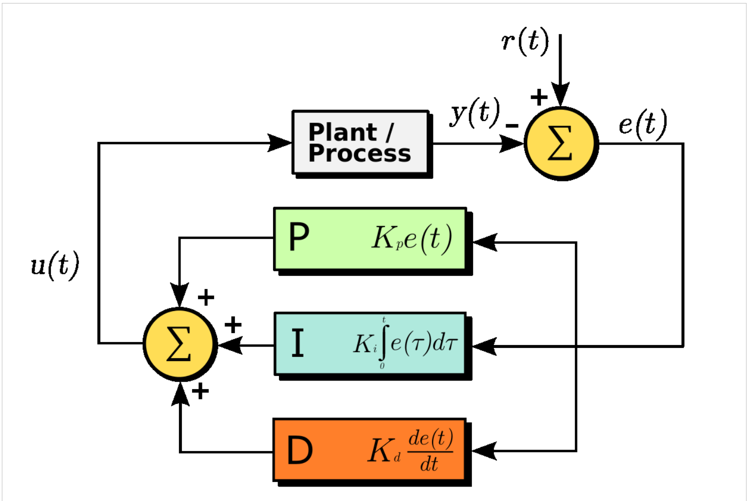
PID controller

A **proportional-integral-derivative controller** (**PID controller**) is a control loop feedback mechanism (controller) widely used in industrial control systems. A PID controller calculates an *error* value as the difference between a measured process variable and a desired setpoint. The controller attempts to minimize the *error* by adjusting the process through use of a manipulated variable.



A block diagram of a PID controller in a feedback loop

The PID controller algorithm involves three separate constant parameters, and is accordingly sometimes called **three-term control**: the proportional, the integral and derivative values, denoted *P, I,* and *D.* Simply put, these values can be interpreted in terms of time: *P* depends on the *present* error, *I* on the accumulation of *past* errors, and *D* is a prediction of *future* errors, based on current rate of change. The weighted sum of these three actions is used to adjust the process via a control element such as the position of a control valve, a damper, or the power supplied to a heating element.

In the absence of knowledge of the underlying process, a PID controller has historically been considered to be the best controller. By tuning the three parameters in the PID controller algorithm, the controller can provide control action designed for specific process requirements. The response of the controller can be described in terms of the responsiveness of the controller to an error, the degree to which the controller overshoots the setpoint, and the degree of system oscillation. Note that the use of the PID algorithm for control does not guarantee optimal control of the system or system stability.

Some applications may require using only one or two actions 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 due to the control action.

History and applications

PID controllers date to 1890s governor design. PID controllers were subsequently developed in automatic ship steering. One of the earliest examples of a PID-type controller was developed by Elmer Sperry in 1911, while the first published theoretical analysis of a PID controller was by Russian American engineer Nicolas Minorsky, (Minorsky 1922). Minorsky was designing automatic steering systems for the US Navy, and based his analysis on observations of a helmsman, noting the helmsman controlled the ship based not only on the current error, but also on past error as well as the current rate of change; this was then made mathematical by Minorsky. His goal was stability, not

PID theory developed by observing the action of helmsmen.

general control, which simplified the problem significantly. While proportional control provides stability against small disturbances, it was insufficient for dealing with a steady disturbance, notably a stiff gale (due to droop), which required adding the integral term. Finally, the derivative term was added to improve control.

Trials were carried out on the USS *New Mexico*, with the controller controlling the *angular velocity* (not 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.

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.

In the early history of automatic process control the PID controller was implemented as a mechanical device. These mechanical controllers used a lever, spring and a mass and were often energized by compressed air. These pneumatic controllers were once the industry standard.

Electronic analog controllers can be made from a solid-state or tube amplifier, a capacitor and a resistor. 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. Nowadays, electronic controllers have largely been replaced by digital controllers implemented with microcontrollers or FPGAs.

