Proportional-Integral-Derivative PID Controls

Dr M.J. Willis

Dept. of Chemical and Process Engineering
University of Newcastle

e-mail: mark.willis@ncl.ac.uk

Written: 17th November, 1998
Updated: 6th October, 1999

Aims and Objectives

The PID algorithm is the most popular feedback controller used within the process industries. It has been successfully used for over 50 years. It is a robust easily understood algorithm that can provide excellent control performance despite the varied dynamic characteristics of process plant. These lecture notes,

- introduce the Proportional- Integral- Derivative (PID) control algorithm.
- discuss the role of the three modes of the algorithm.
- highlight different algorithm structures.
- Discuss methods that have evolved over the last 50 years as aids in control loop tuning.

After completion of this section of the course a student should be capable of approaching a loop tuning problem in a competent and efficient manner and have sufficient knowledge to effectively tune a PID control algorithm.

The Proportional-Integral-Derivative (PID) algorithm

As the name suggests, the PID algorithm consists of three basic modes, the Proportional mode, the Integral and the Derivative modes. When utilising this algorithm it is necessary to decide which modes are to be used (P, I or D ?) and then specify the parameters (or settings) for each mode used. Generally, three basic algorithms are used P, PI or PID.

A Proportional algorithm

The mathematical representation is,

$\frac{mv(s)}{e(s)} = k_c$ (Laplace domain) or $mv(t) = mv_{ss} + k_c e(t)$ (time domain) (3)

The proportional mode adjusts the output signal in direct proportion to the controller input (which is the error signal, e). The adjustable parameter to be specified is the controller gain, k_c. *This is not to be confused with the process gain, k_p.* The larger k_c the more the controller output will change for a given error. For instance, with a gain of 1 an error of 10% of scale will change the controller output by 10% of scale. Many instrument manufacturers use Proportional Band (PB) instead of k_c.[1]

The time domain expression also indicates that the controller requires calibration around the steady-state operating point. This is indicated by the constant term mv_{ss}. This represents the 'steady-state' signal for the mv and is used to ensure that at zero error the cv is at setpoint. In the Laplace domain this term disappears, because of the ‘deviation variable’ representation.

A proportional controller reduces error but does not eliminate it (unless the process has naturally integrating properties), i.e. an offset between the actual and desired value will normally exist.

A proportional integral algorithm

The mathematical representation is,

$$\frac{mv(s)}{e(s)} = k_c\left[1+\frac{1}{T_i s}\right] \text{ or } mv(t) = mv_{ss} + k_c\left[e(t)+\frac{1}{T_i}\int e(t)dt\right] \quad (4)$$

The additional integral mode (often referred to as reset) corrects for any offset (error) that may occur between the desired value (setpoint) and the process

[1] This is defined as the range over which the error must change in order to drive the controller output over full range. The PB also tells you how large the error has to be before the manipulated variable reaches 0 or 100%. The PB is generally centered around the setpoint causing the output to be at 50% when the setpoint and the process output are equal.

output automatically over time[2]. The adjustable parameter to be specified is the integral time (Ti) of the controller.

Where does the term reset come from?

Reset is often used to describe the integral mode. Reset is the time it takes for the integral action to produce the same change in mv as the P modes initial (static) change. Consider the following figure,

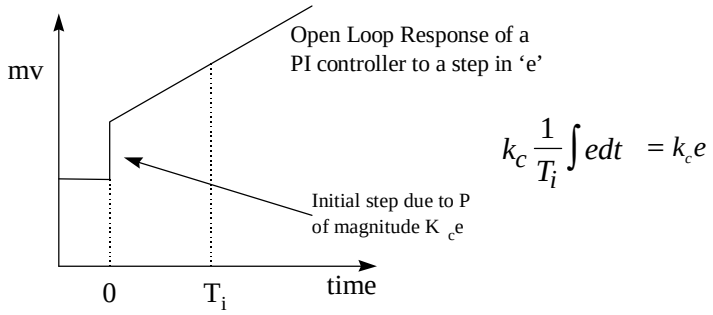

Figure (1) The response of a PI algorithm to a step in error

Figure (1) shows the output that would be obtained from a PI controller given a step change in error. The output immediately steps due to the P mode. The magnitude of the step up is $K_c e$. The integral mode then causes the mv to 'ramp'. Over the period 'time 0 to time T_I' the mv again increases by $K_c e$.

