Teaching Your PI Controller to Behave (Part VII)

Dave (Wisconsin) Wilson Apr 13, 2013

Dave Wilson, Motion Products Evangelist, Texas Instruments

So far we have discussed the PI tuning problem in the context of a linear system. This is because under steady-state conditions when the system transients settle out, you will probably be operating in the linear region, and *that's* when your feedback loop will either oscillate or not. Therefore, performing a small-signal (linear) analysis will tell you how stable your system will be when it is not saturated. But in most real-life scenarios, the system *will* saturate because of limits on your voltage and/or current, especially under large transient conditions. This saturation effect can play some real mind games with your PI controller; especially the integrator. Since the maximum torque your motor can produce is determined by your current limit, the acceleration of the system is also limited. But the integrator in the PI speed controller doesn't know this, and it thinks it can make the motor speed up even faster by increasing its output. This increased integrator output can't help the situation since the system is already saturated. All it does is create a very large output that will cause the system to overshoot when it *does* come out of saturation. For this reason, most PI integrators employ techniques to keep them from needlessly integrating while the system is saturated.

One technique which is commonly used is *integrator clamping*, which establishes limits for the maximum and minimum values that the integrator output can have. An example of this is illustrated in Figure 1. The most common scenario is to set the clamp values to be equal to the PI output limit values. However, there is nothing that says that the integrator limits *must* equal the PI output limits, and many designs use different clamp values based on the specific application. As you might expect, the tighter you set the clamp limits, the less windup you will have to deal with when the system finally does come out of saturation.

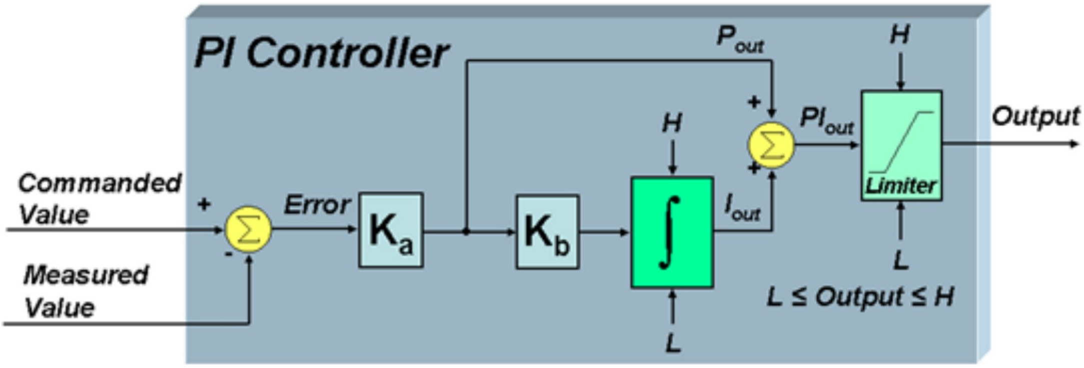

Figure 1. PI Controller with Static Integrator Clamping

However, a more robust clamping scheme is shown in Figure 2. The thinking behind *dynamic integrator clamping* is based on the rationale that if the system is already saturated by the P gain output, then why allow the integrator to continue integrating since it can’t help anyway. Only during conditions where changes in the integrator output would affect changes in the PI controller output is the integrator allowed to integrate the error signal.

