PID controller

A **proportional–integral–derivative controller** (**PID controller** or **three-term controller**) is a control loop mechanism employing feedback that is widely used in industrial control systems and a variety of other applications requiring continuously modulated control. A PID controller continuously calculates an *error value* $e(t)$ as the difference between a desired setpoint (SP) and a measured process variable (PV) and applies a correction based on proportional, integral, and derivative terms (denoted *P*, *I*, and *D* respectively), hence the name.

In practical terms it automatically applies an accurate and responsive correction to a control function. An everyday example is the cruise control on a car, where ascending a hill would lower speed if only constant engine power were applied. The controller's PID algorithm restores the measured speed to the desired speed with minimal delay and overshoot by increasing the power output of the engine.

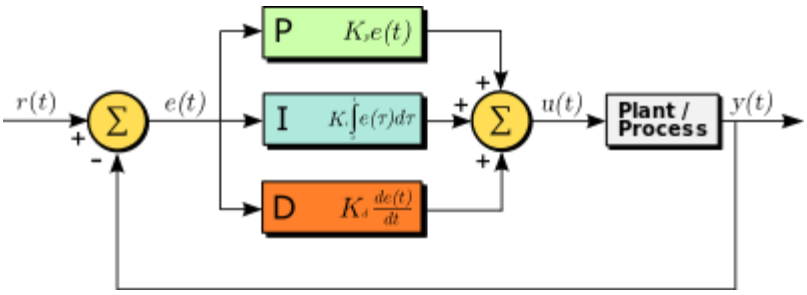
The first theoretical analysis and practical application was in the field of automatic steering systems for ships, developed from the early 1920s onwards. It was then used for automatic process control in the manufacturing industry, where it was widely implemented in pneumatic, and then electronic, controllers. Today the PID concept is used universally in applications requiring accurate and optimized automatic control.

Contents

Fundamental operation

The distinguishing feature of the PID controller is the ability to use the three *control terms* of proportional, integral and derivative influence on the controller output to apply accurate and optimal control. The block diagram on the right shows the principles of how these terms are generated and applied. It shows a PID

controller, which continuously calculates an *error value* $e(t)$ as the difference between a desired setpoint $\mathrm{SP} = r(t)$ and a measured process variable $\mathrm{PV} = y(t)$: $e(t) = r(t) - y(t)$, and applies a correction based on proportional, integral, and derivative terms. The controller attempts to minimize the error over time by adjustment of a *control variable* $u(t)$, such as the opening of a control valve, to a new value determined by a weighted sum of the control terms.



A block diagram of a PID controller in a feedback loop. *r*(*t*) is the desired process value or setpoint (SP), and *y*(*t*) is the measured process value (PV).

In this model:

- Term **P** is proportional to the current value of the SP − PV error $e(t)$. For example, if the error is large and positive, the control output will be proportionately large and positive, taking into account the gain factor "K". Using proportional control alone will result in an error between the setpoint and the actual process value because it requires an error to generate the proportional response. If there is no error, there is no corrective response.
- Term **I** accounts for past values of the SP − PV error and integrates them over time to produce the **I** term. For example, if there is a residual SP − PV error after the application of proportional control, the integral term seeks to eliminate the residual error by adding a control effect due to the historic cumulative value of the error. When the error is eliminated, the integral term will cease to grow. This will result in the proportional effect diminishing as the error decreases, but this is compensated for by the growing integral effect.
- Term **D** is a best estimate of the future trend of the SP − PV error, based on its current rate of change. It is sometimes called "anticipatory control", as it is effectively seeking to reduce the effect of the SP − PV error by exerting a control influence generated by the rate of error change. The more rapid the change, the greater the controlling or damping effect.[1]

Tuning – The balance of these effects is achieved by loop tuning to produce the optimal control function. The tuning constants are shown below as "K" and must be derived for each control application, as they depend on the response characteristics of the complete loop external to the controller. These are dependent on the behavior of the measuring sensor, the final control element (such as a control valve), any control signal delays and the process itself. Approximate values of constants can usually be initially entered knowing the type of application, but they are normally refined, or tuned, by "bumping" the process in practice by introducing a setpoint change and observing the system response.

Control action – The mathematical model and practical loop above both use a *direct* control action for all the terms, which means an increasing positive error results in an increasing positive control output correction. The system is called *reverse* acting if it is necessary to apply negative corrective action. For instance, if the valve in the flow loop was 100–0% valve opening for 0–100% control output – meaning that the controller action has to be reversed. Some process control schemes and final control elements require this reverse action. An example would be a valve for cooling water, where the fail-safe mode, in the case of loss of signal, would be 100% opening of the valve; therefore 0% controller output needs to cause 100% valve opening.

Mathematical form

The overall control function $u(t) = K_\mathrm{p} e(t) + K_\mathrm{i} \int_0^t e(t')\,dt' + K_\mathrm{d} \frac{de(t)}{dt}$,

where K_p, K_i, and K_d, all non-negative, denote the coefficients for the proportional, integral, and derivative terms respectively (sometimes denoted *P*, *I*, and *D*).

In the *standard form* of the equation (see later in article), K_i and K_d are respectively replaced by $K_\mathrm{p}/T_\mathrm{i}$ and $K_\mathrm{p}T_\mathrm{d}$; the advantage of this being that T_i and T_d have some understandable physical meaning, as they represent the integration time and the derivative time respectively.

$$u(t) = K_\mathrm{p}\left(e(t) + \frac{1}{T_\mathrm{i}}\int_0^t e(t')\,dt' + T_\mathrm{d}\frac{de(t)}{dt}\right),$$

Selective use of control terms

Although a PID controller has three control terms, some applications need only one or two terms to provide appropriate control. This is achieved by setting the unused parameters to zero and is called a PI, PD, P or I controller in the absence of the other control actions. PI controllers are fairly common in applications where derivative action would be sensitive to measurement noise, but the integral term is often needed for the system to reach its target value.

Applicability

The use of the PID algorithm does not guarantee optimal control of the system or its control stability . Situations may occur where there are excessive delays: the measurement of the process value is delayed, or the control action does not apply quickly enough. In these cases lead–lag compensation is required to be effective. The response of the controller can be described in terms of its responsiveness to an error, the degree to which the system overshoots a setpoint, and the degree of any system oscillation. But the PID controller is broadly applicable since it relies only on the response of the measured process variable, not on knowledge or a model of the underlying process.

History

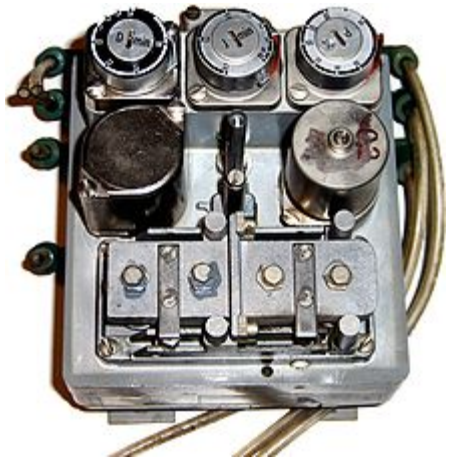
Origins

Continuous control, before PID controllers were fully understood and implemented, has one of its origins in the centrifugal governor, which uses rotating weights to control a process. This had been invented by Christiaan Huygens in the 17th century to regulate the gap between millstones in windmills depending on the speed of rotation, and thereby compensate for the variable speed of grain feed.[2][3]

Early PID theory was developed by observing the actions of helmsmen in keeping a vessel on course in the face of varying influences such as wind and sea state.

With the invention of the low-pressure stationary steam engine there was a need for automatic speed control, and James Watt's self-designed "conical pendulum" governor, a set of revolving steel balls attached to a vertical spindle by link arms, came to be an industry standard. This was based on the millstone-gap control concept.[4]

Rotating-governor speed control, however, was still variable under conditions of varying load, where the shortcoming of what is now known as proportional control alone was evident. The error between the desired speed and the actual speed would increase with increasing load. In the 19th century, the theoretical basis for the operation of governors was first described by James Clerk Maxwell in 1868 in his now-famous paper *On Governors*. He explored the mathematical basis for control stability, and progressed a good way towards a solution, but made an appeal for mathematicians to examine the problem.[5][4] The problem was examined further in 1874 by Edward Routh, Charles Sturm, and in 1895, Adolf Hurwitz, all of whom contributed to the establishment of control stability criteria.[4] In subsequent applications, speed governors were further refined, notably by American scientist Willard Gibbs, who in 1872 theoretically analyzed Watt's conical pendulum governor.