Integral wind-up

When a controller that possesses integral action receives an error signal for significant periods of time the integral term of the controller will increase at a rate governed by the integral time of the controller. This will eventually cause the manipulated variable to reach 100 % (or 0 %) of its scale, i.e. its maximum or minimum limits. This is known as integral wind-up. A sustained error can occur due to a number of scenarios, one of the more common being control system 'override'. Override occurs when another controller takes over control of a particular loop, e.g. because of safety reasons. The original controller is not switched off, so it still receives an error signal, which through time, 'winds-up' the integral component unless something is done to stop this occurring. There are many techniques that may be used to stop this

[2] Different control manufacturers use different definitions for the integral mode of a controller. It can be defined as minutes, minutes/repeat or repeats per minute. The difference is very important to note so as to ensure problems do not occur during a tuning exercise. *Remember the 'name game'*. T_i is the integral time (minutes), if specified as repeats / minute then it is $1/T_i$ that must be entered into the controller, while minutes / repeat is again T_i. This is confusing and is compounded by the fact that manufacturers are not consistent !

happening. One method is known as ‘external reset feedback’ (Luyben, 1990). Here, the signal of the control valve is also sent to the controller. The controller possess logic that enables it to integrate the error when its signal is going to the control value, but breaks the loop if the override controller is manipulating the valve.

A Proportional Integral Derivative algorithm

The mathematical representation is,

$$\frac{mv(s)}{e(s)} = k_c\left[1+\frac{1}{T_i s}+T_D s\right] \text{ or } mv(t) = mv_{ss} + k_c\left[e(t)+\frac{1}{T_i}\int e(t)dt + T_D\frac{de(t)}{dt}\right] \qquad (5)$$

Derivative action (also called rate or pre-act) *anticipates* where the process is heading by looking at the time rate of change of the controlled variable (its derivative). T_D is the ‘rate time’ and this characterises the derivative action (with units of minutes). In theory derivative action should always improve dynamic response and it does in many loops. In others, however, the problem of noisy signals makes the use of derivative action undesirable (differentiating noisy signals can translate into excessive mv movement).

Derivative action depends on the slope of the error, unlike P and I. If the error is constant derivative action has no effect.

Revision Exercise

Use Matlab / Simulink to explore the effect a step change in error has on the various modes of an ideal PID control algorithm. Assume that k_c = 1, T_i = 10 mins and T_D = 5mins.

PID algorithms can be different

Not all manufactures produce PID’s that conform to the ideal 'textbook' structure. So before commencing tuning it is important to know the configuration of the PID algorithm! The majority of ‘text-book’ tuning rules are only valid for the ideal architecture. If the algorithm is different then the controller parameters suggested by a particular tuning methodology will have to be altered.

Ideal PID

The mathematical representation of this algorithm is:

$$\frac{mv(s)}{e(s)} = k_c\left[1+\frac{1}{T_i s}+T_D s\right]$$

One disadvantage of this ideal 'textbook' configuration is that a sudden change in setpoint (and hence e) will cause the derivative term to become very large and thus provide a “derivative kick” to the final control element - this is undesirable. An alternative implementation is

$$mv(s) = k_c\left[1+\frac{1}{T_i s}\right]e(s) + T_D s cv(s)$$

The derivative mode acts on the measurement and not the error. After a change in setpoint the output will move slowly avoiding "derivative kick" after setpoint changes. This is therefore a standard feature of most commercial controllers.

Series (interacting) PID

The mathematical representation of this algorithm is:

$$\frac{mv(s)}{e(s)} = k_c\left[1+\frac{1}{T_i s}\right]T_D s$$

As with the ideal implementation the series mode can include either derivative on the error or derivative on the measurement. In which case, the mathematical representation is,

$$\frac{mv(s)}{e(s)} = k_c\left[1+\frac{1}{T_i s}\right] \quad where\ e(s) = SP - T_D s cv(s)$$

Parallel PID

The mathematical description is,

$$mv(s) = k_c e(s) + \frac{1}{T_i s}e(s) + T_D s e(s)$$

The proportional gain only acts on the error, whereas with the ideal algorithm it acts on the integral and derivative modes as well.

Revision Exercises

1. Draw the block diagram representation of the ideal, series (interacting) and parallel PID control laws.
2. Write down the 'time-domain' mathematical representation of the ideal (without derivative kick) , series (interacting) and parallel PID control laws.

