Teaching Your PI Controller to Behave (Part VI)

Dave (Wisconsin) Wilson Apr 5, 2013

Dave Wilson, Motion Products Evangelist, Texas Instruments

I am writing this particular blog installment while lying by a swimming pool in a resort on the Riviera Maya. There are lots of beautiful distractions which are competing for my attention, including my gorgeous 20 month old granddaughter who is playing at my feet. So I have to disclaim any liability related to the quality of this particular blog!

So far in this series we have discussed how to set the coefficients for the PI controllers in a cascaded speed control loop. We saw that Kb is used for pole-zero cancellation within the current loop. Kc and Kd are indexed to the velocity feedback filter's time constant, and are calculated by first selecting a suitable damping factor which is related to the responsiveness and damping of the system. We also saw that Ka sets the current controller's bandwidth. But what should that bandwidth be?

It turns out that we have two competing effects vying for control of where we set Ka. On the bottom end of the frequency range, we have the velocity feedback filter pole with some really nice tan lines that wants to push the current controller pole to higher frequencies so as to not interfere with our nice tuning procedure. But we can only go so high before we start running into other problems. Let's take a look at both ends of the frequency spectrum in an effort to understand these issues, and hopefully gain some insight into how to judiciously set Ka. But first, let's rewrite the open-loop transfer function of the velocity loop to once again include the current controller's transfer function:

$$GH(s) = \frac{K\,K_c K_d\left(1+\dfrac{s}{K_d}\right)}{s^2\left(1+\dfrac{L}{K_a}s\right)(1+\tau s)} \qquad \text{Equ. 1}$$

where: K is a coefficient that contains several terms related to the motor and load

Kc and Kd are the PI coefficients for the velocity loop

L is the motor inductance

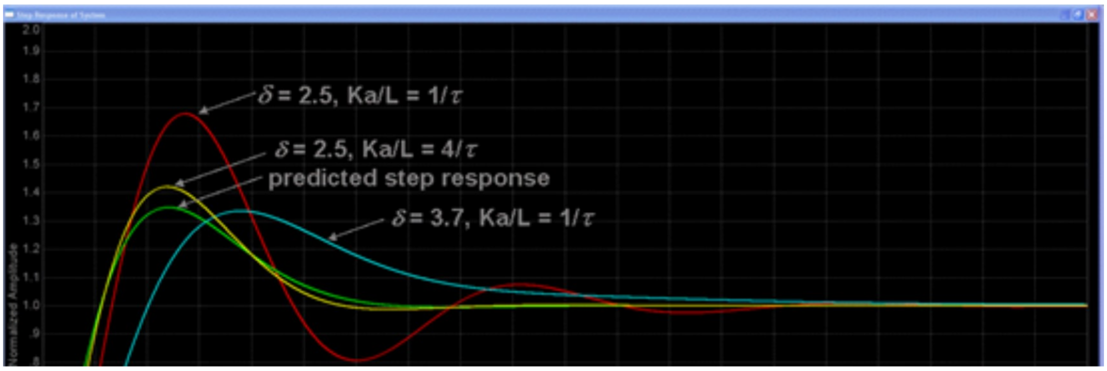
Ka is one of the PI coefficients for the current loop

τ is the time constant of the velocity feedback filter

s is the Laplace frequency variable

Our tuning procedure assumes that the current controller's closed-loop gain is always unity, which implies it has infinite bandwidth. But in reality, as long as the current controller's bandwidth is at least 10x higher than the velocity loop unity gain frequency, our tuning procedure is still pretty good at predicting the system response. If this condition isn't satisfied, the current controller pole starts interfering with the velocity loop phase margin, resulting in a more underdamped response than our tuning procedure would otherwise indicate.

Figure 1 shows an example case to illustrate this point. The green curve represents what the tuning procedure predicts the normalized step response should be for a system with a damping factor of 2.5. The red curve shows what happens when the current controller's bandwidth is reduced to equal the velocity filter's bandwidth. The system is still stable, but the damping is much less than predicted from our tuning procedure. At this point, we have two options…either increase the damping factor (and consequently lower the frequency response of the velocity loop), or increase the current loop bandwidth by increasing Ka. The cyan curve shows the first option where we increase the damping factor just enough to bring the overshoot down to the predicted value. Unfortunately this increases the step response transient time as well. The yellow curve shows the latter option where we put the damping factor back to 2.5, and increase the current controller's pole to be 10x higher than the velocity loop unity gain frequency. As you can see, the actual response more closely resembles the predicted value. The higher the current controller's bandwidth is, the closer the response will resemble the predicted response.



