated an output that lagged the input by 12.5 mSec, the phase would be -90°. Bode plots are shown on a semilog scale with log frequency on the horizontal and gain and phase shown separately on the vertical. Figure 3 shows an example Bode plot.

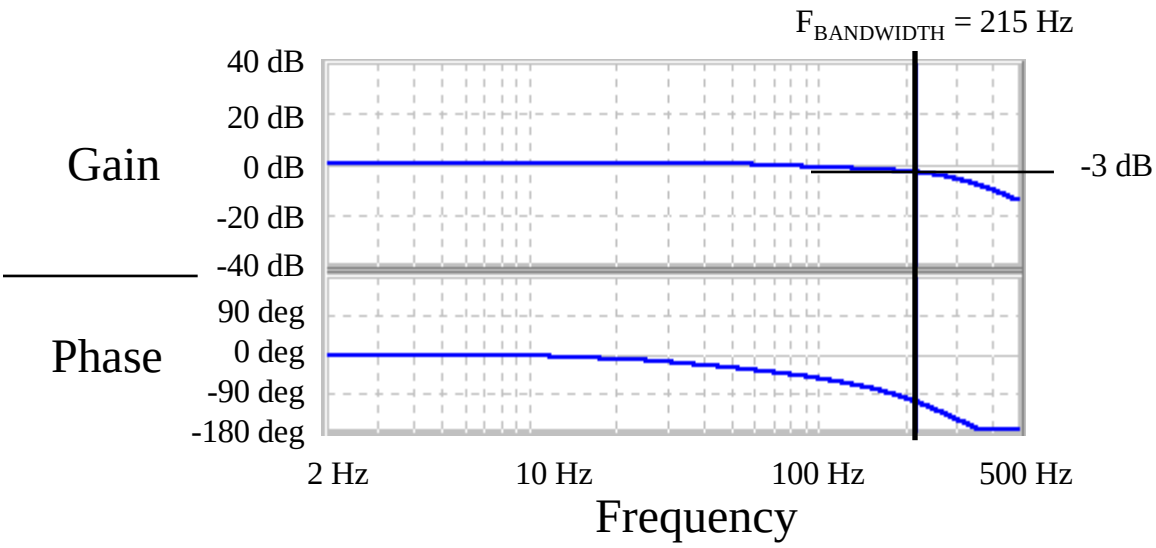
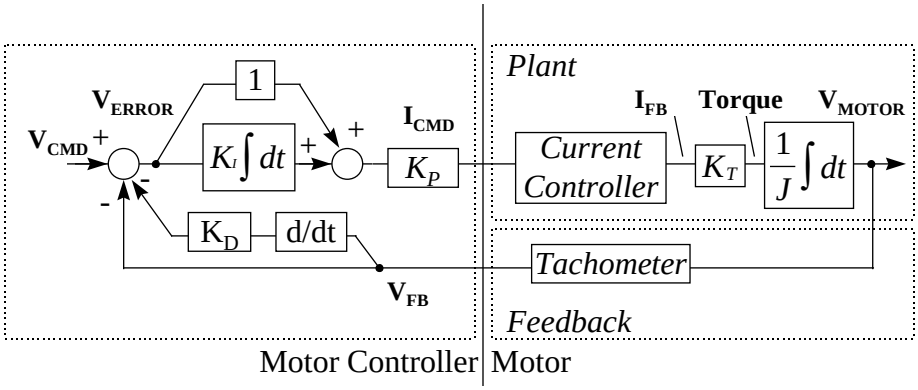


Figure 3: Bode plot from PID controller

In last month's column we discussed the gains of the PID controller shown in Figure 4. The final gains were K_{VP} = 1.3, K_{VI} = 100, K_{VD} = 0.0004. The oscilloscope display of the step response for the final tuning parameters is recalled in Figure 1. Figure 3 is the Bode plot for the same system. In that sense Figure 3 corresponds to Figure 1.



PID velocity control system
Figure 4

Bode plots for motion systems typically have a gain of 0 dB (input = output) at low frequencies and decline at higher frequencies. This is expected: at low frequencies the output will precisely follow the input. At high frequencies, the motion system cannot keep up. This is why gain plots for servo systems usually look like a mountain ridge. They start of 0 dB on the left, stay flat as they move right, and at some frequency start falling off.

Like a scope plot, responsiveness and stability can be read from a Bode plot. Most of this information is in the gain plot. Responsiveness is usually measured by *bandwidth*. Bandwidth is the frequency where the gain falls to -3 dB (i.e., where Output = 0.7 * Input) as shown in Figure 3. Higher bandwidth corresponds to faster settling times. Stability is measured in *peaking*, the characteristic where the gain plot first climbs a few dB before starting to fall. Peaking in the Bode plot corresponds to overshoot in the scope plot. More than 1 or 2 dB is usually considered excessive; the system in Figure 3 has less than 1 dB of peaking.

Bode plots are useful at several levels. First, they provide understanding about control theory. For example, in the coming months Bode plots will be used in this column to explain how servo gains work and

where problems such as mechanical resonance originate. In this case, having the ability to simulate Bode plots is sufficient. However, the best situation is to have the instrument to measure the Bode plot on the actual machine. This instrument, called a Dynamic Signal Analyzer or DSA, is a relative of the spectrum analyzer. They are available from numerous companies including Hewlett-Packard, Schlemberger, and DSP Technologies. Be warned though that measuring Bode plots takes a certain commitment. It's often a challenge to connect these devices into motion systems.