Most modern PID controllers in industry are implemented in programmable logic controllers (PLCs) or as a panel-mounted digital controller. Software implementations have the advantages that they are relatively cheap and are flexible with respect to the implementation of the PID algorithm. PID temperature controllers are applied in industrial ovens, plastics injection machinery, hot stamping machines and packing industry.

Variable voltages may be applied by the time proportioning form of pulse-width modulation (PWM)—a cycle time is fixed, and variation is achieved by varying the proportion of the time during this cycle that the controller outputs +1 (or −1) instead of 0. On a digital system the possible proportions are discrete—e.g., increments of 0.1 second within a 2 second cycle time yields 20 possible steps: percentage increments of 5%; so there is a discretization error, but for high enough time resolution this yields satisfactory performance.

Control loop basics

Further information: Control system

A familiar example of a control loop is the action taken when adjusting hot and cold faucets to fill a container with water at a desired temperature by mixing hot and cold water. The person touches the water in the container as it fills to sense its temperature. Based on this feedback they perform a control action by adjusting the hot and cold faucets until the temperature stabilizes as desired.

The sensed water temperature is the process variable (PV). The desired temperature is called the setpoint (SP). The input to the process (the water valve position), and the output of the PID controller, is called the manipulated variable (MV) or the control variable (CV). The difference between the temperature measurement and the setpoint is the error (e) and quantifies whether the water in the container is too hot or too cold and by how much.

After measuring the temperature (PV), and then calculating the error, the controller decides how to set the tap position (MV). The obvious method is **proportional** control: the tap position is set in proportion to the current error. A more complex control may include **derivative** action. This also considers the rate of temperature change: adding extra hot water if the temperature is falling, and less on rising temperature. Finally **integral** action uses the average temperature in the past to detect whether the temperature of the container is settling out too low or too high and set the tap proportional to the past errors. An alternative formulation of integral action is to change the current tap position in steps proportional to the current error. Over time the steps add up (which is the discrete time equivalent to integration) the past errors.

Making a change that is too large when the error is small will lead to overshoot. If the controller were to repeatedly make changes that were too large and repeatedly overshoot the target, the output would oscillate around the setpoint in either a constant, growing, or decaying sinusoid. If the amplitude of the oscillations increase 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.

In the interest of achieving a gradual convergence to the desired temperature (SP), the controller may damp the anticipated future oscillations by tempering its adjustments, or reducing the loop gain.

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. Changes in feedwater temperature constitute a disturbance to the faucet temperature control process.

In theory, a controller can be used to control any process which 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, flow rate, chemical composition, weight, position, speed and practically every other variable for which a measurement exists.

PID 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:

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

where

K_p : Proportional gain, a tuning parameter

K_i : Integral gain, a tuning parameter

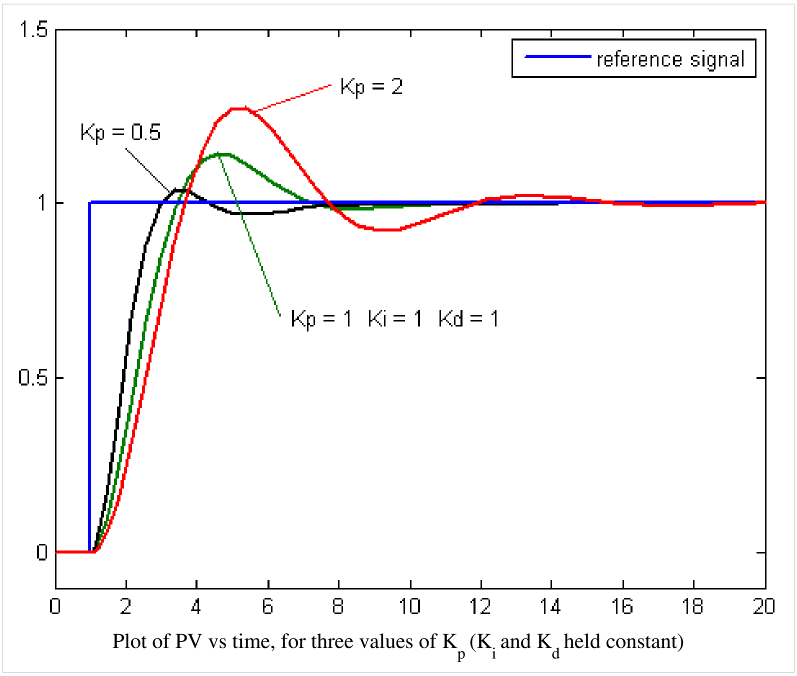
K_d : Derivative gain, a tuning parameter

