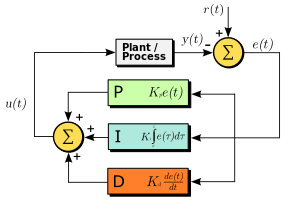
# PID Control

## Introduction to PID Controllers

An excellent introduction is provided on <https://en.wikipedia.org/wiki/PID_controller>. Here it is repeated for your convenience:

[](https://en.wikipedia.org/wiki/File:PID_en_updated_feedback.svg)

A [block diagram](https://en.wikipedia.org/wiki/Block_diagram) of a PID controller in a feedback loop

A **proportional–integral–derivative controller** (**PID controller**) is a [control loop](https://en.wikipedia.org/wiki/Control_loop) [feedback mechanism](https://en.wikipedia.org/wiki/Feedback_mechanism) ([controller](https://en.wikipedia.org/wiki/Controller_(control_theory))) commonly used in [industrial control systems](https://en.wikipedia.org/wiki/Industrial_control_system). A PID controller continuously calculates an *error value* as the difference between a desired [setpoint](https://en.wikipedia.org/wiki/Setpoint_(control_system)" \o "Setpoint (control system)) and a measured [process variable](https://en.wikipedia.org/wiki/Process_variable). The controller attempts to minimize the error over time by adjustment of a *control variable*, such as the position of a [control valve](https://en.wikipedia.org/wiki/Control_valve), a [damper](https://en.wikipedia.org/wiki/Damper_(flow)), or the power supplied to a heating element, to a new value determined by a weighted sum:

u(t) = K_p e(t) + K_i \int_{0}^{t}e(\tau)d\tau + K_d \frac{de(t)}{dt}

where K_p, K_i, and K_d, all non-negative, denote the coefficients for the [proportional](https://en.wikipedia.org/wiki/Proportional_control), [integral](https://en.wikipedia.org/wiki/Integral), and [derivative](https://en.wikipedia.org/wiki/Derivative) terms, respectively (sometimes denoted *P,* *I,* and *D*). In this model,

* *P* accounts for present values of the error (e.g. if the error is large and positive, the control variable will be large and negative),
* *I* accounts for past values of the error (e.g. if the output is not sufficient to reduce the size of the error, the control variable will accumulate over time, causing the controller to apply a stronger action), and
* *D* accounts for possible future values of the error, based on its current rate of change.[[1]](https://en.wikipedia.org/wiki/PID_controller#cite_note-1)

As a PID controller relies only on the measured process variable, not on knowledge of the underlying process, it is broadly applicable.[[2]](https://en.wikipedia.org/wiki/PID_controller#cite_note-ben93p48-2) By tuning the three parameters of the model, a PID controller can deal with specific process requirements. The response of the controller can be described in terms of its responsiveness to an error, the degree to which the system [overshoots](https://en.wikipedia.org/wiki/Overshoot_(signal)) a setpoint, and the degree of any system oscillation. The use of the PID algorithm does not guarantee [optimal control](https://en.wikipedia.org/wiki/Optimal_control) of the system or even its [stability](https://en.wikipedia.org/wiki/Nyquist_stability_criterion).

Some applications may require using only one or two terms to provide the appropriate system control. This is achieved by setting the other parameters to zero. A PID controller will be called a PI, PD, P or I controller in the absence of the respective control actions. PI controllers are fairly common, since derivative action is sensitive to measurement noise, whereas the absence of an integral term may prevent the system from reaching its target value.

For [discrete time](https://en.wikipedia.org/wiki/Discrete_time) systems, the term PSD, for **proportional-summation-difference**, is often used.[[3]](https://en.wikipedia.org/wiki/PID_controller#cite_note-3)

# PID Control Example: Cruise Control

Objective is to control the speed of a car to a given set-point, say 120 km/hr

The input is the gas injection rate, which contributes to the acceleration.

The car's acceleration is determined by its mass and the net force on it



where  is due to air resistance,  is the car's weight in the backward direction ( is the angle of incline the car is driving on; while going up-hill [] the car's weight acts to slow it down, while it does the opposite while going downhill), and  is a frictional loss term which increases linearly with the car's forward velocity.

How does the car's velocity depend on the input (motor force)?



So the velocity we're actually at depends in a complicated way on the history of the velocity as well as the motor force, wind speed, and incline.

Let's program this to explore it. Suppose for simplicity that:



Now make a MATLAB program to simulate how the velocity changes for certain fixed values of Fmotor. Use 1000 timesteps between 0 and 50 s, and assume that there is a 2 second control delay (due to a combination of the accelerator not producing car force instantly, the sensors taking time to read, etc.).

(see the file PIDDemo.m for a solution)

Procedure for tuning the control loop (from page 1074 of The Art of Electronics Third Edition):

1. Start with the I & D terms at 0, and Increase the P term until oscillations start, then back off a bit. The system should now show overshoot & ringing, but not sustained oscillations from a change in setpoint.
2. Increase the *D* term until the response is critically damped
3. Increase the I term until you've achieved the minimum settling time.

This procedure will give "pretty good" results, but isn't the optimum tuning.

Note: If your controller measures everything that affects the system's control variable (i.e., the present and future wind speed, friction, delay, road slope, etc.) then you can write a formula for the optimum input and do much better than PID control. The key benefit of PID control is its robustness: it works reasonably well even if you *don't* measure most of the things that are affecting the system's output. The only thing the PID controller needs to measure to control the velocity is the velocity itself.

Note 2: If there's no delay time, then pure proportional control is sufficient. PID especially shines when there's delay between input and output.

Note 3: See what happens when you turn up the gains very high and then change the system. Gains too high lead to instability, while those too low give poor performance. Even though PID control is robust, proper tuning is essential for good performance.

Note 4: Since we're controlling the velocity by setting the acceleration, we have 

So compared to controlling the acceleration by setting the acceleration, every term acts somewhat like its integral. For this reason you may consider adding in a second derivative control term:



If instead we were controlling the position, we'd have:



As such, you may consider adding in two additional derivative terms. Note that these derivative terms after integration don't behave the same as proportional or integral terms because they can't carry information about the setpoint through the derivative. Nevertheless, additional derivative terms are helpful in stabilizing higher order systems. This is presented for completion only - for this course PID control will be fine.