Transactions of the Institute of Measurement and Control
http://tim.sagepub.com/

PID algorithms and their computer implementation

D.W. Clarke

Transactions of the Institute of Measurement and Control 1984 6: 305
DOI: 10.1177/014233128400600605

The online version of this article can be found at:
http://tim.sagepub.com/content/6/6/305

Published by:
SAGE
http://www.sagepublications.com

On behalf of:
Institute of Measurement and Control
The Institute of Measurement and Control

Additional services and information for *Transactions of the Institute of Measurement and Control* can be found at:

Email Alerts: http://tim.sagepub.com/cgi/alerts

Subscriptions: http://tim.sagepub.com/subscriptions

Reprints: http://www.sagepub.com/journalsReprints.nav

Permissions: http://www.sagepub.com/journalsPermissions.nav

Citations: http://tim.sagepub.com/content/6/6/305.refs.html

>> Version of Record - Oct 1, 1984

What is This?

Downloaded from tim.sagepub.com at Scientific library of Moscow State University on February 18, 2014

PID algorithms and their computer implementation

by D. W. Clarke*, MA, DPhil, CEng, FIEE

Three-term or PID regulators are deservedly popular in industrial control systems. They give satisfactory performance for a wide class of processes, they are readily understood by technical plant personnel, and they are easy to implement using analogue or digital hardware. Stable elimination of offset due to load-disturbances is a principal control requirement, as in many cases processes are operated continuously at a fixed set-point. Often this is achieved by the PI terms alone, though derivative action can be added for plant with excessive phase lag. In practice, however, nonlinear behaviour such as actuator saturation is a critical component of overall performance.

This report discusses various forms of the PID regulator (positional, incremental, cascade, etc) and their closed-loop properties. Practical features such as avoidance of 'integral windup' and 'derivative kick' are described. The ways in which the PID algorithm can be discretised for use in a digital computer are outlined, together with appropriate Fortran coding. The effect of choice of sample interval on the discretised performance is analysed. Often the interval is chosen to be as short as possible given the computational power available; the influence of this on quantisation noise and offset is discussed.

Various filters are used in a discrete PID algorithm, including the anti-aliasing filter and a filter on derivative action. The properties of these filters are investigated and some interesting nonlinear filters, such as 'spike' filters are presented.

1 Introduction

The classical PID algorithm has the following form

$$u(t) = K\left\{e(t) + \frac{1}{T_i}\int e(t)\,\mathrm{d}t + T_d\frac{\mathrm{d}e}{\mathrm{d}t}(t)\right\} \qquad \ldots(1)$$

where

$u(t)$ is the control action or manipulated variable at time t,
$e(t)$ is the error (or deviation) $w(t) - y(t)$: the difference between the setpoint $w(t)$ and the measured output (or measured variable),
K is the gain and
T_i, T_d are the integral and derivative times, respectively.

In Laplace transforms, the controller (1) is

$$u(s) = K\left\{1 + \frac{1}{sT_i} + sT_d\right\}e(s) \qquad \ldots(2)$$

Practical actuators and measuring devices have limits on the allowable range of values and often the corresponding variables are scaled in the range 0–100%. For example, a valve modulating a flow rate is nominated to have a value of 0% when closed and 100% when fully open. The limits generally correspond to 3–15 lbf/in^{2}g for pneumatic systems or 4–20 mA for electrical actuators or transducers, these being industry-standard values. Scaling of variables in this way has several advantages: the appropriate gain K is easier to choose, the PID algorithm can be designed to have sensible numerical properties, and operators find the displayed variables more understandable.

With these limits in mind, chemical engineers often use the term 'proportional band' for $1/K$, such that 100% PB gives 100% control action for 100% error and 50% PB gives 100% action for 50% error, etc. The integral time T_i is sometimes called the 'reset time', being the time in which the integral component of $u(t)$ reaches 100% for a constant 100% error.

1.1 Elementary properties of PID regulators

Consider the plant modelled by Fig 1 in which $G_1(s)$, $G_2(s)$ are forward-path transfer functions and $\Gamma(s)$ is an unmeasurable load disturbance. The objective of the controller $C(s)$ is to maintain the average value $\bar{y}$ of the output $y(t)$ equal to the setpoint w despite possible step changes in $\Gamma(t)$.

Solving for the closed loop gives:

$$y(s) = \frac{CG_1G_2}{1 + CG_1G_2}w(s) - \frac{G_2}{1 + CG_1G_2}\Gamma(s) \qquad \ldots(3)$$

For the case where w and Γ are constants W and L, it is seen that a proportional control law $C(s) = K$ will produce a constant output:

$$Y = \frac{KK_p}{1 + KK_p}W - \frac{K_2}{1 + KK_p}L$$

where $K_1 = G_1(0)$, $K_2 = G_2(0)$, and $K_p = K_1K_2$ is the plant gain. Hence with a proportional controller, the mean output Y is not equal to W unless KK_p is unacceptably large; this is the well known offset problem.