Pneumatic PID (three-term) controller. The magnitudes of the three terms (P, I and D) are adjusted by the dials at the top.

About this time, the invention of the Whitehead torpedo posed a control problem that required accurate control of the running depth. Use of a depth pressure sensor alone proved inadequate, and a pendulum that measured the fore and aft pitch of the torpedo was combined with depth measurement to become the pendulum-and-hydrostat control. Pressure control provided only a proportional control that, if the control gain was too high, would become unstable and go into overshoot with considerable instability of depth-holding. The pendulum added what is now known as derivative control, which damped the oscillations by detecting the torpedo dive/climb angle and thereby the rate-of-change of depth.[6] This development (named by Whitehead as "The Secret" to give no clue to its action) was around 1868.[7]

Another early example of a PID-type controller was developed by Elmer Sperry in 1911 for ship steering, though his work was intuitive rather than mathematically-based.[8]

It was not until 1922, however, that a formal control law for what we now call PID or three-term control was first developed using theoretical analysis, by Russian American engineer Nicolas Minorsky.[9] Minorsky was researching and designing automatic ship steering for the US Navy and based his analysis on observations of a helmsman. He noted the helmsman steered the ship based not only on the current course error but also on past error, as well as the current rate of change;[10] this was then given a mathematical treatment by Minorsky.[4] His goal was stability, not general control, which simplified the problem significantly. While proportional control provided stability against small disturbances, it was insufficient for dealing with a steady disturbance, notably a stiff gale (due to steady-state error), which required adding the integral term. Finally, the derivative term was added to improve stability and control.

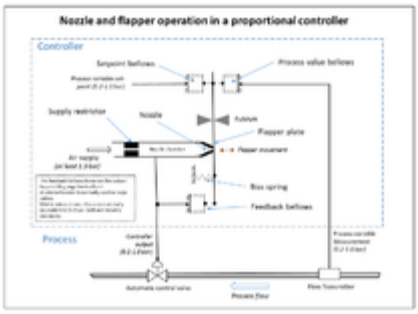
Trials were carried out on the USS *New Mexico*, with the controllers controlling the *angular velocity* (not the angle) of the rudder. PI control yielded sustained yaw (angular error) of ±2°. Adding the D element yielded a yaw error of ±1/6°, better than most helmsmen could achieve.[11]

The Navy ultimately did not adopt the system due to resistance by personnel. Similar work was carried out and published by several others in the 1930s.

Industrial control

The wide use of feedback controllers did not become feasible until the development of wideband high-gain amplifiers to use the concept of negative feedback. This had been developed in telephone engineering electronics by Harold Black in the late 1920s, but not published until 1934.[4] Independently, Clesson E Mason of the Foxboro Company in 1930 invented a wide-band pneumatic controller by combining the nozzle

and flapper high-gain pneumatic amplifier, which had been invented in 1914, with negative feedback from the controller output. This dramatically increased the linear range of operation of the nozzle and flapper amplifier, and integral control could also be added by the use of a precision bleed valve and a bellows generating the integral term. The result was the "Stabilog" controller which gave both proportional and integral functions using feedback bellows.[4] The integral term was called *Reset*.[12] Later the derivative term was added by a further bellows and adjustable orifice.



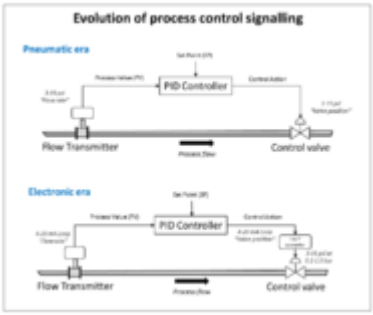
Proportional control using nozzle and flapper high gain amplifier and negative feedback

From about 1932 onwards, the use of wideband pneumatic controllers increased rapidly in a variety of control applications. Air pressure was used for generating the controller output, and also for powering process modulating devices such as diaphragm-operated control valves. They were simple low maintenance devices that operated well in harsh industrial environments and did not present explosion risks in hazardous locations. They were the industry standard for many decades until the advent of discrete electronic controllers and distributed control systems.

With these controllers, a pneumatic industry signaling standard of 3–15 psi (0.2–1.0 bar) was established, which had an elevated zero to ensure devices were working within their linear characteristic and represented the control range of 0-100%.

In the 1950s, when high gain electronic amplifiers became cheap and reliable, electronic PID controllers became popular, and the pneumatic standard was emulated by 10-50 mA and 4–20 mA current loop signals (the latter became the industry standard). Pneumatic field actuators are still widely used because of the advantages of pneumatic energy for control valves in process plant environments.

Most modern PID controls in industry are implemented as computer software in distributed control systems (DCS), programmable logic controllers (PLCs), or discrete compact controllers.



Showing the evolution of analog control loop signaling from the pneumatic to the electronic eras

Electronic analog controllers

Electronic analog PID control loops were often found within more complex electronic systems, for example, the head positioning of a disk drive, the power conditioning of a power supply, or even the movement-detection circuit of a modern seismometer. Discrete electronic analog controllers have been largely replaced by digital controllers using microcontrollers or FPGAs to implement PID algorithms. However, discrete analog PID controllers are still used in niche applications requiring high-bandwidth and low-noise performance, such as laser-diode controllers.[13]

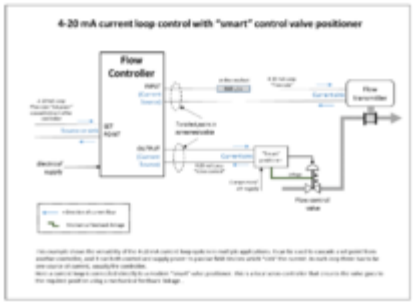
Control loop example

Consider a robotic arm[14] that can be moved and positioned by a control loop. An electric motor may lift or lower the arm, depending on forward or reverse power applied, but power cannot be a simple function of position because of the inertial mass of the arm, forces due to gravity, external forces on the arm such as a load to lift or work to be done on an external object.

- The sensed position is the process variable (PV).

- The desired position is called the setpoint (SP).
- The difference between the PV and SP is the error (e), which quantifies whether the arm is too low or too high and by how much.
- The input to the process (the electric current in the motor) is the output from the PID controller. It is called either the manipulated variable (MV) or the control variable (CV).

By measuring the position (PV), and subtracting it from the setpoint (SP), the error (e) is found, and from it the controller calculates how much electric current to supply to the motor (MV).



Current loops used for sensing and control signals. A modern electronic "smart" valve positioner is shown, which will incorporate its own PID controller.

Proportional

The obvious method is **proportional** control: the motor current is set in proportion to the existing error. However, this method fails if, for instance, the arm has to lift different weights: a greater weight needs a greater force applied for the same error on the down side, but a smaller force if the error is on the upside. That's where the integral and derivative terms play their part.

Integral

An **integral** term increases action in relation not only to the error but also the time for which it has persisted. So, if the applied force is not enough to bring the error to zero, this force will be increased as time passes. A pure "I" controller could bring the error to zero, but it would be both slow reacting at the start (because the action would be small at the beginning, needing time to get significant) and brutal (the action increases as long as the error is positive, even if the error has started to approach zero).

Derivative

A **derivative** term does not consider the error (meaning it cannot bring it to zero: a pure D controller cannot bring the system to its setpoint), but the rate of change of error, trying to bring this rate to zero. It aims at flattening the error trajectory into a horizontal line, damping the force applied, and so reduces overshoot (error on the other side because of too great applied force). Applying too much impetus when the error is small and decreasing will lead to overshoot. After overshooting, if the controller were to apply a large correction in the opposite direction and repeatedly overshoot the desired position, the output would oscillate around the setpoint in either a constant, growing, or decaying sinusoid. If the amplitude of the oscillations increases with time, the system is unstable. If they decrease, the system is stable. If the oscillations remain at a constant magnitude, the system is marginally stable.

Control damping

In the interest of achieving a controlled arrival at the desired position (SP) in a timely and accurate way, the controlled system needs to be critically damped. A well-tuned position control system will also apply the necessary currents to the controlled motor so that the arm pushes and pulls as necessary to resist external forces trying to move it away from the required position. The setpoint itself may be generated by an external system, such as a PLC or other computer system, so that it continuously varies depending on the work that the robotic arm is expected to do. A well-tuned PID control system will enable the arm to meet these changing requirements to the best of its capabilities.

