Teaching Your PI Controller to Behave (Part X)

Dave (Wisconsin) Wilson May 15, 2013

Dave Wilson, Motion Products Evangelist, Texas Instruments

Throughout this series we have discussed a practical and efficient way to tune the PI controllers in a cascaded velocity loop by simply choosing a factor that determines the desired damping of the system. From this information, plus a rudimentary knowledge of some of the motor and load parameters, the PI coefficients for the velocity loop and the inner current loops can be calculated. We developed this technique by first ignoring certain aspects of the control loop, such as the pole of the current controller. But in blog 6 we reintroduced the current controller's pole back into the discussion, and evaluated its impact on system performance.

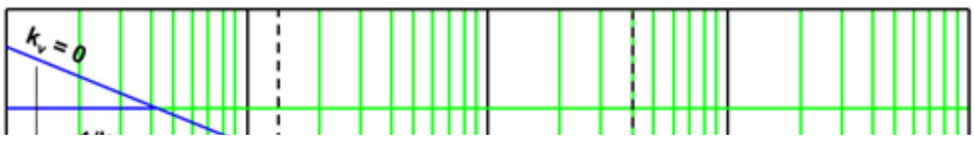
Today, let's look at the other end of the frequency spectrum. At low frequencies, viscous damping may affect your velocity loop response times by changing the phase margin at the unity-gain frequency. When viscous damping is present, it siphons off part of the motor's torque to do work to move fluid. Since viscous torque is directly proportional to the speed of the load, we can rewrite the load's transfer function to be:

$$\frac{\omega(s)}{T(s)} = \frac{1}{Js + k_v} = \frac{1}{k_v\left(\frac{J}{k_v}s + 1\right)}$$

Equ. 1

where k_v is the viscous damping factor

As you can see from Equ. 1, adding the viscous damping term moves the pole that was at s = 0 to s = k_v/J. Figure 1 shows how the addition of viscous damping changes the load's Bode plot.



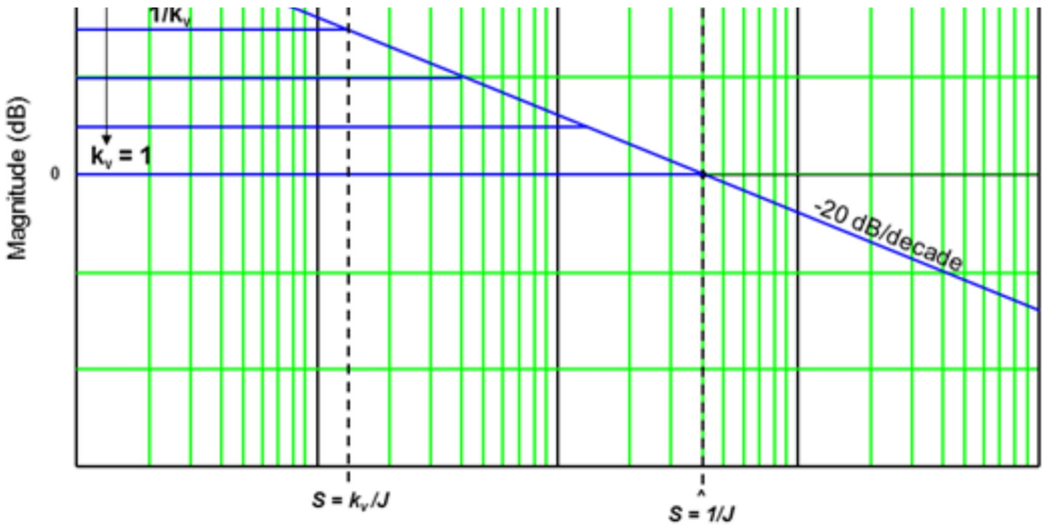
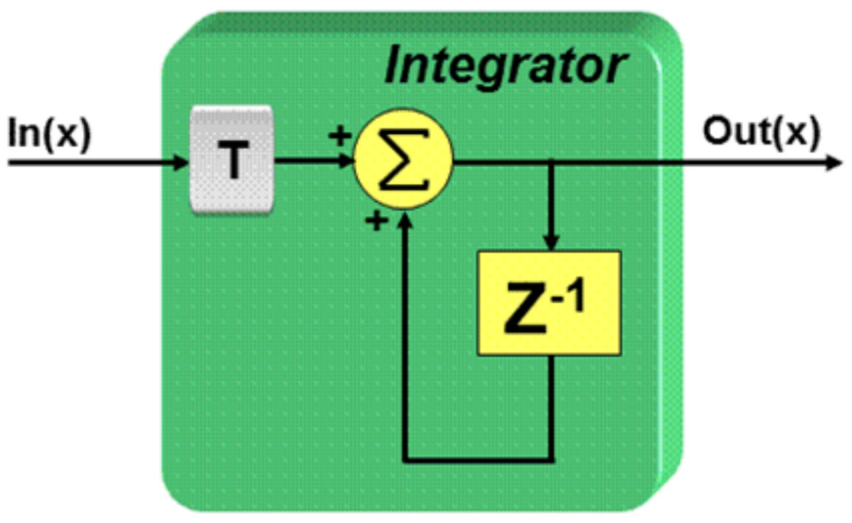