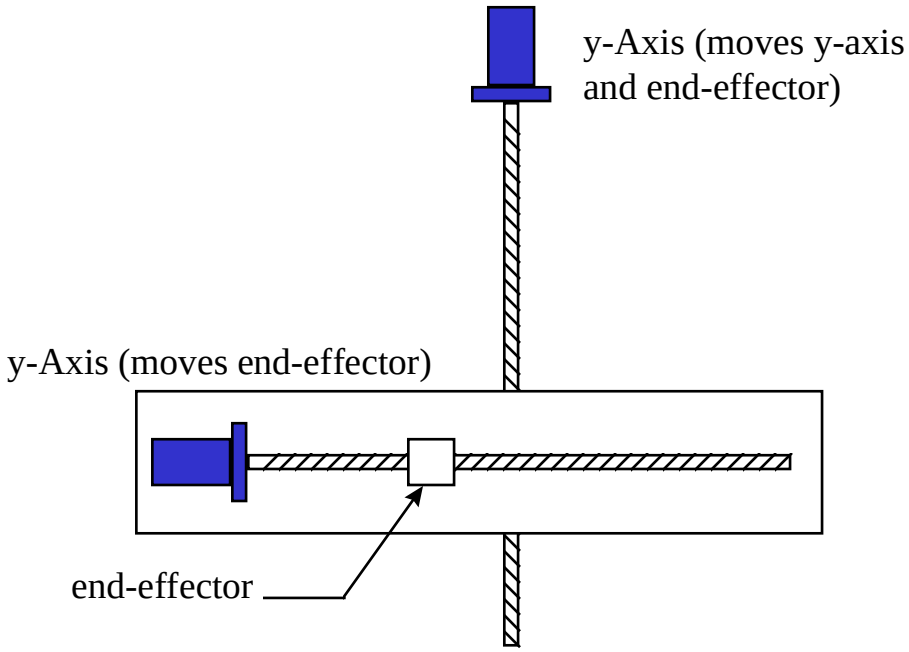
Practice tuning

The best way to learn is by doing. The simulation above was run on a stand-alone simulation package called *ModelQ*. It is available at no charge at {web site}. After installation, click "Run" and at top left and select the March model from the combo-box at top center. Select the "Constants" tab and start adjusting KVP, KVI, and KVD. After you have entered a set of constants with a good time-domain response, run a Bode plot by pressing "GO" near center-bottom. *ModelQ* will automatically find the -3 dB point for you by clicking the auto-find button with the magnifying glass near the bottom right of the screen. Continue to experiment until you see the relationships between bandwidth and settling time, and between peaking in the Bode plot and overshoot.

Because the example system is not tuned as tightly as possible, you should be able to improve the responsiveness without causing instability. Put another way, you can increase the bandwidth without causing peaking. However at some frequency, in this case about 200 Hz, you should notice that this is no longer true. As with almost all servo systems, there is a frequency above which you cannot increase the bandwidth without reducing stability. This trade-off of responsiveness and stability is difficult. Proper tuning requires an in-depth knowledge of the application and of motion control systems. Remember that most servo machines close position loops around velocity loops; we have only discussed the velocity loop so far. However, the implications discussed here are similar whether the system is in velocity or position control. At some point the engineer must trade stability off against responsiveness.

Sidebar: Pick and Place

Pick-and-place machines demand high response. These machines are used in electronic board assembly and in numerous *back-end* semiconductor processes (processes after the wafer has been diced.) Often, the most demanding axes of these machines are the x-traverse and y-traverse axes. These axes move an end effector such as a part holder (see Figure). Depending on the process, the x and y axes may move the part from a bin to a position on the table, or from the table to a bin. Because the x and y axes are often a limiting factor in overall machine speed, this application demands that these axes be tuned for very high bandwidth, frequently over 200 Hz. Because these machines move many thousands of parts per hour, saving just a few milliseconds in settling time (that is, raising the bandwidth) can significantly improve the overall speed of the machine.



Simple x-y stage

(Note to editor: this sketch shows two motors connected to lead screws)

Adding difficulty, many pick-and-place operations require an elimination of overshoot. This may be because a part is placed next to a barrier and overshoot may cause contact between the barrier and the part. Put together, the requirements for very high bandwidth and high stability make pick-and-place machines a demanding application.

This article was the basis for the April, 1999 edition of
"20-minute Tune-Up" printed in PT Design Magazine.

PDFF Velocity Control: Extending PI

April, 1999

Last months, we discussed PID velocity control. These controllers are common in analog motor drives. They are simple to implement and they provide good performance. However, tuning PID velocity controllers is complex because it's iterative —first you tune "P", then "I", then add some "D" to cure overshoot, then add more "P", and so on.[1] The process can be unwieldy on the factory floor.

In digital systems, many manufacturers use PI (proportional-integral) velocity loops, eliminating the derivative ("D") term. PI loops are easy to tune: first adjust "P", then "I; you're done. PI is ideal for applications that demand the maximum responsiveness such as pick-and-place machines. But PI control has a weakness—it does not provide good low frequency "stiffness."

Stiffness and Disturbance

Stiffness measures how well a system overcomes disturbances. Disturbances come in many varieties. Friction is one example. Another, which occurs in printing machines, is the force generated when a printing plate contacts the paper. Disturbances test the controller; they try to move the motor away from its commanded velocity. When a control system does a good job overcoming disturbances, we say it is "stiff."

Friction is a special disturbance because it is a low frequency effect. Often it is important that the controller be stiff in the presence of low-frequency disturbances like friction. In integrating velocity controllers, low frequency stiffness comes from the integral gain. The problem with PI is that integral gain must remain relatively small to avoid excessive overshoot. PDFF velocity control was developed to combat this problem.

PDFF Velocity Control

Figure 1 shows a PDFF controller. Like the PI controller, it has an integral gain (K_{VI}) and a proportional gain (K_{VP}). PDFF adds the gain K_{VFR}, which allows the user to raise the integral gain in some applications. When an application requires the maximum responsiveness, you don't need much integral gain. Here, you set K_{VFR} high. (Although it's not easy to see in Figure 1, when you select K_{VFR} = 100%, PDFF is identical to PI!) When the application requires maximum low-frequency stiffness, set K_{VFR} low; this allows much higher integral gain without inducing overshoot. Unfortunately, it also makes the system slower in responding to the command. The majority of motion control applications are in the middle and K_{VFR} = 65% usually gives good results.

[1] This discussion applies to PID velocity control, not PID position control. PID position control is a technique that will be discussed in a future column.

