This article was the basis for the February, 2000 edition of "20-minute Tune-Up" printed in PT Design Magazine.

Controlling Current

February, 2000

Background

Electric servo motors need current to produce motion. Permanent magnet (PM) motors are built with a combination of magnets and current-carrying coils. The electro-magnetism produced by the current interacts with the field from the magnets to produce the torque that moves the motor. If you want well-controlled torque, you have to have well-controlled current.

Electric motor drives (sometimes called inverters or amplifiers) use power transistors to deliver energy. These transistors produce voltage; current is generated by the electrical circuit that is formed from the drive voltage and the motor windings (see Fig. 1). Because the power transistors produce voltage, a current loop is required to achieve precise control of current. A current loop compares the current command to the feedback and adjusts the drive voltage to minimize the error. Not all drives use current loops. Many drives output a voltage and rely on motor parameters to limit the current. This works in some applications, but usually not with higher power (> 500 W) or on more demanding machines.

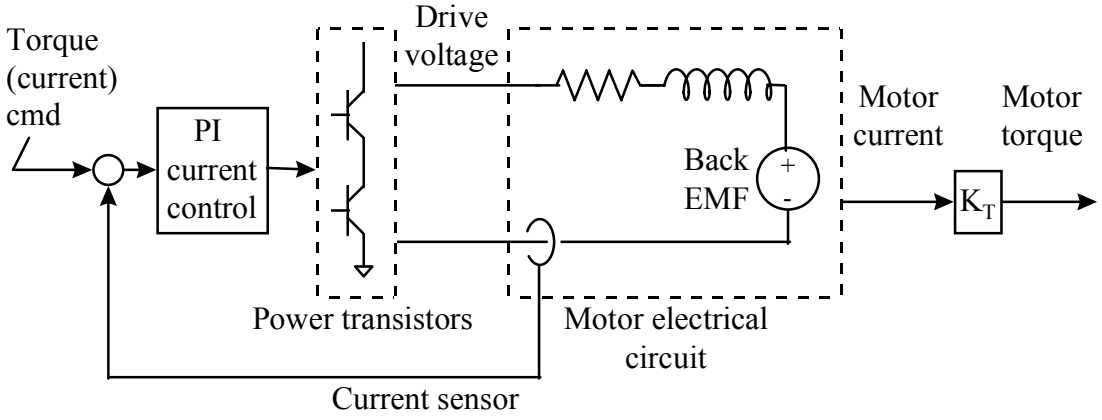


Figure 1. The current loop of an electric servo motor.

Controlling torque with a brush motor is straight-forward. The motor produces torque that is approximately proportional to the winding current; that constant of proportionality is called the torque constant, or K_T. With brush motors, the current command is independent of motor position or speed. With brushless motors, controlling torque is more complicated. For the smoothest torque, the drive must produce sine waves of current in three motor windings in a process called *commutation*. Commutation includes the algorithms to determine the right amount of current for each winding based on the motor position and speed, and the amount of torque needed at that instant. Commutation is complicated enough that a digital signal processor (DSP) or microprocessor is usually

required. Torque is no longer proportional to current in any one winding as it was with the brush motor, but similar relationships are present: the servo loop commands torque, and the drive applies voltage. A current loop is still the best way to regulate the voltage in order to produce the currents that make the torque.

When the electric servo motor industry started, drives were analog. As the years have progressed, drives have converted to digital control. Early on, commutation and, then, velocity loops were implemented digitally; current loops are among the most complex operations carried out in drives and, accordingly, were the last to become digital. One reason is that these loops must be updated at very high rates, typically 16 kHz; this is several times faster than most for other processes in the drive. This is why the DSP, with its high computational speed, is the processor of choice for most all-digital drives.

If you purchase your drive and motor from one vendor, you may be spared the task of commissioning the current loop; most motor/drive vendors take on this responsibility, especially for brushless systems. However, you should understand how the current loops work because they may impact the ultimate performance of your machine. Ask your vendor for the bandwidth of the current loop. This is a measure of responsiveness that most vendors provide. Generally, applications work well if the current loop has a bandwidth of at least 800 Hz. Fig. 2 shows the step response of a fast (800 Hz) and a slow (200 Hz) current loop; the slower current loop adds about four times the delay.

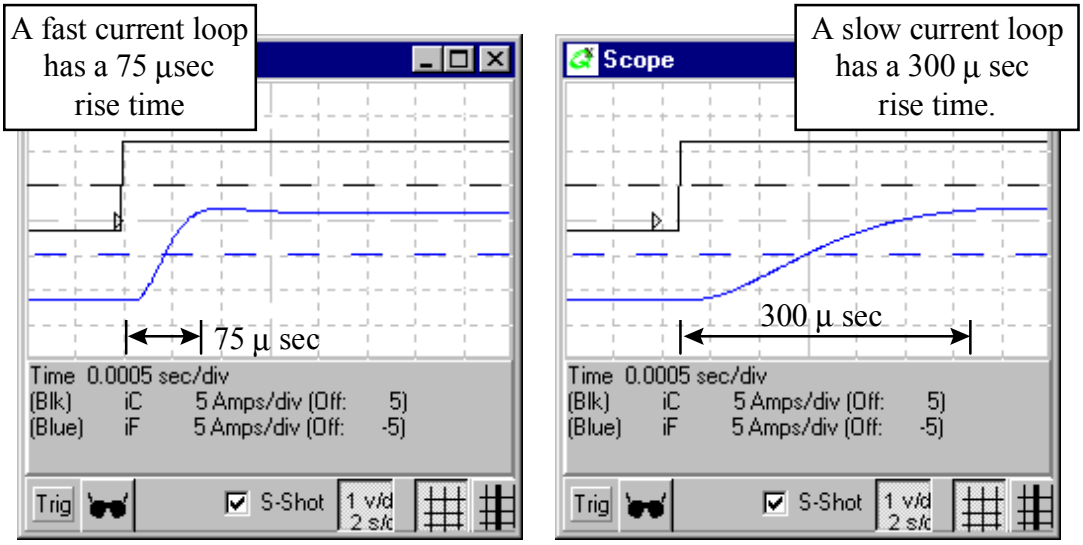


Figure 2. Current loop step at 50 µs/div: a) 800 Hz current loop has much less delay than b) 200 Hz loop

Current loops must be fast; otherwise, they will reduce system performance. The current loop is the middle of the velocity loop; in a sense, the current loop acts like a filter, providing a delayed response to the velocity-loop output. As discussed in earlier installments of this column (See 20-Minute Tune-Up, May 1999), this delay decreases the stability of the velocity loop. If you are using slow current loops, the only way to avoid velocity-loop instability is to reduce servo gains; that makes the system slower than it could be. As you can see by comparing Figs. 3a and 3b, a slow *current loop* can

dramatically degrade the response of a fast *velocity loop*. (Note: Fig. 2 shows a current loop and Fig. 3 shows a velocity loop; see that the time scale changed from 50 µs to 10 ms per division to display the slower velocity loop.)

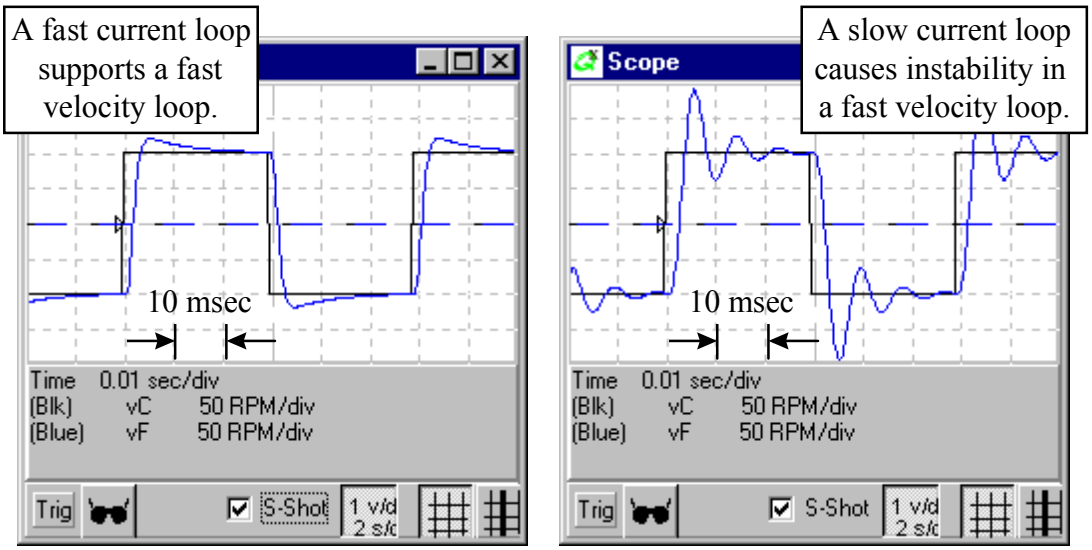


Figure 3. Velocity step for a velocity loop with two current loops. a) An 800 Hz loop supports fast velocity controller while b) a 200 Hz loop does not.

The details of current loop operation can be involved. The process of torque production includes commutation, the control of power transistors, and the current loops themselves; trying to understand the impact of the current loop on the performance of the machine can be challenging. Usually, the current loop can be simplified to look like a filter. This filter should have a similar Bode plot to the current loop itself. For example, if your vendor tells you that the current loop on a given drive has a bandwidth of 800 Hz, then you can probably use a two-pole low-pass filter with an 800 Hz bandwidth as a simplified model. Such a model helps explain the impact of the current loop. In fact, the model of a current loop used in the simulations for this column is just such a filter. The bandwidth of the filter is 800 Hz and the damping ratio (ζ) of the filter is 0.7 (see Fig. 4); this is typical for a high-performance servo drive.

{Note to editor: Fig 4 can be cut if the article runs over}

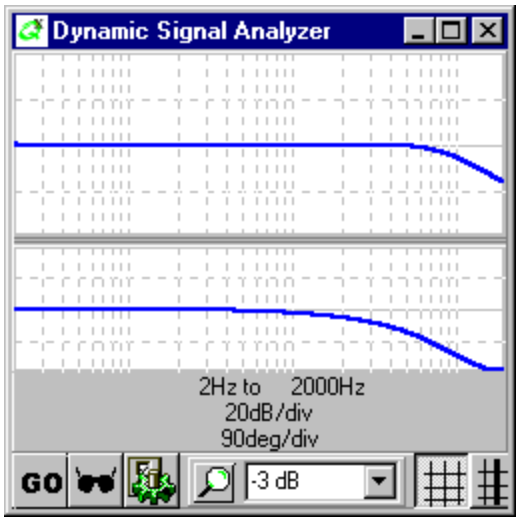


Figure 4. Bode plot of current loop, used to clarify impact of current loop on velocity control.