3. Suppose that the controller settings for an ideal PID algorithm are given by, k_c, T_i, T_D. Work out the conversion factors required to ensure that a parallel implementation of the PID algorithm will provide the same mv signal given the same error signal.

Controller tuning

Controller tuning involves the selection of the best values of k_c, Ti and T_D (if a PID algorithm is being used). This is often a subjective procedure and is certainly process dependent. A number of methods have been proposed in the literature over the last 50 years. However, recent surveys indicate,

- 30 % of installed controllers operate in manual.
- 30 % of loops increase variability.
- 25 % of loops use default settings.
- 30 % of loops have equipment problems.

A possible explanation for this is lack of understanding of process dynamics, lack of understanding of the PID algorithm or lack of knowledge regarding effective tuning procedures. This section of the notes concentrates on PID tuning procedures. The suggestion being that if a PID can be properly tuned there is much scope to improve the operational performance of chemical process plant.

When tuning a PID algorithm, generally the aim is to match some preconceived 'ideal' response profile for the closed loop system. The following response profiles are typical.

Servo Control

For a unit step change in setpoint (0 - 1) the two response profiles shown in figure 2 could be obtained (depending upon the process dynamics and controller settings),

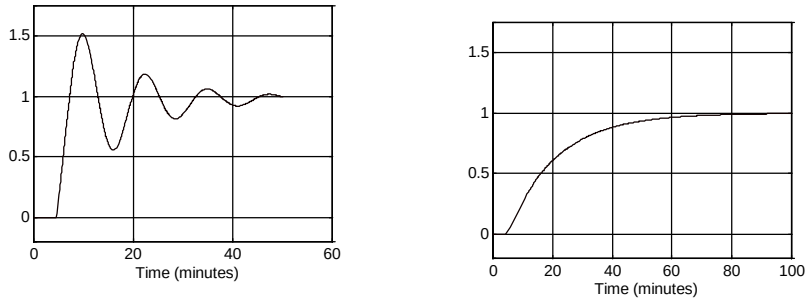

Figure (2) Underdamped (LHS) and overdamped (RHS) system response to a unit change in setpoint (PI control).

Terms used to describe underdamped response characteristics are,

- **Overshoot:** this is the magnitude by which the controlled variable 'swings' past the setpoint. 5/10% overshoot is normally acceptable for most loops.
- **Rise time:** the time it takes for the process output to achieve the new desired value. One-third the dominant process time constant would be typical.
- **Decay ratio:** this is the ratio of the maximum amplitude of successive oscillations.
- **Settling time:** the time it takes for the process output to die to between, say +/- 5% of setpoint.

These characteristics are often used as objectives during a tuning exercise.

Regulatory Control

For a unit step change in the dv, the following type of response profile may be desired,

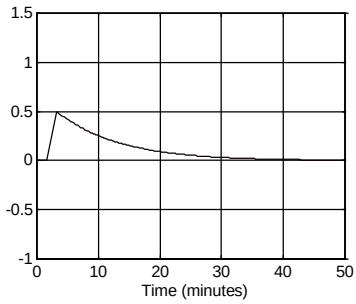

Figure (3) Disturbance rejection (a typical response profile)

i.e. the disturbance initially causes the process to move away from the desired value (which is set to zero in this figure). The controller then adjusts the mv so that the cv slowly moves back to setpoint. In other words the impact that the disturbance has on the closed loop system is eliminated and the system returns to the desired value. A transfer function that could be used to model this behaviour is,

$$\frac{cv(s)}{dv(s)} = \frac{\lambda s}{\lambda s + 1} \qquad (6)$$

where the constant λ models the ‘peak’ effect of the disturbance as well as the speed at which the system returns to steady-state.

Tuning Rules

Rules of thumb

The following rules of thumb are intended to give “ball-park” figure controller settings. The settings[1] assume a series algorithm, the others are for ideal PID

Loop Type	PB(%)	I (mins)	D (mins)
Liquid level	**< 100**	**10**	**-**
Temperature	**20 - 60**	**2 - 15**	**I/4**
Flow	**150**	**0.1**	**-**
Liquid Pressure[1]	**50 - 500**	**0.005 - 0.5**	**-**
Gas Pressure[1]	**1- 50**	**0.1 - 50**	**0.02 - 0.1**
Chromatograph[1]	**100 - 2000**	**10 - 120**	**0.1 - 20**

Often, with level systems exact setpoint following is not essential, hence proportional control is often used. Temperature loop dynamics can be slow because of process heat transfer lags. Deadtime is possible, especially in heat exchangers and temperature is not normally noisy. Consequently PID control is normally preferred. Flow loop dynamics are generally fast (of the order of seconds). Control valve dynamics are normally the slowest in the loop. Flow systems are noisy. However, noise can often be dealt with simply by reducing the gain.

