PID Tuning Summary

Gary Lewis
Mentor, Kell Robotics, Team 1311

Introduction

This is a quick guide assembled from many sources. It assumes the reader has reviewed basic material on PID control, and knows some basic definitions.

Manual Tuning

One manual tuning method:

1. Set Ki and Kd values to zero
2. Increase Kp until oscillation
3. Set Kp to about half of that value
4. Increase Ki to correct any offset, if needed
5. Increase Kd to improve settling time, if needed

A fast PID system usually overshoots. If you cannot tolerate any overshoot, a lower value of Kp may be required.

Ki will be needed in cases where the system has an inherent bias that keeps it from reaching the setpoint. For example, an elevator system that needs power applied to the motor to just remain in position.

Results of changes in the parameters are shown below.

Parameter to Increase	Rise time	Overshoot	Settling time	Steady-state error	Stability
Kp	Decrease	Increase	Small Increase	Decrease	Degrade
Ki	Decrease	Increase	Increase	Large Decrease	Degrade
Kd	Small Decrease	Decrease	Decrease	Minor	Improve (if small Kd)

Ziegler–Nichols Method

Ziegler and Nichols described a practical tuning method that results in rapid recovery from disturbance, and results in some overshoot, so it is not applicable to all situations. A simple summary is below.

1. Set Ki and Kd values to zero
2. Increase Kp until you get a stable oscillation
3. That Kp is the critical gain Kc
4. Measure the period of the oscillation to get the critical period Tc
5. Set parameters as shown below

Control Type	Kp	Ki	Kd
P	0.50Kc		
PI	0.45Kc	1.2Kp/Tc	
PD	0.80Kc		KpTc/8
PID	0.60Kc	2Kp/Tc	KpTc/8

PID with the Talon SRX
Note that the Talon SRX effectively has its own convention for distance and time. There is no capability for setting distance per count from the encoder. You can set the encoder counts per revolution (codesPerRev) which allows reading revolutions, but we'll just stick with the raw counts for this discussion. The timing is not based on seconds, but on 100ms (0.1 second). That means that we measure:

Distance in counts
Time in tenths of seconds
Speed in counts per tenth of a second

Also, instead of a throttle value to the motor that is in the range of 0 to 1, the internal PID system on the Talon SRX uses 0-1023, with 1023 as full throttle. These conventions impact the calculations involving setting up the PID parameters.

Simple Procedure for Talon Position PID Tuning
An initial value for Kp can be determined from the desired throttle setting for a distance error. For:

error = distance error in counts
throttle = desired throttle setting for that error (as a percent or fraction)

Then:

Kp = (throttle*1023)/error

Let's say that you want 50% throttle when the device is 1000 counts from the desired location. Then you would have Kp = (0.5*1023)/1000 = 0.5115. When you have an input error of 1000, this will yield a value for (Kp*error) of 511.5, which is half throttle.

This is a good starting point; increase it (doubling works), until oscillation, and then back off.

- If acceleration is too rapid, add Kd, typically 10 to 100 times Kp.
- If the target is never reached, add Ki, starting with Kp/100.
- The previous general discussion of PID tuning can be used for refinement.

Simple Procedure for Talon Speed PID Tuning
For speed control, it is advisable to use the feed forward term (Kf). This provides an approximate throttle setting, based on the speed setting, and then the PID parameters can control about that point.

Set Kf based on max desired speed corresponding to max throttle, so that the throttle setting from feed forward is Kf*(Speed Setting). This means:

Kf = 1023/(max speed in counts/100ms)

So if your max speed will be 6000 counts per second, then the counts per 100ms will be 600 and

Kf = 1023/600 = 1.705

Then, if you request a speed of 300 counts/100ms, the resulting throttle (from Kf) will be:

Throttle = Kf*speed = 1.705*300 = 511.5, or half throttle.

Now we add in Kp (proportional gain) in a manner similar to the position control. We take a given error (in counts/100ms) and decide what change in throttle we want. Let's say the worst error we have seen is 200 counts/100ms and we want a 20% throttle change in response to that. Then we calculate:

Kp = (throttle*1023)/error = (0.2*1023)/200 = 1.023

If needed, use the previous methods to refine Kp, Ki, Kd.