Laboratory

Ready to take a look at the impact of current loops for yourself? Then log on to www.ptdesign.com and download the free control system simulator, ModelQ. After installation select the February 2000 model in the combo-box at top center and click "Run." You'll see a velocity controller operating. Initially, the current-loop bandwidth will be high and the system does a good job following the step. You should see a response similar to Fig. 3a. Now change the current loop bandwidth (the constant *CLoop* on the left of the screen) from 800 Hz to 200 Hz; notice how the system becomes much less stable, as shown in Fig. 3b. If this system used 200 Hz current loop, you would have to reduce the velocity loop gains (KVP and KVI) to achieve stability. Unfortunately, and that would slow down the whole system.

This article was the basis for the March, 2000 edition of
"20-minute Tune-Up" printed in PT Design Magazine.

Gearing Servos

March, 2000

Background

Gearboxes let a small motor to do a big motor's job. Electric motors are sized according to the torque they output; more torque implies larger size and higher cost motors. Because gearboxes multiply torque, they reduce the torque required from the motor. In a majority of servo applications, a geared motor will cost less and be smaller than a non-geared motor, even when you add in the cost and size of the gearbox.

Another reason to use a gear box is to reduce problems of resonance, the self-sustained oscillations that occur frequently on servo systems (see "20-Minute Tune-up," Oct 1999). Gearboxes help resonance by reducing the *reflected* inertia of the load. Reflected inertia is the inertia that the motor senses from the input side of a gearbox. Consider the schematic of Fig. 1. A gearbox with a reduction of N:1 will reduce the reflected inertia of the load by a factor of N^2. For example, a gear reduction of 2:1 will reduce the reflected load inertia to just $(1/2)^2$ or 25% of the actual load inertia.

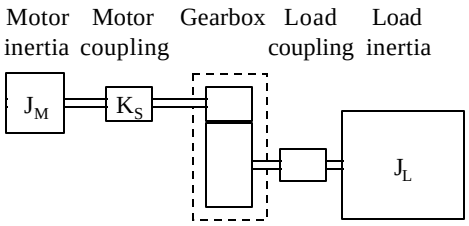


Figure 1: Mechanical schematic of motor and load.

Resonance is particularly difficult to deal with when the load inertia is much larger than the motor inertia. Consider an example: if a load inertia of 0.002 kg-m^2 is driven directly by a motor with a rotor inertia of 0.0002 kg-m^2, the load/motor inertia ratio would be 10:1. As shown in Fig. 2a, such high ratios make a servo system difficult to control, even with modest response rates (about 60Hz bandwidth) shown in Fig. 2.

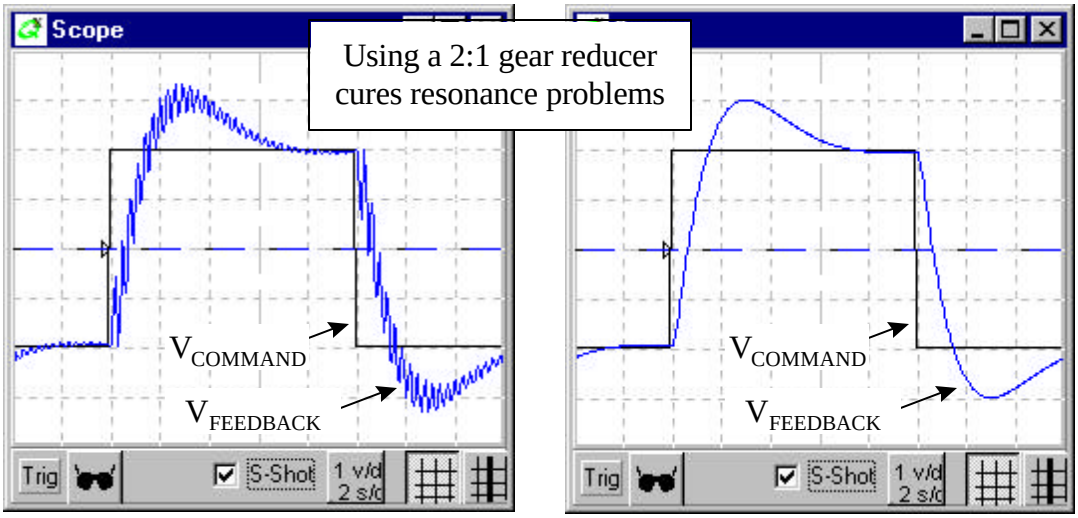


Figure 2: Resonance is removed by changing the gear reduction from a) 1:1 to b) 2:1.

Problems with mechanical resonance have increased substantially in recent years. A major reason is the use of low-inertia brushless motors to replace brush DC motors. Because of their unique construction (see "20-Minute Tune-up," Sept. 1999), brushless motor rotors are commonly 5-8 times lighter than equivalently-sized, older brush motors. This is usually considered an advantage as it reduces weight and allows faster acceleration. However, a machine where a brush motor was originally used, might have had a reasonable 2:1 load-to-motor inertia ratio; but when the design is upgraded with a brushless motor, the load-to-inertia ratio may grow to 10:1 used for Fig 2a. The second reason for more problems with resonance is that modern machines requires faster response; this, in turn, demands servo gains be turned up higher than in the past. These higher gains also make machines more sensitive to resonance.

One common misconception about the load-to-motor inertia ratio is that it is optimized when the inertias are *matched*. Matched inertia means that the motor inertia is equal to the reflected load inertia. Experience does show that more responsive control systems require smaller load-to-motor inertia ratios. Load-to-motor ratios of 3-5 are common in servo applications and many applications will work even with larger ratios. The most responsive applications require the load inertia be no larger than about 70% of the motor inertia. So, while matched inertia is a good target, the maximum tolerable load-to-motor inertia ratio will vary between applications.

While gearboxes do help servos perform better, they are not without problems. One problem is *lost motion*. Lost motion is the difference between the actual output shaft movement and what it would have been had the gearbox provided ideal gearing. Most of lost motion comes from backlash, the phenomenon where the small clearance between gear teeth causes a discontinuity in motion when torque is reversed. Backlash causes inaccuracy especially when using the motor feedback sensor to close the servo loops. Most applications require that the backlash be a fraction of the system overall accuracy. Gearboxes also display mechanical *compliance* or springiness so that the input shaft winds up a little when transmitting

high torque. However, this effect can be ignored in many machines because the motor-gearbox coupling usually is the dominant source of mechanical compliance.

High gear reduction can require high-speed motors. For typical applications, avoid gearing that requires more than 3,000 to 4,000 RPM from the motor. High-speeds gearboxes will often be noisy and have reduced life due to wear. Different types of gear geometries (see "Brushing up," Nov, 1999) have been developed which can reduce noise, error, and backlash, although these improvements can reduce life or raise cost. Fortunately, advances in servo gearhead technology, such as wide use of helical gears. promise to one day improve the performance of gearboxes without significant impact on cost.

Laboratory

Ready to take a look at the impact of reducing reflected inertia for yourself? Then log on to www.ptdesign.com and download the free control system simulator, *ModelQ*. After installation select the March 2000 model in the combo-box at top center and click "Run." You'll see a scope display of a velocity controller operating with the velocity command in black and the motor feedback velocity in blue. The compliance (K_S = 3000 Nm/rad) is chosen to represent the motor coupling. Initially, the gear reduction will be one-to-one and the system is almost unstable because of the resonance generated by a 10:1 load-to-motor (0.002 kg-m^2 to 0.0002 kg-m^2) inertia ratio. You should see a response similar to Fig. 2a.

Let's try reducing the inertia ratio by using a 2:1 gear box. Notice in Table 1 that in this case the reflected load inertia (J_L/N^2) is 3.5 times the motor inertia. Notice also that changing the gearing also requires that a tuning gain be changed. Without a gearbox, K_{VP} = 0.4 was found to work well. However, the gearbox reduces the total inertia (J_T) by about three times, and the loop gain, K_{VP}, must fall in the same proportion to maintain the servo performance (K_{VP} = 0.13). The result is shown in Fig 2b; the inertia ratio is greatly improved and resonance has been eliminated. Experiment yourself; you should see that the higher you set the servo gains (especially K_{VP}), the larger gear reduction you will need to avoid resonance.

J_M (kg-m^2)	J_L (kg-m^2)	N	J_L/N^2	$J_T = J_M+J_L/N^2$	K_{VP}	Inertia ratio	Resonance ?
0.0002	0.002	1	0.002	0.0022	0.4	10.0	Yes
0.0002	0.002	2	0.0005	0.0007	0.13	3.5	No

Table 1: Using a gearbox to change inertia ratio

Thanks to Dan Moss (dmoss@baysidemotion.com) of Bayside Motion Group for his contributions to this article.

This article was the basis for the May, 2000 edition of
"20-minute Tune-Up" printed in PT Design Magazine.

Varying inertia

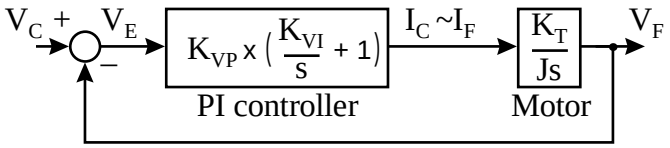
May, 2000

Background

Servo systems can do a great job controlling motion. When properly applied, they respond quickly to the command and they reject disturbances. But to achieve good results, servo gains have to be tuned well; it's a process that requires time and skill. Unfortunately, tuning depends heavily on the load inertia. In fact, a servo machine can be tuned for top performance with one load inertia, and may become sluggish or even unstable when the load inertia changes.

Load inertia can vary for any number of reasons. A motor that moves a work piece will see an abrupt drop in inertia when the work piece is set down. In web-handling applications, inertia of the roll drive increases when material is added to a roll. In robots, the inertia sensed by the shoulder motor increases when the robot arm is fully extended. In all these cases, the motor can see a large variation in inertia during normal operation of the machine. What surprises many servo designers is that the inertia acts just like a servo gain; when it varies, servo dynamics can change dramatically.