Figure 1. Effect of Viscous Damping (k_V) on the Load Bode Plot

From Figure 1, we see that as viscous damping increases from zero, low frequency gain plateaus to a value of $1/k_V$. The net effect on the phase plot is to add more phase margin at lower frequencies, since the phase lag of a load with viscous damping will *always* be less than a load with inertia only. As a result, stability should actually *improve* for non-zero values of k_V. However, the response time may take a hit, depending on where the velocity open-loop unity-gain frequency is with respect to the pole frequencies shown in Figure 1. So if your system response is sluggish and the motor doesn't seem to put out as much torque as it is rated for, you might want to investigate how much viscous damping your system has.

Before closing out this series, I want to address one more important topic. I have seen and experienced situations myself where the PI coefficients were calculated correctly, but when they were incorporated into the code, the motor either screamed like a banshee or sat there like a lump and did nothing. So what happened? In many cases, it was one of the following problems:

1. Forgetting to account for the sampling frequency effect on the GAIN of the integrator term. Figure 2 shows how to implement a typical integrator for a digital PI controller. To scale the output to match what an analog integrator would provide, we must first multiply the input signal by the sampling period "T". In order to avoid two separate multiply operations, most code examples simply lump T together with the integrator gain term. If T is not accounted for, you will end up with an integrator gain that is MUCH higher than you anticipated.



$$Out(x) = Out(x-1) + In(x)*T$$

or...

```
Out += In*T;
```

(where "T" is the time in seconds between consecutive executions of the above code segment)

Figure 2. Typical Implementation of a Digital Integrator

2. So far we have assumed there are no limitations on the number format itself. If you are using a floating-point processor, then you don't have to worry about the fractional component of your PI terms. But most motor control applications are implemented on fixed-point machines for cost reasons. The good news is that TI has developed a linkable library that is in the ROM of most of our C2000 processors which solves this problem for you. It is called the "IQ Math" library which stands for "Integer Quotient". This allows you to handle floating point values with ease on a fixed-point machine without suffering from the performance hit you typically get with a full-blown floating-point package. IQ math creates a new variable type in your code which is designated by a "Q" followed by a number. For example, let's say you have a 32 bit variable which is typed as a "Q24" variable type. This means that any variable of this type is assumed to have a 24-bit fractional component, and an 8-bit integer part. But what occasionally happens is that someone copies TI example code into their design without realizing that our coefficients are represented in IQ format. For example, if you calculate *your* I-gain to be 10,000 ($00002710 in Q0 format), but you didn't realize that the TI code assumes the variable is in Q24 format, you will end up with an integrator gain of 0.596E-3 instead of 10,000. Obviously, a big difference! If you make the same mistake for all of

the PI coefficients, the motor will most likely just sit there and do nothing since all the gains are way too low. So, a word to the wise; make sure you know what numerical format your coefficients are represented by.