The offset is eliminated if either $G_1(0)$ or $C(0) = \infty$; this is achieved if the transfer function contains at least one integrator. As it is rare to have integrating action in G_1 (more commonly in G_2), the control requirement implies that $C(s)$ must itself contain an integrator. Hence $C(s) = 1/sT_i$ removes offset in a time proportional to T_i: the smaller T_i is, the greater is the potential speed. Unfortunately, the use of integral control alone will frequently lead to closed-loop instability, as the controller adds 90° of phase lag to the forward path. However, the combination of proportional and integral terms in the form:

$$u(s) = K\left(\frac{1 + sT_i}{sT_i}\right) \qquad \ldots(4)$$

will generally give a stable closed loop provided that T_i is appropriately chosen. Note that the frequency response of

* Department of Engineering Science, University of Oxford.

Downloaded from tim.sagepub.com at Scientific library of Moscow State University on February 18, 2014

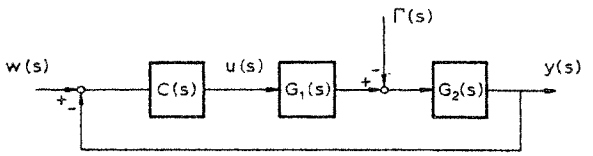


Fig 1 Control loop with a load disturbance

Eqn (4) (putting $s = j\omega$) behaves as K/sT_i for low frequencies but as K for high frequencies where $\omega T_i \gg 1$. Hence for large-enough T_i, a closed loop which is stable under proportional gain K is also stable for the full PI controller. For example, it is easily shown that a plant $G(s) = K_p/s(1 + sT_p)$ – such as a speed-control system or a coupled-tank process – is stably controlled by $C(s)$ given by Eqn (4) provided that $T_i > T_p$.

Derivative action can be used where there is excessive phase lag in the controlled process (eg, with order $\geqslant 3$ or with significant deadtime). In practice it is much less employed than the other terms, probably because it is difficult to tune for the potential extra performance or because measurement or plant noise is undesirably amplified at high frequencies. In cases where overshoot is to be avoided, as with batch chemical reactors whose exothermic behaviour is strongly dependent on temperature, the derivative term gives useful 'anticipation' such that the control signal becomes negative (ie, producing cooling) *before* reaching the setpoint.

With all three terms, the PID regulator can be written:

$$C(s) = K(1 + sT_i + s^2 T_i T_d)/sT_i \qquad \ldots(5)$$

This provides one controller pole at the origin (for offset removal) and two zeros given by the roots of:

$$s^2 + s/T_i + 1/T_i T_d = 0 \qquad \ldots(6)$$

The choice of T_i and T_d can place these zeros at the designer's discretion, so that if an analytic model of the process is available, root-locus methods could be used to obtain satisfactory performance. Typically one zero would be used to cancel a dominant process pole. As most chemical processes have real poles, there is no general reason to allow the roots of Eqn (6) to be complex. However, mechanical systems often exhibit underdamped complex poles, and a good design strategy is to attempt to cancel these poles with controller zeros so that the undesirable mode is eliminated from the loop.

1.2 Incremental (velocity) algorithms

Sometimes the integration in a PID algorithm is achieved externally to the main controller. For example, a valve positioned by a stepper motor *changes* its position according to the number of steps demanded. For non-critical processes this increases availability, as failure of the controller means that the valve stem maintains its last position and thus allows the plant to remain operational. This is dependent, of course, on the type of plant; in general, controller failure should result in valves opening or closing according to predetermined safety requirements.

Suppose that a stepper motor requires N steps to traverse its complete range ($u(t) = 0 \ldots 100\%$). The action of the main controller is to request a *rate* $u_1(t)$ of r steps per second. Hence in a time interval Δt a number of steps $r\Delta t$ steps are demanded, giving a change (or *increment*) ΔS in the total number of steps S:

$$\Delta S = r\Delta t, \quad \text{or} \quad S(t) = \Sigma r \Delta t$$

Suppose r_{max} is the maximum stepping rate, corresponding to a 100% demand $u_1(t)$. Now as $S_{max} = N$ corresponds to the maximum control $u(t)$ of 100%, then

$$u = \frac{S}{N} \cdot 100\%, \quad \text{and} \quad u_1 = \frac{r}{r_{max}} \cdot 100\%$$

So:

$$\frac{Nu(t)}{100\%} = \sum \frac{u_1(t) r_{max}}{100\%} \Delta t$$

In the limit, as $\Delta t \to 0$ this gives:

$$u(t) = \int \frac{r_{max}}{N} u_1(t)\, dt, \quad \text{or} \quad u(s) = \frac{1}{sN/r_{max}} u_1(s) \qquad \ldots(7)$$

This corresponds to an integrator with integral time N/r_{max}.

As the function of the *overall* control action between $e(t)$ and $u(t)$ is to be PID, the main controller must have a transfer function

$$u_1(s) = K'(1 + sT_i + s^2 T_i T_d)\, e(t)$$

as shown in Fig 2.