Consider the PI velocity-loop block diagram below. The tuning variables are a proportional gain, K_{VP}, and an integral gain K_{VI} with an integral shown as *1/s*. The controller produces commanded current (I_C) from velocity error (V_E). Usually a current controller produces actual current (I_F) which is very close to I_C. Here we will assume we can ignore its effect so that $I_C \sim I_F$. I_F drives the motor through the torque constant (K_T) and the inertia, J.

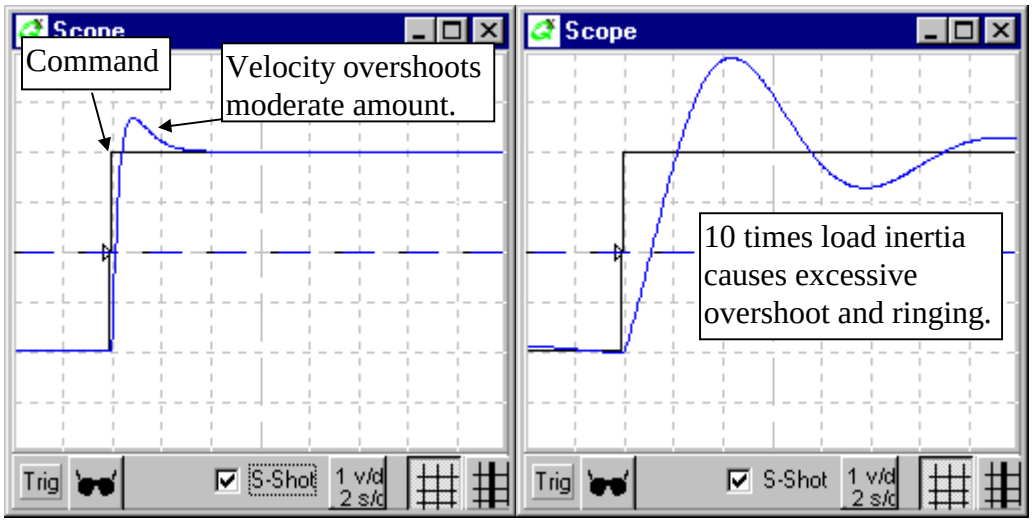


Notice that the inertia (J) appears in the denominator of the motor model; it divides down the effect of the current almost identically to the way K_{VP} raises it. In fact, as we will see below, J acts as much like a servo gain as K_{VP} does.

Incidentally, this discussion focuses on a velocity controller. The structure is ideal for studying the effects of inertia variation because it's so simple. Of course, servo systems are often configured in more complex control loops such as position and tension controllers. However, the effects of inertia variation are about the same in those structures as they are in the velocity loop, and the velocity loop is easier to understand.

The following figure shows the step response of the control loop for a well-tuned set of gains (K_{VP} = 0.72, K_{VI} = 100) when the inertia is 0.002 kg-m^2. The motor response displays a moderate amount of overshoot which is typical for PI velocity controllers. The

figure at right shows what happens when the total inertia increases by a factor of 10. Now the overshoot is excessive (about 50%) and there is ringing. Such a response would be inappropriate for many applications.



Effects of changing inertia: A well-tuned system (left) looses stability with 10 times total inertia (right). Vertical scale: 5 RPM/div. Horizontal scale: 0.02 sec/div.

There are several ways to deal with changing inertia. One step is to use a larger motor. If the motor inertia is large, it reduces the impact of variation of the load. This is because the effect on system stability is based on the relative change of total (motor and load) inertia. Consider the example above. It was devised by assuming the motor inertia was 0.0002 kg-m^2 and the initial load inertia was 0.0018 kg-m^2. So, to observe a 10:1 (0.002 to 0.02 kg-m^2) change in total inertia, the load inertia would have to increase to 0.0198 kg-m^2. A larger motor, say 0.002 kg-m^2, would have reduced the relative increase to (0.002 + 0.0198)/(0.002 + 0.0018) or a factor of 5.7, down from 10.

Another step is to use *gain scheduling*, a technique where the servo gains are increased in proportion to total inertia change. For example, were the gain K_{VP}, raised by the amount the total (motor and load) inertia changed, there would be no change in the dynamics of the control loop.

Gain scheduling is easy to accomplish on digital drives if the load inertia can be predicted. This is common in industrial controllers. The machine may know the inertia of an object being moved, or it may be able to measure the changing diameter of a roll accumulating material. In this case, the servo gains (here, just K_{VP}) can be adjusted appropriately. Gain scheduling works best with digital drives because total inertia can be calculated by the controlling computer or PLC which can then adjust gains in the servo drives appropriately.

Be aware that two factors often limit the ability to schedule gains. First, the resolution of the position feedback device (see Tune-up, July 1999) may not be able to support the highest gains, which occur with the highest values of load inertia. Second, mechanical resonance may also limit the maximum gain (see Tune-up, Oct 1999, Feb 2000, and March 2000). Both of these problems can be avoided by tuning the system

with the maximum load inertia so that the system is fully characterized with the highest servo gains.

Laboratory

Want to learn more about inertia variation? Then log onto www.ptdesign.com and download this month's ModelQ simulation program. Launch the program, select "May 2000 Varying Inertia" from the combo box at top center and click "Run." You should be looking at the well-tuned system shown above. Now, click in the box at left next to "J" (total inertia). Type in 0.02 and notice the result is the overshoot and ringing shown above. Now schedule the gain: click in the box next to "K_{VP}" and set the value to 7.2 (up a factor of 10). Notice that the original dynamic performance returns.

You can experiment more with inertia variation. For example, reduce the inertia; notice the system, with a gain of $K_{VP} = 0.72$, will become unstable when the total inertia falls below about 0.0002 kg-m^2. You can also adjust the gains (K_{VI} and K_{VP}) so they can better tolerate inertial changes. Generally, lower gains are move forgiving.

This article is the July, 2000 installment 20-minute Tune-up which is published monthly in PT Design

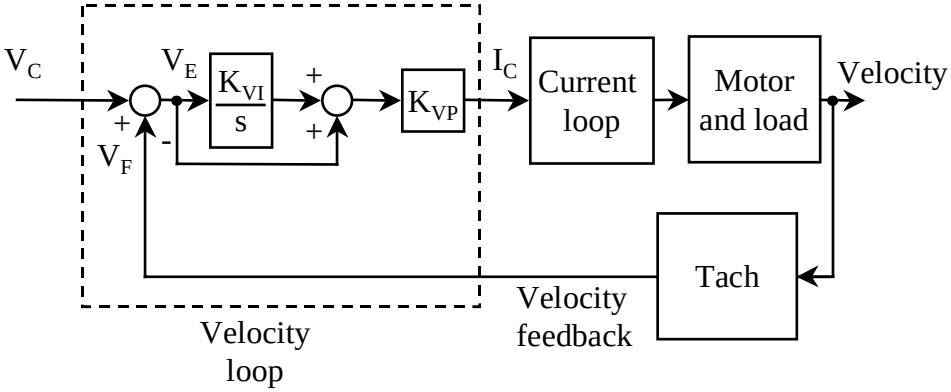
Cascaded position-velocity loops

July, 2000

This column begins a three-part series on position loops. This month we will cover the cascaded position-velocity loop. The next two installments will discuss the PID position loop and how feed-forward is used in these two loops.

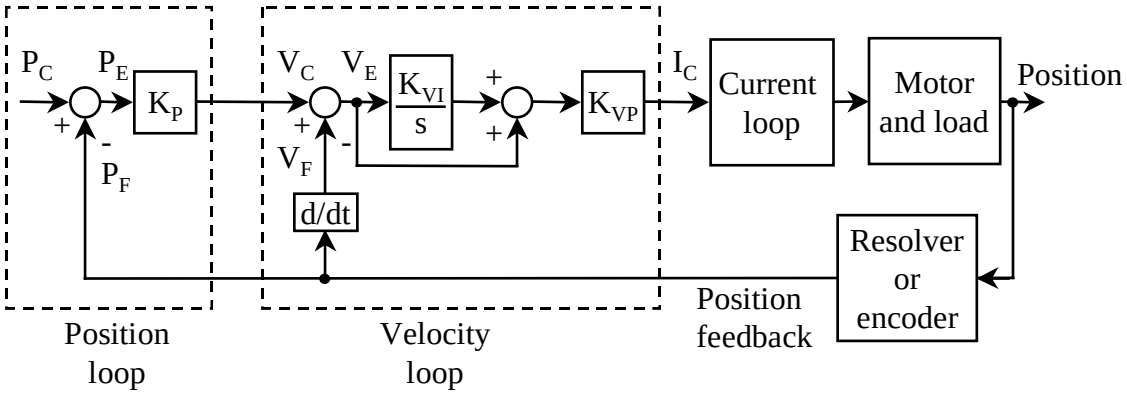
Theory

The velocity loop is the most basic servo control loop. In its simplest implementation, an analog command is compared to an analog tachometer output to generate current which produces torque which, in turn, will speed or slow the motor to satisfy the velocity command. The most common velocity loop is the proportional-integral or PI velocity loop. This loop has two gains, a proportional gain (K_{VP}) which reacts to the velocity error and an integral gain (K_{VI}) which reacts to the integral of velocity error. (Note that 1/s is the conventional depiction of an integration).



Block diagram of velocity loop.

A velocity loop fulfills many of the needs of a servo loop. It reacts to rapidly changing commands and it provides resistance to high-frequency load disturbances. However, a velocity loop cannot ensure that the machine stays in position over long periods of time. Since most machines require position control, the velocity loop must be augmented. One of the most common configurations of servo loops is to place a proportional position loop in series or *cascade* with a PI velocity controller. This configuration is shown below.



Block diagram of cascaded position-velocity loops

The position loop compares a position command to a position feedback signal, and calculates the position error or *following error*. Position error is scaled by a constant (K_P) to generate a velocity command. The principle here is simple: more position error generates a larger velocity command. A larger velocity command, fed to the velocity loop, creates more torque which, in turn, moves the motor to satisfy the position error.

Years ago the position and velocity loops had separate sensors. An encoder or resolver provided feedback for the position loop; an analog tachometer provided velocity feedback. Today, most servo systems rely solely on the position feedback sensor. Now the velocity loop derives velocity from sensed position; this is done by differentiation (velocity is the derivative of position) and is shown as *d/dt* in the block diagram.

Tuning in Zones

Tuning is the process of setting control loop gains to achieve optimal performance. Higher gains improve responsiveness, but move the system closer to instability. Normally, the goal of tuning is to raise the gains as high as possible without causing instability.