e : Error $= SP - PV$

t : Time or instantaneous time (the present)

τ: Variable of integration; takes on values from time 0 to the present t .

Proportional term



Plot of PV vs time, for three values of K_p (K_i and K_d held constant)

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:

$$P_{\text{out}} = K_p e(t)$$

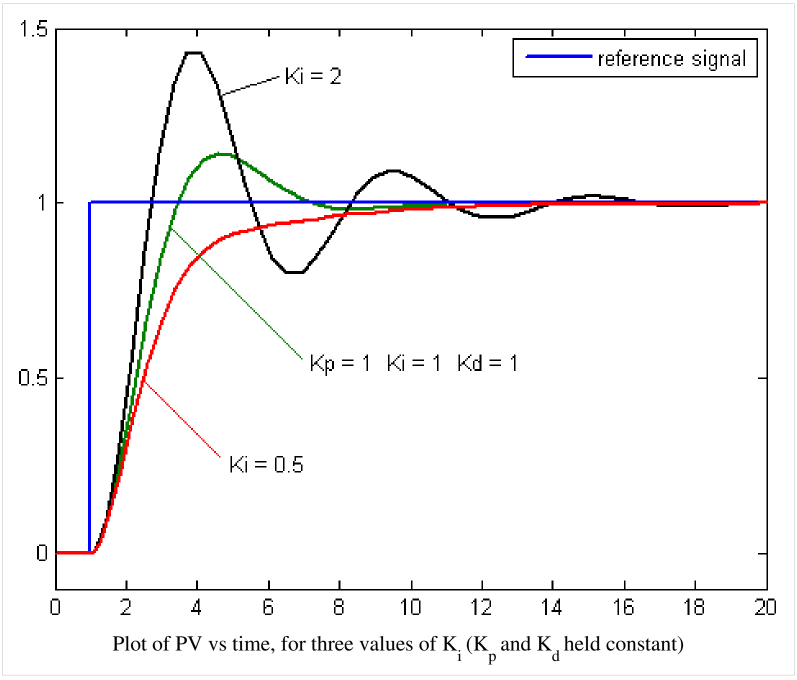
A high proportional gain results in a large change in the output for a given change in the error. If the proportional gain is too high, the system can become unstable (see the section on loop tuning). In contrast, a small gain results in a small output response to a large input error, and a less responsive 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.Wikipedia:Citation needed

Droop

Because a non-zero error is required to drive it, a proportional controller generally operates with a steady-state error, referred to as *droop*.[1] Droop is proportional to the process gain and inversely proportional to proportional gain. Droop may be mitigated by adding a compensating bias term to the setpoint or 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.