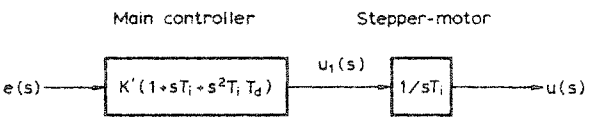


Fig 2 The incremental form of a PID controller

Note that, even though the stepper motor appears to have its own integral time of T_i', an arbitrary integral time of the overall PID controller can be chosen simply by equating:

$$K'(1 + sT_i + s^2 T_i T_d)/sT_i = K(1 + sT_i + s^2 T_i T_d)/sT_i$$

If the stepper motor had infinitely fine resolution and fast stepping rate, the performance of the incremental algorithm would be the same as the usual positional form. However, the effect of the stepper motor is to quantise the allowable levels of $u(t)$, possibly causing 'dither' (hence choose large N) and to limit the slew-rate of the output to a maximum of $(r_{max}/N) \cdot 100\%$. This may be acceptable as rate-limiting is often used in awkward plant; otherwise r_{max} should be large, for rate limiting imposes a phase lag if a limit is attained. The algorithm is, in general, sensitive to noise and should use filtering if derivative action is required.

1.3 Cascade and feedforward control

Cascade control is used to linearise the effect of an actuator or to control some intermediate variable so that the overall control loop is easier to handle. A simple example of the first application is a valve-step positioner which overcomes the inherent stiction or backlash of the valve. The second use is in, say, controlling product composition in a distillation column. A selected tray temperature is used as an intermediate variable because the primary output variable is measured typically by a chromatograph with significant deadtime. Not only does this allow for tighter control but disturbances can be detected more

Downloaded from tim.sagepub.com at Scientific library of Moscow State University on February 18, 2014

rapidly and regulated using the inner loop. In these cases PID regulators can be used in both loops, but more often simple proportional control is sufficient for the inner loop. If both loop controllers have integral action, special precautions need to be adopted in their desaturation (qv), particularly if incremental algorithms are used.

Feedforward control can be used if there is some measurable disturbance $v(t)$ acting on the process, such as variations in the quality of feed input to a distillation column. A signal which (partly) cancels its effect can be added to the control action $u(t)$, typically being $K_f v(t)$ where K_f needs to be tuned for the best effect. Considerable improvements in control quality can be achieved using feedforward, though the appropriate value of K_f for this is often difficult to find and may vary with time. Note that integral action must *not* be applied to $v(t)$; if an incremental algorithm is used in the main controller, the *derivative* $K_f \cdot dv/dt$ must be added to $u_1(t)$ rather than the full-value $K_f v(t)$.

2 Practical considerations

When a PID regulator is to be applied to an industrial process there are several important issues which must be resolved:

1. Choice of 'positional' or 'incremental' form; this depends on the type of actuator used and whether a constant control $u(t)$ is desirable on controller failure.
2. Provision for manual/automatic operation with 'bumpless transfer' when switching over from an operator chosen value of $u(t)$ to initiate PID control. This involves ensuring that the initial PID control signal would not differ significantly from the previous manual setting, it being presumed that the operator has chosen this sensibly.
3. Avoidance of 'derivative kick' and possibly 'proportional kick' which would otherwise occur on step changes in the setpoint $w(t)$. This sudden large control action might tend to drive the plant into a non-linear region. To avoid derivative kick, for example, derivative action should only be on the output $y(t)$, not the error, giving:

   $$u(s) = K\ (1 + 1/sT_i)\, e(s) - sT_d y(s)$$

   Similarly, eliminating proportional kick has the corresponding term acting on y, with only the integral term acting on the error. This does not affect the stability or disturbance–reduction property of the PID algorithm, but might slow the response to step changes in demanded $w(t)$.
4. Measurement conditioning/filtering. For example, a differential-pressure flow transducer gives a signal proportional to (flow)^2, which would need to be rooted to maintain linearity in the loop. Similarly, electrical heating in which the generated heat depends on u^2 might need linearisation of the output. Filtering of noise and derivative action is a complex issue treated later in detail.
5. Output signal amplitude or rate limiting. Actuators have inherent amplitude limits; rate limits are sometimes required on start-up of sensitive plant.
6. Use of 'local' (internal) or 'remote' setpoint. If, for example, the controller is part of the inner loop in a cascade configuration, its setpoint is chosen by the outer loop rather than by the operator.
7. Whether one or more feedforward signals are to be added, and the extent of the dynamics allowable in the compensation (eg, $P + D$).
8. Avoidance of integral 'windup' by using integral *desaturation*. If, typically during start-up, the error signal is large for a long time, then $\int e(t)\, dt$ itself becomes large. Thus, even when $y(t)$ has attained the set-point $w(t)$, the control action would remain saturated until the integral term has 'unwound'. This invokes a sustained and highly undesirable overshoot. Desaturation can be achieved by limiting the integral term to some value, though which particular value should be chosen is debatable. Typically 100% is taken, using the argument that control action should start to drop from its limit when the error $e(t) = w(t) - y(t)$ changes sign. This still involves some overshoot (unless derivative action is also included); Shinskey (1979) provides a useful guide to alternative strategies if overshoot is to be avoided.

