


f (<https://www.facebook.com/Control-Solutions-Minnesota-Inc-272608333650812>) | **in**

(<https://www.linkedin.com/company/control-solutions-minnesota-inc>) | 

(<https://www.youtube.com/channel/UC4cfJLYA5laABCNiTG25Z3A>)



PID for Dummies

PID for Dummies

"I personally have a few hundred dollars worth of books on controllers, PID algorithms, and PID tuning. Since I am an engineer, I stand a chance of understanding some of it. But where do you go if you want to understand PID without a PhD? **Finn Peacock** (<http://www.pidtuning.net>) has written some very good material about PID which simplifies understanding. What follows here is an excerpt from his *Idiots Guide to PID Algorithms*, reprinted with his permission. I don't make a dime off this endorsement. I simply provide this introduction because I found Finn's material to be an excellent simplified discussion of PID." -- Jim Hogenson.



(RemoteMonitoring.html)

Did you ever wish you could monitor something simple without installing a whole expensive monitoring system?

Did you ever wish that one particular piece of equipment you're keeping an eye on could just send you a text message when something gets out of line?

Did you ever wish you had a simple way to just send a text to a piece of equipment and get a reply telling you how things were right then?

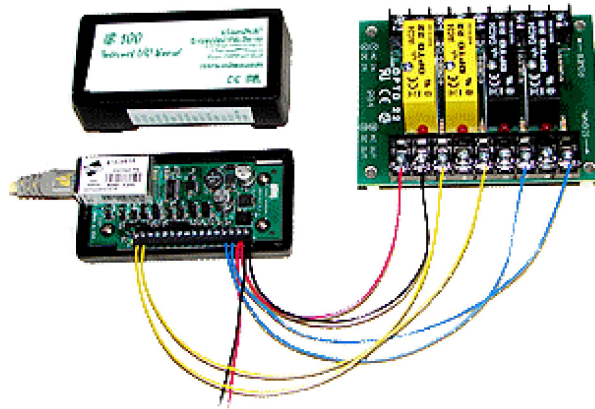
All of these are possible with the Control Solutions i.Report Text Messenger.

The i.Report Text Messenger is a self-contained remote monitoring and alarm notification system that monitors data points in one or more Modbus or BACnet devices and alerts you and others via SMS when an event has been detected that somebody should know about.

The i.Report Text Messenger offers simple yet powerful remote monitoring without the need for a monitoring service - no contract required.

Find out more about this exciting remote monitoring option and get yours **here**. (RemoteMonitoring.html)

IB-100 Internet I/O Board with Web Server and Modbus TCP Built In! Plus Script Basic Built In!



(Servers.html)

The IB-100 provides a simple and cost effective web appliance for facility management and remote monitoring. The IB-100 web server includes an impressive collection of pre-programmed web pages, but also allow the user to put a custom HTML "wrapper" around the embedded web site. The IB-100 is programmable using Script Basic!.

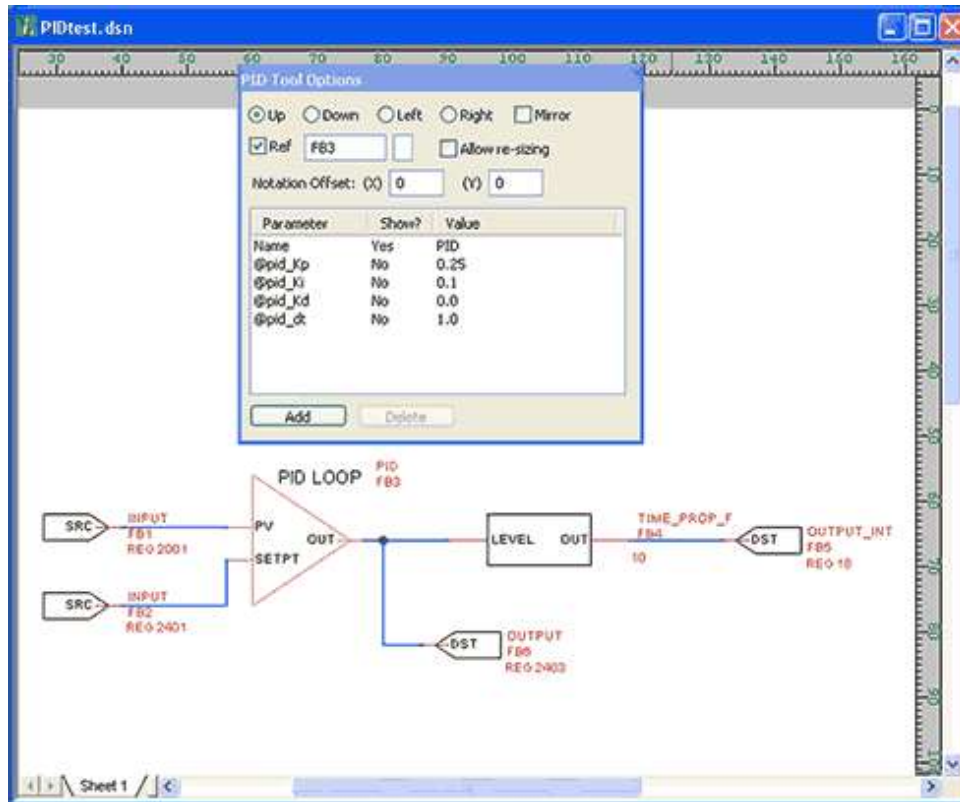
ValuPoint® Programmable I/O with Modbus Built In!