Response to disturbances

If a controller starts from a stable state with zero error (PV = SP), then further changes by the controller will be in response to changes in other measured or unmeasured inputs to the process that affect the process, and hence the PV. Variables that affect the process other than the MV are known as disturbances. Generally, controllers are used to reject disturbances and to implement setpoint changes. A change in load on the arm constitutes a disturbance to the robot arm control process.

Applications

In theory, a controller can be used to control any process that has a measurable output (PV), a known ideal value for that output (SP), and an input to the process (MV) that will affect the relevant PV. Controllers are used in industry to regulate temperature, pressure, force, feed rate,[15] flow rate, chemical composition (component concentrations), weight, position, speed, and practically every other variable for which a measurement exists.

Controller theory

> *This section describes the parallel or non-interacting form of the PID controller. For other forms please see the section Alternative nomenclature and PID forms.*

The PID control scheme is named after its three correcting terms, whose sum constitutes the manipulated variable (MV). The proportional, integral, and derivative terms are summed to calculate the output of the PID controller. Defining $u(t)$ as the controller output, the final form of the PID algorithm is

$$u(t) = \mathrm{MV}(t) = K_\mathrm{p} e(t) + K_\mathrm{i} \int_0^t e(\tau)\,d\tau + K_\mathrm{d} \frac{de(t)}{dt},$$

where

K_p is the proportional gain, a tuning parameter,
K_i is the integral gain, a tuning parameter,
K_d is the derivative gain, a tuning parameter,
$e(t) = \mathrm{SP} - \mathrm{PV}(t)$ is the error (SP is the setpoint, and PV(t) is the process variable),
t is the time or instantaneous time (the present),
τ is the variable of integration (takes on values from time 0 to the present t).

Equivalently, the transfer function in the Laplace domain of the PID controller is

$$L(s) = K_\mathrm{p} + K_\mathrm{i}/s + K_\mathrm{d} s,$$

where s is the complex frequency.

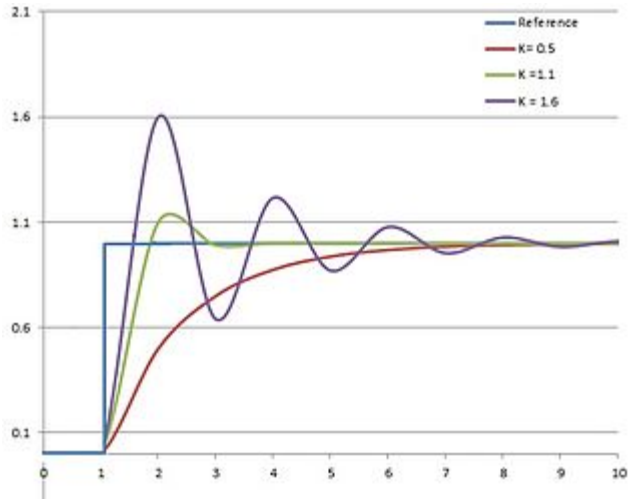
Proportional term

The proportional term produces an output value that is proportional to the current error value. The proportional response can be adjusted by multiplying the error by a constant K_p, called the proportional gain constant.

The proportional term is given by

$$P_{\text{out}} = K_{\text{p}} e(t).$$

A high proportional gain results in a large change in the output for a given change in the error. If the proportional gain is too high, the system can become unstable (see the section on loop tuning). In contrast, a small gain results in a small output response to a large input error, and a less responsive or less sensitive controller. If the proportional gain is too low, the control action may be too small when responding to system disturbances. Tuning theory and industrial practice indicate that the proportional term should contribute the bulk of the output change.



Response of PV to step change of SP vs time, for three values of K_{p} (K_{i} and K_{d} held constant)

Steady-state error

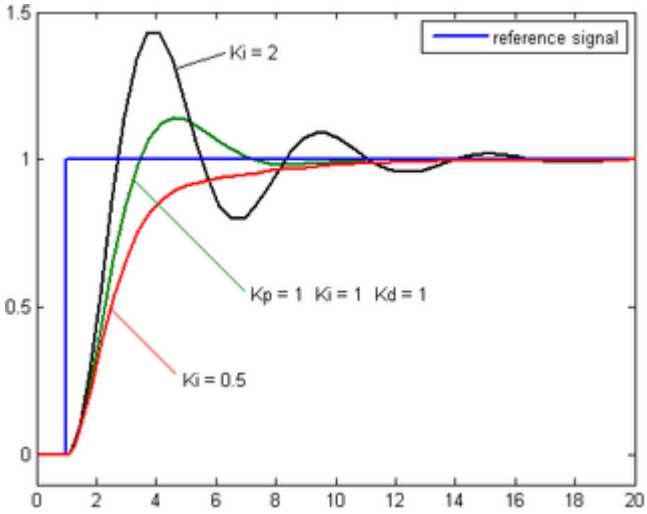
The **steady-state error** is the difference between the desired final output and the actual one.[16] Because a non-zero error is required to drive it, a proportional controller generally operates with a steady-state error.[a] Steady-state error (SSE) is proportional to the process gain and inversely proportional to proportional gain. SSE may be mitigated by adding a compensating bias term to the setpoint AND output or corrected dynamically by adding an integral term.

Integral term

The contribution from the integral term is proportional to both the magnitude of the error and the duration of the error. The integral in a PID controller is the sum of the instantaneous error over time and gives the accumulated offset that should have been corrected previously. The accumulated error is then multiplied by the integral gain (K_{i}) and added to the controller output.

The integral term is given by

$$I_{\text{out}} = K_{\text{i}} \int_0^t e(\tau)\, d\tau.$$



Response of PV to step change of SP vs time, for three values of K_{i} (K_{p} and K_{d} held constant)

The integral term accelerates the movement of the process towards setpoint and eliminates the residual steady-state error that occurs with a pure proportional controller. However, since the integral term responds to accumulated errors from the past, it can cause the present value to overshoot the setpoint value (see the section on loop tuning).

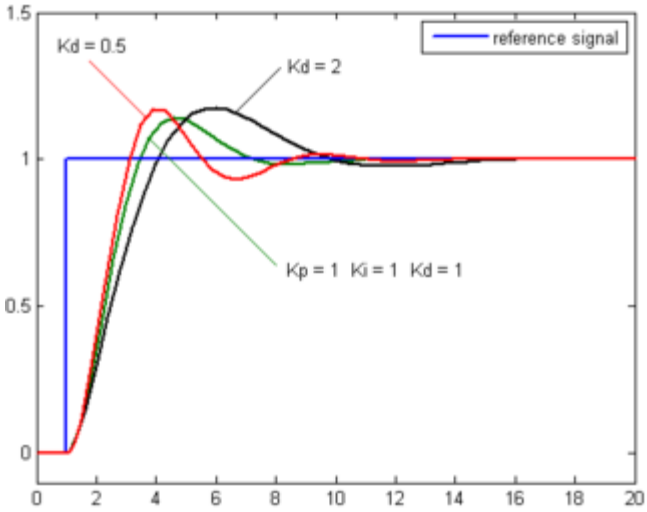
Derivative term

The derivative of the process error is calculated by determining the slope of the error over time and multiplying this rate of change by the derivative gain K_{d}. The magnitude of the contribution of the derivative term to the overall control action is termed the derivative gain, K_{d}.

The derivative term is given by

$$D_{\text{out}} = K_{\text{d}} \frac{de(t)}{dt}.$$

Derivative action predicts system behavior and thus improves settling time and stability of the system.[17][18] An ideal derivative is not causal, so that implementations of PID controllers include an additional low-pass filtering for the derivative term to limit the high-frequency gain and noise. Derivative action is seldom used in practice though – by one estimate in only 25% of deployed controllers – because of its variable impact on system stability in real-world applications.



Response of PV to step change of SP vs time, for three values of K_{d} (K_{p} and K_{i} held constant)

Loop tuning

Tuning a control loop is the adjustment of its control parameters (proportional band/gain, integral gain/reset, derivative gain/rate) to the optimum values for the desired control response. Stability (no unbounded oscillation) is a basic requirement, but beyond that, different systems have different behavior, different applications have different requirements, and requirements may conflict with one another.

PID tuning is a difficult problem, even though there are only three parameters and in principle is simple to describe, because it must satisfy complex criteria within the limitations of PID control. There are accordingly various methods for loop tuning, and more sophisticated techniques are the subject of patents; this section describes some traditional manual methods for loop tuning.