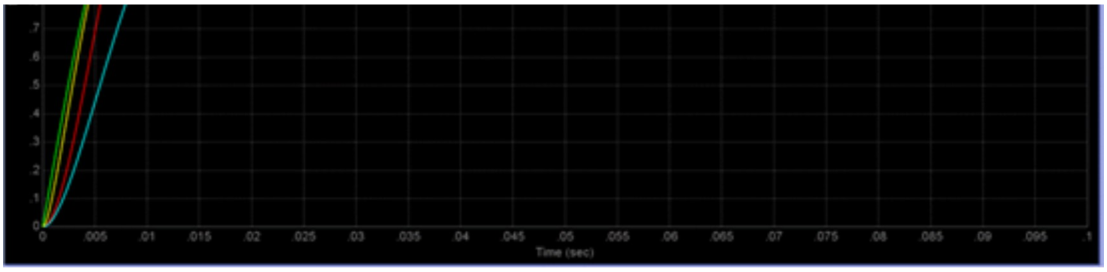


Figure 1. Normalized Small Signal Step Response

So from this exercise, we might conclude that the best strategy is to set the current controller bandwidth to be as high as possible. But is this really the best course of action? As my dear ol' grandma used to say, "Never use a 1 GHz op-amp when a 1 MHz op-amp will do the job." (Well, she never actually said it in those exact words, but you get the point). Usually high bandwidth in *any* system results in unruly and obnoxious behavior, and should only be used if absolutely necessary. In this case, high current loop bandwidth often results in undue stress on your motor, since high frequency *current* transients and noise translate into high frequency *torque* transients and noise. This can even manifest itself as *audible* noise! But there is also another limit on your current loop bandwidth: the *sampling frequency*.

Take a look at Figure 2 which shows a digital Field Oriented Control (FOC) based Variable Frequency Drive (VFD). To simplify the discussion, we will assume that the entire control loop is clocked by a common sampling signal, although in real-world applications we often choose to sample the velocity loop at a much lower frequency than the current loop to save processor bandwidth.

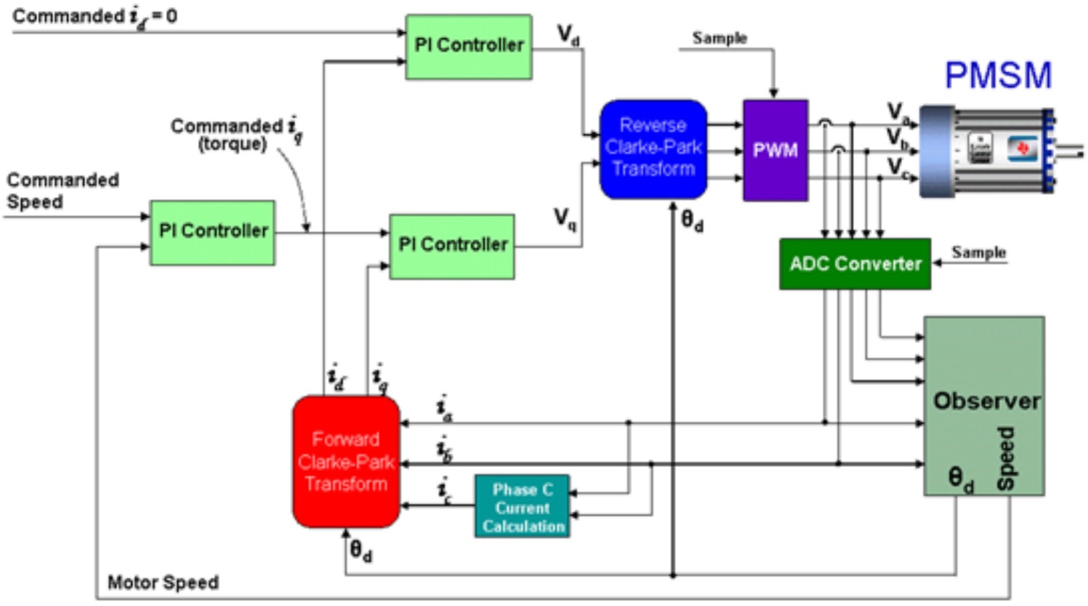


Figure 2. Digital Field Oriented Control System for a PMSM.