(ProgrammableIO.html)

Control Solutions' ValuPoint® VP4-2310 provides a mix of I/O points and a soft PLC with graphical programming that includes PID blocks. The VP4-2310 can operate as Modbus RTU master or slave.

ValuPoint® is Programmable Using i.CanDrawIt®

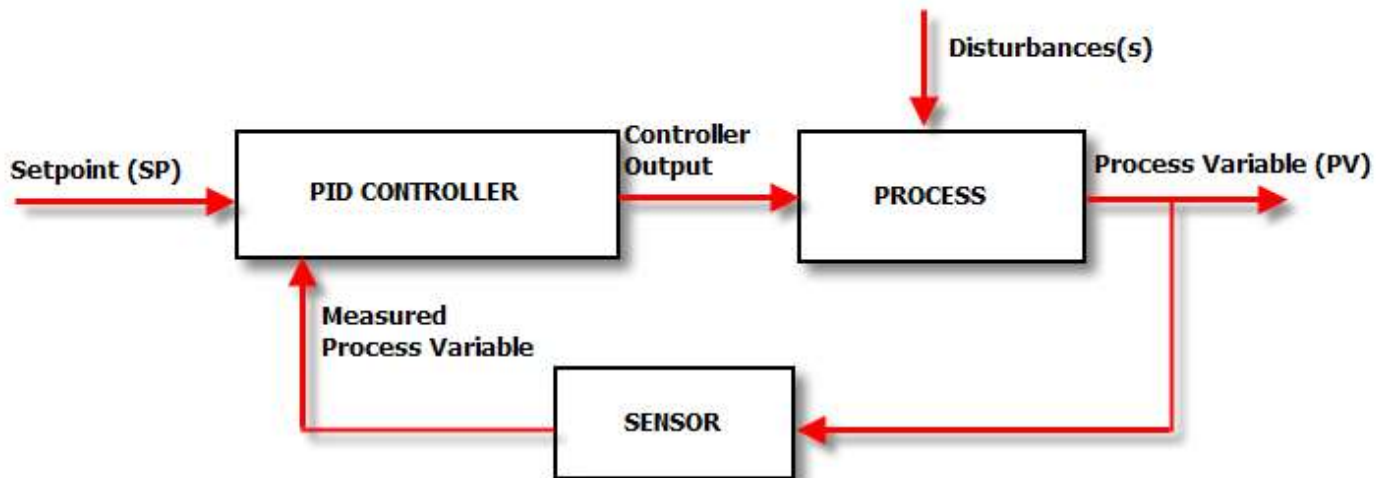


(ProgrammableIO.html)

Control Solutions' own graphical programming package includes PID function blocks ready to drop onto your drawing. Once you draw out the "schematic" of how your program should run, you are just two click away from executable code to load into your ValuPoint controller..

Anatomy Of A Feedback Control System

Here is the classic block diagram of a process under PID Control.



What's going on this diagram?

The **Setpoint (SP)** is the value that we **want** the process to be.

For example, the temperature control system in our house may have a SP of 22°C. This means that *“we want the heating and cooling process in our house to achieve a steady temperature of as close to 22°C as possible”*

The PID controller looks at the setpoint and compares it with the actual value of the **Process Variable (PV)**. Back in our house, the box of electronics that is the PID controller in our Heating and Cooling system looks at the value of the temperature sensor in the room and sees how close it is to 22°C.

If the SP and the PV are the same – then the controller is a very happy little box. It doesn't have to do anything, it will set its output to zero.