Designing and tuning a PID controller appears to be conceptually intuitive, but can be hard in practice, if multiple (and often conflicting) objectives such as short transient and high stability are to be achieved. PID controllers often provide acceptable control using default tunings, but performance can generally be improved by careful tuning, and performance may be unacceptable with poor tuning. Usually, initial designs need to be adjusted repeatedly through computer simulations until the closed-loop system performs or compromises as desired.

Some processes have a degree of nonlinearity and so parameters that work well at full-load conditions don't work when the process is starting up from no-load; this can be corrected by gain scheduling (using different parameters in different operating regions).

Stability

If the PID controller parameters (the gains of the proportional, integral and derivative terms) are chosen incorrectly, the controlled process input can be unstable, i.e., its output diverges, with or without oscillation, and is limited only by saturation or mechanical breakage. Instability is caused by *excess* gain, particularly in the presence of significant lag.

Generally, stabilization of response is required and the process must not oscillate for any combination of process conditions and setpoints, though sometimes marginal stability (bounded oscillation) is acceptable or desired.

Mathematically, the origins of instability can be seen in the Laplace domain.[19]

The total loop transfer function is:

$$H(s) = \frac{K(s)G(s)}{1 + K(s)G(s)}$$

where

$K(s)$ is the PID transfer function and
$G(s)$ is the plant transfer function

The system is called unstable where the closed loop transfer function diverges for some s.[19] This happens for situations where $K(s)G(s) = -1$. Typically, this happens when $|K(s)G(s)| = 1$ with a 180 degree phase shift. Stability is guaranteed when $K(s)G(s) < 1$ for frequencies that suffer high phase shifts. A more general formalism of this effect is known as the Nyquist stability criterion.

Optimal behavior

The optimal behavior on a process change or setpoint change varies depending on the application.

Two basic requirements are *regulation* (disturbance rejection – staying at a given setpoint) and *command tracking* (implementing setpoint changes) – these refer to how well the controlled variable tracks the desired value. Specific criteria for command tracking include rise time and settling time. Some processes must not allow an overshoot of the process variable beyond the setpoint if, for example, this would be unsafe. Other processes must minimize the energy expended in reaching a new setpoint.

Overview of tuning methods

There are several methods for tuning a PID loop. The most effective methods generally involve the development of some form of process model, then choosing P, I, and D based on the dynamic model parameters. Manual tuning methods can be relatively time-consuming, particularly for systems with long loop times.

The choice of method will depend largely on whether or not the loop can be taken offline for tuning, and on the response time of the system. If the system can be taken offline, the best tuning method often involves subjecting the system to a step change in input, measuring the output as a function of time, and using this response to determine the control parameters.

Choosing a tuning method

Method	Advantages	Disadvantages
Manual tuning	No math required; online.	Requires experienced personnel.
Ziegler–Nichols [b]	Proven method; online.	Process upset, some trial-and-error, very aggressive tuning.
Tyreus Luyben	Proven method; online.	Process upset, some trial-and-error, very aggressive tuning.
Software tools	Consistent tuning; online or offline - can employ computer-automated control system design (*CAutoD*) techniques; may include valve and sensor analysis; allows simulation before downloading; can support non-steady-state (NSS) tuning.	Some cost or training involved.[21]
Cohen–Coon	Good process models.	Some math; offline; only good for first-order processes.
Åström-Hägglund	Can be used for auto tuning; amplitude is minimum so this method has lowest process upset	The process itself is inherently oscillatory.

Manual tuning

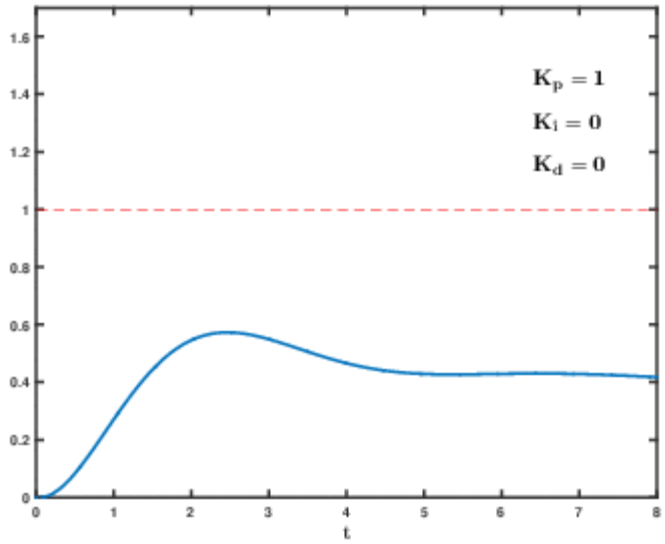
If the system must remain online, one tuning method is to first set K_i and K_d values to zero. Increase the K_p until the output of the loop oscillates, then the K_p should be set to approximately half of that value for a "quarter amplitude decay" type response. Then increase K_i until any offset is corrected in sufficient time for the process. However, too much K_i will cause instability. Finally, increase K_d, if required, until the loop is acceptably quick to reach its reference after a load disturbance. However, too much K_d will cause excessive response and overshoot. A fast PID loop tuning usually overshoots slightly to reach the setpoint more quickly; however, some systems cannot accept overshoot, in which case an overdamped closed-loop system is required, which will require a K_p setting significantly less than half that of the K_p setting that was causing oscillation.

Effects of *increasing* a parameter independently[22][23]

Parameter	Rise time	Overshoot	Settling time	Steady-state error	Stability
K_p	Decrease	Increase	Small change	Decrease	Degrade
K_i	Decrease	Increase	Increase	Eliminate	Degrade
K_d	Minor change	Decrease	Decrease	No effect in theory	Improve if K_d small

Ziegler–Nichols method

Another heuristic tuning method is known as the Ziegler–Nichols method, introduced by John G. Ziegler and Nathaniel B. Nichols in the 1940s. As in the method above, the K_i and K_d gains are first set to zero. The proportional gain is increased until it reaches the ultimate gain, K_u, at which the output of the loop starts to oscillate constantly. K_u and the oscillation period T_u are used to set the gains as follows:



Effects of varying PID parameters (K_p,K_i,K_d) on the step response of a system

Ziegler–Nichols method

Control Type	K_p	K_i	K_d
P	$0.50K_u$	—	—
PI	$0.45K_u$	$0.54K_u/T_u$	—
PID	$0.60K_u$	$1.2K_u/T_u$	$3K_uT_u/40$

These gains apply to the ideal, parallel form of the PID controller. When applied to the standard PID form, only the integral and derivative gains K_i and K_d are dependent on the oscillation period T_u.

Cohen–Coon parameters

This method was developed in 1953 and is based on a first-order + time delay model. Similar to the Ziegler–Nichols method, a set of tuning parameters were developed to yield a closed-loop response with a decay ratio of 1/4. Arguably the biggest problem with these parameters is that a small change in the process parameters could potentially cause a closed-loop system to become unstable.

Relay (Åström–Hägglund) method

Published in 1984 by Karl Johan Åström and Tore Hägglund,[24] the relay method temporarily operates the process using bang-bang control and measures the resultant oscillations. The output is switched (as if by a relay, hence the name) between two values of the control variable. The values must be chosen so the process will cross the setpoint, but need not be 0% and 100%; by choosing suitable values, dangerous oscillations can be avoided.

As long as the process variable is below the setpoint, the control output is set to the higher value. As soon as it rises above the setpoint, the control output is set to the lower value. Ideally, the output waveform is nearly square, spending equal time above and below the setpoint. The period and amplitude of the resultant oscillations are measured, and used to compute the ultimate gain and period, which are then fed into the Ziegler–Nichols method.

Specifically, the ultimate period T_u is assumed to be equal to the observed period, and the ultimate gain is computed as $K_u = 4b/\pi a$, where a is the amplitude of the process variable oscillation, and b is the amplitude of the control output change which caused it.

There are numerous variants on the relay method.[25]

First with dead time model

The transfer function for a first-order process, with dead time, is:

$$y(s) = \frac{k_p e^{-\theta s}}{\tau_p s + 1} * u(s)$$

where k_p is the process gain, τ_p is the time constant, θ is the dead time, and u(s) is a step change input. Converting this transfer function to the time domain results in:

$$y(t) = k_p \Delta u (1 - e^{\frac{-t-\theta}{\tau_p}})$$

using the same parameters found above.

It is important when using this method to apply a large enough step change input that the output can be measured; however, too large of a step change can affect the process stability. Additionally, a larger step change will ensure that the output is not changing due to a disturbance (for best results, try to minimize disturbances when performing the step test).