Ziegler Nichols closed loop method

The method is straightforward. First, set the controller to P mode only. Next, set the gain of the controller (k_c) to a small value. Make a small setpoint (or load) change and observe the response of the controlled variable. If k_C is low the response should be sluggish. Increase k_C by a factor of two and make another small change in the setpoint or the load. Keep increasing k_C (by a factor of two) until the response becomes oscillatory. Finally, adjust k_C until a response is obtained that produces continuous oscillations. This is known as the ultimate gain (ku). Note the period of the oscillations (Pu). The control law settings are then obtained from the following table,

	k_C	T_i	T_D
P	ku/2		
PI	Ku/2.2	Pu/1.2	
PID	Ku/1.7	Pu/2	Pu/8

Practical use of the technique

It is unwise to force the system into a situation where there are continuous oscillations as this represents the limit at which the feedback system is stable. Generally, it is a good idea to stop at the point where some oscillation has been obtained. It is then possible to approximate the period (Pu) and if the gain at this point is taken as the ultimate gain (ku) , then this will provide a more conservative tuning regime.

Cohen - Coon

This method depends upon the identification of a suitable process model (plant identification has been covered in previous lectures). Cohen-Coon recommended the following settings to give responses having ¼ decay ratios, minimum offset and other favourable properties,

	k_c	T_i	T_D
P	$\frac{1}{k_p}\frac{\tau}{\theta}(1+\frac{\theta}{3\tau})$		
PI	$\frac{1}{k_p}\frac{\tau}{\theta}(\frac{9}{10}+\frac{\theta}{12\tau})$	$\theta\frac{30+3(\theta/\tau)}{9+20(\theta/\tau)}$	
PID	$\frac{1}{k_p}\frac{\tau}{\theta}(\frac{4}{3}+\frac{\theta}{4\tau})$	$\theta\frac{32+6(\theta/\tau)}{13+8(\theta/\tau)}$	$\theta\frac{4}{11+2(\theta/\tau)}$

In the table k_p is the process gain, τ the process time constant and θ the process time delay.

Practical use of the technique

If the process delay is small (in the limit as it approaches zero) increasingly large controller gains will be predicted. The method is therefore not suitable for systems where there is zero or virtually no time delay.

Direct synthesis

This is a model based tuning technique. It uses an identified process model in conjunction with a user specified closed loop response characteristic. An advantage of this approach is that it provides insight into the role of the 'model' in control system design. A disadvantage of the approach is that a PID controller may not be realised unless an appropriate model form is used to synthesise the control law.

Tuning for servo control

Let the symbol Gp represent the process dynamics and Gc the controller dynamics. If all other dynamic elements within the loop are ignored then the following closed loop transfer function can be derived,

$$\frac{cv}{SP} = \frac{G_c G_p}{1 + G_c G_p} \tag{6}$$

this can be re-arranged to give an expression for the feedback control law as,

$$G_c = \frac{1}{G_p}\left(\frac{\frac{cv}{SP}}{1-\frac{cv}{SP}}\right) \tag{7}$$

In other words, the controller comprises the inverse of the process model (common to model based design techniques) as well as a specification for the closed loop response characteristic, cv/SP.

A process model can be obtained through plant identification. The closed loop response characteristic, cv/SP must be specified. A simple specification is,

$$\frac{cv}{SP} = \frac{1}{\lambda s + 1} \tag{8}$$

λ is a user specified closed loop time constant.

Substituting this into equation (7) and re-arranging gives,

` $$G_c = \frac{1}{G_p}\left(\frac{1}{\lambda s}\right) = \frac{\tau_p s + 1}{k_p \lambda s} = \frac{\tau_p}{k_p \lambda}\left(1 + \frac{1}{\tau_p s}\right) \tag{9}$$

where it has been assumed that the process transfer function is,

$$G_p(s) = \frac{k_p}{\tau_p s + 1} \tag{10}$$

ie. first order, no dead-time.