However, if there is a disparity between the SP and the PV we have an error and corrective action is needed. In our house this will either be cooling or heating depending on whether the PV is higher or lower than the SP respectively.

Let's imagine the temperature PV in our house is higher than the SP. It is too hot. The air-con is switched on and the temperature drops.

The sensor picks up the lower temperature, feeds that back to the controller, the controller sees that the “temperature error” is not as great because the PV (temperature) has dropped and the air con is turned down a little.

This process is repeated until the house has cooled down to 22°C and there is no error.

Then a disturbance hits the system and the controller has to kick in again.

In our house the disturbance may be the sun beating down on the roof, raising the temperature of the air inside.

So that's a really, really basic overview of a simple feedback control system. Sounds dead simple eh?

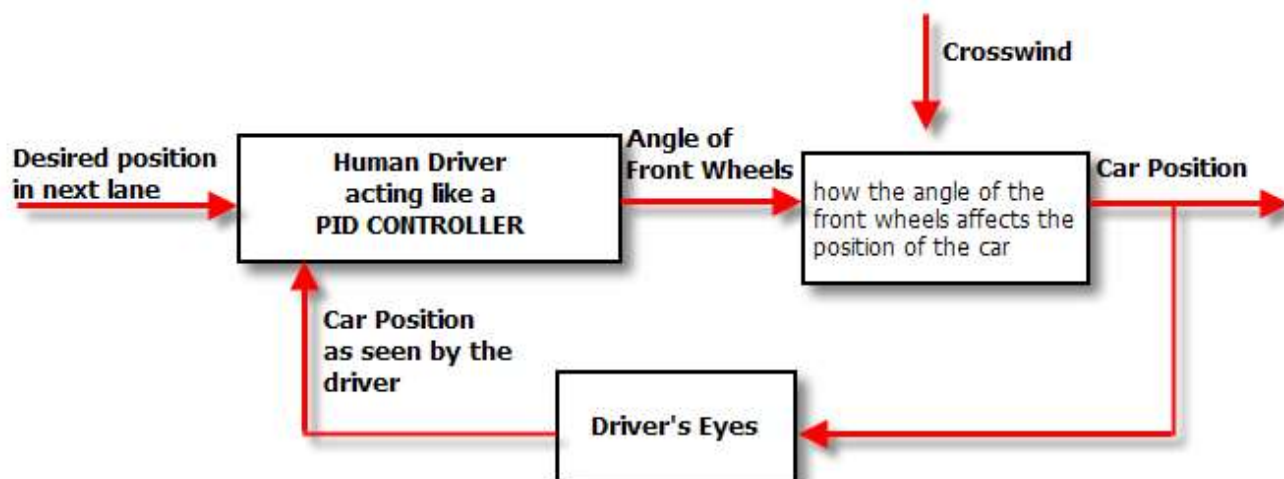
Understanding the controller

Unfortunately, in the real world we need a controller that is a bit more complicated than the one described above, if we want top performance from our loops. To understand why, we will be doing some "thought experiments" where we are the controller.

When we have gone through these thought experiments we will appreciate why a PID algorithm is needed and why/how it works to control the process.

We will be using the analogy of changing lanes on a freeway on a windy day. We are the driver, and therefore the controller of the process of changing the car's position.

Here's the Block Diagram we used before, with the labels changed to represent the car-on-windy-freeway control loop.



Notice how important closing the feedback loop is. If we removed the feedback loop we would be in "open loop control", and would have to control the car's position with our eyes closed!

Thankfully we are under "Closed loop control" -using our eyes for position feedback.