One way to determine the parameters for the first-order process is using the 63.2% method. In this method, the process gain (k_p) is equal to the change in output divided by the change in input. The dead time (θ) is the amount of time between when the step change occurred and when the output first changed. The time constant (τ_p) is the amount of time it takes for the output to reach 63.2% of the new steady-state value after the step change. One downside to using this method is that the time to reach a new steady-state value can take a while if the process has large time constants. [26]

Tuning software

Most modern industrial facilities no longer tune loops using the manual calculation methods shown above. Instead, PID tuning and loop optimization software are used to ensure consistent results. These software packages will gather the data, develop process models, and suggest optimal tuning. Some software packages can even develop tuning by gathering data from reference changes.

Mathematical PID loop tuning induces an impulse in the system and then uses the controlled system's frequency response to design the PID loop values. In loops with response times of several minutes, mathematical loop tuning is recommended, because trial and error can take days just to find a stable set of loop values. Optimal values are harder to find. Some digital loop controllers offer a self-tuning feature in which very small setpoint changes are sent to the process, allowing the controller itself to calculate optimal tuning values.

Another approach calculates initial values via the Ziegler–Nichols method, and uses a numerical optimization technique to find better PID coefficients.[27]

Other formulas are available to tune the loop according to different performance criteria. Many patented formulas are now embedded within PID tuning software and hardware modules.[28]

Advances in automated PID loop tuning software also deliver algorithms for tuning PID Loops in a dynamic or non-steady state (NSS) scenario. The software will model the dynamics of a process, through a disturbance, and calculate PID control parameters in response.[29]

Limitations

While PID controllers are applicable to many control problems, and often perform satisfactorily without any improvements or only coarse tuning, they can perform poorly in some applications and do not in general provide *optimal* control. The fundamental difficulty with PID control is that it is a feedback control system, with *constant* parameters, and no direct knowledge of the process, and thus overall performance is reactive and a compromise. While PID control is the best controller in an observer without a model of the process, better performance can be obtained by overtly modeling the actor of the process without resorting to an observer.

PID controllers, when used alone, can give poor performance when the PID loop gains must be reduced so that the control system does not overshoot, oscillate or hunt about the control setpoint value. They also have difficulties in the presence of non-linearities, may trade-off regulation versus response time, do not react to changing process behavior (say, the process changes after it has warmed up), and have lag in responding to large disturbances.

The most significant improvement is to incorporate feed-forward control with knowledge about the system, and using the PID only to control error. Alternatively, PIDs can be modified in more minor ways, such as by changing the parameters (either gain scheduling in different use cases or adaptively modifying them based on performance), improving measurement (higher sampling rate, precision, and accuracy, and low-pass filtering if necessary), or cascading multiple PID controllers.

Linearity

Another problem faced with PID controllers is that they are linear and symmetric. Thus, performance of PID controllers in non-linear systems (such as HVAC systems) is variable. For example, in temperature control, a common use case is active heating (via a heating element) but passive cooling (heating off, but no cooling), so overshoot can only be corrected slowly – it cannot be forced downward. In this case the PID should be tuned to be overdamped, to prevent or reduce overshoot, though this reduces performance (it increases settling time).

Noise in derivative

A problem with the derivative term is that it amplifies higher frequency measurement or process noise that can cause large amounts of change in the output. It is often helpful to filter the measurements with a low-pass filter in order to remove higher-frequency noise components. As low-pass filtering and derivative control can cancel each other out, the amount of filtering is limited. Therefore, low noise instrumentation can be important. A nonlinear median filter may be used, which improves the filtering efficiency and practical performance.[30] In some cases, the differential band can be turned off with little loss of control. This is equivalent to using the PID controller as a PI controller.

Modifications to the algorithm

The basic PID algorithm presents some challenges in control applications that have been addressed by minor modifications to the PID form.

Integral windup

One common problem resulting from the ideal PID implementations is integral windup. Following a large change in setpoint the integral term can accumulate an error larger than the maximal value for the regulation variable (windup), thus the system overshoots and continues to increase until this accumulated error is unwound. This problem can be addressed by:

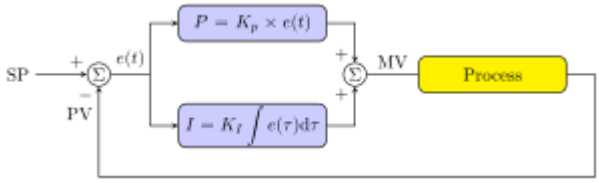
- Disabling the integration until the PV has entered the controllable region
- Preventing the integral term from accumulating above or below pre-determined bounds
- Back-calculating the integral term to constrain the regulator output within feasible bounds.[31]

Overshooting from known disturbances

For example, a PID loop is used to control the temperature of an electric resistance furnace where the system has stabilized. Now when the door is opened and something cold is put into the furnace the temperature drops below the setpoint. The integral function of the controller tends to compensate for error by introducing another error in the positive direction. This overshoot can be avoided by freezing of the integral function after the opening of the door for the time the control loop typically needs to reheat the furnace.

PI controller

A **PI controller** (proportional-integral controller) is a special case of the PID controller in which the derivative (D) of the error is not used.



Basic block of a PI controller

The controller output is given by

$$K_P \Delta + K_I \int \Delta \, dt$$

where Δ is the error or deviation of actual measured value (PV) from the setpoint (SP).

$$\Delta = SP - PV.$$

A PI controller can be modelled easily in software such as Simulink or Xcos using a "flow chart" box involving Laplace operators:

$$C = \frac{G(1 + \tau s)}{\tau s}$$

where

$G = K_P$ = proportional gain
$\frac{G}{\tau} = K_I$ = integral gain

Setting a value for G is often a trade off between decreasing overshoot and increasing settling time.

The lack of derivative action may make the system more steady in the steady state in the case of noisy data. This is because derivative action is more sensitive to higher-frequency terms in the inputs.

Without derivative action, a PI-controlled system is less responsive to real (non-noise) and relatively fast alterations in state and so the system will be slower to reach setpoint and slower to respond to perturbations than a well-tuned PID system may be.

Deadband

Many PID loops control a mechanical device (for example, a valve). Mechanical maintenance can be a major cost and wear leads to control degradation in the form of either stiction or backlash in the mechanical response to an input signal. The rate of mechanical wear is mainly a function of how often a device is activated to make a change. Where wear is a significant concern, the PID loop may have an output deadband to reduce the frequency of activation of the output (valve). This is accomplished by modifying the controller to hold its output steady if the change would be small (within the defined deadband range). The calculated output must leave the deadband before the actual output will change.

Setpoint step change

The proportional and derivative terms can produce excessive movement in the output when a system is subjected to an instantaneous step increase in the error, such as a large setpoint change. In the case of the derivative term, this is due to taking the derivative of the error, which is very large in the case of an instantaneous step change. As a result, some PID algorithms incorporate some of the following modifications:

Setpoint ramping
: In this modification, the setpoint is gradually moved from its old value to a newly specified value using a linear or first-order differential ramp function. This avoids the discontinuity present in a simple step change.

Derivative of the process variable
: In this case the PID controller measures the derivative of the measured process variable (PV), rather than the derivative of the error. This quantity is always continuous (i.e., never has a step change as a result of changed setpoint). This modification is a simple case of setpoint weighting.

Setpoint weighting
: Setpoint weighting adds adjustable factors (usually between 0 and 1) to the setpoint in the error in the proportional and derivative element of the controller. The error in the integral term must be the true control error to avoid steady-state control errors. These two extra parameters do not affect the response to load disturbances and measurement noise and can be tuned to improve the controller's setpoint response.

Feed-forward

The control system performance can be improved by combining the feedback (or closed-loop) control of a PID controller with feed-forward (or open-loop) control. Knowledge about the system (such as the desired acceleration and inertia) can be fed forward and combined with the PID output to improve the overall system performance. The feed-forward value alone can often provide the major portion of the controller output. The PID controller primarily has to compensate for whatever difference or *error* remains between the setpoint (SP) and the system response to the open-loop control. Since the feed-forward output is not affected by the process feedback, it can never cause the control system to oscillate, thus improving the system response without affecting stability. Feed forward can be based on the setpoint and on extra measured disturbances. Setpoint weighting is a simple form of feed forward.