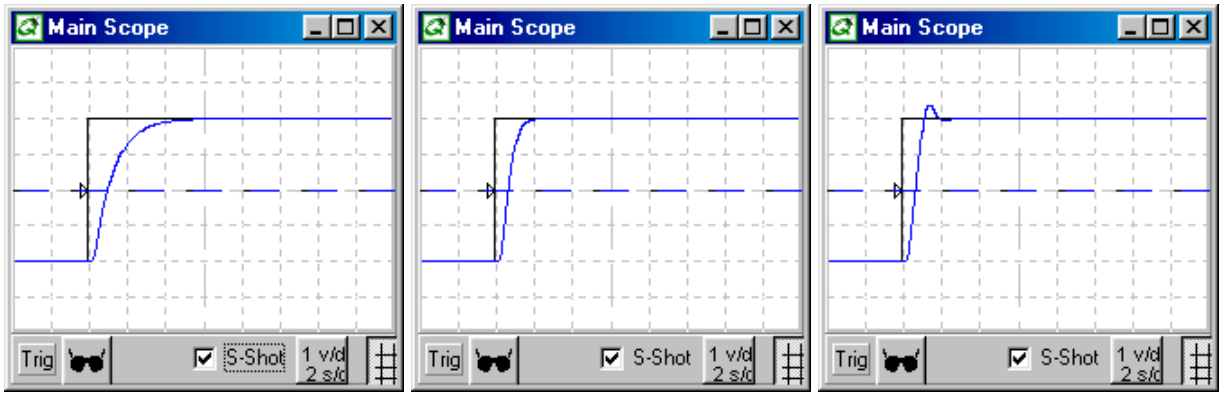
Tuning cascaded position-velocity loops can be challenging because there are three servo gains: the position loop gain (K_P), and the velocity integral (K_{VI}) and proportional (K_{VP}) gains. Using a "hit-or-miss" method of tuning with three gains can absorb a lot of time. Fortunately, each of the gains plays a different role in the servo system. Once you understand those roles, you can tune the gains independently, saving time and ensuring consistency.

Each of the gains operates in one of three frequency "zones." The highest frequency zone is covered by the velocity-loop proportional gain; typically, this zone ranges from about 30 to 100 Hz, although it can be much higher. The velocity-loop integral gain is most important for frequencies between about 10 and 30 Hz. The position loop gain covers everything below that.

Zone 1: Velocity-loop proportional gain

Tuning begins at the highest zone. Start by eliminating the lower zones: set K_{VI} to zero and put the system in velocity mode to eliminate the position-loop gain. Inject a square-wave velocity command; adjust the command magnitude to be as large as possible

without saturating the current controller; you should ensure that the current controller remains below its maximum at all times. Usually a command magnitude of 0–250 RPM works well. Now, raise the velocity-loop proportional gain as high as possible without generating overshoot in the velocity response. The three figures below show the servo system response to a square-wave command when K_{VP} is too a) low, b) about right, and c) too high.



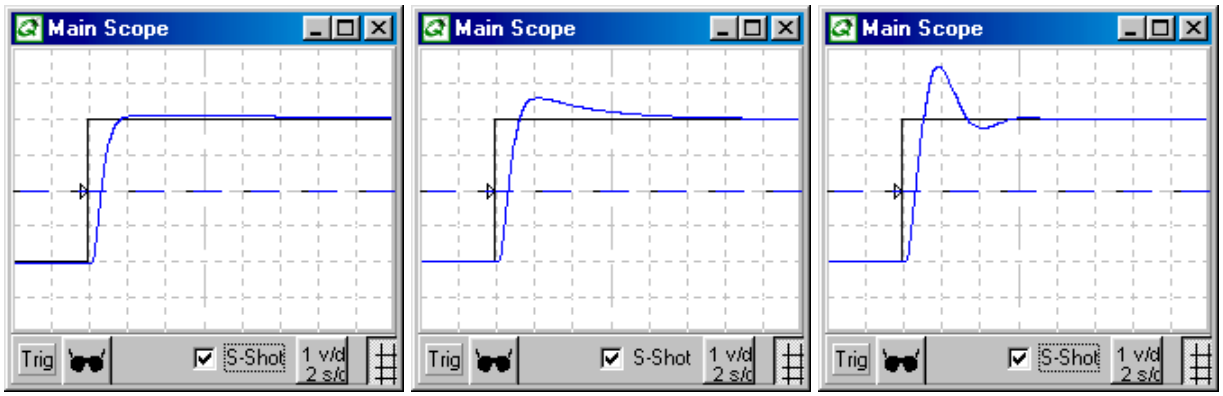
a) b) c)

Zone 1: Velocity step response with K_{VI} = 0 and K_{VP} a) low (0.6), b) about right (1.1) and c) high (2.0). Horizontal scale is 5mSec/div and vertical scale is 50 RPM/div.

Zone 1 is the hardest zone to tune. This is because two common problems seen in servo systems, resonance (20-minute tune-up, October, 1999) and audible noise (20-minute tune-up, July, 1999), are excited by this gain. You may need to use low-pass filters to reduce these two problems. While low-pass filters are helpful, you should always minimize the their use because they cause instability and force lower values for the servo gains.

Zone 2: Velocity-loop integral gain

Now that the velocity-loop proportional gain is set, it's time to tune the velocity-loop integral gain. This gain is easily tuned. Leaving the command signal as it was, raise the integral gain from zero until there is about 15% overshoot. The responses for three integral gain values are shown below.

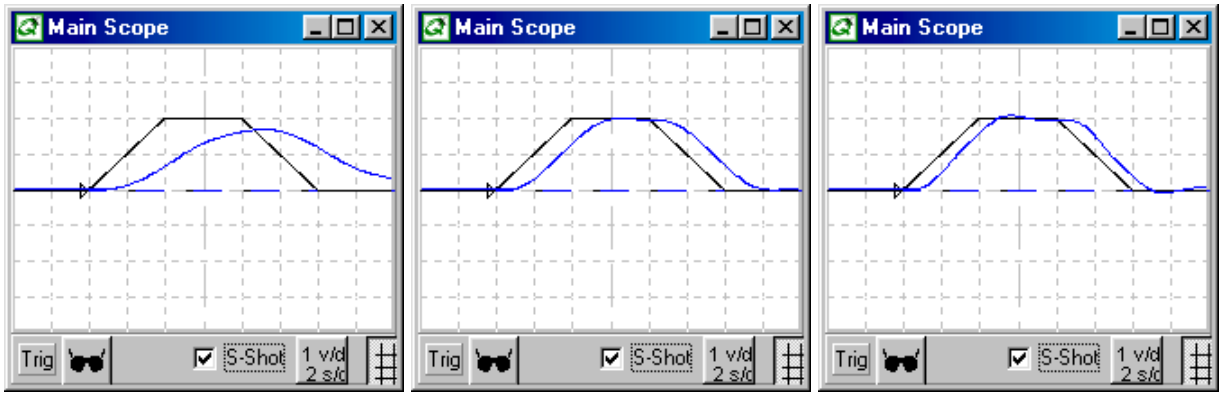


a) b) c)

Zone 2: Velocity step response with K_{VP} = 1.1 and K_{VI} a) low (20), b) about right (100) and c) high (420). Horizontal scale is 5mSec/div and vertical scale is 50 RPM/div.

Zone 3: Position-loop gain

The final zone to tune is the position-loop gain. To complete this step, you must set the controller to operate in position mode. Most controllers provide a variable that will let you make this change. Now set the command to produce a point-to-point move with the acceleration rate set at the maximum that the servo loop will see in normal operation. Finally adjust the position-loop gain up until just before the response starts to overshoot. The figures below depict the response to a rapid trapezoidal command; note that since most applications require zero overshoot in normal operation, just the little overshoot apparent in "c)" is enough to eliminate this high tuning gain value (600).



a) b) c)

Zone 3: Velocity trapezoid response with K_{VP} = 1.1, K_{VI} = 100, and K_P a) low (100), b) medium (250) and c) high (600). Horizontal scale is 5mSec/div and vertical scale is 50 RPM/div.

Laboratory

Want to tune a cascaded position-velocity loop yourself? Then log onto www.motionsystemdesign.com and download this month's ModelQ simulation program. Launch the program, select July's model, and click "Run." You should be looking at the square wave response with just the velocity-loop proportional gain. The value is set a little low (0.6), so raise it until the square wave response just overshoots, and then reduce the gain to eliminate the overshoot (K_{VP} = 1.1). Now raise K_{VI}, adjusting for about 15% overshoot (K_{VI} = 100). This completes the tuning of the velocity loop.

To tune the position loop, you must configure this model as a position controller. First, set the gain K_{VF} to 0% (it was set to 100% to simulate a velocity controller). Now select a trapezoidal velocity command by choosing "Trapezoid" in the waveform generator at bottom left. Finally, raise K_P to just below where the system overshoots (K_P = 250).

This article is the August, 2000 installment 20-minute Tune-up which is published monthly in PT Design

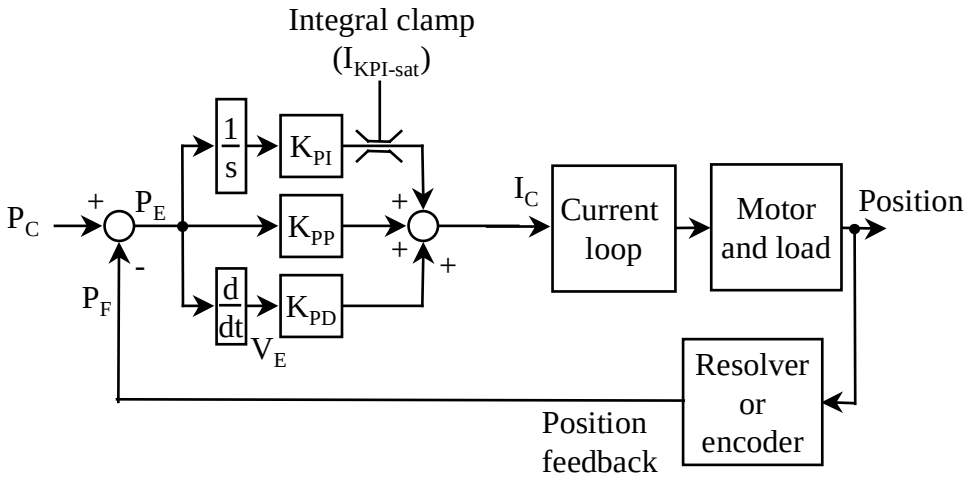
PID position loops

August, 2000

This column is the second a three-part series on position loops. This month we will discuss the PID position loop. Next month we will take up the use of feed-forward in these two loops.