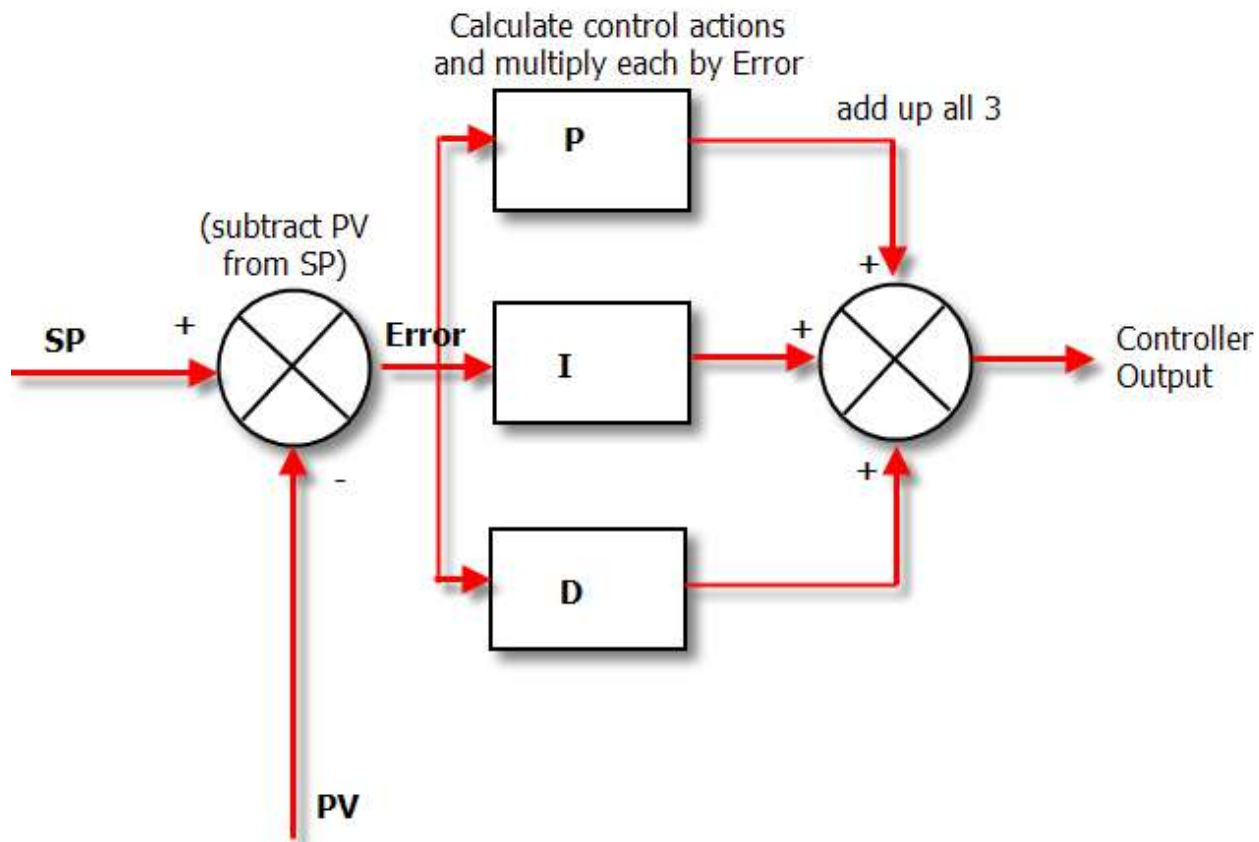
As we saw in the house-temperature example the controller takes the both the PV and SP signals, which it then puts through a black box to calculate a controller output. That controller output is sent to an actuator which moves to actually control the process.

We are interested here in what the black box actually does, which is that it applies 1, 2 or 3 calculations to the SP and Measured PV signals. These calculations, called the "Modes of Control" include:

- Proportional (P)
- Integral (I)
- Derivative (D)

Under The Hood Of The PID Controller

Here's a simplified block diagram of what the PID controller does:



It is really very simple in operation. The PV is subtracted from the SP to create the Error. The error is simply multiplied by one, two or all of the calculated P, I and D actions (depending which ones are turned on). Then the resulting "error x control actions" are added together and sent to the controller output.

These 3 modes are used in different combinations:

P – Sometimes used

PI - Most often used

PID – Sometimes used

PD – rare as hen's teeth but can be useful for controlling servomotors.

Derivatives

Go into the control room of a process plant and ask the operator:

"What's the derivative of reactor 4's pressure?"

And the response will typically be:

"Bugger off smart arse!"

However go in and ask:

"What's the rate of change of reactor 4's pressure?"

And the operator will examine the pressure trend and say something like:

"About 5 PSI every 10 minutes"

He's just performed calculus on the pressure trend! (don't tell him though or he'll want a pay raise)

So derivative is just a mathematical term meaning rate-of-change. That's all there is to it.

Integrals without the Math

Is it any wonder that so many people run scared from the concept of integrals and integration, when this is a typical definition?

Integral

From Wikipedia, the free encyclopedia

This article is about the concept of integrals in calculus. For the set of numbers, see [integer](#). For other uses, see [List of integrals](#).

Integration is a fundamental concept in [mathematics](#), specifically in the field of [calculus](#) and, more broadly, [mathematical analysis](#). Given a [function](#) f of a [real variable](#) x and an [interval](#) $[a, b]$ of the [real line](#), the **integral**

$$\int_a^b f(x) dx,$$

is defined informally to be the signed [area](#) of the region in the xy -plane bounded by the [graph](#) of f , the x -axis, and the vertical lines $x = a$ and $x = b$.