For example, in most motion control systems, in order to accelerate a mechanical load under control, more force is required from the actuator. If a velocity loop PID controller is being used to control the speed of the load and command the force being applied by the actuator, then it is beneficial to take the desired instantaneous acceleration, scale that value appropriately and add it to the output of the PID velocity loop controller. This means that whenever the load is being accelerated or decelerated, a proportional amount of force is commanded from the actuator regardless of the feedback value. The PID loop in this situation uses the

feedback information to change the combined output to reduce the remaining difference between the process setpoint and the feedback value. Working together, the combined open-loop feed-forward controller and closed-loop PID controller can provide a more responsive control system.

Bumpless operation

PID controllers are often implemented with a "bumpless" initialization feature that recalculates the integral accumulator term to maintain a consistent process output through parameter changes.[32] A partial implementation is to store the integral gain times the error rather than storing the error and postmultiplying by the integral gain, which prevents discontinuous output when the I gain is changed, but not the P or D gains.

Other improvements

In addition to feed-forward, PID controllers are often enhanced through methods such as PID gain scheduling (changing parameters in different operating conditions), fuzzy logic, or computational verb logic (ftp://www.cc.uah.es/pub/Alumnos/G_Ing_Informatica/Comp_Flexible/Articulos/CI.pdf).[33][34] Further practical application issues can arise from instrumentation connected to the controller. A high enough sampling rate, measurement precision, and measurement accuracy are required to achieve adequate control performance. Another new method for improvement of PID controller is to increase the degree of freedom by using fractional order. The order of the integrator and differentiator add increased flexibility to the controller.[35]

Cascade control

One distinctive advantage of PID controllers is that two PID controllers can be used together to yield better dynamic performance. This is called cascaded PID control. Two controllers are in cascade when they are arranged so that one regulates the set point of the other. A PID controller acts as outer loop controller, which controls the primary physical parameter, such as fluid level or velocity. The other controller acts as inner loop controller, which reads the output of outer loop controller as setpoint, usually controlling a more rapid changing parameter, flowrate or acceleration. It can be mathematically proven that the working frequency of the controller is increased and the time constant of the object is reduced by using cascaded PID controllers..

For example, a temperature-controlled circulating bath has two PID controllers in cascade, each with its own thermocouple temperature sensor. The outer controller controls the temperature of the water using a thermocouple located far from the heater, where it accurately reads the temperature of the bulk of the water. The error term of this PID controller is the difference between the desired bath temperature and measured temperature. Instead of controlling the heater directly, the outer PID controller sets a heater temperature goal for the inner PID controller. The inner PID controller controls the temperature of the heater using a thermocouple attached to the heater. The inner controller's error term is the difference between this heater temperature setpoint and the measured temperature of the heater. Its output controls the actual heater to stay near this setpoint.

The proportional, integral, and differential terms of the two controllers will be very different. The outer PID controller has a long time constant – all the water in the tank needs to heat up or cool down. The inner loop responds much more quickly. Each controller can be tuned to match the physics of the system *it* controls – heat transfer and thermal mass of the whole tank or of just the heater – giving better total response.

Alternative nomenclature and forms

Standard versus parallel (ideal) form

The form of the PID controller most often encountered in industry, and the one most relevant to tuning algorithms is the *standard form*. In this form the K_p gain is applied to the I_{out}, and D_{out} terms, yielding:

$$u(t) = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau)\,d\tau + T_d \frac{d}{dt} e(t) \right)$$

where

T_i is the *integral time*
T_d is the *derivative time*

In this standard form, the parameters have a clear physical meaning. In particular, the inner summation produces a new single error value which is compensated for future and past errors. The proportional error term is the current error. The derivative components term attempts to predict the error value at T_d seconds (or samples) in the future, assuming that the loop control remains unchanged. The integral component adjusts the error value to compensate for the sum of all past errors, with the intention of completely eliminating them in T_i seconds (or samples). The resulting compensated single error value is then scaled by the single gain K_p to compute the control variable.

In the parallel form, shown in the controller theory section

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\,d\tau + K_d \frac{d}{dt} e(t)$$

the gain parameters are related to the parameters of the standard form through $K_i = K_p/T_i$ and $K_d = K_p T_d$. This parallel form, where the parameters are treated as simple gains, is the most general and flexible form. However, it is also the form where the parameters have the weakest relationship to physical behaviors and is generally reserved for theoretical treatment of the PID controller. The standard form, despite being slightly more complex mathematically, is more common in industry.

Reciprocal gain, a.k.a. proportional band

In many cases, the manipulated variable output by the PID controller is a dimensionless fraction between 0 and 100% of some maximum possible value, and the translation into real units (such as pumping rate or watts of heater power) is outside the PID controller. The process variable, however, is in dimensioned units such as temperature. It is common in this case to express the gain K_p not as "output per degree", but rather in the reciprocal form of a *proportional band* $100/K_p$, which is "degrees per full output": the range over which the output changes from 0 to 1 (0% to 100%). Beyond this range, the output is saturated, full-off or full-on. The narrower this band, the higher the proportional gain.

Basing derivative action on PV

In most commercial control systems, derivative action is based on process variable rather than error. That is, a change in the setpoint does not affect the derivative action. This is because the digitized version of the algorithm produces a large unwanted spike when the setpoint is changed. If the setpoint is constant then changes in the PV will be the same as changes in error. Therefore, this modification makes no difference to the way the controller responds to process disturbances.

Basing proportional action on PV

Most commercial control systems offer the *option* of also basing the proportional action solely on the process variable. This means that only the integral action responds to changes in the setpoint. The modification to the algorithm does not affect the way the controller responds to process disturbances. Basing proportional action on PV eliminates the instant and possibly very large change in output caused by a sudden change to the setpoint. Depending on the process and tuning this may be beneficial to the response to a setpoint step.

$$\mathrm{MV(t)} = K_p \left(-PV(t) + \frac{1}{T_i} \int_0^t e(\tau)\,d\tau - T_d \frac{d}{dt} PV(t) \right)$$

King[36] describes an effective chart-based method.

Laplace form

Sometimes it is useful to write the PID regulator in Laplace transform form:

$$G(s) = K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s}$$

Having the PID controller written in Laplace form and having the transfer function of the controlled system makes it easy to determine the closed-loop transfer function of the system.

Series/interacting form

Another representation of the PID controller is the series, or *interacting* form

$$G(s) = K_c \left(\frac{1}{\tau_i s} + 1\right)(\tau_d s + 1)$$

where the parameters are related to the parameters of the standard form through

$K_p = K_c \cdot \alpha$, $T_i = \tau_i \cdot \alpha$, and
$T_d = \frac{\tau_d}{\alpha}$

with

$$\alpha = 1 + \frac{\tau_d}{\tau_i}.$$

This form essentially consists of a PD and PI controller in series. As the integral is required to calculate the controller's bias this form provides the ability to track an external bias value which is required to be used for proper implementation of multi-controller advanced control schemes.

Discrete implementation

The analysis for designing a digital implementation of a PID controller in a microcontroller (MCU) or FPGA device requires the standard form of the PID controller to be *discretized*.[37] Approximations for first-order derivatives are made by backward finite differences. The integral term is discretized, with a sampling time Δt,

as follows,

$$\int_0^{t_k} e(\tau)\,d\tau = \sum_{i=1}^{k} e(t_i)\,\Delta t$$

The derivative term is approximated as,

$$\frac{de(t_k)}{dt} = \frac{e(t_k) - e(t_{k-1})}{\Delta t}$$

Thus, a *velocity algorithm* for implementation of the discretized PID controller in a MCU is obtained by differentiating $u(t)$, using the numerical definitions of the first and second derivative and solving for $u(t_k)$ and finally obtaining:

$$u(t_k) = u(t_{k-1}) + K_p \left[\left(1 + \frac{\Delta t}{T_i} + \frac{T_d}{\Delta t}\right) e(t_k) + \left(-1 - \frac{2T_d}{\Delta t}\right) e(t_{k-1}) + \frac{T_d}{\Delta t} e(t_{k-2}) \right]$$

s.t. $T_i = K_p/K_i, T_d = K_d/K_p$

Pseudocode

Here is a simple software loop that implements a PID algorithm:[38]

```
previous_error := 0
integral := 0

loop:
    error := setpoint − measured_value
    integral := integral + error × dt
    derivative := (error − previous_error) / dt
    output := Kp × error + Ki × integral + Kd × derivative
    previous_error := error
    wait(dt)
    goto loop
```

In this example, two variables that will be maintained within the loop are initialized to zero, then the loop begins. The current *error* is calculated by subtracting the *measured_value* (the process variable, or PV) from the current *setpoint* (SP). Then, *integral* and *derivative* values are calculated, and these and the *error* are combined with three preset gain terms – the proportional gain, the integral gain and the derivative gain – to derive an *output* value.