Plot of PV vs time, for three values of K_i (K_p and K_d held constant)

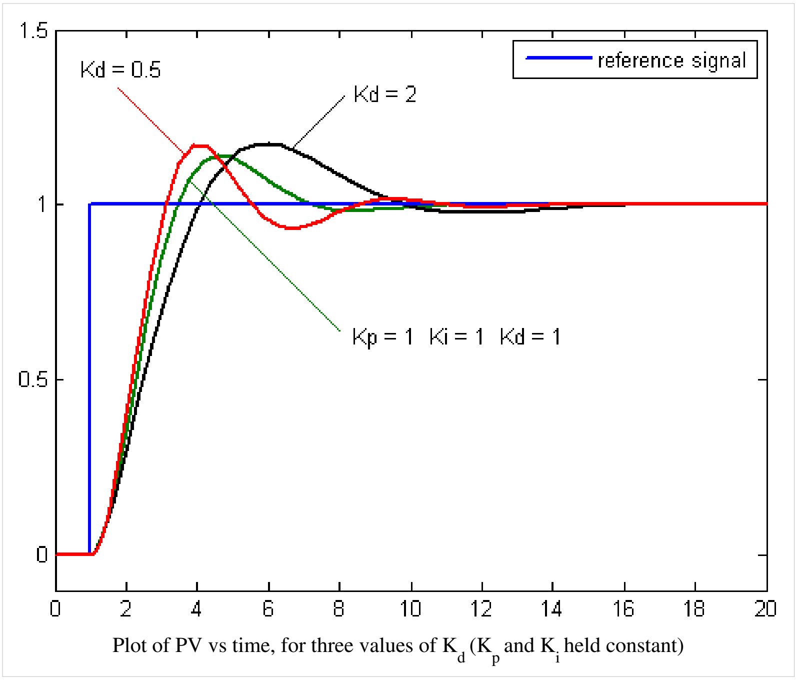
The integral term is given by:

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

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.



Plot of PV vs time, for three values of K_d (K_p and K_i held constant)

The derivative term is given by:

$$D_{\text{out}} = K_d \frac{d}{dt} e(t)$$

Derivative action predicts system behavior and thus improves settling time and stability of the system. 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 20% of deployed controllers - because of its variable impact on system stability in real-world applications.

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.Wikipedia:Citation needed

Optimum behavior

The optimum 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 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 inefficient, particularly if the loops have response times on the order of minutes or longer. Wikipedia:Citation needed

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. Wikipedia:Citation needed

Choosing a tuning method

Method	Advantages	Disadvantages
Manual tuning	No math required; online.	Requires experienced personnel. Wikipedia:Citation needed
Ziegler–Nichols	Proven method; online.	Process upset, some trial-and-error, very aggressive tuning. Wikipedia:Citation needed
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.[2]
Cohen–Coon	Good process models.	Some math; offline; only good for first-order processes. Wikipedia:Citation needed

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 *over-damped* 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.Wikipedia:Citation needed

Effects of *increasing* a parameter independently

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

For more details on this topic, see Ziegler–Nichols method.

Another heuristic tuning method is formally 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. K_u and the oscillation period P_u are used to set the gains as shown:

Ziegler–Nichols method

Control Type	K_p	K_i	K_d
P	$0.50K_u$	-	-
PI	$0.45K_u$	$1.2K_p/P_u$	-
PID	$0.60K_u$	$2K_p/P_u$	$K_pP_u/8$

These gains apply to the ideal, parallel form of the PID controller. When applied to the standard PID form, the integral and derivative time parameters T_i and T_d are only dependent on the oscillation period P_u. Please see the section "Alternative nomenclature and PID forms".

PID 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.

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.[3]

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.

Limitations of PID control

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 feed*back* 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 in particular 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 does this so much, that a physical controller cannot have a true derivative term, but only an approximation with limited bandwidth. 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. So low noise instrumentation can be important. A nonlinear median filter may be used, which improves the filtering efficiency and practical performance.[4] 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 PID algorithm

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

Integral windup

For more details on this topic, see 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:

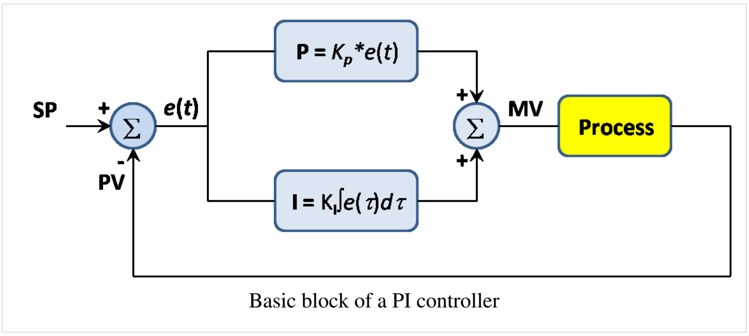
- Increasing the setpoint in a suitable ramp
- Disabling the integration until the PV has entered the controllable region
- Preventing the integral term from accumulating above or below pre-determined bounds
- Back-calculating the integral term to constrain the regulator output within feasible bounds.

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 this 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

$G/\tau = K_I$ = 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 a deadband 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.

Set Point 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:

Set point 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 set point weighting.

Set point weighting

Set point weighting uses different multipliers for 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 set point 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 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.

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 an appropriate integral accumulator term to maintain a consistent process output through parameter changes.

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. 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.Wikipedia:Please clarify

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. In cascade control there are two PIDs arranged with one PID controlling the setpoint of another. 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 provenWikipedia:Citation needed that the working frequency of the controller is increased and the time constant of the object is reduced by using cascaded PID controllers.Wikipedia:Vagueness.

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 PID forms

Ideal versus standard PID 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:

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

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 addition of the proportional and derivative components effectively predicts 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 scaled by the single gain K_p.

In the ideal parallel form, shown in the controller theory section

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

the gain parameters are related to the parameters of the standard form through $K_i = \frac{K_p}{T_i}$ 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 least physical interpretation 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

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 form of a temperature $1/K_p$ which is "degrees per full output". This is the range over which the output changes from 0 to 1 (0% to 100%).

Basing derivative action on PV

In most commercial control systems, derivative action is based on PV rather than error. This is because the digitized version of the algorithm produces a large unwanted spike when the SP is changed. If the SP is constant then changes in PV will be the same as changes in error. Therefore this modification makes no difference to the way the controller responds to process disturbances.

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

Basing proportional action on PV

Most commercial control systems offer the option of also basing the proportional action on PV. This means that only the integral action responds to changes in SP. The modification to the algorithm does not affect the way the controller responds to process disturbances. The change to proportional action on PV eliminates the instant and possibly very large change in output on a fast change in SP. Depending on the process and tuning this may be beneficial to the response to a SP 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[5] describes an effective chart-based method.

Laplace form of the PID controller

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.

PID Pole Zero Cancellation

The PID equation can be written in this form:

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

When this form is used it is easy to determine the closed loop transfer function.

$$H(s) = \frac{1}{s^2 + 2\zeta\omega_0 s + \omega_0^2}$$

If

$$\frac{K_i}{K_d} = \omega_0^2$$

$$\frac{K_p}{K_d} = 2\zeta\omega_0$$

Then

$$G(s)H(s) = \frac{K_d}{s}$$

While this appears to be very useful to remove unstable poles, it is in reality not the case. The closed loop transfer function from disturbance to output still contains the unstable poles.

Series/interacting form

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

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

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, and it made early (analog) controllers easier to build. When the controllers later became digital, many kept using the interacting form.

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.* Approximations for first-order derivatives are made by backward finite differences. The integral term is discretised, 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:

```
previous_error = 0
integral = 0
start:
  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 start
```

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 (or 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.

Notes

[1] The only exception is where the target value is the same as the value obtained when the proportional gain is equal to zero.