In an analog system, any change in the motor feedback signals immediately starts having an effect on the output control voltages. But with the digital control system of Figure 2, the motor signals are sampled via the ADC at the beginning of the PWM cycle, the control calculations are performed, and the resulting control voltages are deposited into double-buffered PWM registers. These values sit unused in the PWM registers until they are clocked to the PWM output at the start of the next PWM cycle. From a system modeling perspective, this looks like a sample-and-hold function with a sampling frequency equal to the PWM update rate frequency. The fixed time delay from the sample-and-hold shows up as a lagging phase angle which gets progressively worse at higher frequencies. Figure 3 shows a normalized frequency plot of the phase delay for a sample-and-hold function, where the sampling frequency is assumed to be 1. As you can see, the phase delay reaches down into frequencies much lower than the sampling frequency. For example, at one decade below the sampling frequency, the S&H is still affecting a phase shift of -18 degrees.

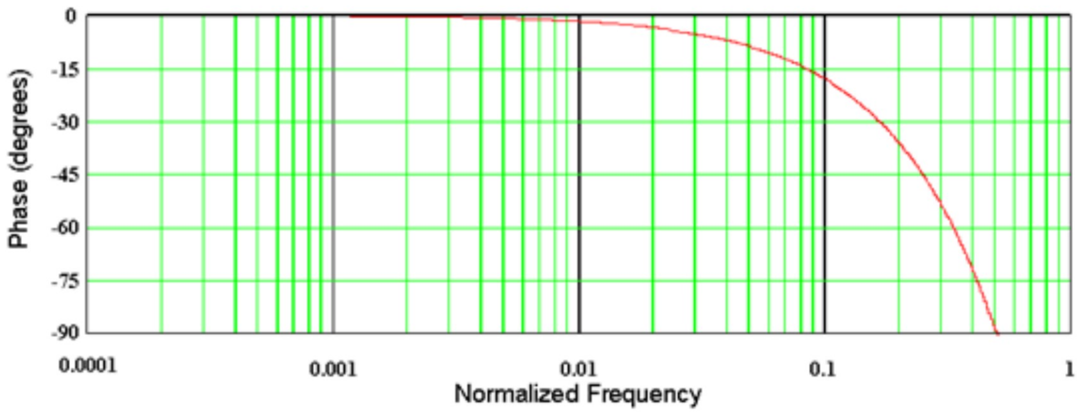


Figure 3. Phase Lag Plot for a Sample-and-Hold, Fs = 1

Since the current controller processes higher bandwidth signals than the velocity loop, it is usually the current loop that suffers most by the S&H effect of the PWM module. Since the PWM S&H is in series with the signal path for the current loop, its magnitude and phase contributions add directly to the open-loop response for the current controller. If we rewrite the equation for the open-loop response of the current controller (assuming Kb = R/L), we end up with the following function:

$$GH_c(s) = PI(s)\cdot\frac{I(s)}{V(s)} = \left(\frac{K_a}{L}\right)\left(\frac{1}{s}\right)$$

Equ. 2

This is simply an integrator function with gain of Ka/L. Unity gain will obviously occur when s = Ka/L. The single pole at s = 0 implies that the unity gain frequency will have a 90 degree phase shift, which also implies a 90 degree phase margin. However, for a digital control system, you must add the phase lag shown in Figure 3. To do this, calculate for the following frequency ratio:

$$\frac{K_a\, T_s}{2\pi L} \quad \textit{where Ts is the sampling period}$$

Equ. 3

Then use Figure 3 to determine how much phase lag you must subtract off of your phase margin. For example, if KaTs/(2π L) = 0.1, then you must subtract off 18 degrees from your phase margin, leaving you with a comfortable 72 degrees. In most designs you probably want to keep Ka/(2π L) at least an order of magnitude below the sampling frequency. So using this assumption as well as the constraints from the velocity loop tuning procedure, we can now write a general “rule of thumb” expression for the range of Ka:

$$\frac{10L}{\delta\tau} < K_a < \frac{\pi L}{5T_s}$$

Equ. 4

where: *L* is the motor inductance

τ is the time constant of the velocity filter

δ is the damping factor

Ts is the sampling period

In most designs this still leaves a fairly broad range for the value of Ka. In my experience, I have found that erring on the side of the current bandwidth being too low is usually better than being too high. Snappy waveforms look nice on an oscilloscope, but your application may not need (or want) all that bandwidth.

