PID CONTROLLER – SIMPLY EXPLAINED

What is PID?

PID (Proportional Integral Differential) is an algorithm. Think of PID as a simple spring. A spring has an original length, which when disturbed, by expansion or contraction, tends to regain its original length in the shortest possible time. Similarly, a PID algorithm in a system has a set-value of a particular physical quantity to be controlled, called a 'set point', which when altered due to some reason, the system controls the other necessary features in it, to get back to the original set point in the shortest time possible. PID controllers are used wherever there is a need to control a physical quantity and to make it equal to a specified value. Example, Cruise controller in cars, Robots, Temperature regulators, Voltage regulators, etc.

How does PID work?

The system calculates the 'error', or 'deviation' of the physical quantity from the set point, by measuring the current value of that physical quantity using a sensor(s). To get back to the set point, this 'error' should be minimized, and should ideally be made equal to zero. Also, this process should happen as quickly as possible. Ideally, there should be zero lag in the response of the system to the change in its set point.

Why implement PID?

Consider the cruise control in a car. It is a feature in all the cars these days in which, when a button is pressed, the speed of the car is immediately changed, to equal the set speed of the cruise controller. Why use a PID algorithm for this case? Is it not possible to just have an if-else condition which increases the speed by 10 kmph if the current speed is less than the set speed, or decrease the speed by 10 kmph if the current speed is more than the set speed?

Think of it.

Say the set speed is 50 kmph, and the current speed is 47 kmph. When the button is pressed, the if-else condition decides that the current speed is less than the set speed, so it has to be incremented by 10 kmph, whereas ideally, it has to be incremented only by 3 kmph. So it pumps the gas, accelerates and increments it by 10 kmph, only for it to realize the next instant, that it's current speed is 57 kmph, which is more than the set point, so it has to decrease the speed by 10 kmph, whereas ideally it has to be decreased by only 7 kmph.

This method is okay when the deviation and the increment/decrement value of the physical quantity from the set point is low, but as the deviation becomes greater, using this method, clearly damages the system. The problem is that, the amount by which the speed should be controlled, is constant and does not depend on the deviation of the speed from the set point.

Implementing PID

i) Error Term (e):

This is equal to the difference between the set point and the current value of the quantity being controlled.

e = set_point – current_value

ii) Proportional Term (P):

This term is proportional to the error.

P = e

This value is responsible for the magnitude of change required in the physical quantity to achieve the set point.

iii) Integral Term (I):

This term is the sum of all the previous error values.

I = I + previous_I_value

This value is responsible for the quickness of response of the system to the change from the set point.

iv) Differential Term (D):

This term is the difference between the instantaneous error from the set point, and the error from the previous instant.

D = e – previous_e_value

This value is responsible to slow down the rate of change of the physical quantity when it comes close to the set point.

Equation:

PID_value = (Kp*P) + (Ki*I) + (Kd*D)

Where:

Kp is the constant, which used to vary the magnitude of the change required to achieve the set point.

Ki is the the constant, which is used to vary the rate at which the change should be brought in the physical quantity to achieve the set point.

Kd is the constant, which is used to vary the stability of the system.

Start with a small value for **Kp** and **Kd** and let **Ki** be equal to zero. If the error values are high, increase the **Kp** value. If the system is not fast enough in responding to a change, increase the **Ki** value by small amounts. If the system oscillates about the set point, increase the **Kd** value by considerable amounts, depending on the extent of oscillation.

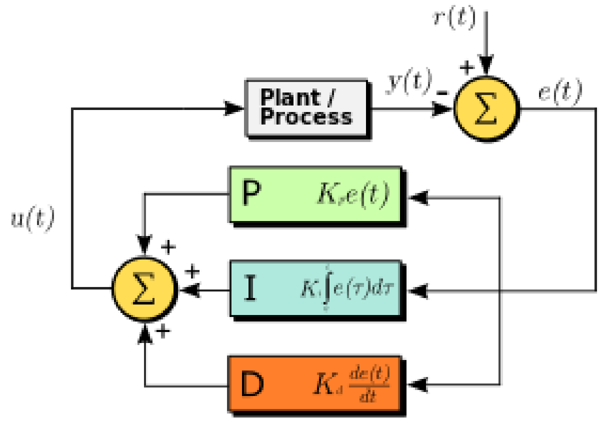
The **PID_value** is directly fed into the required part, which controls that particular physical quantity, for example, the accelerator in the case of a cruise control.

Lastly, PID doesn’t guarantee effective results just by simple implementation of a code, it requires constant tweaking based on the circumstances, once correctly tweaked it yields exceptional results. The PID implementation also involves a settling time, hence effective results can be seen only after a certain time from the start of the run of the robot. Also to obtain a fairly accurate output it is not always necessary to implement all the three expressions of PID. If implementing just PI results yields a good result we can skip the derivative part.

Read about how to implement PID on a basic Line Follower robot.

For more information about PID Controllers and PID algorithms, follow these links:

- Wikipedia
- Control Tutorials
- CalTech



PID Controller Block Diagram

5 THOUGHTS ON "PID CONTROLLER – SIMPLY EXPLAINED"