3. In some cases, the PI coefficients for an AC Induction Motor drive are calculated correctly and the motor runs just fine… until the motor speed exceeds its rated value, at which point the controller response becomes underdamped. The reason for this can be found back in blog 3 of this series. The motor transfer function between q-axis current and electromagnetic torque for an AC induction motor is repeated below.

$$Mtr(s) = \frac{3}{4} P \frac{Lm^2}{Lr} I_d$$

Equ. 2

For speeds up to the rated speed, Id usually stays constant. But to achieve speeds beyond the rated speed, you need to reduce the flux in the machine by lowering the value of Id. You can probably see where I'm going with this. When you reduce Id, you reduce the gain of the entire velocity loop gain. If it is reduced too much, you can end up moving the unity-gain frequency to a lower value where the effect of the two poles at s=0 have more effect on reducing the phase margin. To keep the system response from changing, the Kc (or Kp) term in the speed PI controller can be increased in direct proportion to the reduction in Id.

4. Finally, the scaling of the PI coefficients throughout this series has been done assuming we want to represent real system values throughout the signal chain. For example, the output of the velocity PI controller equals the actual input reference current in amps for the current controller. The output of the current controller equals the actual voltage applied to the motor winding. But in many designs, the PI controller outputs are normalized to "Per Unit" (PU) scaling where a value of 1 represents the maximum value possible, and a value of -1 represents the minimum value possible. For example, a current regulator's output might be scaled in such a way that 1 corresponds to 100% PWM, and -1 corresponds to 0% PWM. The advantage of this approach is that you can quickly adapt your code to different systems by simply changing your per unit scaling values. To convert from real-value scaling to PU scaling for Ka, Kb, Kc, and Kd (including the sampling period for the integrators), use the following formulae:

$$K_{a\_scaled} = K_a \frac{I_{scale}}{V_{scale}}$$

Equ. 3

.

where: *Iscale* is the actual current value corresponding 1.0

Vscale is the actual voltage value corresponding to 1.0

$$K_{b\_scaled} = K_b T_i$$

Equ. 4

where: T_i corresponds to the sampling period of the current loop.

$$K_{c\_scaled} = K_c \frac{Vel_{scale}}{I_{scale}} \frac{2}{P}$$

Equ. 5

where: Vel_{scale} = the actual velocity value in rad/sec corresponding to 1.0

Iscale = the actual current value corresponding to 1.0

P = the number of rotor poles on the motor

$$K_{d\_scaled} = K_d T_v$$

Equ. 6

where: *Tv* corresponds to the sampling period of the velocity loop

The PU scaling for traditional parallel PI structures is:

$$K_{p\_scaled} = K_{c\_scaled}$$

Equ. 7

$$K_{i\_scaled} = K_c K_d \frac{Vel_{scale}}{I_{scale}} \frac{2}{P} T_v$$

Equ. 8

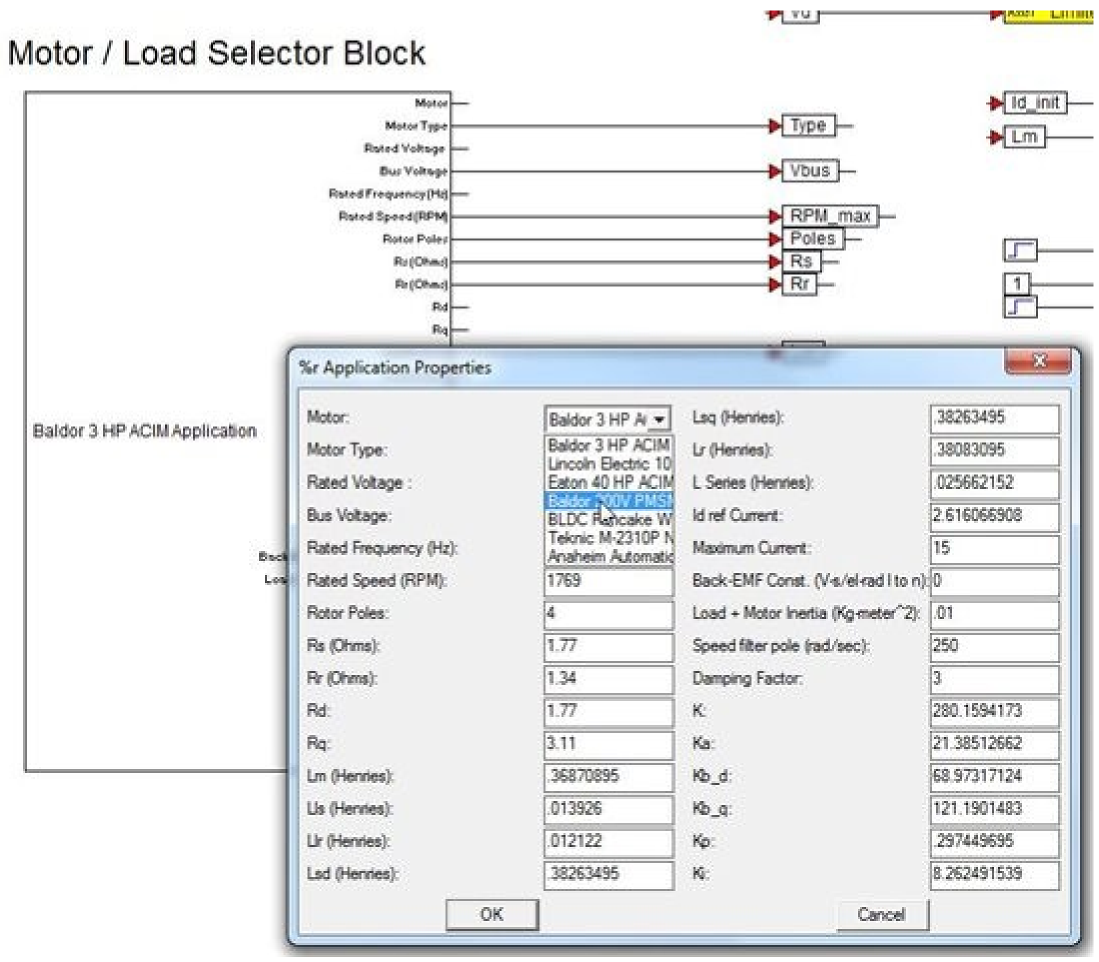
where: Vel_{scale} = the actual velocity value in rad/sec corresponding to 1.0

Iscale = the actual current value corresponding to 1.0

P = the number of rotor poles on the motor

Tv corresponds to the sampling period of the velocity loop

To assist you with the design of your FOC based speed controller, I have created a VisSim simulation (FOC.vsm) which accepts inputs from an Excel spreadsheet (Motor.xlsx.) If you open the spreadsheet, you can see the parameters for several different motor/load examples as well as the PI controller coefficients which are automatically calculated. You can also add your own motor and load parameters in the spreadsheet if you want. To create a text output file which is required by the VisSim simulation, you must perform a "Save As" option, and in the "Save as Type" field, select "CSV (Comma delimited) (*.csv)." Assuming that this file is in the same folder as "FOC.vsm", you can then open "FOC.vsm" and simply click the run icon on the toolbar at the top of the page to run the simulation. You can see the speed response (as well as plots of other system variables) by scrolling over to the right side of the simulation window. The simulation creates a step change in the commanded velocity at t=0 which saturates the system, and another step change at t=0.5sec which doesn't saturate the system, allowing you to evaluate the system response under both conditions. If you would like to evaluate another motor /load example, simply double-click on the Motor/Load Selector Block in the bottom-left corner of the simulation window and choose a different option, as shown below.