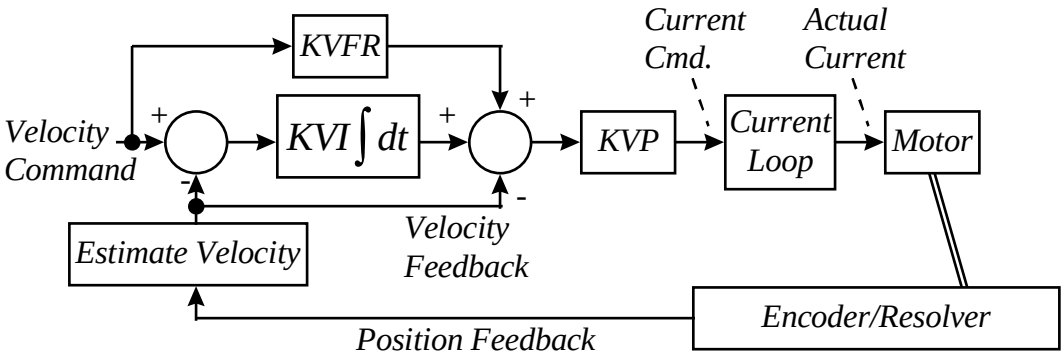


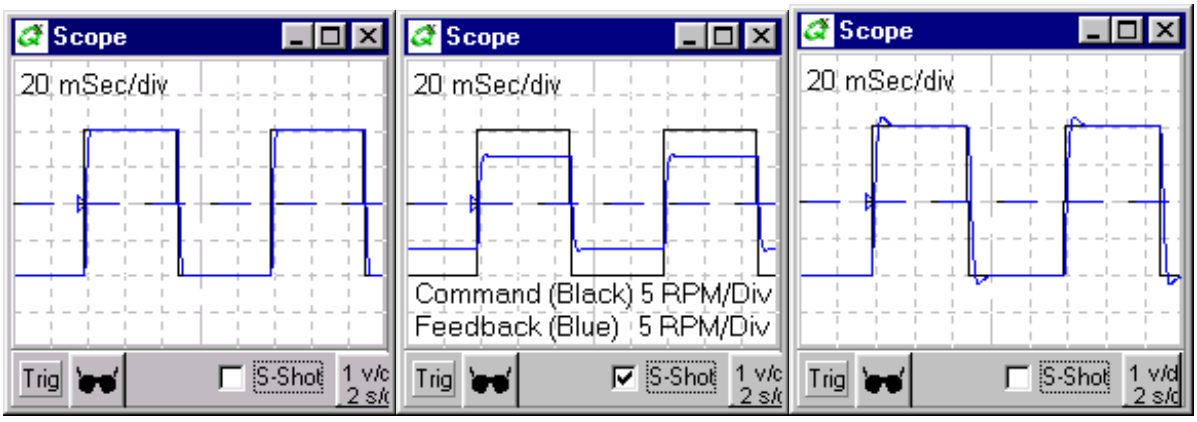
Figure 1: Block diagram of PDFF-based velocity loop.

Tuning PDFF

The following three-step procedure shows how to tune PDFF velocity control:

1. Tune K_{VP}
   Command a step velocity. Temporarily configure PDFF as a proportional-only controller by setting K_{VFR} = 100% and K_{VI} = 0. Raise K_{VP} as high as possible without causing overshoot or oscillation.
2. Select K_{VFR} for the application.
   - Set K_{VFR} to 100% for applications which require the greatest response
   - Set K_{VFR} to 65% for general applications
   - Set K_{VFR} to 0% for applications which require the greatest low-frequency stiffness
3. Raise K_{VI} for 5% overshoot.

Figure 2 compares the command and the feedback for each of these steps for a PDFF controller. First K_{VP} was raised to just below the level causing overshoot (K_{VP}=1.2). Then K_{VFR} was set to 65%, assuming a general application. Finally, K_{VI} was raised for 5% overshoot (K_{VI} =220). Use this same process for PI control, but skip step 2—remember PI is just the special case of PDFF where K_{VFR} = 100%. PI works well in applications where low-frequency stiffness is not critical.



Step 1: K_{VFR} =100, K_{VI} = 0. Raise K_{VP} to max (Value = 1.2)

Step 2: Select K_{VFR} for application. (Value = 65%)

Step 3: Raise K_{VI} for 5% overshoot. (Value = 220)

Figure 2: 3 Steps for tuning PDFF

Practice tuning

The simulations above were run on a stand-alone simulation package called *ModelQ*. It is available at *www.ptdesign.com*. After installation, click "Run" and at top left, and select the April model from the combo-box at top center. Select the "Constants" tab and start adjusting K_{VP}, K_{VI} and K_{VFR}. After you have entered a set of constants with a good step response, run a Bode plot (press "GO" near center-bottom) and check the bandwidth. Try tuning K_{VI} with different values of K_{VFR}. Notice that increasing K_{VFR} causes overshoot which can be removed by reducing K_{VI}. Draw more Bode plots and watch the bandwidth go up as you increase K_{VFR}. By the way, you can also measure the responsiveness by watching the settling time to a step. As the bandwidth goes up, the settling time goes down.

Application

Applications where a motor drives a worm gear often require maximum low-frequency stiffness. This is because worm gear mechanisms usually have a lot of friction. Without a high integral gain, worm gear systems pull in to position at the end of a move slowly. When people use PI controllers for worm-gear mechanisms, they sometimes are not pleased with the long settling times the sometimes result. PDFF control can help this problem because reducing K_{VFR}=0 allows high integral gains, usually 8 - 10 times larger than the integral gain of a PI controller.

This article was the basis for the June, 1999 edition of
"20-minute Tune-Up" printed in PT Design Magazine.

Measuring the Margin

June, 1999

In last month's column, we discussed control loop stability. Simply put, instability comes from a sign reversal in the loop. This sign reversal comes from an accumulation of phase and gain around the control loop adding up to -1, or a gain of 0 dB at 180°. This month we will continue to investigate instability by developing phase margin, a numerical measure of stability.

What is Phase Margin?

Phase margin (PM) is one way of measuring stability. PM is measured at a frequency called the "gain crossover;" that's the frequency where the gain is 0 dB. For control systems, there will always be a frequency where the gain is 0 dB. The loop gain for control loops is usually high at low frequency, and low at high frequency. (For motion systems this means it is a lot easier to move inertia at low frequencies.) So somewhere in the middle frequencies, the gain crosses through 0 dB; that's the gain crossover frequency.

PM measures how far the loop phase lag is from 180° at the gain crossover. If the phase were 180° at gain crossover, the system would be unstable. The margin of stability is how far the actual phase is from 180°. So, PM is defined as follows:

$$PM \equiv \text{Loop Phase}_{\text{Gain Crossover}} + 180°$$

If PM is zero, the system is unstable. If it's too low, the system will be marginally stable: it will overshoot excessively and ring. The right amount of PM varies, but it usually falls between 50° and 80°.

The example system of Figure 1 shows how PM is measured. At one frequency, here 60.5 Hz, the phase and gain around the loop add up to 0 dB at -114°. Since the gain is 0 dB, we know this is the gain crossover frequency. The difference between the actual phase (-14°-6°-90°-4° = -114°) and instability (-180°) is the PM (66°).