Theory

The velocity loop is the most basic servo control loop. However, since a velocity loop cannot ensure that the machine stays in position over long periods of time, most applications require position control. There are two common configurations used for position control: the cascaded position-velocity loop, as discussed last month, and the PID position controller, as shown below.



Block diagram of PID position loop

<Note to editor: can you replace my "1/s" in the block diagram with an integral symbol followed by "dt?">

The position loop compares a position command to a position feedback signal, and calculates the position error, P_E. In a PID controller, current command is generated with three gains: P_E is scaled by the proportional gain (K_{PP}), the integral of P_E is scaled by the integral gain (K_{PI}), and the derivative of P_E is scaled by the derivative gain (K_{PD}).

Tuning in Zones

Tuning is the process of setting control loop gains to achieve optimal performance. Higher gains improve responsiveness, but move the system closer to instability. Tuning PID position loops can be challenging because there are three servo gains: K_{PP}, K_{PI}, and K_{PD}. As with the cascaded position-velocity loop, each of the gains

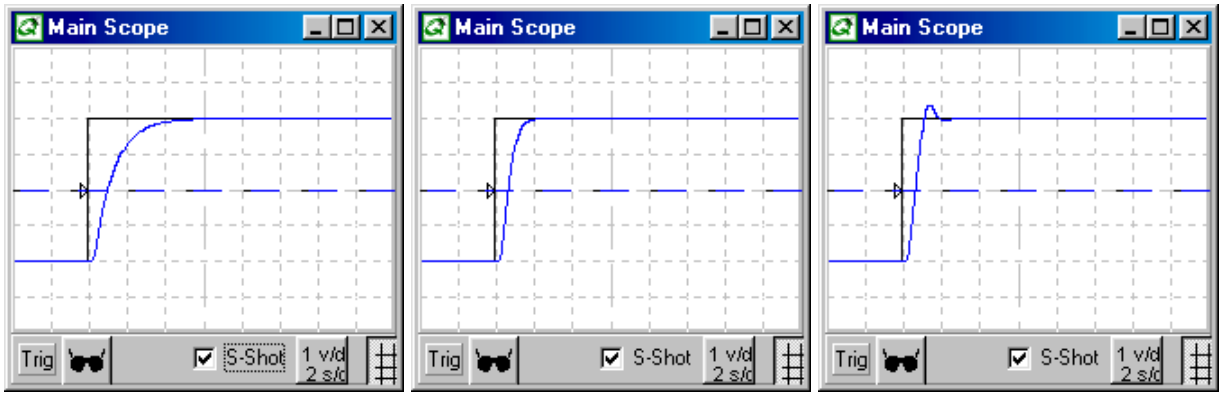
plays a different role in the servo system. Once you understand those roles, you can tune the gains independently, saving time and ensuring consistency.

Each of the gains operates in one of three frequency "zones." The highest frequency zone is covered by the derivative gain (K_{PD}); typically, this zone ranges from about 30 to 100 Hz, although it can be much higher. The proportional gain (K_{PP}) is most important for frequencies between about 10 and 30 Hz. The position loop gain covers all frequencies below that.

Zone 1: Derivative gain

Begin tuning the highest frequency zone. Start by eliminating the lower zones: zero K_{PI} and, if possible, K_{PP}. Many PID controllers do not allow K_{PP} to be zeroed. If that's your case, fix K_{PP} at a fairly low value while tuning K_{PD}. Now, prepare a trapezoidal point-to-point move. These commands have three segments: acceleration, cruise, and deceleration. When tuning Zone 1, you should set the acceleration and deceleration rates as high as the controller will allow. In fact, a square velocity command (unlimited acceleration) is ideal.

If the commanded move puts the current controller in saturation (that is, commands more than the controller can produce), reduce the peak velocity of the move. Usually a peak velocity of 0–250 RPM works well. Now, raise the derivative gain as high as possible without generating overshoot in the velocity response. The three figures below show the servo system response to a square-wave command when K_{PD} is a) too low, b) about right, and c) too high. Note that if you cannot zero the proportional gain (K_{PP}), expect some overshoot to a square wave. Overshoot due to K_{PD} occurs on a much shorter time scale and so is easily distinguished from overshoot caused by K_{PP}.



a) b) c)

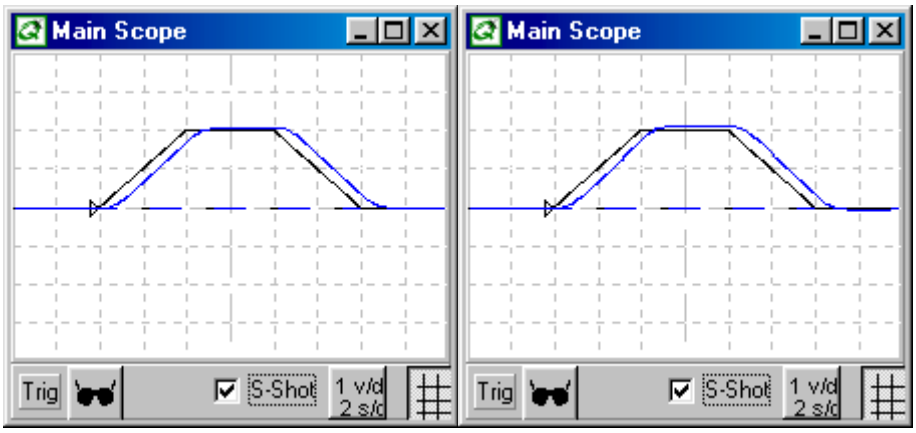
Zone 1: Velocity step command (black) and response (blue) with K_{PP} and K_{PI} = 0, and K_{PD} a) low (0.6), b) about right (1.1) and c) high (2.0). Horizontal scale is 5mSec/div and vertical scale is 50 RPM/div.

Zone 1 is the hardest zone to tune. This is because two common problems seen in servo systems, resonance (20-minute tune-up, October, 1999) and audible noise (20-minute tune-up, July, 1999), are excited by this gain. You may need to use low-pass filters to reduce these two problems. While low-pass filters are helpful, you should

always minimize the their use because they cause instability and force lower value servo gains.

Zone 2: Proportional gain

Now that the derivative gain is set, it's time to tune the proportional gain. First, modify the position command. Lower the acceleration and deceleration to the highest rates that will be seen in the application under normal operation. Raise the integral gain until a slight amount of overshoot appears, and then reduce the gain to remove the overshoot. The responses for two proportional gain values are shown below with K_{PP}: a) about right and b) too high.

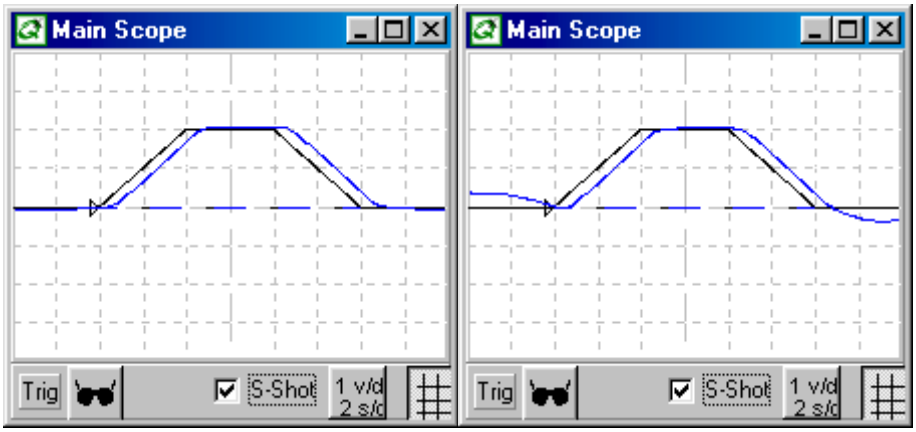


a) b)

Zone 2: Trapezoidal position profile with K_{PD} = 1.1 and K_{PP} a) about right (20.0) and b) a little high (40.0). Horizontal scale is 5mSec/div and vertical scale is 50 RPM/div.

Zone 3: Integral gain

The final zone to tune is the position-loop gain. Tuning integral gain is difficult because even a small amount causes overshoot. Several methods have been developed to deal with this shortcoming. First, most controllers allow you to clamp the maximum current the integral term can command. This makes sense because the primary reason to use integral gain in many applications is to overcome frictional loads; when that it the case, there is no need to allow current generated by the integral term to be much larger than the maximum friction load. Another technique is to force the integral to zero anytime the motor is commanded to move. Both of these techniques allow the integral gain to be raised to higher values than it otherwise could be. The figures below show the integral gain set to a level just below causing overshoot (K_{PI} = 5) and to value so high (K_{PI} = 10) that it causes oscillations at zero speed. Both figures depict a system with the integral zeroed when motion is commanded, and a clamp on the integral output of 2 amps.



a) b) c)

Zone 3: Velocity trapezoid response with K_{VP} = 1.1, K_{VI} = 2.0, and K_P a) about right (5.0) and b) high (10.0). Horizontal scale is 5mSec/div and vertical scale is 50 RPM/div.

Laboratory

Want to tune a PID position loop yourself? Then log onto www.motionsystemdesign.com and download this month's ModelQ simulation program. Launch the program, select August's model, and click "Run." You should be looking at the square-wave response with just the derivative gain (K_{PD}). The value is set a little low (0.6), so raise it until the square wave response just overshoots, and then reduce the gain to eliminate the overshoot (K_{PD} = 1.1).

Now select a trapezoidal velocity command by choosing "Trapezoid" in the waveform generator at bottom left. Now raise K_{PP}, adjusting to the highest value that does not cause overshoot (K_{PP} = 10). Finally, raise K_{PI} to 5 to get a complete set of PID tuning gains.

George Ellis is a Senior Scientist at Kollmorgen. [note to editor: we use "Kollmorgen" rather than "Kollmorgen, Inc."]. *His book, "Control System Design Guide," 2nd edition, was published in May, 2000 by Academic Press. He can be reached at gellis@kollmorgen.com*

This article is the Sept., 2000 installment 20-minute Tune-up which is published monthly in PT Design

Feed-forward in position-velocity loops

September, 2000

This column concludes a three-part series on position loops. This month we will cover feed-forward gains.