3 A simple PI and PID configuration

A particularly elegant implementation of the PI regulator which has many of the attributes described in the previous section consists basically of a simple lag in positive feedback form as shown in Fig 3.

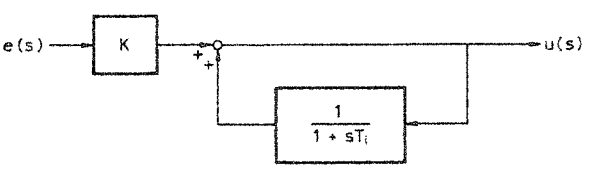


Fig 3 A lag in feedback ≡ PI

Here

$u(s) = Ke(s) + u(s)/(1 + sT_i)$,

or:

$u(s) = K\ \{(1 + sT_i)/sT_i\}\ e(s)$ — in PI form.

The power of the configuration is seen best when including output limits: $u_{min} < u(t) < u_{max}$, an auto/manual switch, and derivative action, as shown in Fig 4.

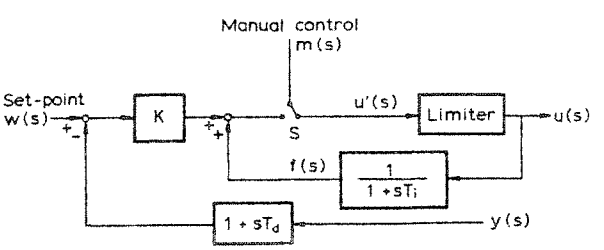


Fig 4 Practical PID block diagram

It is seen that the set-up has the following properties:

(1) When the demand $u'(t)$ is within the linear range and not limited, then $u = u'$ so the above analysis is valid and the control is given by:

$$u(s) = K\ [(1 + sT_i)/sT_i\ \{w(s) - (1 + sT_d)\, y(s)\}\ ]$$

This is PI action on the error $e(s) = w(s) - y(s)$, but the total action on y is

$$u(s) = -K\ [(1 + 1/sT_i)(1 + sT_d)]\, y(s) \qquad \ldots(8)$$

This, therefore, avoids derivative kick but is not quite in the

Downloaded from tim.sagepub.com at Scientific library of Moscow State University on February 18, 2014

classical form of a PID regulator given by Eqn (1). However, the algorithm gives approximately the same behaviour for small values of T_d. In some ways the above factored form is easier to tune as the zeros are given by $-1/T_i$ and $-1/T_d$; as such it is employed by some leading users of PID regulators. Of course it does not allow for complex zeros, and technical staff familiar with tuning classical PID coefficients may have some initial difficulty with it.

(2) When the demand $u'(t)$ is outside the allowable limits, the control $u(t)$ is set to the corresponding clipped value. If sustained for some time (eg about 3–5 time constants T_i) the signal $f(t)$ reaches the limited value of $u(t)$. When the output of the gain element K changes sign (ie, when $e(t)$ changes sign if $T_d = 0$) $u'(t)$ immediately drops from the limited value and hence $u(t)$ also drops away – the desired integral desaturation property. This is seen to be even more appropriate when $f(t)$ has not reached a steady level, for note that $f(t)$ always lies *within* the bounds of $u(t)$.

(3) The switch S is the manual/auto switch. If connected to $m(t)$ the signal is still limited (as a check on the operator), and if $m(t)$ is constant for an interval $f(t)$ will attain its value (the lag in the feedback loop having unity gain). A good operator will switch to automatic when $e(t)$ is approximately zero and $y(t)$ is sensibly constant; in such a case there will be bumpless transfer.

The controller of Eqn (8) is called *interacting* as adjusting the values of T_i and T_d modify the effective integral and derivative times as defined by Eqn (1). The transfer function of Eqn (8) can be written

$$K\{1 + s(T_i + T_d) + s^2 T_i T_d\}/sT_i$$

which, when compared with the conventional form of Eqn (1):

$$K'\{1 + sT_i' + s^2 T_i' T_d'\}/sT_i'$$

gives

actual integral time $= T_i' = T_i + T_d$

actual differential time $= T_d' = (T_i T_d)/(T_i + T_d)$

actual gain factor $= K' = K(1 + T_d/T_i)$

It can be shown that the maximum effective T_d' is obtained when $T_d = T_i$, whereupon:

$$T_i' = 2T_i, \quad T_d' = T_d/2 = T_i'/4, \quad \text{and} \quad K' = 2K.$$

In most cases a maximum differential time of ¼ the integral time is sufficient except for the most exacting control requirements.

When used to control a simple lag with deadtime, Shinskey (1979) shows that a noninteracting form of PID can produce as little as ½ the integral square error following a step in load disturbance compared with the above interacting form. The corresponding controller is, though, more sensitive to an incorrectly modelled plant. If a noninteracting form is required, the structure shown in Fig 5 can be used.