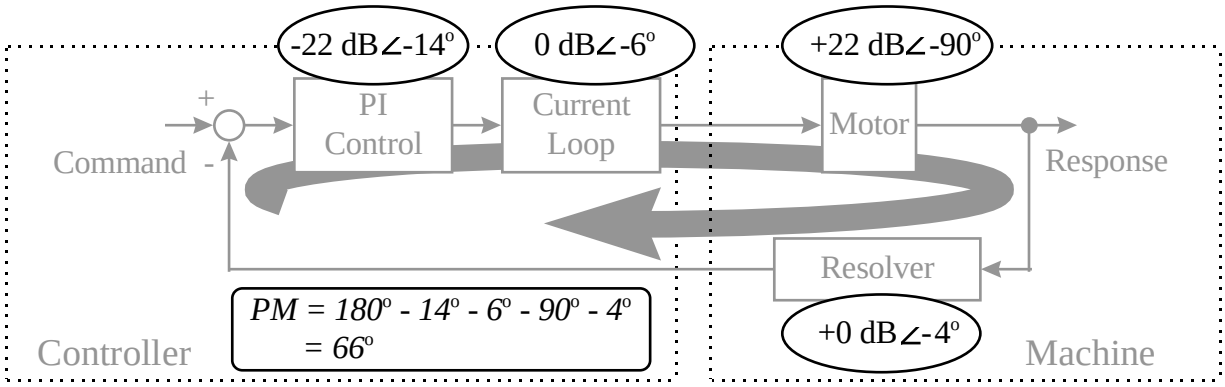


Figure 1: Control loop measured at 60.5 Hz, the gain crossover frequency for this system

Finding gain crossover requires a plot of the loop phase and gain, often called "the open loop." This is just the sum of the gain and phase around the loop over a wide range of frequencies. "Open loop" is really a

misnomer—we don't open the loop in modern controllers. Figure 2 shows the "open loop" Bode plot for the system in Figure 1.

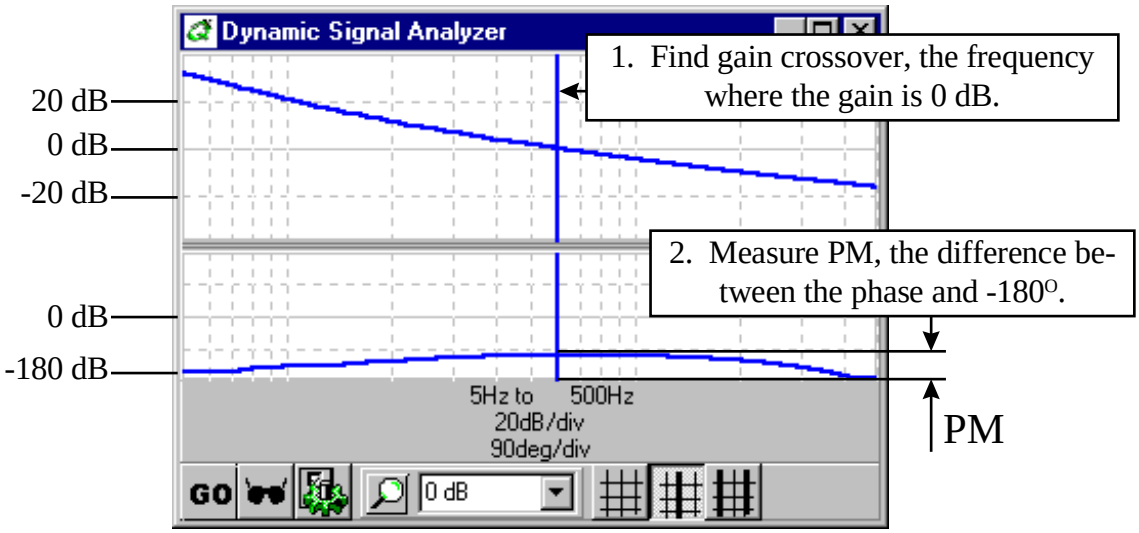
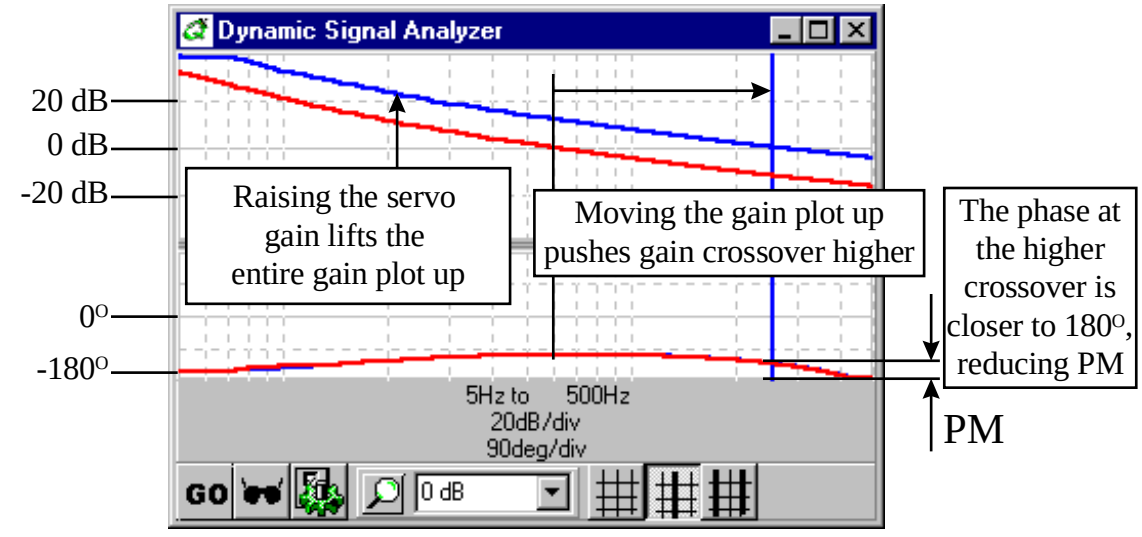


Figure 2: Open Loop" plot with PM measured

Use PM to understand tuning

So let's use PM to understand tuning a little better. Consider our control system from Figure 1. If we can increase the gains, we enjoy the benefit of a more responsive system. But increasing the gain also causes instability as Figure 3 explains. Here there are two "open loop" gains. The red trace shows the system with the original servo constants and the blue one shows those gains increased by a factor of 4 which is equivalent to 12 dB. Increasing the servo gain by 12 dB has the effect of increasing the loop gain by 12 dB. This is why the blue trace is 12 dB above the red trace. Raising the loop gain has the effect of moving the crossover frequency to the right, here to about 250 Hz. At 250 Hz, the phase is 140°, so the PM falls to 39°.



Open Loop" plot with PM measured

So, what's the effect of dropping the PM from 66° to 39°? Figure 4 shows the effect in the time domain. The original (PM = 66°) system is in Figure 4a and Figure 4b shows the raised servo gain (PM = 39°). The loss of PM would make the system in Figure 4b almost useless in most industrial applications.