Theory

Motion control systems use feed forward gains to speed up the response to rapidly changing position commands. Feed forward is often used in the cascaded position/velocity loop; here a faster velocity loop is enclosed within a slower position loop. The feed forward path speeds response by taking the command around the slower position loop, directly to the velocity loop.

With or without feed forward, servo systems rely on high loop gains to achieve good performance. High loop gains provide two functions: they help the controller respond to the command and they help reject disturbances. Unfortunately, as we have seen, servo gain values are limited because, when they are too high, they cause instability. Since feed-forward gains work outside the loop, they do not cause instability. Feed-forward, in parallel with high loop gains, can greatly improve the dynamics of a servo system.

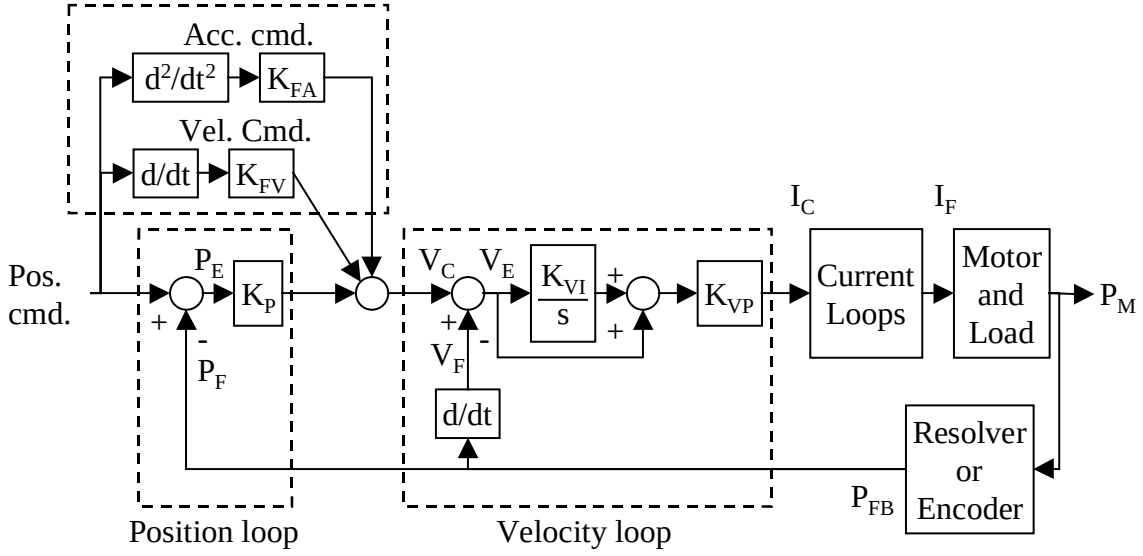


Figure 1. Cascaded position and velocity loop with feed-forward.

Velocity vs. acceleration feed-forward

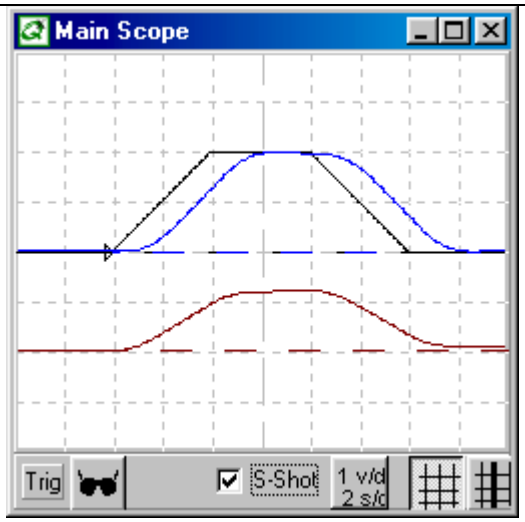
The velocity feed forward path connects the velocity profile to the velocity loop through the scaling gain, K_{FV}. When the position profile abruptly changes, the velocity feed forward path immediately shuttles that change to the velocity command. This greatly speeds the system response when compared to relying solely on the position loop. Velocity feed forward can reduce the time to make a quick move by a factor of two or more. The primary shortcoming of velocity feed forward is that it induces overshoot.

This can be cured one of two ways: reduce the loop gains or add acceleration feed forward. Reducing the loop gain works, but at the cost of reducing the system disturbance response.

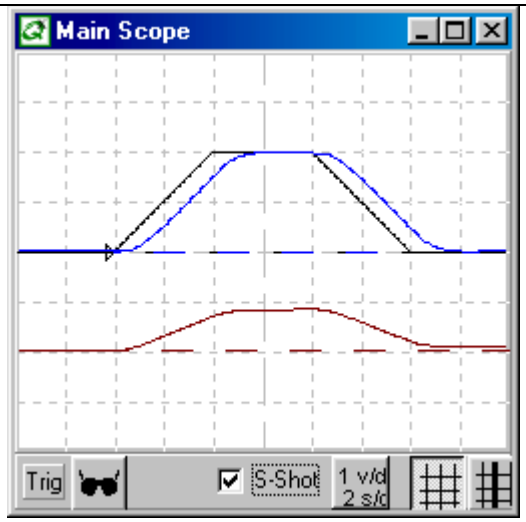
The acceleration feed forward path connects the acceleration profile to the velocity loop through the scaling gain, K_{FA}. Acceleration feed forward eliminates the overshoot caused by velocity feed forward, but it does so without reducing loop gains. This allows high velocity feed forward (giving fast command response) and high loop gains (giving solid disturbance rejection) at the same time. Unfortunately, many position controllers do not support acceleration feed forward; in that case, the controls engineer will have to decide between command and disturbance response.

Take a look at the difference

Figure 2 shows the benefits of velocity feed-forward. Figure 2a has the servo gains tuned up about as high as possible without generating overshoot. The black waveform is the velocity command and blue is the response; the brown waveform is the position or "following" error (P_E in Figure 1). Less following error is almost always an advantage. Figure 2b shows that adding 50% velocity feed-forward improves response, cutting the following error by about 30%. Unfortunately, two loop gains (K_P and K_{VI}) had to be reduced to rid the response of overshoot caused by the velocity feed forward (see Table 1).



Response of position loop without feed-forward.



50% velocity feed-forward reduces following error (brown) by about 30%.

Figure 2. Response of position loop with and without velocity feed-forward. Black is velocity profile and blue is feedback velocity (50 RPM/div); brown is following error (0.2 Rev/div).

Acceleration feed forward allows even faster command response and with less effect on servo gains. For example, Figure 3a shows the effect of adding 25% acceleration feed-forward to the system of Figure 2b. The command response improves, as shown by the reduction of following error. Also, the addition of acceleration feed-forward allowed the servo gains to be increased: K_P was be restored to its original value and K_{VI} was raised

substantially. Compared to the non-feed-forward system of Figure 2a, the acceleration and velocity feed-forward gains allowed about double the command response (as evidenced by the reduced following error) with almost the same disturbance response.

Figure 3b shows the command response can be further improved by the use of more feed-forward. Here, the system has 70% velocity feed-forward and 50% acceleration feed-forward. The command response is greatly enhanced; the feedback is very close to the command trace and the following error is reduced again. Unfortunately, such high feed-forward gains require that servo gain K_{VI} be reduced to 25 (1/4 of its original value), so the system is somewhat softer responding to disturbances.

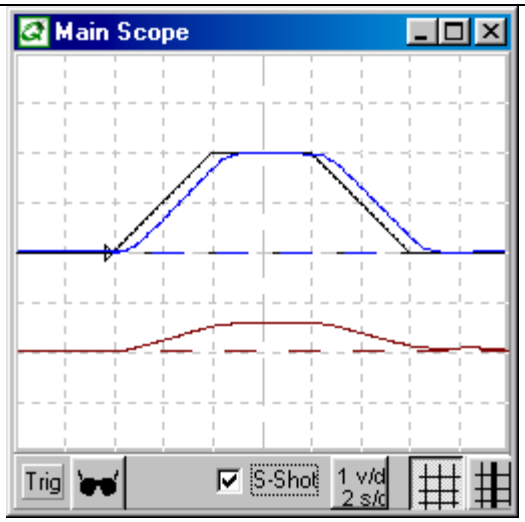


Figure 3a. Response of position loop with velocity and acceleration feed-forward. Black is velocity profile and blue is actual velocity (50 RPM/div); brown is following error (0.2 Rev/div).

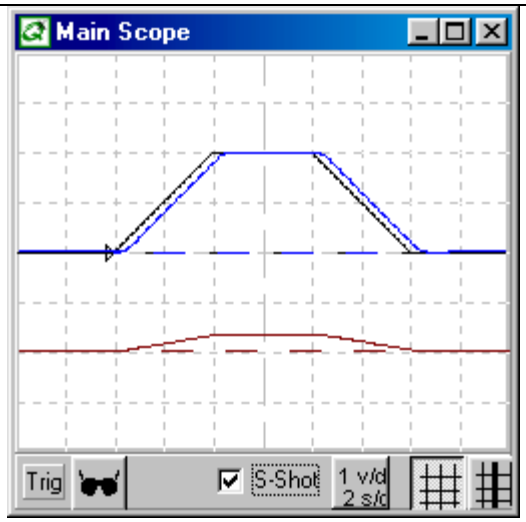


Figure 3. Response of position loop with more velocity and acceleration feed-forward. Black is velocity profile and blue is actual velocity (50 RPM/div); brown is following error (0.2 Rev/div).

Figure 3. Response of position loop with acceleration feed-forward. Black is velocity profile and blue is actual velocity (50 RPM/div); brown is following error (0.2 Rev/div).

Laboratory

Want to tune feed-forward gains yourself? Then log onto www.motionsystemdesign.com and download the most recent version of the ModelQ simulation program. Launch the program and select July's model. In the constants grid at left, set K_P=250, K_{VP}=1.1, and K_{VI}=100. In the waveform generator (at bottom), in the Waveform combo box (on the left), select Trapezoid. Click "Run." This is the final position loop from July's model which was tuned up with high servo gains, but without feed-forward. Display the following error on the scope by selecting the "Scale" tab on the scope (at left), and then the "Add..." button; double click on "PE" and click "Close."

Now, repeat the results for Figures 2b, 3a, and 3b by entering the values shown in Table 1. Experiment for yourself. Notice how more velocity feed-forward improves command

response, but causes more overshoot. Reducing the servo gains K_P and K_{VI} reduces that overshoot at the expense of reducing disturbance response. Notice how acceleration feed-forward reduces the overshoot as well, allowing servo gains.