In the real world, this is D-to-A converted and passed into the process under control as the manipulated variable (MV). The current error is stored elsewhere for re-use in the next differentiation, the program then waits until dt seconds have passed since start, and the loop begins again, reading in new values for the PV and the setpoint and calculating a new value for the error.[38]

Note that for real code, the use of "wait(dt)" might be inappropriate because it doesn't account for time taken by the algorithm itself during the loop, or more importantly, any preemption delaying the algorithm.

See also

- Control theory

Notes

a. The only exception is where the target value is the same as the value obtained when the controller output is zero.

b. A common assumption often made for Proportional-Integral-Derivative (PID) control design, as done by Ziegler and Nichols, is to take the integral time constant to be four times the derivative time constant. Although this choice is reasonable, selecting the integral time constant to have this value may have had something to do with the fact that, for the ideal case with a derivative term with no filter, the PID transfer function consists of two real and equal zeros in the numerator.[20]

References

1. Araki, M. "PID Control" (http://www.eolss.net/ebooks/Sample%20Chapters/C18/E6-43-03-03.pdf) (PDF).
2. Hills, Richard L (1996), *Power From the Wind*, Cambridge University Press
3. Richard E. Bellman (December 8, 2015). *Adaptive Control Processes: A Guided Tour* (https://books.google.com/books?id=iwbWCgAAQBAJ&q=%22Centrifugal+Governor%22+Huygens&pg=PA36). Princeton University Press. ISBN 9781400874668.
4. Bennett, Stuart (1996). "A brief history of automatic control" (https://web.archive.org/web/20160809050823/http://ieeecss.org/CSM/library/1996/june1996/02-HistoryofAutoCtrl.pdf) (PDF). *IEEE Control Systems Magazine*. **16** (3): 17–25. doi:10.1109/37.506394 (https://doi.org/10.1109%2F37.506394). Archived from the original (http://ieeecss.org/CSM/library/1996/june1996/02-HistoryofAutoCtrl.pdf) (PDF) on 2016-08-09. Retrieved 2014-08-21.
5. Maxwell, J. C. (1868). "On Governors" (https://upload.wikimedia.org/wikipedia/commons/b/b1/On_Governors.pdf) (PDF). *Proceedings of the Royal Society*. **100**.
6. Newpower, Anthony (2006). *Iron Men and Tin Fish: The Race to Build a Better Torpedo during World War II*. Praeger Security International. ISBN 978-0-275-99032-9. p. citing Gray, Edwyn (1991), *The Devil's Device: Robert Whitehead and the History of the Torpedo*, Annapolis, MD: U.S. Naval Institute, p. 33.
7. Sleeman, C. W. (1880), *Torpedoes and Torpedo Warfare* (https://archive.org/stream/torpedoestorpedo00sleerich#page/136/mode/2up/search/depth), Portsmouth: Griffin & Co., pp. 137–138, "which constitutes what is termed as the secret of the fish torpedo."
8. "A Brief Building Automation History" (https://web.archive.org/web/20110708104028/http://www.building-automation-consultants.com/building-automation-history.html). Archived from the original (http://www.building-automation-consultants.com/building-automation-history.html) on 2011-07-08. Retrieved 2011-04-04.
9. Minorsky, Nicolas (1922). "Directional stability of automatically steered bodies". *J. Amer. Soc. Naval Eng*. **34** (2): 280–309. doi:10.1111/j.1559-3584.1922.tb04958.x (https://doi.org/10.1111%2Fj.1559-3584.1922.tb04958.x).
10. Bennett 1993, p. 67 (https://books.google.com/books?id=VD_b81J3yFoC&pg=PA67)
11. Bennett, Stuart (June 1986). *A history of control engineering, 1800-1930*. IET. pp. 142–148 (https://books.google.com/books?id=1gfKkqB_fTcC&pg=PA142). ISBN 978-0-86341-047-5.
12. Shinskey, F Greg (2004), *The power of external-reset feedback* (https://classes.engineering.wustl.edu/2009/spring/che433/2009-LAB/Control%20Theory/external-reset.pdf) (PDF), Control Global
13. Neuhaus, Rudolf. "Diode Laser Locking and Linewidth Narrowing" (http://www.toptica.com/fileadmin/user_upload/Articles_Application_Notes/toptica_AP_1012_laser_locking_2012.pdf) (PDF). Retrieved June 8, 2015.

14. "Position control system" (http://www.ee.hacettepe.edu.tr/~solen/ELE356/exp1.pdf) (PDF). Hacettepe University Department of Electrical and Electronics Engineering.
15. Kebriaei, Reza; Frischkorn, Jan; Reese, Stefanie; Husmann, Tobias; Meier, Horst; Moll, Heiko; Theisen, Werner (2013). "Numerical modelling of powder metallurgical coatings on ring-shaped parts integrated with ring rolling". *Material Processing Technology*. **213** (1): 2015–2032. doi:10.1016/j.jmatprotec.2013.05.023 (https://doi.org/10.1016%2Fj.jmatprotec.2013.05.023).
16. Lipták, Béla G. (2003). *Instrument Engineers' Handbook: Process control and optimization* (4th ed.). CRC Press. p. 108. ISBN 0-8493-1081-4.
17. "Introduction: PID Controller Design" (http://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID). University of Michigan.
18. Tim Wescott (October 2000). "PID without a PhD" (http://igor.chudov.com/manuals/Servo-Tuning/PID-without-a-PhD.pdf) (PDF). EE Times-India.
19. Bechhoefer, John (2005). "Feedback for Physicists: A Tutorial Essay On Control". *Reviews of Modern Physics*. **77** (3): 783–835. Bibcode:2005RvMP...77..783B (https://ui.adsabs.harvard.edu/abs/2005RvMP...77..783B). CiteSeerX 10.1.1.124.7043 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.124.7043). doi:10.1103/revmodphys.77.783 (https://doi.org/10.1103%2Frevmodphys.77.783).
20. Atherton, Drek P (December 2014). "Almost Six Decades in Control Engineering". *IEEE Control Systems Magazine*. **34** (6): 103–110. doi:10.1109/MCS.2014.2359588 (https://doi.org/10.1109%2FMCS.2014.2359588). S2CID 20233207 (https://api.semanticscholar.org/CorpusID:20233207).
21. Li, Y., et al. (2004) CAutoCSD - Evolutionary search and optimisation enabled computer automated control system design, Int J Automation and Computing, vol. 1, No. 1, pp. 76-88. ISSN 1751-8520 (https://www.worldcat.org/search?fq=x0:jrnl&q=n2:1751-8520).
22. Kiam Heong Ang; Chong, G.; Yun Li (2005). "PID control system analysis, design, and technology" (http://eprints.gla.ac.uk/3817/1/IEEE3.pdf) (PDF). *IEEE Transactions on Control Systems Technology*. **13** (4): 559–576. doi:10.1109/TCST.2005.847331 (https://doi.org/10.1109%2FTCST.2005.847331). S2CID 921620 (https://api.semanticscholar.org/CorpusID:921620).
23. Jinghua Zhong (Spring 2006). "PID Controller Tuning: A Short Tutorial" (https://web.archive.org/web/20150421081758/http://saba.kntu.ac.ir/eecd/pcl/download/PIDtutorial.pdf) (PDF). Archived from the original (http://saba.kntu.ac.ir/eecd/pcl/download/PIDtutorial.pdf) (PDF) on 2015-04-21. Retrieved 2011-04-04.
24. Åström, K.J.; Hägglund, T. (July 1984). "Automatic Tuning of Simple Regulators" (https://lup.lub.lu.se/record/8601786). *IFAC Proceedings Volumes*. **17** (2): 1867–1872. doi:10.1016/S1474-6670(17)61248-5 (https://doi.org/10.1016%2FS1474-6670%2817%2961248-5).
25. Hornsey, Stephen (29 October 2012). "A Review of Relay Auto-tuning Methods for the Tuning of PID-type Controllers" (http://www2.warwick.ac.uk/fac/cross_fac/iatl/reinvention/issues/volume5issue2/hornsey). *Reinvention*. **5** (2).
26. Bequette, B. Wayne (2003). *Process Control: Modeling, Design, and Simulation*. Upper Saddle River, New Jersey: Prentice Hall. p. 129. ISBN 978-0-13-353640-9.
27. Heinänen, Eero (October 2018). *A Method for automatic tuning of PID controller following Luus-Jaakola optimization* (https://dspace.cc.tut.fi/dpub/bitstream/handle/123456789/26690/Heinanen.pdf) (PDF) (Master's Thesis ed.). Tampere, Finland: Tampere University of Technology. Retrieved Feb 1, 2019.
28. Li, Yun; Ang, Kiam Heong; Chong, Gregory C.Y. (February 2006). "Patents, software, and hardware for PID control: An overview and analysis of the current art" (http://eprints.gla.ac.uk/3816/1/IEEE2pdf.pdf) (PDF). *IEEE Control Systems Magazine*. **26** (1): 42–54. doi:10.1109/MCS.2006.1580153 (https://doi.org/10.1109%2FMCS.2006.1580153). S2CID 18461921 (https://api.semanticscholar.org/CorpusID:18461921).
29. Soltesz, Kristian (January 2012). *On Automation of the PID Tuning Procedure* (http://www.control.lth.se/documents/2012/sol2012lic.pdf) (PDF) (Licentiate theis). Lund university. 847ca38e-93e8-4188-b3d5-8ec6c23f2132 (http://lup.lub.lu.se/record/2293573).