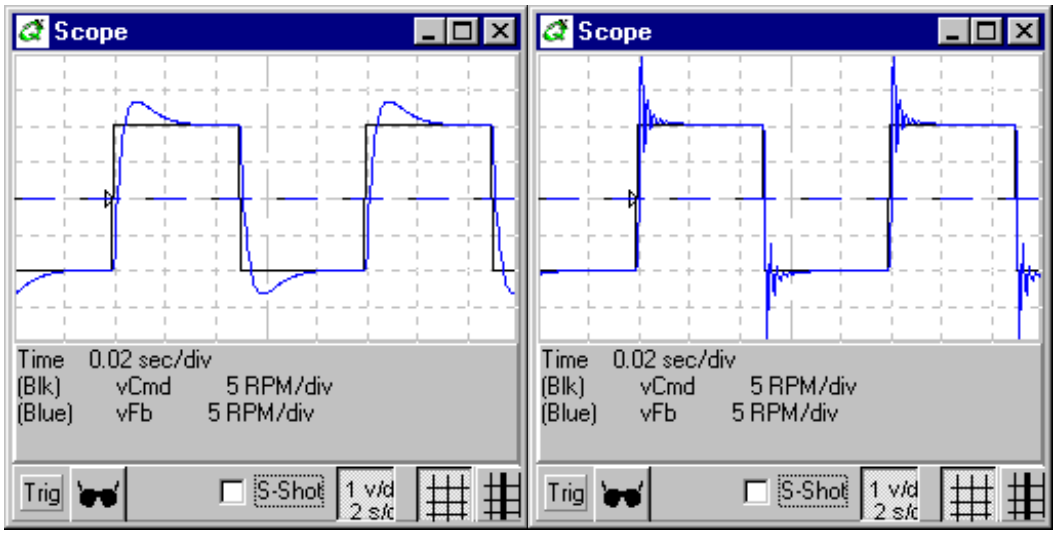


Figure 4 (a) Original gains (PM = 66°) and (b) with 4x higher gains (PM = 39°)

Experiment

The best way to learn is by doing. A simulation for the above system was run on a stand-alone software package called *ModelQ*. It is available at no charge at www.ptdesign.com. After installation, click "Run" and at top left and select the May/June model from the combo-box at top center. You can generate a Bode plot of the loop gain by pressing "GO" near the center bottom. Click "Open" near the center of the screen to see the "open loop" gain. In a few seconds a Bode plot of the loop gain will appear on the dynamic signal analyzer (DSA). Notice that the gain (top plot) crosses 0 dB at about 60 Hz, just as it did in the discussion above. Turn on the cursors and move the cursor to the gain crossover selecting "0 dB" in the combo box next to the magnifying glass and then clicking on the magnifying glass. Notice that the cursor box shows the frequency, phase, and gain. Notice that the phase is 115° indicating a PM of 65°.

Now raise the gain K_{VP} from 0.72 to 2.88, a factor of four or 12 dB. Run a new Bode plot. Notice how the gain plot moves up 12 dB and this moves the gain crossover frequency up. Click on the magnifying glass again. Notice that crossover is now at 254 Hz and the phase is -141°indicating 39°PM. Also note how the scope plot at left shows that the stability is degraded by the reduction in PM.

Application

Applications that demand the highest performance need to reduce unnecessary phase lag in the system. Any reduction in phase lag will increase the PM. In our simple example, just changing the feedback device from resolver to encoder would have helped the PM. The phase loss by the resolver was just 4° at 60 Hz, but phase lag often grows as frequency increases. The phase lag at 254 Hz from the resolver is 20°. One reason why the highest responding systems are more likely to use encoders is that they add almost no phase lag. Control system designers can take other steps to reduce unnecessary phase lag. For example, use high response current loops, high sample rate digital drives, and take out unnecessary noise filters. Often in control systems, filters are used to cover over problems like poor wiring or low-resolution encoders. Filters cover over the noise but at the expense of adding phase lag. Using filters wisely helps ensure that the system will achieve the response rate of which it is capable.

This article was the basis for the July, 1999 edition of
"20-minute Tune-Up" printed in PT Design Magazine.

Encoder Resolution and Noise

July, 1999

Servos can be noisy. One of the main culprits is the resolution limitations of the position feedback device which may be either an encoder or a resolver coupled with a resolver-to-digital (R/D) converter. In both cases, the servo controller knows the position imperfectly--to within a *least-significant bit* or LSB. When the servo controller tries to estimate velocity from that position signal, noise pulses are generated. The control loop generates current spikes from the pulses and those spikes cause noise.

Theory

Most modern servo controllers sense position rather than velocity. Because position is fed back digitally from an encoder or resolver with an R/D converter, the signal is quantized to a least-significant bit or LSB. For example, a 1000-line encoder can provide up to 4000 counts for each revolution of the motor using a technique called "quadrature." The resolution of such an encoder is 1/4000 of a revolution. Figure 1 shows a true position (blue trace) vs. a quantized position (brown).

Digital drives cannot watch the position continuously. Instead, they can only *sample* the position at regular intervals, say, every 250 µs. At each sample, most drives estimate velocity from position using a simple formula: $V \approx (P_N - P_{N-1})/T$ where P_N is the most recent position, P_{N-1} is the position from the last sample, and T is the interval between samplings.

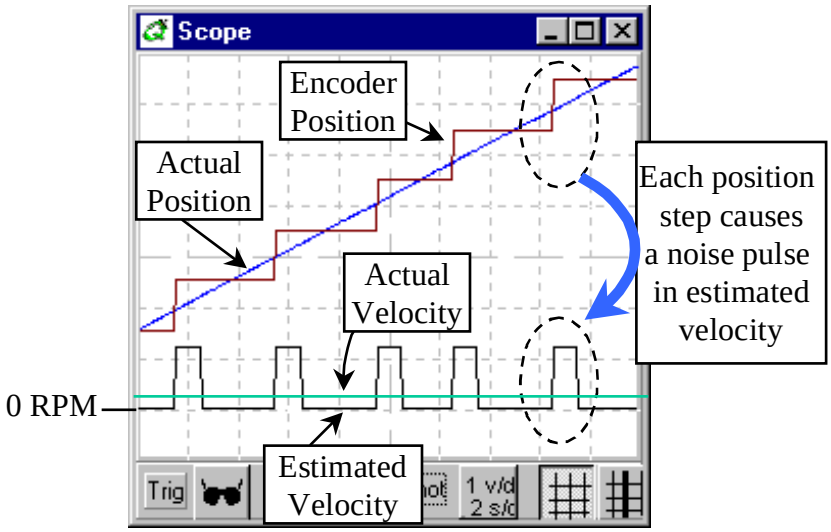


Figure 1: Quantized position and the effects on velocity estimation.

How resolution effects the system

The technique of estimating velocity from the two positions works well except for the problem of resolution. As Figure 1 shows, the estimated velocity (black trace) is a noisy version of the true velocity (green trace). The noise from the velocity estimation travels though the servo loop and produces current spikes. Those current spikes are responsible for most of the noise in many servo systems.