Up until now, we have only dealt with “small signal” conditions (i.e., linear operation with no saturation effects). But in the real world, step transient responses almost always saturate the system’s voltage or current levels, which tend to lengthen the response times. When this happens, you can increase the PI gains all you want, but it won’t speed up the response. In fact, it will usually just make the overshoot worse, since the integrator is acting on a gained-up error signal, which it will just have to dump eventually. So how do we deal with this problem? Are we doomed to simply using low integrator gains? It turns out that there is another solution which doesn’t involve changing your integrator gains, which I will discuss in my next blog. Until then…

Keep Those Motors Spinning,

Dave

8 comments 0 members are here

Ros River over 10 years ago

Thank you very much for the perfect bridge between highschool theory and the real industrial application. Could you please answer two question regarding the provided Excel-Spreadsheet:

1. The Back-EMF Constant from the motor datasheet doesn't meet the Excel-value: in the Teknic Motor M-2310 datasheet I find Ke=4.64, in your calculation - 0,0063954, for Baldor BSM80N-233 - 34.7 and 24.6, the sheet - 0.0957 . I understand that you converted rpm in rad/sec, so the coefficients 60, pi and 0.7 (rms/peak) should be somewhere in the equation, but it doesn't work together.

2. How did you find out the speed filter pole, for example the 250rad/sec for Teknic motor? The value of maximum deceleration from the Teknik datasheet is 250rad/sec2, so the speed can't physically change faster than that. What happens to that value in an application with an encoder?

Thanks

Dave (Wisconsin) Wilson over 10 years ago

Hi Ros,

Thanks for your feedback on my PI series, and also for your excellent questions.

Regarding your first question, so many people have inqured about this that I am thinking about amending my blog to address it. Basically, you need to understand what units the datasheet back-EMF value is listed as, and then convert that to "volts (peak, line-to-neutral) per electrical radians per second". The Teknic motor is listed as 4.64 volts (peak, line-to-line) per 1000 mechanical revolutions per minute. So here is the conversion process for the Teknic motor:

To convert volts (line-to-line) to volts (line-to-neutral), divide by sqrt(3).

To convert 1k mechanical revolutions per minute to mechanical revolutions per second, multiply by 60, and then divide by 1000.

To convert from mechanical revolutions per second to electrical cycles per second, divide by 4 (since there are 8 poles)

To convert from cycles to radians, divide by 2*pi.

So, 4.64V peak, line-to-line, per 1000 RPM translates to 6.3954 mV peak, line-to-neutral per electrical radians per second.

Regarding the second question, the speed filter pole setting is application dependent. Most applications require some kind of speed feedback filtering due to the nature by which speed feedback signals are obtained. Whether it is from an encoder or an observer, you usually have to filter the signal. In my case, I was using the FAST observer which is part of our InstaSPIN-FOC product. With InstaSPIN-FOC, the filter value is selectable. For my system, I was able to get satisfactory performance with a pole value of 250 rad/sec2, but admittedly, this was a subjective decision.

Thanks,

Dave

Rock Meng over 10 years ago

First , thanks Dave . I have learned a lot about how to set PID parameter . But I still have some question .

1. For example , Equ3 and Equ4 is disappeared in chapter 12 of User's Guide , the expression is differrent that how to calculate the Ka(current controller Kp in User's Guide) .

2. In the software code of CTRL_setParams(CTRL_Handle handle,USER_Params *pUserParams)

```
// set the default Id PID controller parameters

Kp = _IQ(0.1);

Ki = _IQ(pUserParams->ctrlPeriod_sec/0.004);

Kd = _IQ(0.0);

outMin = _IQ(-0.95);

outMax = _IQ(0.95);

 // set the default speed PID controller parameters

 Kp = _IQ(0.02*pUserParams->maxCurrent*pUserParams->iqFullScaleFreq_Hz/pUserParams->iqFullScaleCurrent_A);

 Ki = _IQ(2.0*pUserParams->maxCurrent*pUserParams->iqFullScaleFreq_Hz*pUserParams->ctrlPeriod_sec/pUserParams->iqFullScaleCurrent_A);

 Kd = _IQ(0.0);

outMin = _IQ(-1.0);

outMax = _IQ(1.0);
```

why the default id(iq and speed) PID controller parameters set like that ? I have no knowledge about the theory . can I changed the default values and how to change it .