30. Li, Y. and Ang, K.H. and Chong, G.C.Y. (2006) PID control system analysis and design - Problems, remedies, and future directions (http://eprints.gla.ac.uk/3815/1/IEEE_CS_PID_01580152.pdf). IEEE Control Systems Magazine, 26 (1). pp. 32-41. ISSN 0272-1708 (https://www.worldcat.org/search?fq=x0:jrnl&q=n2:0272-1708)
31. Cooper, Douglas. "Integral (Reset) Windup, Jacketing Logic and the Velocity PI Form" (http://www.controlguru.com/2008/021008.html). Retrieved 2014-02-18.
32. Cooper, Douglas. "PI Control of the Heat Exchanger" (http://www.controlguru.com/wp/p71.html). *Practical Process Control by Control Guru*. Retrieved 2014-02-27.
33. Yang, T. (June 2005). "Architectures of Computational Verb Controllers: Towards a New Paradigm of Intelligent Control". *International Journal of Computational Cognition*. **3** (2): 74–101. CiteSeerX 10.1.1.152.9564 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.152.9564).
34. Liang, Yilong; Yang, Tao (2009). "Controlling fuel annealer using computational verb PID controllers" (http://dl.acm.org/citation.cfm?id=1719210&preflayout=tabs). *Proceedings of the 3rd International Conference on Anti-Counterfeiting, Security, and Identification in Communication*: 417–420.
35. Tenreiro Machado JA, et al. (2009). "Some Applications of Fractional Calculus in Engineering" (https://doi.org/10.1155%2F2010%2F639801). *Mathematical Problems in Engineering*. **2010**: 1–34. doi:10.1155/2010/639801 (https://doi.org/10.1155%2F2010%2F639801). hdl:10400.22/4306 (https://hdl.handle.net/10400.22%2F4306).
36. King, Myke (2011). *Process Control: A Practical Approach*. Wiley. pp. 52–78. ISBN 978-0-470-97587-9.
37. "Discrete PI and PID Controller Design and Analysis for Digital Implementation" (https://www.scribd.com/doc/19070283/Discrete-PI-and-PID-Controller-Design-and-Analysis-for-Digital-Implementation). Scribd.com. Retrieved 2011-04-04.
38. "PID process control, a "Cruise Control" example" (http://www.codeproject.com/Articles/36459/PID-process-control-a-Cruise-Control-example). CodeProject. 2009. Retrieved 4 November 2012.

- Bequette, B. Wayne (2006). *Process Control: Modeling, Design, and Simulation*. Prentice Hall PTR. ISBN 9789861544779.

Further reading

- Liptak, Bela (1995). *Instrument Engineers' Handbook: Process Control*. Radnor, Pennsylvania: Chilton Book Company. pp. 20–29. ISBN 978-0-8019-8242-2.
- Tan, Kok Kiong; Wang Qing-Guo; Hang Chang Chieh (1999). *Advances in PID Control*. London, UK: Springer-Verlag. ISBN 978-1-85233-138-2.
- King, Myke (2010). *Process Control: A Practical Approach* (http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0470975873.html). Chichester, UK: John Wiley & Sons Ltd. ISBN 978-0-470-97587-9.
- Van Doren, Vance J. (July 1, 2003). "Loop Tuning Fundamentals" (http://old.controleng.com/article/268148-Loop_Tuning_Fundamentals.php.html). *Control Engineering*.
- Sellers, David. "An Overview of Proportional plus Integral plus Derivative Control and Suggestions for Its Successful Application and Implementation" (https://web.archive.org/web/20070307161741/http://www.peci.org/library/PECI_ControlOverview1_1002.pdf) (PDF). Archived from the original (http://www.peci.org/library/PECI_ControlOverview1_1002.pdf) (PDF) on March 7, 2007. Retrieved 2007-05-05.
- Graham, Ron; Mike McHugh (2005-10-03). "FAQ on PID controller tuning" (https://web.archive.org/web/20050206113949/http://www.tcnj.edu/~rgraham/PID-tuning.html). Mike McHugh. Archived from the original (http://www.tcnj.edu/~rgraham/PID-tuning.html) on February 6, 2005. Retrieved 2009-01-05.

- Aidan O'Dwyer (2009). *Handbook of PI and PID Controller Tuning Rules* (http://cyxtp.ucoz.ru/pdf/Aidan_O_Dwyer_Handbook_of_PI_and_PID_Controller_Tuning_Rules.pdf) (PDF) (3rd ed.). Imperial College Press. ISBN 978-1-84816-242-6.

External links

- PID tuning using Mathematica (http://reference.wolfram.com/mathematica/ref/PIDTune.html)
- PID tuning using Python (https://apmonitor.com/pdc/index.php/Main/ProportionalIntegralDerivative)
- Principles of PID Control and Tuning (http://www.eurotherm.com/temperature-control/principles-of-pid-control-and-tuning)
- Introduction to the key terms associated with PID Temperature Control (http://www.eurotherm.com/temperature-control/pid-control-made-easy)

PID tutorials

- PID Control in MATLAB/Simulink and Python with TCLab (http://apmonitor.com/pdc/index.php/Main/ArduinoTemperatureControl)
- What's All This P-I-D Stuff, Anyhow? (http://electronicdesign.com/analog/whats-all-p-i-d-stuff-anyhow) Article in Electronic Design
- Shows how to build a PID controller with basic electronic components (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.154.240&rep=rep1&type=pdf) (pg. 22)
- PID Without a PhD (https://www.wescottdesign.com/articles/pid/pidWithoutAPhd.pdf)
- PID Control with MATLAB and Simulink (http://www.mathworks.com/discovery/pid-control.html)
- PID with single Operational Amplifier (http://www.postreh.com/vmichal/papers/PID-Radio.pdf)
- Proven Methods and Best Practices for PID Control (http://www.controlguru.com/pages/table.html)
- Principles of PID Control and Tuning (http://www.eurotherm.com/temperature-control/principles-of-pid-control-and-tuning)
- PID Tuning Guide: A Best-Practices Approach to Understanding and Tuning PID Controllers (http://controlstation.com/portfolio-items/pid-tuning-guide-a-best-practices-approach-to-understanding-and-tuning-pid-controllers/?portfolioID=7052)
- Michael Barr (2002-07-30), *Introduction to Closed-Loop Control* (https://web.archive.org/web/20100209125831/http://embedded.com/story/OEG20020726S0044), Embedded Systems Programming, archived from the original (https://www.embedded.com/story/OEG20020726S0044) on 2010-02-09
- Jinghua Zhong, Mechanical Engineering, Purdue University (Spring 2006). "PID Controller Tuning: A Short Tutorial" (https://web.archive.org/web/20150421081758/http://saba.kntu.ac.ir/eecd/pcl/download/PIDtutorial.pdf) (PDF). Archived from the original (http://saba.kntu.ac.ir/eecd/pcl/download/PIDtutorial.pdf) (PDF) on 2015-04-21. Retrieved 2013-12-04.
- Introduction to P,PI,PD & PID Controller with MATLAB (http://vivekbose.com/introduction-to-pid-controller-with-detailed-ppipd-pd-control)

Online calculators

- PID tutorial, free PID tuning tools, advanced PID control schemes, on-line PID simulators (http://www.pidlab.com/)
- Online PID Tuning applet from University of Texas Control Group (http://www.che.utexas.edu/course/che360/Process_Tuner.html)

- Online PID tuning application (https://pidtuner.com)