Resolution noise is generated from the proportional gain in a velocity loop. In fact, for the purposes of understanding resolution noise, we can think of the servo loop as a simple proportional controller as shown in Figure 2. While position loops, integrating velocity loops, and full PID positions are much more

common in industry, only the proportional gain generates noise. Our simple loop in Figure 2 forms a velocity error and then amplifies it with K_{VP} and filters it.

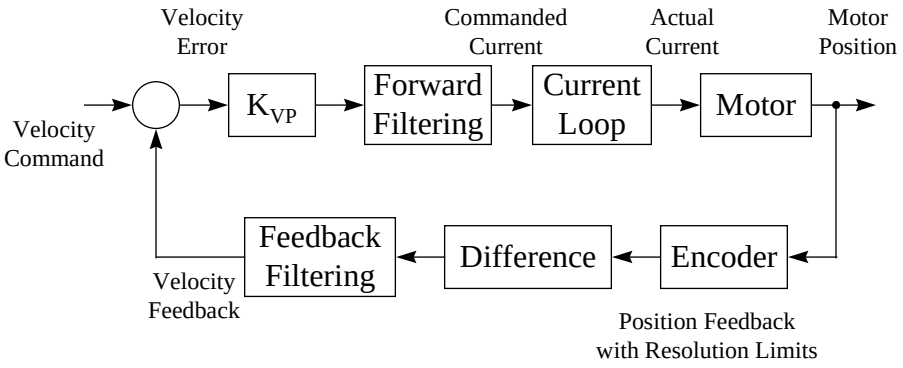


Figure 2: The proportional velocity loop

Fast response means more noise

The most responsive feedback systems require the largest gains. When a servo system has to move quickly, K_{VP} will be relatively large. All control loops have a proportional gain, even those not called K_{VP}. For example, PID position loops use the "D" gain for the same purpose as our velocity loop K_{VP}. So, responsive systems have a large K_{VP} (or equivalent), and K_{VP} amplifies noise. That is one reason the most responsive servos need the highest feedback resolution.

High inertia means more noise

Large load inertia also means the system will be noisier. Think of a motor as a control element. When the inertia gets large, the motor gain is low; it's hard to accelerate. When you have a low motor gain, you must have an even larger electronic gain to compensate. Raising inertia forces raising K_{VP}, making the system noisier. Systems that are highly responsive or have large inertias need more highly resolved feedback signals.

Filtering the noise

An easy way to help the noise problem is to add filtering in the loop. Filtering can be added in three places: filter the feedback signal directly, filter the signal in the forward path, and use the current loop as an implicit filter. These filters are shown in Figure 3.–Filtering reduces noise amplitude of the current either directly (forward filter and current loop), or indirectly filtering the velocity signal (feedback filter). Unfortunately, filtering also causes overshoot and instability. If filtering cannot fix the noise problem without degrading servo performance, designers must increase the resolution of the feedback.

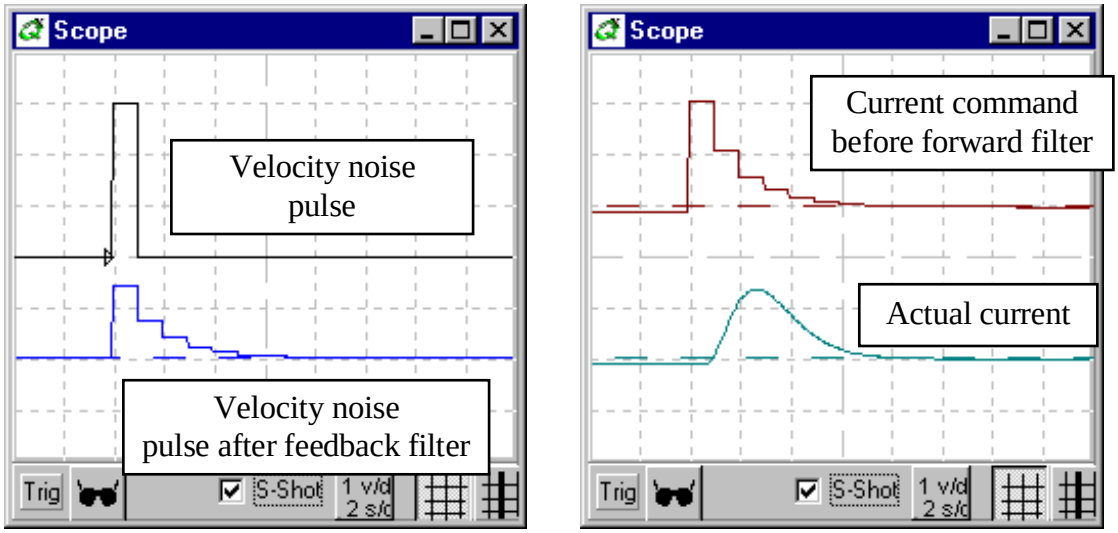


Figure 3: Noise reduction a) from feedback filter and b) from current filter and current loop.

Laboratory

The best way to learn is by doing. A simulation for the above system was run on a stand-alone software package called *ModelQ* which is available at no charge at www.ptdesign.com. After installation, click "Run" at top left and select the July model from the combo-box at top center. The controller shows a step response. The command (black) and response (blue) are at the top and the current (brown) is at the bottom. The more noise on the current signal, the noisier the servo system. The default configuration uses essential infinite resolution (10^{10} lines) and so there is no measurable resolution noise. However, when you input a realistic value for resolution (2500, for example), you will see the noise grow.

Notice also that when you increase the inertia (by double, for example) you need to increase the K_{VP} gain by the same amount to maintain the servo performance. When this occurs, servo performance is restored, but noise increases proportionally with the gain. System responsiveness can be increased by raising the gains. For example, K_{VP} can be raised to about 1.1 before servo (that is, non resolution) problems start to occur. But raising K_{VP} also increases noise, demonstrating that larger inertia and higher responsiveness increase system sensitivity to resolution noise.

You can add in filters and change the filter frequencies for the forward filters (LPF1, LPF2) and the current loop. The feedback filter is fixed at 400 Hz. Notice how using more filters, and using filters with low bandwidths (say 150 Hz), helps the noise but degrades the servo performance.

Application

Applications that demand the highest responsiveness or have very large load inertias need high resolution. If enough filtering to solve the noise problem were used, the required response rate could not be reached Fortunately, there are other techniques to increase resolution. If an encoder is used, increase the line count. Sometimes this may not be an option because encoder top speed is reduced by higher line counts. Some servo controllers support 1/T or clock pulse counting, a technique which increases resolution by improving on the simple difference. If using a resolver, interpolate an analog “ILSB” signal from some R/D converters to increase resolution. A final option is using a sine encoder, which allows division of each encoder line into 1000 positions, increasing the resolution well beyond any~~d~~ equivalent standard encoder.