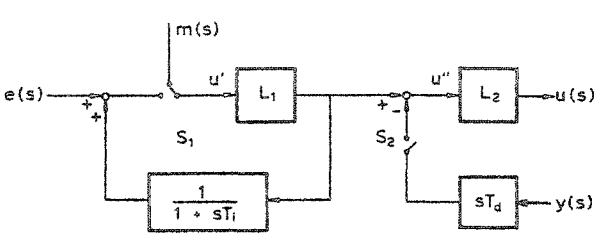


Fig 5 Noninteracting PID implementation

Here L_1 and L_2 are limiters: L_1 to provide the desaturation property for the integral term in the first loop and L_2 to limit the final control value. L_2 could be omitted if the desired limits correspond to the practical actuator limits. Switches S_1 and S_2 are for auto/manual transfer.

4 Discretising the PID algorithm

4.1 The effect of sampling

When implementing a PID controller in a digital computer a sequence $\{\ldots y(t), y(t+1)\ldots\}$ of measured outputs is available at sample intervals h, and a corresponding sequence of controls $\{\ldots u(t), u(t+1)\ldots\}$ is generated. The measurements are not only corrupted by the usual plant noise but there is also noise generated by the quantisation levels of the A–D converter. Similarly, the control signal to the plant is quantised by the D–A converter. Typically 12 bits (1 part in 4096) is specified for and A–D converter and 10 bits (1 part in 1024) is used in the D–A converter, though low-cost microcomputer systems may employ only 8 bits in both. The reason for the lower precision generally used in D–A converters is that the low-pass filtering nature of most processes removes the generated noise. However quantisation in A–D converters can produce problems, particularly in the derivative component of PID controllers, and these will be discussed in detail later.

When using a discrete-time approximation to a continuous-time controller, the sample interval h is an important parameter. In principle, the interval should be chosen to be as short as possible, but often numerical difficulties due to round-off may then occur (particularly with fixed-point implementations). When controlling many loops there can be an unacceptable computational burden using a small value of h, though increasing computer power and the tendency to decentralise loops makes this less of a problem nowadays. Even so, it is important when considering discretisation that the properties of the continuous-time controller should be reflected reasonably accurately by the derived algorithm, even for relatively large values of h. Some approaches are better than others in this respect.

In deriving discrete-time controllers the Z-transform is used, though in the following z^{-1} is treated mainly as the backward shift operator, defined as $z^{-1}x(t) = x(t-1)$ etc. In the same way s can be considered to be the differential operator and $1/s$ to be the integral operator. Approaches to discretisation can then be presented in terms of numerical methods for integrating a signal $x(t)$ in terms of its sampled values.

1 Rectangular integration

Let $I(t-1)$ be the value of the approximately integrated signal at the previous sample instant and $x(t)$ the new input value. Rectangular integration assumes that x has remained constant over the sample interval, giving

$$I(t) = I(t-1) + h \cdot x(t)$$

ie,

$$(1 - z^{-1})I(t) = h \cdot x(t)$$

Comparing this with the continuous-time integral operator

Downloaded from tim.sagepub.com at Scientific library of Moscow State University on February 18, 2014

$I(s) = x(t)/s$ gives the correspondence (the *linear* transform):

$$s \approx (1 - z^{-1})/h, \quad \text{or} \quad s \approx \Delta/h, \quad \text{where} \quad \Delta \equiv 1 - z^{-1}. \qquad \ldots(9)$$

This is the usual approximation to the derivative $dx/dt \approx \{x(t) - x(t-h)\}/h$ used in numerical analysis (the backward difference).

2 Trapezoidal integration

This method is considerably more accurate, approximating the behaviour of x over the interval h by a straight line passing through $x(t)$ and $x(t-1)$:

$$I(t) = I(t-1) + h\{x(t) + x(t-1)\}/2,$$

giving the correspondence:

$$s \approx \frac{2(1-z^{-1})}{h(1+z^{-1})} \qquad \ldots(10)$$

This is known as the *bilinear* transform. Higher-order approximations can be used (eg Simpson's rule) but producing excessive complexity.

4.2 Frequency-response of the quantisations

To see how the above discretisations behave for different sample intervals h, it is useful to consider the frequency response in which s becomes $j\omega$ and $z \equiv \exp(sh)$ becomes $\exp(j\omega h)$. Frequencies up to the Nyquist frequency $\omega_n = \pi/h$ are relevant because of the sampling theorem, and a normalised frequency $x = \omega/\omega_n (x = 0 \ldots 1)$ will be employed. Note that $\arg(j\omega) = 90^\circ$ and $|j\omega| = w$ for all w; the interest here is to see how the frequency response of the discretisations differ from this (see Franklin and Powell, 1980).

$$\begin{aligned}(1)\ (1-z^{-1})/h &= \Delta/h \to \{1 - \exp(-jx\pi)\}/h \\ &= \exp(-jx\pi/2)\{\exp(jx\pi/2)\} \\ &\quad - \exp\{(-jx\pi/2)\}/h \\ &= 2j\exp(-j\theta)\{\exp(j\theta) \\ &\quad - \exp(-j\theta)\}/2jh,\end{aligned}$$

where $\theta \equiv x\pi/2$

$$\begin{aligned}&= j\exp(-j\theta)/(h/2) \\ &= j\omega\exp(-j\theta)\{\sin\theta/\theta\}\end{aligned}$$

This equals the required value of $j\omega$ except that there is a phase error of $-\theta = -x\pi/2 \to -90^\circ$ as $x \to 1$. Similarly, there is a gain factor of $\operatorname{sinc}\theta$ (1 when $x = 0$, 0.637 at $x = 1$). It is interesting that the above analysis indicates that the effect of sampling for small h is to insert a deadtime of $h/2$ into the loop.

$$(2)\ \frac{2(1-z^{-1})}{h(1+z^{-1})} \to j\omega\tan\theta/\theta,$$

using a similar analysis.
Here there is *no* phase error, but the multiplying factor $\to \infty$ as $x \to 1$. Table 1 shows the approximation errors of the discretisations.