[2] 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. http://userweb.eng.gla.ac.uk/yun.li/ga_demo/

[3] Y Li, KH Ang, GCY Chong, Patents, software, and hardware for PID control: An overview and analysis of the current art, Control Systems, IEEE, 26 (1), 42-54. http://eprints.gla.ac.uk/3816/1/IEEE2pdf.pdf

[4] Li, Y. and Ang, K.H. and Chong, G.C.Y. (2006) PID control system analysis and design - Problems, remedies, and future directions. IEEE Control Systems Magazine, 26 (1). pp. 32-41. ISSN 0272-1708 (http://eprints.gla.ac.uk/3815/1/IEEE_CS_PID_01580152.pdf)

[5] King, Myke. *Process Control: A Practical Approach*. Wiley, 2010, p. 52-78

References

- 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 (http://dx.doi.org/10.1111/j.1559-3584.1922.tb04958.x).
- Liptak, Bela (1995). *Instrument Engineers' Handbook: Process Control*. Radnor, Pennsylvania: Chilton Book Company. pp. 20–29. ISBN 0-8019-8242-1.
- Tan, Kok Kiong; Wang Qing-Guo; Hang Chang Chieh (1999). *Advances in PID Control*. London, UK: Springer-Verlag. ISBN 1-85233-138-0.
- 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* (Reed Business Information).
- Sellers, David. "An Overview of Proportional plus Integral plus Derivative Control and Suggestions for Its Successful Application and Implementation" (http://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) on March 7, 2007. Retrieved 2007-05-05.
- Graham, Ron; Mike McHugh (10/03/2005). "FAQ on PID controller tuning" (http://web.archive.org/web/20050206113949/www.tcnj.edu/~rgraham/PID-tuning.html). Mike McHugh. Retrieved 2009-01-05.

External links

- PID tuning using Mathematica (http://reference.wolfram.com/mathematica/ref/PIDTune.html)

PID tutorials

- PID tutorial, free PID tuning tools, advanced PID control schemes, on-line PID simulators (http://www.pidlab.com/)
- 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)
- Online PID Tuning applet from University of Texas Control Group (http://www.che.utexas.edu/course/che360/Process_Tuner.html)
- 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)
- PID Control Primer *Introduction to Closed-Loop Control* (http://www.embedded.com/story/OEG20020726S0044), Embedded Systems Programming, 2002-07-30, retrieved 2013-03-07
- Jinghua Zhong, Mechanical Engineering, Purdue University (Spring 2006). "PID Controller Tuning: A Short Tutorial" (http://saba.kntu.ac.ir/eecd/pcl/download/PIDtutorial.pdf). Retrieved 2013-12-04.