	K_P	K_{VP}	K_{VI}	K_{FV}	K_{FA}
Figure 2a	250	1.1	100	0	0
Figure 2b	170	1.1	60	50%	0
Figure 3a	250	1.1	70	50%	25%
Figure 3b	250	1.1	25	70%	50%

Table 1: Servo and Feed-forward gains for Figures 2 and 3.

Analog PI Velocity Loop

October, 2000

This column begins a two-part series on analog velocity loops. This month we will cover the PI controller.

Theory

Today, most discussion about motion control focuses on digital controls. That's because digital control is new to many people so that there is more need for discussion. However, many applications still use analog servo loops. The differences between analog control and digital control are modest. The main differences are that analog controls do not have to deal with sampling, the process of running servo calculations at fixed intervals of time. Not having to sample removes delay, allowing analog controllers to be more responsive than their digital counterparts. Of course, analog controllers can be more difficult to adjust because servo gains are set with passive electrical components. Beyond that, digital and analog systems have a lot in common.

Figure 1 shows the schematic of a typical analog proportional-integral (PI) velocity controller. The servo gains are adjusted with resistors and capacitors and the execution is done with the op-amp. The most common feedback device for an analog controller is an analog tachometer. The current controller and motor are about the same as with a digital system (See June, 2000).

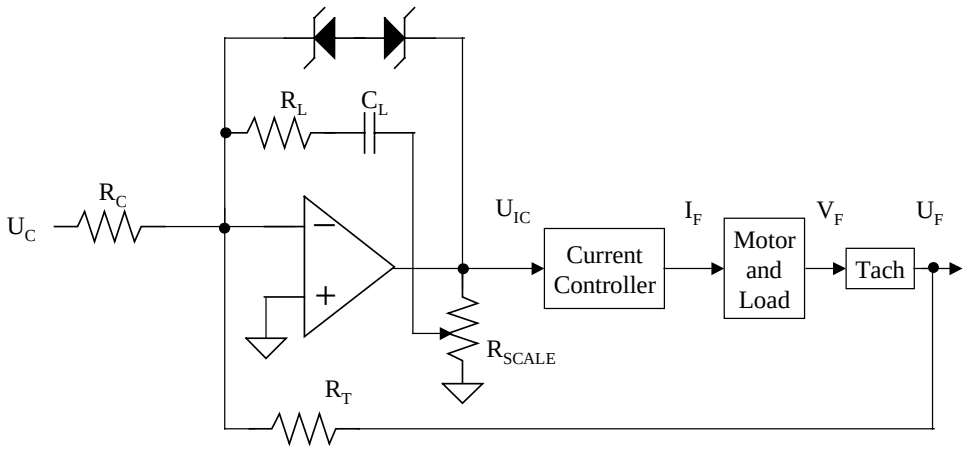


Figure 1. Schematic representation of analog velocity controller

Figure 2 shows a control-loop diagram for the schematic. Notice that this looks very much like a digital PI loop except there are quite a few scaling gains that need to be taken into account. For example, the output of the servo loop is a voltage that commands a current; in this case, 8 volts from the op-amp commands 20 amps to the motor. That scaling constant, I_{PEAK}/V_{PEAK}, measured in amps/volt, is part of the control loop. Similarly, the tachometer scaling, K_{TACH}, measured in volts/kRPM, must also be specified.

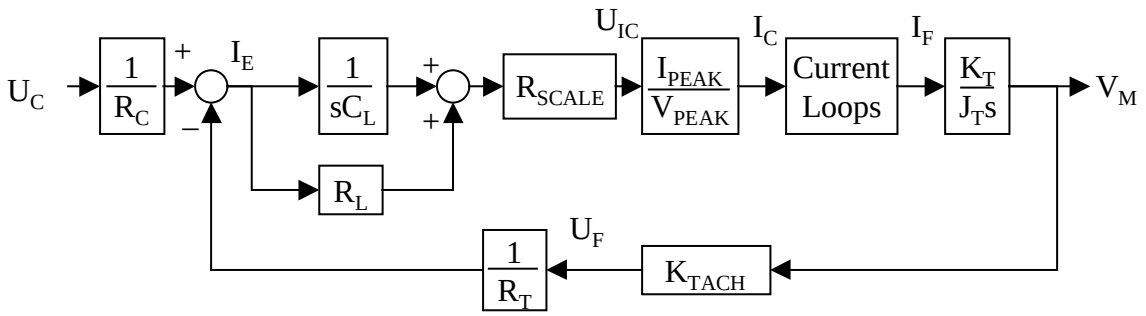
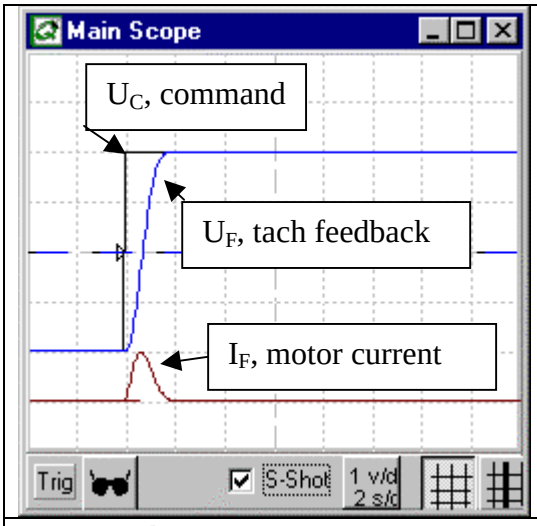


Figure 2. Control-system block diagram

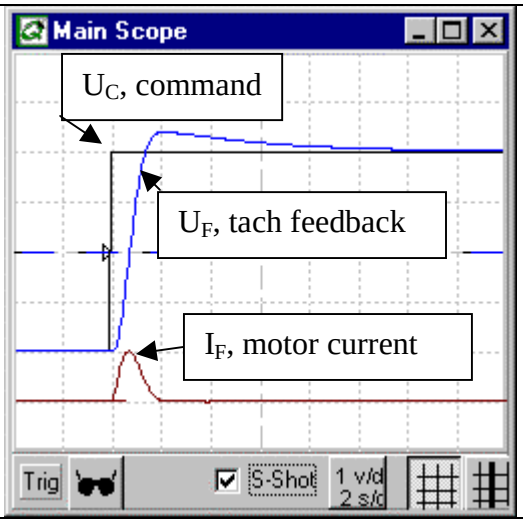
The servo gains are set by a few resistors and a capacitor. R_C scales the command and R_T scales the tachometer feedback. R_L is the key component in tuning the proportional gain and C_L is key in setting the integral gain. The two zener diodes in Figure 1 are not shown in Figure 2. These diodes provide anti-windup, the feature where the gain of the integrator is clamped during large velocity swings. Without anti-windup, the integrator can continue integrating error with large velocity swings; this can cause needlessly large overshoot.

R_{SCALE} is a potentiometer provided as an overall loop gain adjustment. R_{SCALE} is directly in line with the motor and load, K_T/J_T. J_T represents the total (motor plus load) inertia. When J_T changes, it changes the loop gain and that changes the performance of the system. With R_{SCALE}, you can adjust this change out within a reasonable range. Of course, you could make the same sort of adjustment by changing R_L and C_L, but that is less convenient because it can involve soldering in new components.

Tuning an analog PI controller is very much like tuning a digital system. The key is to tune in zones (see June, 2000). With a PI controller, there are two zones: proportional and integral. To tune, apply a small-magnitude velocity step command to the drive. You should adjust the step to be as large as possible without calling for peak current from the drive. If you ask for too much current, you will saturate the current loops; this makes the loop more difficult to tune correctly. Typical step commands are from 50 RPM to about 500 RPM, depending on the loop speed and the amount of inertia. Now, adjust the zones one at a time. First, zero the integral zone (short C_L) and tune R_L to get a little overshoot. Next, add in C_L until the overshoot reaches about 15%. The response with these gain settings are shown in Figure 3. Remember to set the R_{SCALE} pot away from the ends of travel of travel. That way, if the inertia varies up or down in the future, you can use R_{SCALE} to adjust performance.



Zone 1: Short C_L to create a proportional controller. Adjust R_L to 500 kOhm to get maximum response without overshoot. Scaling: U_C and U_F: 50 mV/div; this is equivalent to about 44 RPM for the scaling of this system. I_F: 20 Amps/div.



Zone 2: Leave R_L at 500 kOhm and adjust C_L. Here, a setting of 10 uF yields the 10%-15% overshoot to step that usually indicates good response for a PI controller.

Laboratory

Want to tune an analog PI velocity controller yourself? Then log onto www.motionsystemdesign.com and download the most recent version of the ModelQ simulation program. Launch the program and select October's model. Click "Run." This is the analog PI controller discussed in this column. The command (blue) and response (black) are overlaid; the current is shown below; it's a good idea to monitor current when tuning to ensure that the system remains out of saturation. Remember, if you have questions about constants, click on the button "About this model…."

To tune the system, first tune Zone 1. Set $C_L = 0$ and adjust R_L to where some overshoot starts to appear and then adjust it down. In this case, overshoot occurs at R_L = 600 kOhm, so adjust R_L down to 500 kOhm to eliminate the overshoot. Now add in C_L. Always start with a fairly large value of C_L. Remember that lower values of C_L create larger servo gains. Here, a value of 10 uF gives a good step response for a PI controller: about 15% overshoot. Now, adjust the load inertia down by a factor of two. Notice how this degrades the performance of the system by creating ringing. Finally, adjust R_{SCALE} down from 2 to 1 to account for the inertia change. On the physical amplifier, this would be done by adjusting the pot. Notice how performance is restored. This one of the key functions of R_{SCALE}.

Analog Lead-Lag Velocity Loop

November, 2000

This column concludes a two-part series on analog velocity loops. This month we will cover the lead-lag velocity controller, an extension of last month's PI controller.

Theory

Lead-lag velocity control is a modification of PI control that allows increased system performance. Lead-lag works by adding a lead circuit across the tachometer feedback resistor. This lead path provides a derivative term, which advances the phase of the loop. This phase advance or "lead" allows the proportional gain to increase, typically by about 40%. The integral gain increases by even more. These increases improve command response and disturbance rejection. Lead-lag provides significant enhancement while adding little cost; however it is a little more complicated than PI to tune.

Figure 1 shows the schematic of an analog lead-lag velocity controller; it's equivalent to the proportional-integral (PI) velocity controller from last month with the addition of an R/C lead circuit: R_A and C_A.