TABLE 1: Factors in the approximations

$x = \omega/\omega_n$	Phase error of 1	sinc θ	tan θ/θ
0.1	-9°	0.996	1.008
0.25	-22.5°	0.974	1.055
0.5	-45°	0.900	1.273
0.75	-67.5°	0.784	2.049
1.0	-90°	0.637	∞

It is seen that the phase error due to the first approximation is significant for $\omega > 0.1\,\omega_n$ but the gain error of the second approximation is significant only for $\omega > 0.5\,\omega_n$. As phase error is generally more important than gain error, the second discretisation is to be preferred as it is accurate for considerably wider sample intervals.

4.3 A digital PID algorithm

A straightforward discretisation of Eqn (1) using rectangular integration gives a 'prototype' digital PID controller:

$$U(z^{-1}) = K\left\{1 + h\frac{1}{T_i(1-z^{-1})} + \frac{T_d}{h}(1-z^{-1})\right\}E(z^{-1})$$

This can be implemented piecemeal, or by rearranging to give

$$(1-z^{-1})U(z^{-1}) = (b_0 + b_1 z^{-1} + b_2 z^{-2})E(z^{-1})$$

where

$$b_0 = K(1 + h/T_i + T_d/h); \quad b_1 = -K(1 + 2T_d/h); \quad b_2 = KT_d/h$$

In difference–equation form the algorithm is:

$$u(t) = u(t-1) + b_0 e(t) + b_1 e(t-1) + b_2 e(t-2)$$

To evaluate this at time t requires the new $e(t)$, and old values $e(t-1)$, $e(t-2)$ and $u(t-1)$ to be available. Elementary integral desaturation can be achieved by limiting $u(t)$ and saving the *limited* value rather than the originally calculated value for use as $u(t-1)$ at the next sample.

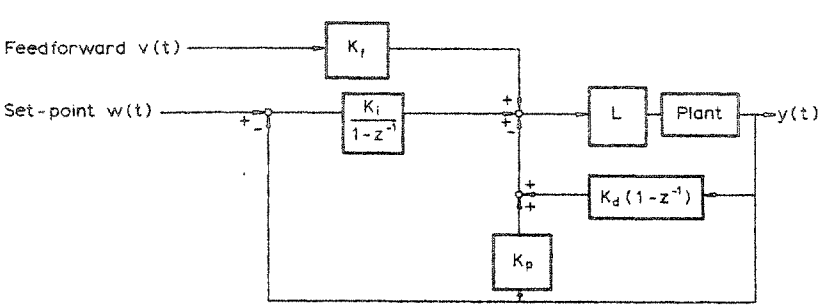


Fig 6 An elementary discrete PID algorithm

Downloaded from tim.sagepub.com at Scientific library of Moscow State University on February 18, 2014

Although the coefficients b_0, b_1 and b_2 can be derived as above from the required K, T_i and T_d to give the same performance for small h, when h becomes larger *different* values may well give better performance. Isermann (1981) uses parameter optimisation to find the best values of the coefficients for any given h, and deduces that small h may not be necessary. For example, although the integral square error does indeed reduce with h, the control variance to achieve the optimum can increase rapidly.

A more practicable algorithm is based on the block diagram of Fig 6, where the three terms have individual gain factors K_p, K_i and K_d. Integral desaturation is achieved by recomputing the integral contribution to just provide saturation if limits on u (t) are exceeded, and bumpless transfer is achieved by computing the integral term so that the PID output would equal that chosen by the operator. Both derivative and proportional kicks are avoided.

Coding for this algorithm is based on the outline of Table 2, and is executed at every sample instant.

The complexity of this can be compared with the preferred method below.

