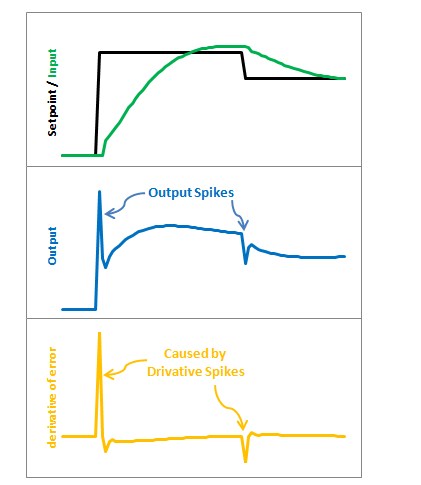
**Improving the Beginner’s PID – Derivative Kick**

(This is Modification #2 in a [larger series](http://brettbeauregard.com/blog/2011/04/improving-the-beginner%e2%80%99s-pid-derivative-kick/improving-the-beginners-pid-introduction) on writing a solid PID algorithm)

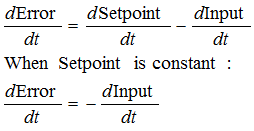
**The Problem**

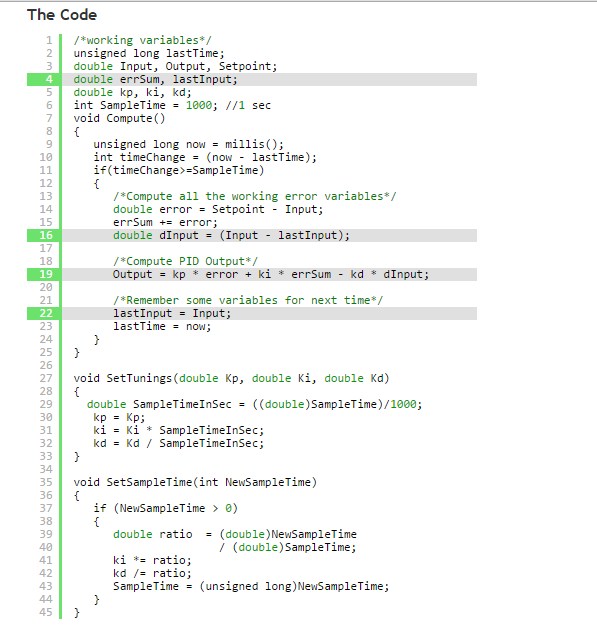
This modification is going to tweak the derivative term a bit. The goal is to eliminate a phenomenon known as “Derivative Kick”.



The image above illustrates the problem. Since error=Setpoint-Input, any change in Setpoint causes an instantaneous change in error. The derivative of this change is infinity (in practice, since dt isn’t 0 it just winds up being a really big number.) This number gets fed into the pid equation, which results in an undesirable spike in the output. Luckily there is an easy way to get rid of this.

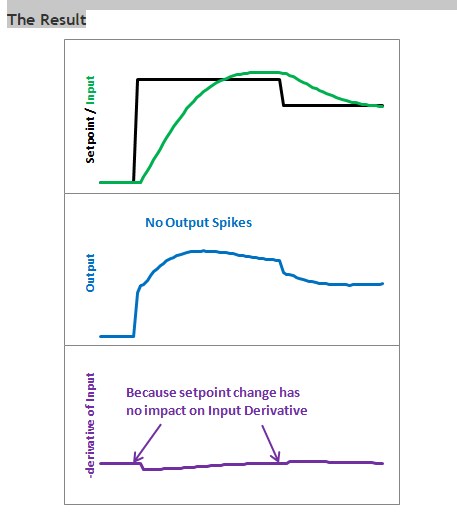
### The Solution

[](http://brettbeauregard.com/blog/wp-content/uploads/2011/03/DonMExplain.png)  
It turns out that the derivative of the Error is equal to negative derivative of Input, EXCEPT when the Setpoint is changing. This winds up being a perfect solution. Instead of adding (Kd \* derivative of Error), we subtract (Kd \* derivative of Input). This is known as using “Derivative on Measurement”



The modifications here are pretty easy. We’re replacing +dError with -dInput. Instead of remembering the lastError, we now remember the lastInput

### The Result



Here’s what those modifications get us. Notice that the input still looks about the same. So we get the same performance, but we don’t send out a huge Output spike every time the Setpoint changes.

This may or may not be a big deal. It all depends on how sensitive your application is to output spikes. The way I see it though, it doesn’t take any more work to do it without kicking so why not do things right?  
[Next >>](http://brettbeauregard.com/blog/2011/04/improving-the-beginner%e2%80%99s-pid-derivative-kick/improving-the-beginner%E2%80%99s-pid-tuning-changes)