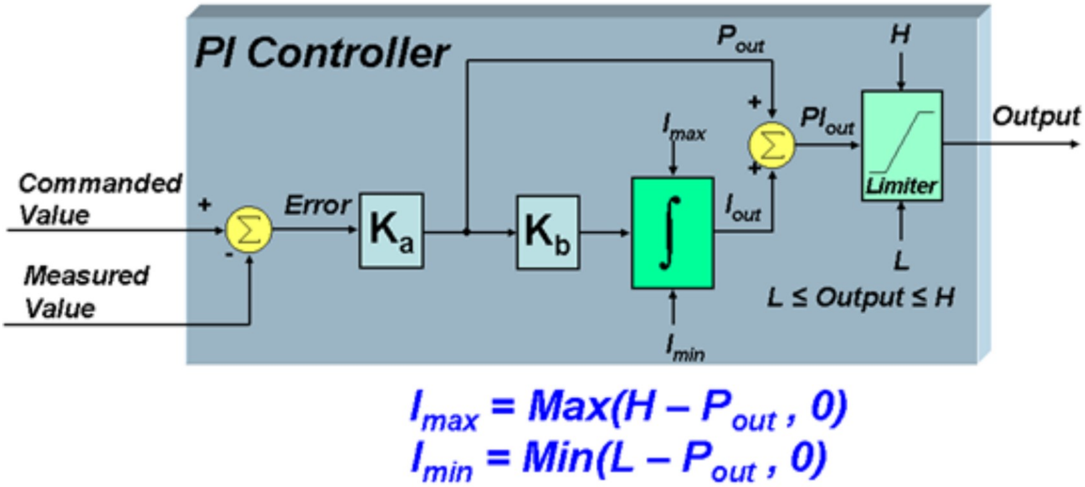

Figure 2. PI Controller with Dynamic Integrator Clamping

The effectiveness of integrator clamping can be seen by the simulated velocity responses in Figure 3. The system is instantly stimulated with a commanded velocity step from zero to a target speed of 1500 RPM, which will obviously cause saturation. Shown are the effects of system overshoot with *no* clamping, *static* clamping where the integrator clamp values equal the output clamp values, and finally *dynamic* clamping. As you can see, no integrator clamping at all is unacceptable as it results in horrendous overshoot which triggers further system saturation and oscillation. Static integrator clamping dramatically improves the system response. However, dynamic clamping improves performance even further, resulting in six times less overshoot than static clamping in this particular example. An important but somewhat obvious point to remember about integrator clamping is that it only improves the overshoot response if the clamp activates. If the commanded stimulation is small enough to keep the system in the linear mode of operation, then the overshoot you get will be determined exclusively by the damping factor.

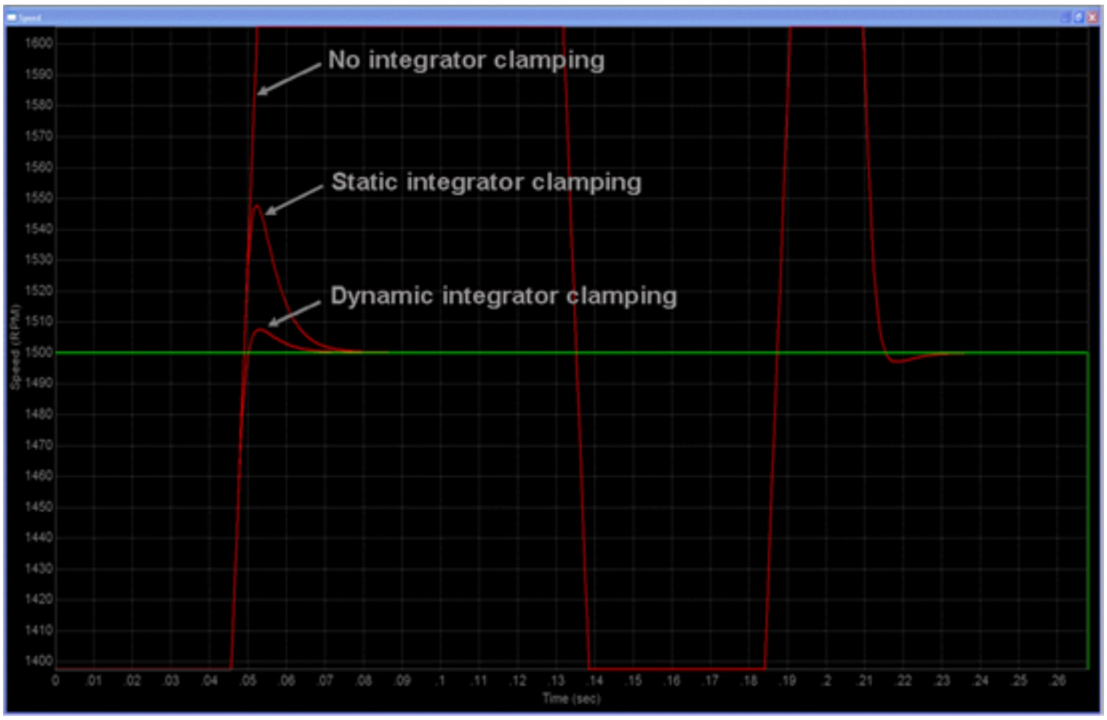

Figure 3. Example Comparison of Integrator Clamping Techniques

Another technique that has been used to mitigate windup is *integrator switching*. With this technique, the integrator is only turned on at the *end* of a commanded profile transition, or when the system output has pulled to within a specified range of the commanded input. In the latter case, the output loading must be small (or at least somewhat predictable) in order to select a meaningful threshold for when to turn the integrator on. But you must also be careful how and when you turn the integrator *off*. If you also reset the integrator output to zero when you switch it off, you can end up with a large and undesirable discontinuity in the output waveform if the integrator output was very large at the time. But not resetting the output to zero can also have undesirable consequences when the integrator is turned back on. For example, if the steady-state integrator output eventually changes polarity when it is turned back on again, the settling time will take much longer than if the integrator output had been reset. My opinion about integrator switching is that it can be an effective technique to minimize windup, but getting good results can be tricky, and it is very much application dependent.