4.4 Discretisation of the simple PI regulator of Section 3

A simple lag with time constant T_1 has an exact Z-transform:

$$\frac{1}{1+sT_1} \rightarrow \frac{(1-\beta)z^{-1}}{1-\beta z^{-1}}, \quad \text{where} \quad \beta = \exp(-h/T_1) \qquad \ldots(11)$$

Placing it in a feedback loop as in Fig 4 with gain factor K' gives:

$$U(z^{-1}) = K'E(z^{-1}) + \frac{(1-\beta)z^{-1}}{1-\beta z^{-1}}E(z^{-1})$$

or

$$(1-z^{-1})U(z^{-1}) = K'(1-\beta z^{-1})E(z^{-1})$$

giving

$$U(z^{-1}) = K'\left\{1 + \frac{(1-\beta)z^{-1}}{1-z^{-1}}\right\}E(z^{-1}) \qquad \ldots(12)$$

Expanding $\exp(-h/T_1)$ as a power series shows that an approximation to $1-\beta$ of order $(h/T_1)^2$ is simply h/T_1, so that

$$U(z^{-1}) = K'\left\{1 + \frac{hz^{-1}}{T_1(1-z^{-1})}\right\}E(z^{-1}) \qquad \ldots(13)$$

which is of the correct PI form. Hence, for small h the pulse-transfer-function (Eqn (11)) in feedback with $\beta = 1 - h/T_i$ will give PI action with gain K' and integral time T_i. The extra delay in the integral computation is necessary both theoretically (see Bristol, 1977) and in the implementation.

It was shown in Section 4.2, though, that trapezoidal integration gives significantly better results than the use of rectangular integration implicit in Eqn (13). Hence, using the corresponding bilinear s→z transformation a more accurate PI algorithm with gain K and integral time T_i is

$$U(z^{-1}) = K\left\{1 + \frac{h(1+z^{-1})}{2T_i(1-z^{-1})}\right\}E(z^{-1})$$

$$= K\frac{(2T_i+h)}{2T_i}\left\{1 + \frac{2hz^{-1}}{(2T_i+h)(1-z^{-1})}\right\}E(z^{-1}) \ldots(14)$$

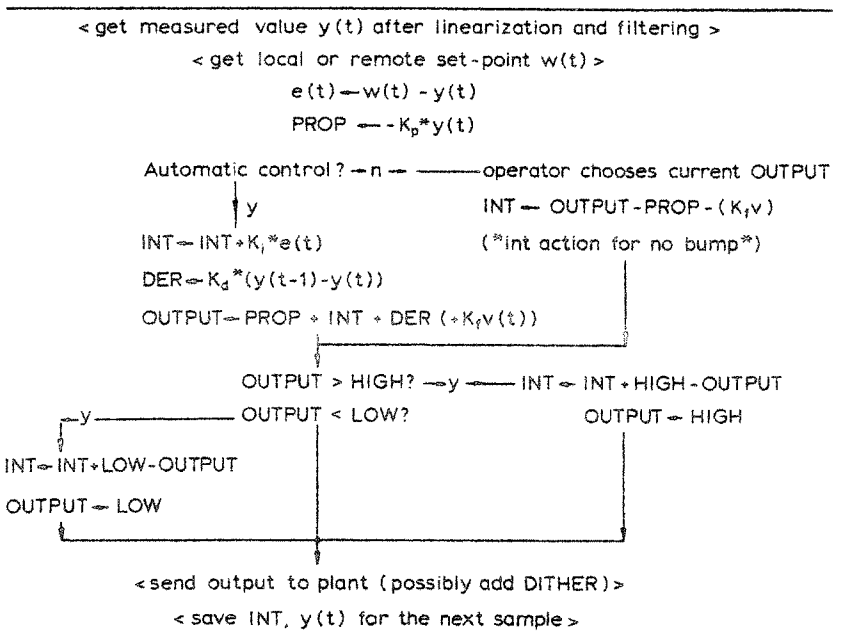
Comparing this with the result of Eqn (12) gives

$$K' = K(1 + h/2T_i) \quad \text{and} \quad \beta = (2T_i - h)/(2T_i + h)$$

[This value of β is a more accurate approximation of order $(h/T_i)^3$ to $\exp(-h/T_i)$]. Hence the discretisation of (Eqn (11)) gives the same action as the PI of Eqn (14) provided that the values of K' and β are chosen as above. Clearly, for small h the properties of the two approximations are indistinguishable, but when T_i approaches h, the trapezoidal method is significantly better. Not only is it more accurate, but it remains stable. Note in particular that if $h > T_i$ then the rectangular integration approach will involve a β greater than 1, thus inducing instability in the algorithm.

The coding of the algorithm follows the flow of Table 3. If 'true' bumpless transfer is required, even if the controller is switched to automatic, far from the set-point, F could be equated to OUTPUT on the manual entry. An outline of the corresponding Fortran coding is given in Table 4.

TABLE 2: Flow of an elementary PID algorithm



Downloaded from tim.sagepub.com at Scientific library of Moscow State University on February 18, 2014

TABLE 3: The feedback PI regulator

```
    If manual control → 1)
    ⟨get y (t) from the input module⟩
    ⟨get local or remote set-point w (t)⟩
    e (t) ← w (t) − y (t)
    OUTPUT ← K' *e (t) + F
(1) If OUTPUT > HIGH = > OUTPUT ← HIGH
    IF OUTPUT < LOW = > OUTPUT ← LOW
    F ← β* F + (1 − β)*OUTPUT
    ⟨send output to plant; save F for next sample⟩
```