Based on this process description, the ideal form of a PI controller results, where,

$$k_c = \frac{\tau_p}{k_p \lambda} \text{ and } T_i = \tau_p \tag{11}$$

What do you do if you want derivative action? The first order model results in a control law that is of the PI type. If you wish to synthesis a PID controller, there are two options

- choose T_D = Ti/4

- model the process using a 2nd order transfer function.

Revision Exercise

Starting with a second order process transfer function show that a PID control structure can be developed using the direct synthesis derivation technique. What are the settings of the PID controller (in terms of the coefficients of the second order process transfer function) ?

Systems with time delay

Throughout this course, our basic assumption has been that we can model systems using the following transfer function,

$$G_p(s) = \frac{k_p e^{-s\theta}}{\tau_p s + 1} \qquad (12)$$

i.e. a first order plus dead-time transfer function. If this were the case, what type of control law would result using the direct synthesis procedure?

Following the derivation presented, the following control law results**,**

$$G_c = \frac{1}{G_p}\left(\frac{e^{-s\theta}}{\lambda s + 1 - e^{-s\vartheta}}\right) \qquad (13)$$

Note that the following response specification was used (as the time delay cannot be removed from the process),

$$\frac{cv}{SP} = \frac{e^{-s\theta}}{\lambda s + 1} \qquad (14)$$

The control law, equation (13) is of non-standard form because of the time-delay terms. Suppose that $e^{-s\theta}$ is approximated by a 1st order Taylor series expansion, i.e.

$$e^{-s\theta} \approx 1 - \theta s$$

Substituting into the denominator of equation (13) and re-arranging gives,

$$G_c = \frac{1}{G_p}\left(\frac{e^{-s\theta}}{(\lambda + \theta)s}\right) \qquad (15)$$

It is not necessary to approximate the time delay in the numerator of equation (13) as this is cancelled by an identical term in the process transfer function, $G_p(s)$ giving,

$$G_c = \frac{\tau_p s + 1}{k_p(\lambda + \theta)s} = \frac{\tau_p}{k_p(\lambda + \theta)}\left(1 + \frac{1}{\tau_p s}\right) \qquad (16)$$

which is the form of an ideal PI controller where,

$$k_c = \frac{\tau_p}{k_p(\theta + \lambda)} \text{ and } T_i = \tau_p \quad (17)$$

Note the intuitive nature of the controller gain calculation: as the process time delay increases the controller gain will decrease.

Tuning for regulatory control

With reference to the closed loop block diagram, for regulatory control the following closed loop transfer function may be derived,

$$\frac{cv}{dv} = \frac{1}{1 + G_c G_p} \quad (18)$$

This closed loop expression can be re-arranged to give an expression for the feedback control law as,

$$G_c = \frac{1}{G_p}\left(\frac{1 - \frac{Y}{d}}{\frac{Y}{d}}\right) \quad (19)$$

Again, the controller consists of the inverse of the process model as well as a specification for the closed loop response characteristic, cv/dv.

The process model is obtained through plant identification however, the closed loop response characteristic, cv/dv, must be specified by the designer. Using the simple specification described earlier,

$$\frac{cv(s)}{dv(s)} = \frac{\lambda s}{\lambda s + 1} \quad (20)$$

where λ is user specified. Substituting this into equation (19) and re-arranging gives,

$$G_c = \frac{1}{G_p}\left(\frac{1}{\lambda s}\right) \quad (21)$$

This is exactly the same form as equation (9) for servo control. Hence the controller gain and integral term for a PI controller is given by,

$$k_c = \frac{\tau_p}{k_p \lambda} \text{ and } T_i = \tau_p \quad (22)$$

Final Remarks

The notes have reviewed PID control, discussed the modes of the various control algorithms, the different structures of algorithms that exist and standard tuning rules. The tuning rules reviewed include, Ziegler-Nichols, Cohen- Coon, and direct synthesis. Remember:

- the tuning rules are only valid for the 'ideal' PID control structure and any prediction of control law settings should be adjusted if an alternative PID implementation is used.
- the tuning rules are only valid for self-regulating processes (i.e open loop stable processes such as those that may be described by the 1^{st} order plus dead-time description).

Luckily most process systems are self-regulating the exception to the rule being level systems. Tuning of level controllers will be the subject of the next section of the notes.