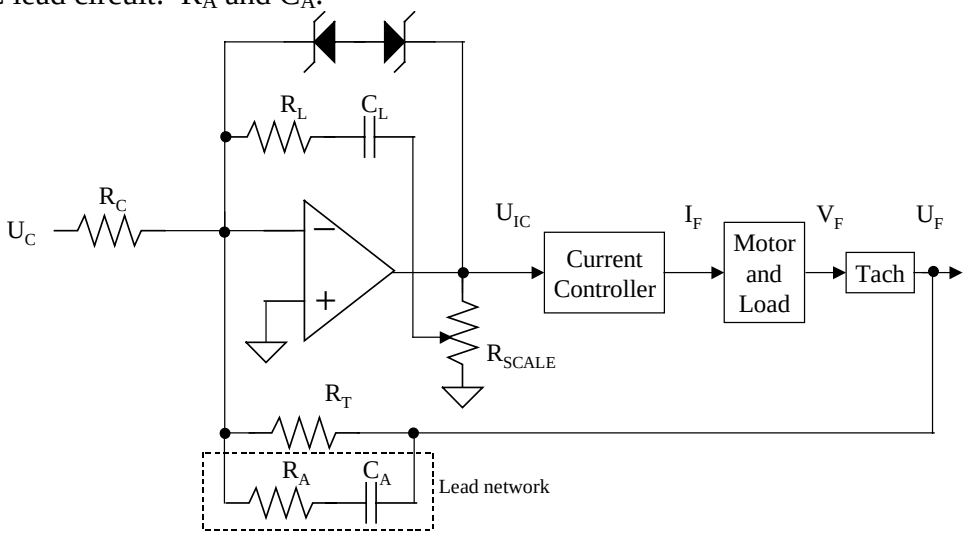
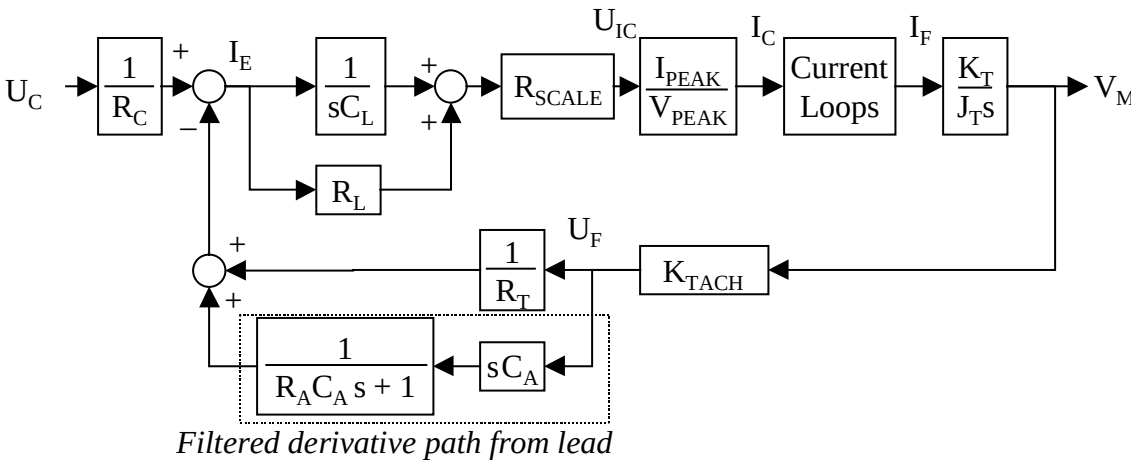


Figure 1. Schematic representation of lead-lag velocity controller

Figure 2 shows a control-loop diagram for the schematic. Notice that this is like the analog PI loop except for the addition of the "filtered derivative" path contributed by R_A and C_A (note that the "s" in "s C_A" indicates a derivative). The presence of a derivative term explains why lead-lag is occasionally referred to a "PID" control. R_A provides a low-pass filter for the derivative. Without R_A, the derivative would be too noisy to be useful in most industrial systems.



Filtered derivative path from lead

Figure 2. Control-system block diagram

Tuning a lead-lag controller is similar to tuning an analog PI controller. In both cases, you tune in zones (see June, 2000). With a lead-lag controller, there are two zones: proportional and integral. As with the PI controller, apply a small-magnitude velocity step command to the drive. Adjust the step to be as large as possible without calling for peak current from the drive. Now, adjust the zones, one at a time.

For Zone 1, zero the integral zone (short C_L) and set R_L as high as possible without generating overshoot. Now, increase R_L by about 40%. This will generate overshoot that will be cured by the lead network. Select the value of R_A; it is a compromise between performance and noise. Smaller R_A will allow larger increases in R_L, but makes the system more susceptible to noise. A common value for servo systems is to set R_A to about 1/3 of the tach scaling resistor, R_T.

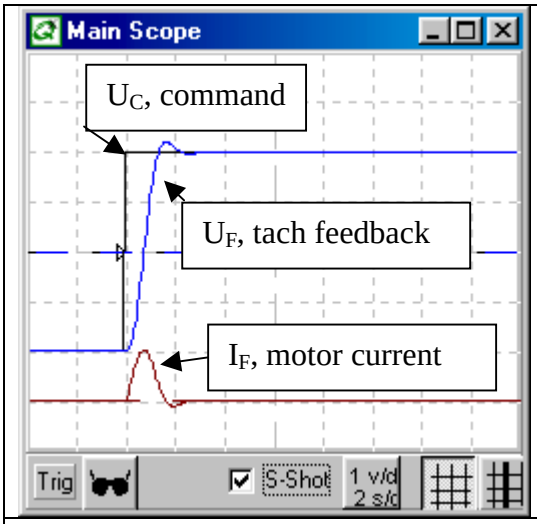
For Zone 2, increase C_L until the overshoot reaches 10-15%. The response with these gain settings is shown in Figure 3. Remember to set the R_{SCALE} pot away from the ends of travel. That way, if the inertia varies up or down in the future, you can use R_{SCALE} to adjust performance.

Static Stiffness

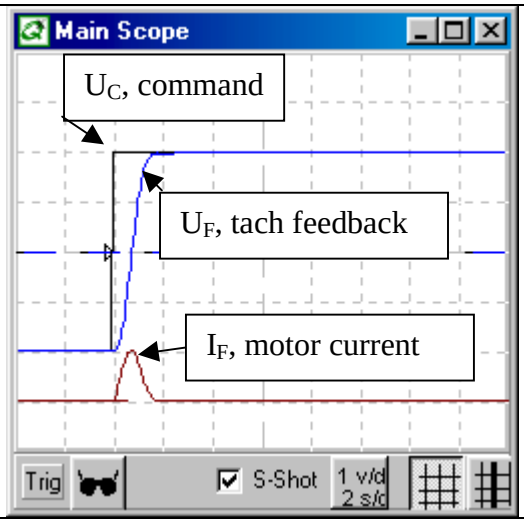
In addition to allowing about 40% higher proportional gain, lead-lag also offers the opportunity for even greater improvement in static stiffness. Static stiffness is the resistance you feel from a servomotor when you apply torque to the shaft by hand while it tries to hold position. Static stiffness is the measure of the system's ability to resist constant torque disturbances such as the effects of Coulomb (sliding) friction or gravity. Higher static stiffness allows a machine to be more accurate in the presence of constant disturbances.

The key servo gain that determines static stiffness is the integral gain, which, here, is set by C_L; smaller values of C_L produce higher static stiffness. For example, in this lead-lag controller, C_L was adjusted to 0.004 uF; in the equivalent PI controller from last month, C_L could be no smaller than 0.01 uF without generating excessive overshoot. So, lead-

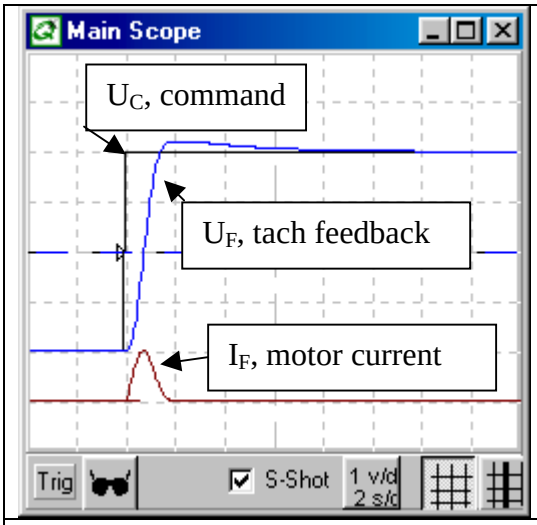
lag, in this case, allowed C_L to be reduced by 40%, producing about 2.5 times more static stiffness than PI control.



Zone 1, Step 1: Short C_L and open C_A to create a proportional controller. Adjust R_L to 500 kOhm to get maximum response without overshoot; then increase it by 40% to 700 kOhm in anticipation of the positive effects of the lead network. Scaling: U_C and U_F: 50 mV/div; this is equivalent to about 44 RPM for the scaling of this system. I_F: 20 Amps/div.



Zone 1, Step 2: Set R_A to 6.8 kOhm, about 1/3 of R_T. Without adjusting R_L, adjust C_A to eliminate the overshoot. Here, a setting of 0.005 uF eliminates all overshoot.



Zone 2: Adjust C_L to get about 10% overshoot in response to a step. Here, a value of 0.004 uF produces little overshoot while still giving 2.5 times more static stiffness than could be attained with the PI controller.

Laboratory

Want to tune an analog lead-lag velocity controller yourself? Then log onto www.motionsystemdesign.com and download the most recent version of the ModelQ simulation program. Launch the program and select November's model. Click "Run." This is the analog lead-lag controller discussed in this column. The command (blue) and response (black) are overlaid; the current is shown below; it's a good idea to monitor current when tuning to ensure that the system remains out of saturation. Remember, if you have questions about constants, click on the button "About this model…."

To tune the system, first tune Zone 1. Set $C_L = 0$ and adjust R_L to where some overshoot starts to appear and then adjust it down. In this case, the highest value of R_L that does not produce overshoot is R_L = 500 kOhm; raise R_L by 40% to 700 kOhm in anticipation of the improvement that will be provided by the lead circuit. Set R_A to 1/3 of R_{TACH} or about 6.8 kOhm. Now, adjust C_A to 0.005 uF to remove overshoot. Finally, add in C_L. Always start with a fairly large value of C_L. Remember that lower values of C_L create larger servo gains. Here, a value of 0.005 uF gives a good step response: about 10% overshoot.