Reference

Application Note: Kollmorgen ASU010H

This article was the basis for the October, 1999 edition of "20-minute Tune-Up" printed in PT Design Magazine.

Mechanical Resonance

October, 1999

Theory

Mechanical resonance is one of the most perplexing problems in motion control. It shows up when the system is being commissioned. Usually, you are tuning the controller, raising the servo gains bit-by-bit. Performance is steadily improving as the gains go up. But, when you try to raise the gains a bit more, the machine starts making noise. Some machines produce a pure tone, similar to the pitch from a tuning fork. Other machines start growling or may go unstable, blaring with an unpleasant sound similar to a fog horn. These are the signs of mechanical resonance.

Resonance is usually caused by compliance or "springiness" between a motor and its load. The motor and load are usually connected by transmission components such as gear boxes, belts, couplings, and lead screws. As shown schematically in Figure 1, each of these components is like a spring; they all twist a little when the motor applies torque. So, while the load is connected to the motor, the connection is not stiff. This lack of stiffness is at the root of many servo problems.

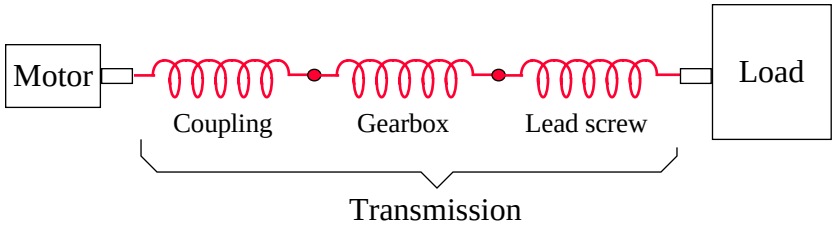


Figure 1. Springiness, shown here schematically, causes most resonance problems.

Figure 1 shows a "two-mass" system; the two masses (motor and load) are connected by a series of springs. Every two-mass system has one frequency where it wants to oscillate; that's its resonant frequency. The resonant frequency is normally well above where the servo controller operates. Resonance worsens as the servo gains rise because higher gains allow the servo controller to reach into higher frequencies. When the servo system reaches to the resonant frequency, the machine starts to resonate. One cure of resonance is raising the resonant frequency. That usually means stiffening the springs.

Stiffer is better

The machine is stiffened by using less compliant components, for example, replacing compliant couplings such as helical couplings with stiffer "bellows" couplings. Shorter, stouter shafts and lead screws can replace their longer, thinner counterparts. Stiffer gearboxes can be used. Using wider belts, belts with steel or Kevlar banding, shorter belts, or multiple belts in parallel increase the stiffness of belt drives. In addition, stiffening the frame of the machine helps reduce resonance problems. When there are several springs in series, the total spring constant, K_{TOTAL}, is:

$$K_{TOTAL} = \frac{1}{\frac{1}{K_{COUPLING}} + \frac{1}{K_{GEARBOX}} + \frac{1}{K_{LEADSCREW}}}$$

In this equation, the most compliant couplings dominate the stiffness. That's why it's important to stiffen the most compliant components. There's little benefit from stiffening the gearbox if the coupling is so compliant that it dominates total spring constant. Fortunately, most vendors of transmission components specify compliance, so looking at the catalog will reduce the guesswork. As you can see from Figure 2, increasing the total machine stiffness from 2000 Nm/Rad to 6000 Nm/Rad can improve resonance quite a bit.

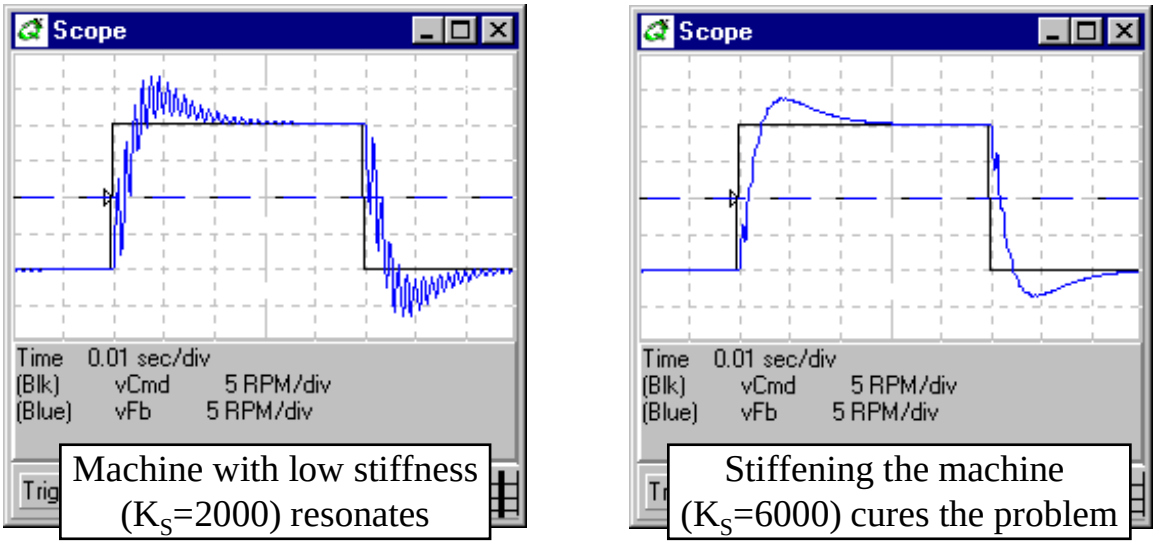


Figure 2. Increasing the mechanical stiffness of a machine often cures resonance problems.

Try enlarging the motor

Another way to reduce resonance problems is to reduce the ratio of load-to-motor inertias. A small motor is more susceptible to resonance; it's harder to control when it's driving a load many times its size. Increasing the motor inertia is usually the easiest way to improve this ratio. That's because by the time resonance problems are identified, the machine is designed and the load inertia has, in most cases, been minimized. Most servo applications will work well if the load is no larger than two times the motor, but the most responsive applications can limit the load to just 70% of the motor. It's prudent, even for the least demanding applications, to limit the load inertia to no more than ten times the motor inertia.