Article Sources and Contributors

PID controller *Source*: http://en.wikipedia.org/w/index.php?oldid=623839514 *Contributors*: 2m man, A. Carty, A.Ou, A3RO, AaronMK, Abarry, Acdx, Ademkader, Aeolian, Ajuneja, Alexius08, Alll, Altzinn, Aluvus, Anaxial, Andrewman327, AnitaS, Anna Lincoln, Applebytom, Attilios, Aturevsk, AutomationBooks, BD2412, Bdijkstra, BeastRHIT, Ben pcc, BenFrantzDale, Benjamander, Bentogoa, Bgwhite, BillyBleach, Billymac00, Bobblewik, Bora Eryilmaz, Bpavel88, BradBeattie, Brews ohare, Briantw, Briceb, Bsodmike, Buesser, CanisRufus, CapitalR, Carmichael, Caslon, Charles Matthews, Chiklit, ChrisGualtieri, ChristianG2, Ciphergoth, Cometstyles, Computationalverb, Control.optimization, Controlexpert, Copeland.James.H, Coredesat, Crinoid, Crohnie, CyberneticSheepCZ, Cyberwriter, DARTH SIDIOUS 2, Dabozz88, Dacera, Dangrayorg, DanielRigal, Dbaechtel, Dedmonds, Dethme0w, Deville, Dhatfield, Dicklyon, Dinhxuanduyet, Dja25, Djgreenfield, Dmcq, Dominick, Drf5n, Dvsphanindra, Dzkd, Ec3243, Ejwong, Elcobbola, Encyclops, Engelec, Epolk, EvergreenFir, FastIkarus, Femto, Fenn fukai, Frank Lofaro Jr., Frappucino, Frze, Fwmchan, Fæ, GB fan, Gabelstaplerfahrer, Gaius Cornelius, GarrettSocling, Gene Nygaard, Giftlite, Glane23, Greatuser, GunnerJr, HK-90, Hankwang, Harriv, HenkvD, Heron, HexaChord, Howie Goodell, Hughjack, Hypergeek14, Icairns, Impsswoon, J.delanoy, JLLCuba, Jbcmf, Jdigangi, Je bonilla, Jeff G., Joel Saks, Jon Mansfield, JonDePlume, JonathonReinhart, Josve05a, Jpmeena, Jrdioko, K6ka, Kanji, Karimnassar, Karthikkumar411, Kbosak, Kerotan, Kku, Kreuzer5, Kvng, KylieTastic, LHOON, Laurentsavaete, Learnfpga, LiDaobing, LieAfterLie, Liu.Yunxia, Liyang, LyonL, Macaddct1984, Magioladitis, Marsian, Masgatotkaca, Mausy5043, Mav, MaykelMoya, Mbbradford, Mbeychok, Mbutts, Md2perpe, Meatmanek, Meestaplu, Michael Hardy, Mikiemike, Miles underwood, Mkratz, Mmeijeri, Mononoke, Motafa, Movedgood, MrOllie, Mrt3366, Mustafa Al-mosawi, N2e, Nave.notnilc, Nbarth, Nigelj, Nitin.mehta, NunoSoares, Nyq, Oddbodz, Oh Snap, Oli Filth, Omegatron, Optikos, Orzetto, Othompson, Panopticon57, Papa November, Patrick, Piano non troppo, Pko, Printer222, ProcessControlGuy, Profr ettore, Pteromys, Pyrilium, Q Science, Ray Van De Walker, Rdautel, Redbeanpaste, Relilles, Requestion, Rich Farmbrough, Robert - Northern VA, Rod57, Rogerwhittakerfan, Ronz, RoyKok, Rspadim, Russvdw, S.borchers, SD5, SMC89, Satellizer, Saxbryn, SchreiberBike, SchreyP, Scs, Seanp, Seaphoto, Sectryan, Sesc, Shadowjams, Shanes, Shimei, Shriramrs31, Signalhead, SilverStar, Sinasgay, Sk wiki, Skorkmaz, Smzoha, Somu2011, Sonett72, Spalding, Spiel496, Spradlig, Stephenb, Steveaa, Strait, Swrkoab, Teoryn, The Anome, TheGrovesy, ThunderStormer, TravTigerEE, Tremilux, Troll1993, Trusilver, Tyvynain, Ulrich67, Unbitwise, Undead warrior, Unregistered.coward, User A1, Van helsing, Varnent, Vary, Vivek.cd, Vizcarra, Wahapure, WakingLili, Warniats, Wbm1058, Wiki13, WikiDao, Wmahan, Wtshymanski, Wywin, Xenodevil, XieChengnuo, Yasirniazkhan, Ywaz, Zephalis, Zerodamage, Zoomzoom1, 852 anonymous edits

Image Sources, Licenses and Contributors

File:PID en updated feedback.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:PID_en_updated_feedback.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:TravTigerEE

File:Scross helmsman.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Scross_helmsman.jpg *License*: Creative Commons Attribution-Sharealike 2.5 *Contributors*: Original uploader was Fishdecoy at en.wikipedia

File:Change with Kp.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Change_with_Kp.png *License*: Public Domain *Contributors*: Skorkmaz

File:Change with Ki.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Change_with_Ki.png *License*: Public Domain *Contributors*: Skorkmaz

File:Change with Kd.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Change_with_Kd.png *License*: Public Domain *Contributors*: Skorkmaz

File:PI controller.png *Source*: http://en.wikipedia.org/w/index.php?title=File:PI_controller.png *License*: Public Domain *Contributors*: Email4mobile (talk)Email4mobile

License