The term "integral" may also refer to the notion of [antiderivative](#), a function F whose [derivative](#) is the given function f . In this case it is called an **indefinite integral**, while the integrals discussed in this article are termed **definite integrals**. Some authors maintain a distinction between antiderivatives and indefinite integrals.

The principles of integration were formulated independently by [Isaac Newton](#) and [Gottfried Leibniz](#) in the late seventeenth century. Through the [fundamental theorem of calculus](#), which they independently developed, integration is connected with [differentiation](#): if f is a continuous real-valued function defined on a [closed interval](#) $[a, b]$, then, once an antiderivative F of f is known, the definite integral of f over that interval is given by

$$\int_a^b f(x) dx = F(b) - F(a).$$

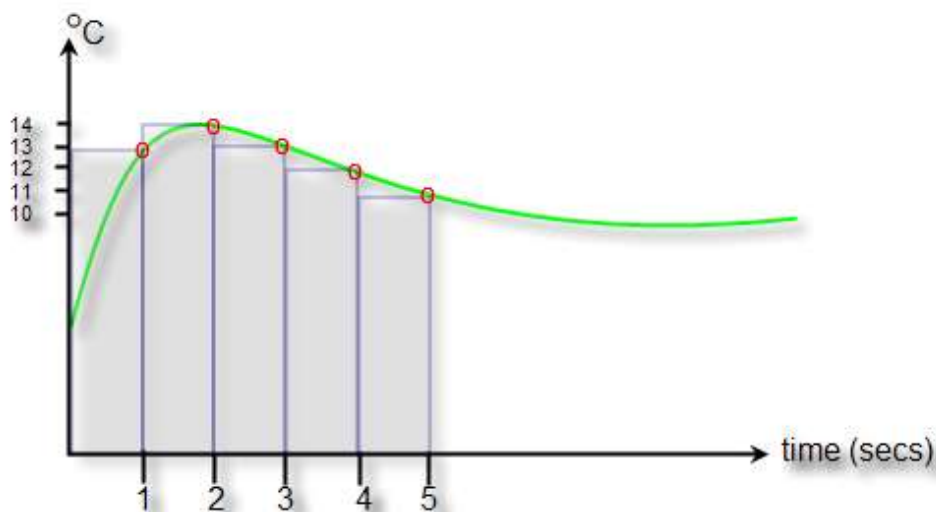
What the!?!?

If you understood that you are a smarter person than me.

Here's a plain English definition:

The integral of a signal is the sum of all the instantaneous values that the signal has been, from whenever you started counting until you stop counting.

So if you are to plot your signal on a trend and your signal is sampled every second, and let's say you are measuring temperature. If you were to superimpose the integral of the signal over the first 5 seconds – it would look like this:



The green line is your temperature, the red circles are where your control system has sampled the temperature and the blue area is the integral of the temperature signal. It is the sum of the 5 temperature values over the time period that you are interested in. In numerical terms it is the sum of the areas of each of the blue rectangles:

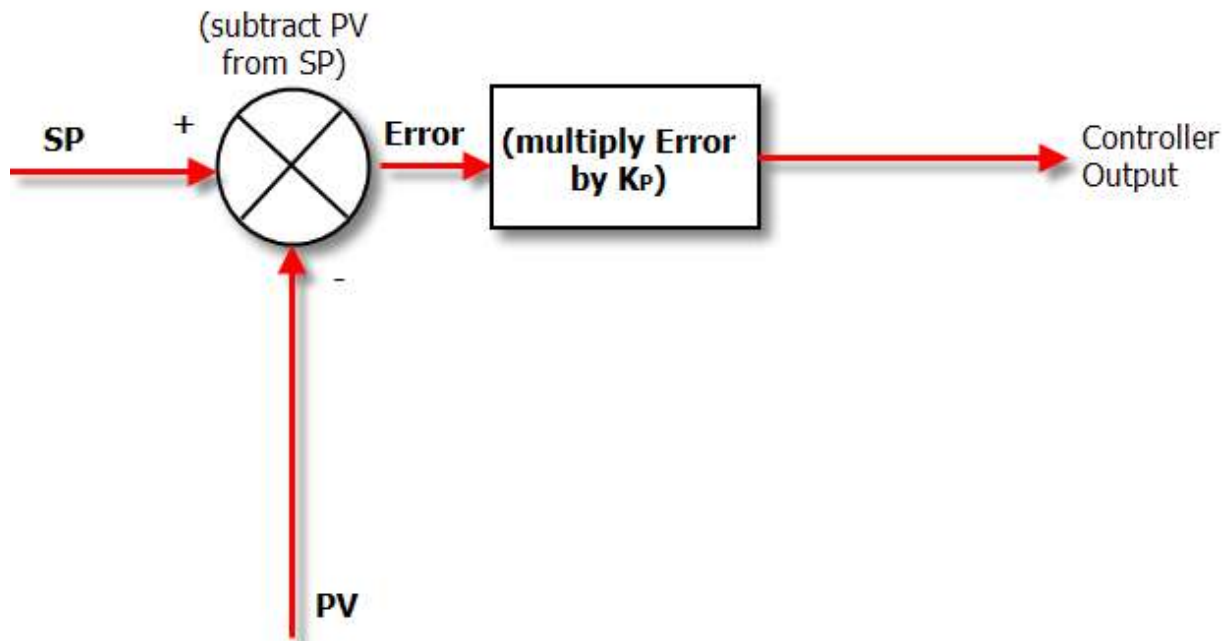
$$(13 \times 1) + (14 \times 1) + (13 \times 1) + (12 \times 1) + (11 \times 1) = 63 \text{ } ^\circ\text{C s}$$

The curious units (degrees Celsius x seconds) are because we have to multiply a temperature by a time – but the units aren't important.

As you can probably remember from school –the integral turns out to be the area under the curve. When we have real world systems, we actually get an approximation to the area under the curve, which as you can see from the diagram gets better, the faster we sample.

Proportional control

Here's a diagram of the controller when we have enabled only P control:

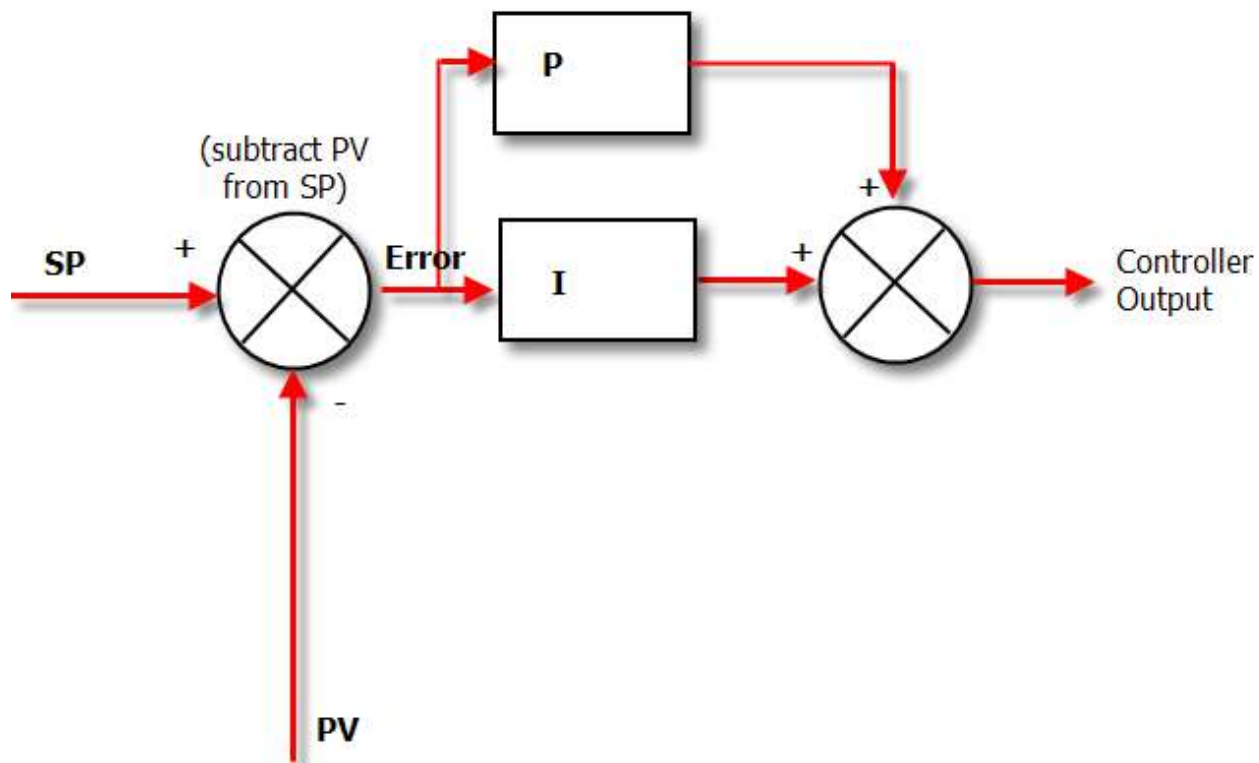


In Proportional Only mode, the controller simply multiplies the Error by the Proportional Gain (K_p) to get the controller output.