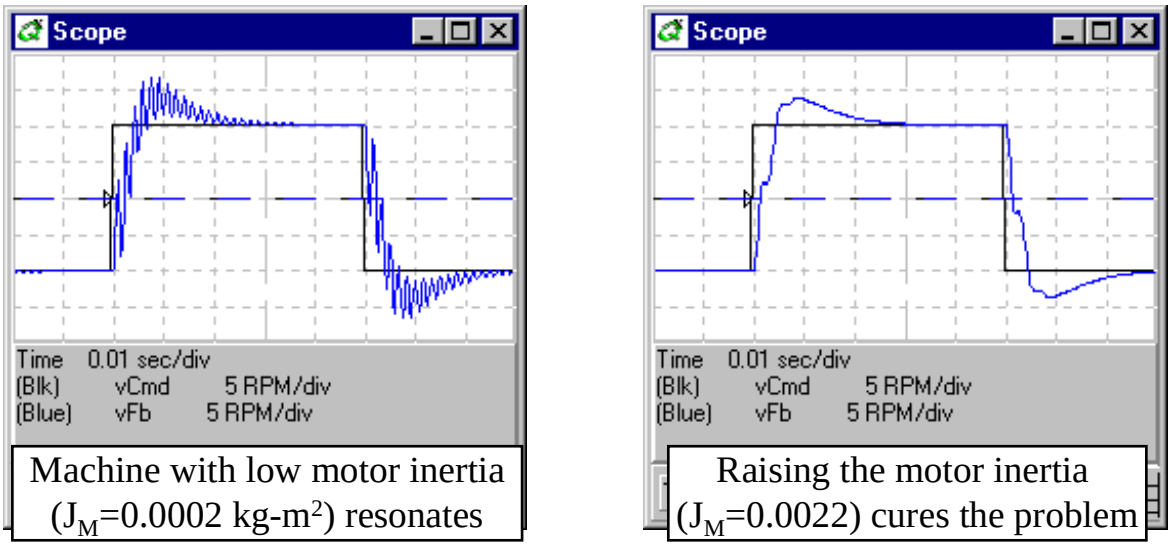


Figure 3. Increasing the motor inertia also can cure resonance problems.

Increasing the motor inertia is not a panacea. The total inertia that has to be accelerated increases and that can be a problem in applications that have fast acceleration/deceleration cycles. The example in Figure 3 doubles the total inertia by increasing the motor inertia.

Next step: electrical options

After the mechanical options have been exhausted, it's time to look to the electrical options. The easiest step is to reduce the servo gains. Lowering servo gains, such as the gains of a PI velocity loop, will help resonance. Actually, for a PI (proportional-integral) velocity controller, resonance problems are due almost exclusively to the "P" gain; for PID position loops, they come from the "D" gain. Reducing those gains will eliminate resonance, but will also degrade the servo performance. The system will respond slower to the command, and it won't react as quickly to mechanical disturbances. A better alternative for most applications is the use of filters.

Low pass filters, placed after the velocity loop, often help resonance by limiting the frequency range of the servo controller. That's similar to reducing servo gains, but low-pass filters have much less negative impact on servo performance. Be aware that using filters will usually cause overshoot. For example, notice how the low-pass filter generates more overshoot in Figure 4 compared to the mechanical cures (Figures 2 and 3.)

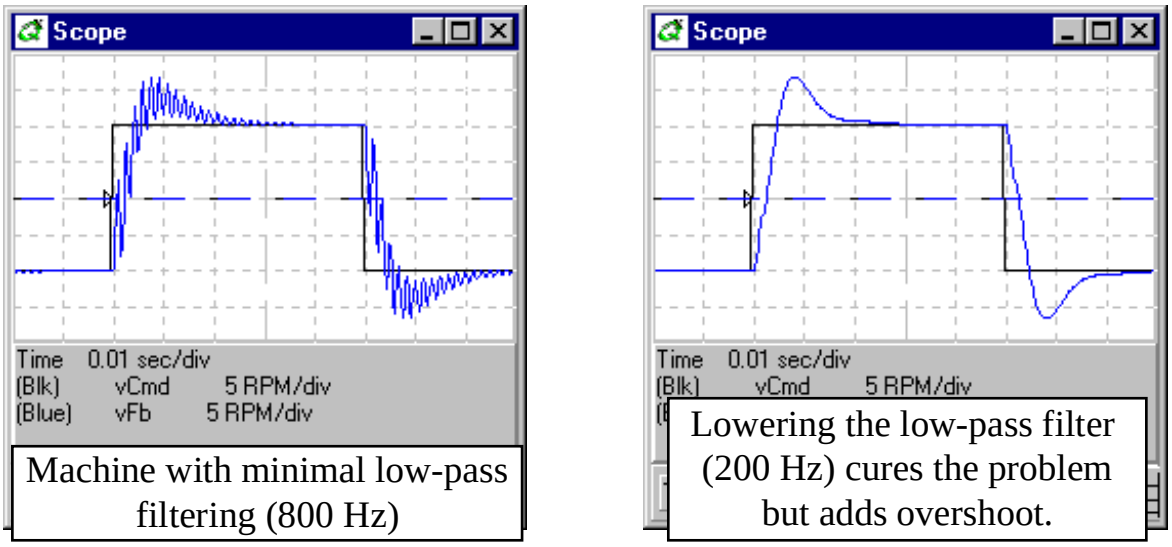


Figure 4. Increasing the motor inertia also can cure resonance problems.

Laboratory

The best way to learn is by doing. A simulation for the above system was run on a stand-alone software package called *ModelQ*, available at no charge from www.ptdesign.com. After installation select the October model from the combo-box at top center and click "Run." You can adjust the stiffness (K_S), the load (J_L) and motor (J_M) inertias, the low-pass filter frequency (LPF), and the servo gains of a PI velocity controller (K_{VP} and K_{VI}). Note that changes in the inertia must be compensated for by the use of higher K_{VP}; when J_M went up 11 times, it increased the total inertia ($J_M + J_L$) by two times, so K_{VP} must go up by two times to maintain servo performance. Each of the figures in this article can be produced by modifying the model gains according to the following table:

Figure	Cure	K_S	J_M	K_{VP}	LPF Hz
2 (Resonating)	(Not cured)	2000	0.0002	0.72	800
2 (Cured)	Stiffen machine	**5000**	0.0002	0.72	800
3 (Cured)	Raise motor inertia	2000	**0.0022**	**1.44**	800
4 (Cured)	Lower LPF frequency	2000	0.0002	0.72	**200**

Table 1. Three ways to cure resonance.

Application

Machine tools are usually built with high mechanical stiffness. The use of large diameter lead screws, stiff gearboxes, and strong couplings all contribute to a stiff coupling between motor and load. This allows machine tool manufacturers to raise their servo gains. High servo gains translate to a more valuable machine. They can cut more parts in an hour because the axes respond faster, and the finishes are smoother because the servos reject disturbances during the cutting process.