In my next blog, I want to double back and talk some more about PI tuning. Now that we know how to put the puzzle together, the obvious question to ask is, “do we have all the pieces?” Up until now, I have assumed the answer is *yes*. But there is one piece that all the other pieces fit around that we haven’t really talked a lot about, and that is **inertia**. While this parameter is sometimes provided on a motor data sheet, it is almost never provided for the load you are driving. Without it, you can't accurately tune your velocity loop unless you rely heavily on trial and error. In my next blog, I want to introduce you to TI's new InstaSPIN-FOC algorithm, and propose a technique for how it can be used to actually *measure* system inertia. You don't want to miss this one! Until then…

Keep Those Motors Spinning,

Dave

www.ti.com/motorblog

3 comments 0 members are here

Marcellus Sauls *over 10 years ago*

Hi Dave,

Thank you so much for your excellent insights in motor control theory. I have tried to implement the PI controllers following the discussions on "teaching your PI to behave". The PI controllers behave well up until the point when I try to introduce saturation on inner current loops and outer speed loop. Is there anyway of systematically dimension the upper and lower limits for the saturation blocks. I am using a BLDC motor from Anaheim Automation BLY172s-24V-4000 with the following specs:

Rated voltage : 24 V

Rated Speed : 4000 RPM

Rated Power : 53 W

Peak Torque : 54 oz-in

Rated Current : 3.5 A

Line to Line Resistance 0.8 ohm

Line to Line Inductance 1.2 mH

Torque constant : 5.03 oz-in-A

Back EMF Voltage : 4.14 VkRPM

Rotor inertia : 0.00068 oz-in-sec^2

Thank you so much in advance

Dave (Wisconsin) Wilson *over 10 years ago*

Hi Marcellus,

Thanks for your kind comments related to my blog series.

Assuming that you are doing Field Oriented Control, setting the clamp limits can be somewhat challenging, especially if you want to do field weakening. Let's start with the speed PI controller…

The speed controller sets the commanded value of Iq for the q-axis current controller. If you are not field weakening (Id = 0), then the q-axis current controller can have all of the available current. Therefore the speed controller clamp limits are +/- Imax, where Imax is determined by the lesser of the inverter or motor steady-state current limits. But if you ARE doing field weakening, Id usually gets first dibs on the available current, and the commanded value for Iq gets what is left over. But remember, it is NOT an algebraic subtraction, but a vector subtraction. So in this case, the speed controller clamp limits are +/- sqrt(Imax^2 – Id^2).

The current PI regulators generate Vd and Vq to drive the motor. We generally assume that “0 volts” corresponds to Vdc/2, thereby allowing us to generate positive and negative voltages. Assuming you are not overmodulating, the clamp limits applied to the current controller outputs must be set in such a way so as to limit the applied phase voltages to values that don’t exceed the power supply limits. Once again, the d-axis current regulator gets first dibs on the available voltage. For example, assuming simple sinusoidal modulation, the clamp values for the d-axis current regulator are +/- Vdc/2. The clamp limits for the q-axis regulator would therefore be +/- sqrt((Vdc/2)^2 – Vd^2).

Hopefully this gives you some insight into how to set the clamp limits.

Best Regards,

Dave

Michael Holmes1 *over 7 years ago*

For current control, you are guaranteed to get integral windup if sqrt(Vd*Vd + Vq*Vq) > maxVoltageRadius because there's not enough voltage to command the required current for (Id,Iq). So, having a hard clamp for Vd using a constant L and H (L <= Vd <= H) independent of what Vq is seems to not take that into account. And aren't you reducing Vd and Vq so as to make their radius right on the voltage boundary when they need to be bigger to successfully command the current? So, if Vd and Vq are being reduced in magnitude later, shouldn't that be reflected by doing something to the integral?