Proportional-Integral-Derivative Control with Derivative Filtering and Integral Anti-Windup for a DC Servo

Kevin M. Passino and Nicanor Quijano

Dept. Electrical Engineering, The Ohio State University
2015 Neil Avenue, Columbus, OH 43210-1272

March 22, 2002

Abstract

First, you will design a proportional controller to meet a given set of specifications. Then, you will design a proportional integral derivative (PID) controller. Next the practical problems of noise and nonlinearities in PID control are considered. The ideas of derivative filtering and anti-windup schemes for PID control are introduced. You are asked to study the effects of derivative filtering and design an anti-windup scheme.

Contents

1 PID Controllers and Root Locus Design

1.1 Proportional Control Design

You will be working with

$$\frac{\theta(s)}{V_{in}(s)}$$

in this lab.

To develop a proportional controller use the following steps:

1. Let θ_d be the reference input to a unity feedback output shaft position control system for the DC servo and
$$e = \theta_d - \theta$$
Choose a proportional controller
$$V_{in} = K_p e$$

2. What is the "system type"? What would you expect to get for the steady state error e_{ss} with a unit step input? A ramp input (1 rad/sec) on θ_d? A parabolic input?

3. Using Matlab, plot the root locus. What value of K_p should you use to get $\zeta = 0.707$ so that you get an overshoot due to a unit step input (1 rad) of about 5%?

4. For the value of K_p that you found in the last step simulate the closed-loop for a unit step reference input and plot it and $\theta(t)$ on the same plot. Also, plot the control input. What is the value of e_{ss}? What is the value of the overshoot? What is the rise time and settling time (1% criterion)? Does this agree with the theory? Derive the theory and compare.

1.2 PID Control Design

Suppose that you use the PID controller

$$V_{in} = K_p e + K_d \dot{e} + K_i \int_0^t e(\tau) d\tau$$

so that

$$\frac{V_{in}(s)}{E(s)} = \frac{K_d s^2 + K_p s + K_i}{s}$$

1. Find K_p, K_d, and K_i so that the zeros of the PID controller are on the $\zeta = 0.707$ line with $\omega_n = 50$ rad/sec.

2. Find the value of K for a root locus (of the unity feedback control system) that puts two closed-loop poles within a ball of radius 5 of the zeros of the PID controller. What are the three closed-loop pole locations for this choice of K?

3. Find the closed-loop unit step response. Also, plot the control input. From the plots, what are the values of the overshoot, rise time, settling time (1% criterion), and steady state error? What is e_{ss} for a ramp and parabola input?

4. Re-design: Repeat the above process, but find K_p, K_d, and K_i to get an overshoot and rise time that is better than what you found in the last step, and better than what you found for the proportional controller above. There is a practical issue, however, that you must consider in solving this problem. In simulation you can achieve any specification that you want to achieve. Why? Increasingly demanding specifications result in unreasonably large control inputs and hence the use of too much control energy. Also, the control inputs will tend to move very fast and hence try to move the motor faster than is physically possible; the linear model breaks down and there is a disconnect between what the theory and implementation wil show. Here, you must also take take into consideration the use of control energy in the following way:

- For your proportional controller design did the value of the control input V_{in} go outside of the range of ± 5 V (what is physically possible for this DC servo)? If so, then reduce the proportional gain until it does not go outside that range for a unit step input to the closed-loop system. If not, then find the upper value of the proportional gain that results in the V_{in} value to just hit the limit of its valid range. Provide this value.
- Repeat this last step but for the PID controller and the root locus gain K.
- Now, return to the re-design problem. Find the best values of rise time and overshoot that you can obtain by tuning K_p, K_d, and K_i (and the root locus gain if you use that) but with the constraint that V_{in} does not go outside of the range of ± 5 V.

2 Effects of Noise and Nonlinearities on PID Control

For practical problems, the PID designs of the previous section often do not work properly since they ignore noise and nonlinear effects. Here, we discuss some commonly encountered problems in PID control in industry and provide solutions to these problems.