3. What's the difference between the USER_MOTOR_RATED_FLUX (which Identified TOTAL flux linkage between the rotor and the stator (V/Hz)) and λr (Mtr(s)=3/4Pλr) . I don't think λr is constant , why λr= Ke ?

4. I also confused about that in your blog and the User's Guide you always talk about a series path topology of a PI controller.

but in the software code it's parallel a path topology of a PI controller .

Thanks .

Vincenzo Pappano *over 8 years ago*

Dave,

Thank you very much for this spot-on blog series. It has brought some sense to an otherwise (apparently) obscure topic. I have a question about Figure 3. The 'normalized frequency' is obtained as the ratio of which values?

Thank you for your time and consideration.

Sincerely,

Vincenzo

Dave (Wisconsin) Wilson *over 8 years ago*

Hello Vincenzo,

Thank you for your feedback. I am glad you are enjoying the series.

All of the frequencies in figure 3 are ratiometric to the the sampling frequency. So if you have a sampling frequency of 20 kHz, then a "normalized frequency" of 0.1 corresponds to 2 kHz. The point of this graph is to show that the phase lag curve resulting from the sample-and-hold process is ratiometric to the sampling frequency, regardless of the actual value of the sampling frequency.

Regards,

Dave

Bui Van Linh *over 1 year ago in reply to Dave (Wisconsin) Wilson*

Hello Dave,

thank you for the wonderful blogs.
There is one point in this blog that is not clear to me. It its the calculation of the phase shift for the sample and hold in figure 3.

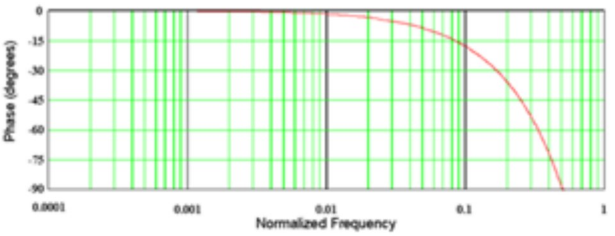
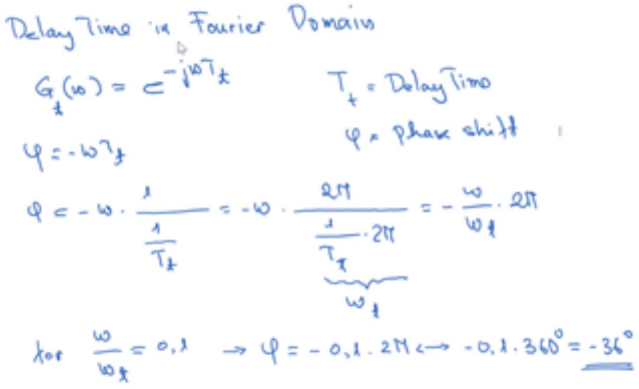


Figure 3. Phase Lag Plot for a Sample-and-Hold, Fs = 1

The phase shift is caused by the delay time and you can calculate it very easily (see my caculation)



Delay Time in Fourier Domain

$G_t(\omega) = e^{-j\omega T_t}$ $\qquad$ T_t = Delay Time

φ = phase shift

$\varphi = -\omega T_t$

$$\varphi = -\omega \cdot \frac{1}{\frac{1}{T_t}} = -\omega \cdot \frac{2\pi}{\underbrace{\frac{1}{T_t} \cdot 2\pi}_{\omega_t}} = -\frac{\omega}{\omega_t} \cdot 2\pi$$

for $\frac{\omega}{\omega_t} = 0,1 \rightarrow \varphi = -0,1 \cdot 2\pi \longleftrightarrow -0,1 \cdot 360^\circ = \underline{-36^\circ}$

With a normalized frequency of 0.1 I have a phase shift of -36 degrees and not -18 degrees as the blog says.

where is my calculation mistake?

Thanks in advance - Bui

Shaoxiong Guo *over 1 year ago in reply to Bui Van Linh*

i have the just same question as you. do you figure it out ?

Bui Van Linh *over 1 year ago in reply to Shaoxiong Guo*

Hello Guo,

I got an answer about this question from TI.

https://e2e.ti.com/support/motor-drivers-group/motor-drivers/f/motor-drivers-forum/1130804/phase-shift-of-sample-and-hold

Regards - Bui