The Proportional Gain is the setting that we tune to get our desired performance from a “P only” controller.

A match made in heaven: The P + I Controller

If we put Proportional and Integral Action together, we get the humble PI controller. The Diagram below shows how the algorithm in a PI controller is calculated.



The tricky thing about Integral Action is that it will really screw up your process unless you know **exactly** how much Integral action to apply.

A good **PID Tuning technique** (<http://www.pidtuning.net>) will calculate exactly how much Integral to apply for your specific process - but how is the Integral Action adjusted in the first place?

Adjusting the Integral Action

The way to adjust how much Integral Action you have is by adjusting a term called “minutes per repeat”. Not a very intuitive name is it?

So where does this strange name come from? It is a measure of how long it will take for the Integral Action to match the Proportional Action.

In other words, if the output of the proportional box on the diagram above is 20%, the repeat time is the time it will take for the output of the Integral box to get to 20% too.

And the important point to note is that the “bigger” integral action, the quicker it will get this 20% value. That is, it will take fewer minutes to get there, so the “minutes per repeat” value will be smaller.

In other words the smaller the “minutes per repeat” is the bigger the integral action.

To make things a bit more intuitive, a lot of controllers use an alternative unit of “repeats per minute” which is obviously the inverse of “minutes per repeat”.

The nice thing about “repeats per minute” is that the bigger it is - the bigger the resulting Integral action is.

Derivative Action – predicting the future

OK, so the combination of P and I action seems to cover all the bases and do a pretty good job of controlling our system. That is the reason that PI controllers are the most prevalent. They do the job well enough and keep things simple. Great.

But engineers being engineers are always looking to tweak performance.

They do this in a PID loop by adding the final ingredient: Derivative Action.

So adding derivative action can allow you to have bigger P and I gains and still keep the loop stable, giving you a faster response and better loop performance.

If you think about it, Derivative action improves the controller action because it predicts what is yet to happen by projecting the current rate of change into the future. This means that it is not using the current measured value, but a future measured value.

The units used for derivative action describe how far into the future you want to look. i.e. If derivative action is 20 seconds, the derivative term will project the current rate of change 20 seconds into the future.