2.1 Derivative Filtering for PID Control

Two problems with implementing the derivative term $K_d s$ ($K_d \dot{e}$) in a PID controller are as follows:

1. The reference input $theta_d$ sometimes has sharp corners (e.g., when it is a square wave) and then $K_d \dot{e} = K_d(\dot{\theta}_d - \dot{\theta})$ will be large, since $\dot{\theta}_d$ is large, resulting in unreasonable size control inputs to the plant if you use a derivative term in the PID controller.
2. There is noise produced by any real sensor such as the encoder or potentiometer that produces the measurement of the output shaft position θ. This noise is typically high frequency noise which implies that it has high values of derivatives of that noise. Hence, the sensor noise will result in $K_d \dot{e} = K_d(\dot{\theta}_d - \dot{\theta})$ being large, since $\dot{\theta}$ is large, so that the plant input can be too large if a derivative term is used in the PID controller.

These two reasons often lead to complications in practical applications and hence in actual industrial control systems the "D term" (derivative term) is often "turned off" (i.e., $K_d = 0$). Other times, however, it is not turned off due to the benefits it can offer, especially its "predictive capability" (why is this term appropriate?) that typically allows you to overcome problems with overshoot in the closed-loop system. Why?

So, if you turn on the derivative term, how do you solve the above two problems? The typical industrial solution to the first problem is to change the form of the derivative term. How? Note that in many applications θ_d is constant for long periods of time so often $\dot{\theta}_d = 0$; hence it is reasonable to implement the derivative term as

$$-K_d \dot{\theta}$$

and hence to avoid having the derivative of θ_d enter into the V_{in} control input, yet still make it possible to give the controller a predictive capability.

The practical solution to the second problem above is to put a first order filter on the derivative term and tune its pole so that the chattering due to the noise does not occur since it attenuates high frequency noise (smooths it out) so that the derivative will not amplify the high frequency noise.

The resulting PID controller when we use both practical modifications is then

$$V_{in}(s) = K_p E(s) - \frac{\alpha K_d s}{s + \alpha} \theta(s) + \frac{K_i}{s} E(s)$$

where α is the location of the root of the "derivative filter" (note that we make the DC gain of the filter unity by putting α in the numerator also).

2.2 An Anti-Windup Scheme for PID Control

Another practical problem that arises in industrial PID control is "integral windup" that is due to the nonlinear effect of plant input saturation that occurs with any real plant. For our DC servo the input V_{in} can only lie in the range of ± 5 V so it is saturated at the extreme values when it is outside that range. In other words, the actual voltage that is input is *not* the commanded one, V_{in}, but is equal to

$$V_{in}^{actual} = \begin{cases} 5 & \text{if} \quad V_{in} \geq 5 \\ V_{in} & \text{if} \quad -5 \leq V_{in} \leq 5 \\ -5 & \text{if} \quad V_{in} \leq -5 \end{cases} \tag{1}$$

This provides a model of the saturation that occurs in the actual plant. If the input V_{in} hits an exteme value and stays there then the plant is essentially running "open loop." Why? During such a saturation condition the error e is integrated by the integral term and hence accumulates ("winds up"). This causes an undesirable effect where even though the control comes off the extreme value the integral term is still large which results in large swings in the control input and hence undesirable slow oscillatory behavior by the plant output. See the handout on PID control which is pp. 10–15 taken from:

> K.J. Astrom and T. Hagglund, Automatic Tuning of PID Controllers, Instrument Society of America, NC, 1988.

2.3 Derivative Filtering and Anti-Windup Scheme Design

Use your best K_p, K_d, and K_i values from above (i.e., your best design).

1. *Derivative Filtering:*Find a value of α such that there is very little performance degradation compared to the case when derivative filtering is not used (but find try to find the smallest such value that has essentially no effect). Show plots to illustrate your comparisons.

2. *Anti-Windup Scheme:* Add to your model a saturation as given in Equation (1) so that your new plant input is V_{in}^{actual} for a given commanded plant input of V_{in}. This is a nonlinear effect so now you need to simulate a nonlinear system (e.g., via Simulink).

   - *Illustrate Anti-Windup Effect:* Use a step input for θ_d with a magnitude of β rad. Find a value of β that will result in integrator windup. Show plots to illustrate the windup that are similar to the ones in the handout (i.e., as in Fig. 2.4 there).
   - *Integrator Windup Scheme Design:* Next, design an anti-windup scheme (like in Fig. 2.5B in the handout on PID control) that will mitigate the integral windup effect (as in Fig. 2.6 in the handout; see also Fig. 2.7). Note that $e_s = V_{in}^{actual} - V_{in}$ in the handout. Find a value of T_t (see the handout) which is the gain in the anti-windup scheme that will significantly reduce the integrator windup effect. Demonstrate the reduction in simulation.