Motor:	Baldor 3 HP A	Lsq (Henries):	.38263495
Motor Type:		Lr (Henries):	.38083095
Rated Voltage :		L Series (Henries):	.025662152
Bus Voltage:		Id ref Current:	2.616066908
Rated Frequency (Hz):		Maximum Current:	15
Rated Speed (RPM):	1769	Back-EMF Const. (V-s/el-rad l to n):	0
Rotor Poles:	4	Load + Motor Inertia (Kg-meter^2):	.01
Rs (Ohms):	1.77	Speed filter pole (rad/sec):	250
Rr (Ohms):	1.34	Damping Factor:	3
Rd:	1.77	K:	280.1594173
Rq:	3.11	Ka:	21.38512662
Lm (Henries):	.36870895	Kb_d:	68.97317124
Lls (Henries):	.013926	Kb_q:	121.1901483
Llr (Henries):	.012122	Kp:	.297449695
Lsd (Henries):	.38263495	Ki:	8.262491539

Figure 3. Motor/Load Selector Block in the FOC.vsm Simulation File

If you would like to take this simulation out for a test drive, but you don't have VisSim installed on your machine, simply follow the instructions below:

1. Go to Visual Solutions' website.

2. In the top right of the browser window, click on "Create account"

3. After you create an account, click on the "Downloads" tab, and select "VisSim Software".

4. Click on the first link on the top of the page that says "VisSim – Core simulator + UML State Charts + Analyze"

5. Run the installer program. This will install the free 60 day trial version of VisSim.

Congratulations…we made it! Thanks for investing your time with me as we have looked at different ways to make your PI control loop behave. I hope you learned some neat tricks that can help you "stay in tune" with your motor control design.

By the way, if you have suggestions for other motor control topics you would like for me to address in a future blog, I would love to hear from you. :-) And of course…

Keep Those Motors Spinning,

Dave

www.ti.com/motorblog

.

10 comments 0 members are here

Shane Colton over 10 years ago

Thanks for yet another awesome motor control reference series! It's great to have such a clear explanation of PI tuning as it relates to motor control. Very glad to see your triumphant return to the motorblog and (coincidentally?!) the release of InstaSPIN FOC.

If you're still taking suggestions for future topics, a great help would be a series on three-phase PWM generation techniques, including/especially SVM and how it differs from/overlaps with carrier-based PWM and third-harmonic injection. (I recall a certain teaser in the last part of your series on single-phase PWM generation.)

Dave (Wisconsin) Wilson over 10 years ago

Thank you Shane for the very encouraging feedback! Yes, I had been missing from the blog scene for a while, and yes, it was because of the "all hands on deck" thrust to get InstaSPIN-FOC released.

The topic you have requested is something that I cover briefly in my motor control seminars, but I agree that a more in-depth discussion would be helpful to many engineers who are using PWM and SVM to drive their motors. Look for it in an upcoming series!

Thanks,

Dave

Alecksey Anuchin over 10 years ago

Thank you for your interesting articles. And what is about saturation of digital PI-controller?

WBR, Alecksey

Dave (Wisconsin) Wilson over 10 years ago

Hi Alecksey,

I just realized that your comment also included a question. Unfortunately, I didn't realize this until over 3 mo. after you asked it. :-(

It is very common to experience saturation in digital PI-controllers, epecially under transient conditions. The saturation limit for the velocity PI controller is set by the current limit of your motor and drive. The saturation limit for the current PI controller(s) is set by the dc bus level. In all cases, the goal is to make the digital PI controller emulate what an analog PI controller would do. When an op-amp saturates, it simply clamps at the maximum output level, and then returns to normal operation once the input stimulus is removed. This is what we would like the digital PI controller to do, making sure that it doesn't wrap around due to the modulo nature of binary arithmetic. In both analog and digital PI controllers, we must be sure to manage the integrator windup that can occur during output saturation. This is discussed in more depth in part 9 of this series.

-Dave

Sivabalan Mohan over 10 years ago

Hi Dave,

Thanks a lot for this blog series. It was very informative and easy to understand. I have a question regarding motor saliency - if Ld and Lq values are different by quite a magnitude, how do the PI parameters change. I would assume that the current controllers for each current component would use the respective L component?

Thanks

Siva

▼ View More