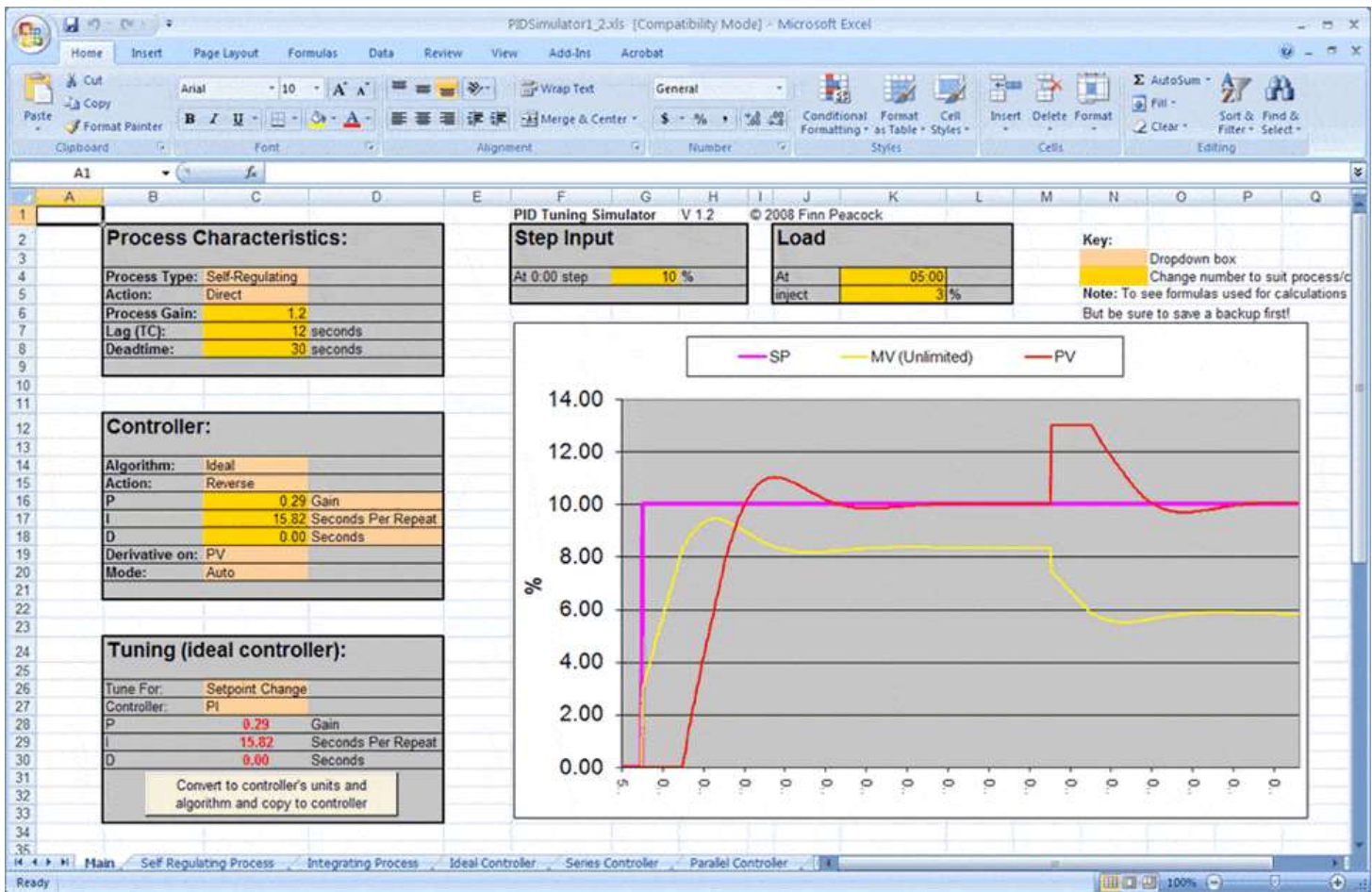
The big problem with D control is that if you have noise on your signal (which looks like a bunch of spikes with steep sides) this confuses the hell out of the algorithm. It looks at the slope of the noise-spike and thinks:

“Holy crap! This process is changing quickly, lets pile on the D Action!!!”

And your control output jumps all over the place, messing up your control.

Of course you can try and filter the noise out, but my advice is that, unless PI control is really slow, don't worry about switching D on.

Another note from Jim: "Whether learning about PID and how the parameters affect performance, or trying to tune a process, simulation is an important tool for getting PID right. Finn Peacock has a simulation tool available that runs in Microsoft Excel. Over the years, I have written a lot of different simulation programs while developing PID algorithms for everything from industrial process controls to scientific research apparatus on the NASA Space Shuttle. If this spread sheet had been available then, it would have saved me a lot of time. You can get a copy of this simulation tool (screen shot below) as part of the Pro package available from Finn Peacock at **www.pidtuning.net** (<http://www.pidtuning.net>). Again, I don't make a dime off this. I just found it valuable and wanted to pass it along."



About our Company

Control Solutions, Inc., a Minnesota corporation founded in 1995, offers a line of control products tailored to facility management, building automation, telecommunications, and remote monitoring & control. Control Solutions has thrived on opportunities to expand, enhance, and support mainstream products and systems. As a result, you can find a little bit of our equipment connected to a lot of other people's equipment, in a lot of different places, doing a lot of different things.

Helpful Links

- 🔗 Home Page (../index.html)
- 🔗 Shop Online (<https://buy.csimn.com>)
- 🔗 Support Ticket (<https://ticket.csimn.com>)

- 🔗 Knowledgebase (<https://info.csimn.com>)
- 🔗 Quotes ([Quotes.html](#))
- 🔗 Contact Us ([ContactUs.html](#))
- 🔗 About Us ([AboutUs.html](#))
- 🔗 Terms & Conditions ([terms.html](#))
- 🔗 Privacy Policy ([privacy.html](#))
- 🔗 Legal ([legal.html](#))

Contact Info

Mailing Address:

PO Box 10789
St. Paul, MN 55110-0789

NEW Shipping Address:

10550 South Avenue
Chisago City, MN 55013

Phone Number:

651-426-4410
800-872-8613

Fax Number:

651-426-4418

E-mail:

sales@csimn.com

Copyright 2020 Control Solutions Minnesota, Inc. All Rights Reserved.