Table 4: Fortran coding of the feedback PI regulator

```
C   THIS PART EVALUATES K' AND BETA, GIVEN TI AND K
    GAINP = GAIN* (1 + H/2/TI)
    BETA = (2*TI-H)/(2*TI + H)
C
C   THIS PART IS ENTERED EVERY SAMPLE
    PROP = GAIN*(SETP − YNEW)
    IF (.NOT. MANUAL) GOTO 10
C   ⟨OBTAIN 'OUTPUT' FROM OPERATOR'S DATABASE⟩
    GOTO 1
10  OUTPUT = PROP + F
1   IF (OUTPUT .GT. OUTHI) OUTPUT = OUTHI
    IF (OUTPUT .LT. OUTLO) OUTPUT = OUTLO
    F = BETA*F + (1.0-BETA)*OUTPUT
    IF (MANUAL) F = OUTPUT − PROP
C   OPTIONAL TO GIVE 'TRUE' BUMPLESS TRANSFER
C   THE DIFFERENTIAL TERM CAN NOW BE ADDED − SEE
    LATER.
```

5 Filtering the derivative term

In principle, the gain of the frequency response of the derivative term grows without bound as the frequency increases. Outside the normal bandwidth of the plant, though, the principal component of $y(t)$ is process and measurement noise. Generally, therefore, some restriction should be imposed on the derivative gain and, typically, a factor of 10 is used. This can be achieved by a derivative filter of the form:

$$d(s) = \frac{sT_d}{1 + \alpha s T_d}\, y(s); \quad \text{where} \quad \alpha = 0.1 \qquad \ldots(15)$$

The effect of this filter is seen most clearly when incorporating it into the interacting PID algorithm of Eqn (8), which becomes (for $w(s) = 0$):

$$u(s) = -K\left[\frac{(1 + sT_i)}{sT_i}\,\frac{[1 + sT_d(1 + \alpha)]}{(1 + \alpha s T_d)}\right] e(s)$$

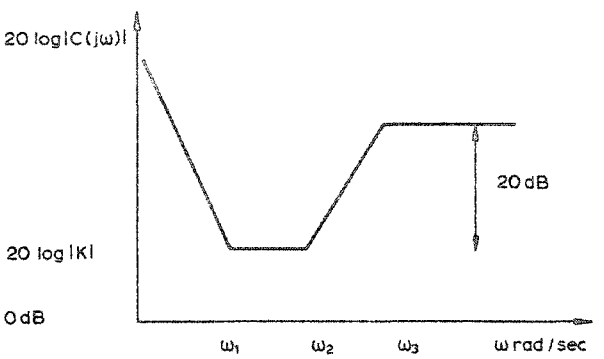


Fig 7 Bode diagram of a PID regulator with filtering

This gives the Bode diagram of Fig 7, in which the break-point frequencies are $\omega_1 = 1/T_i$, $\omega_2 = 1/1.1T_d$ and $\omega_3 = 10/T_d$.

If ω_1 and ω_2 are well separated, the maximum phase advance occurs at $\sqrt{(\omega_2 \omega_3)} = 3.015/T_d$. The corresponding value ψ_{max} is $\sin^{-1}\{1/(1 + 2\alpha)\} = 56°$; this is to be used to obtain sufficient phase margin for plant with excessive lag (such as multiple capacities). Note that increasing the maximum gain factor to 20 makes little difference to ψ_{max}: α becomes 0.05 and the maximum phase lead becomes 65°.

5.1 Discretising the filtered differential term

If the linear transform $s \sim \Delta/h$ is used, Eqn (15) becomes

$$d(t) = \frac{\alpha(1 - z^{-1})}{1 + 0.1\gamma - 0.1\gamma z^{-1}}\, y(t) \quad \text{where} \quad \gamma = T_d/h,$$

or in difference-equation terms

$$d(t) = \gamma\{0.1d(t-1) + y(t) - y(t-1)\}/(1 + 0.1\gamma) \ldots(16)$$

Now Table 1 shows that the linear discretisation has a phase error of $-9°$ at a frequency of $0.1\,\omega_n$ where ω_n is the Nyquist frequency π/h. Hence, the maximum phase lead is reduced to about 45° if ω_n is chosen to be 10 × the frequency giving ψ_{max}.

ie, $\omega_n = 30.15/T_d$, or $h \approx T_d/10$

This implies that if phase advance of up to 45° is required, h must be less than $T_d/10$ or equivalently $\gamma > 10$ using the simple linear discretisation.

Using the bilinear transform, Eqn (15) becomes

$$d(t) = \frac{2\gamma(1 - z^{-1})}{1 + 0.2\gamma + (1 - 0.2\gamma)z^{-1}}\, y(t); \quad \text{where} \quad \gamma = T_d/h,$$

with a corresponding difference equation:

$$d(t) = \{(0.2\gamma - 1)\,d(t-1) - + 2\gamma\,[y(t) - y(t-1)]\}/ \times (1 + 0.2\gamma) \qquad \ldots(17)$$

ie,

$$d(t) = d_0 d(t-1) + d_1(y(t) - y(t-1)) \qquad \ldots(18)$$

Here, for example, if $\gamma = 20$ this gives

$$d(t) = 0.6d(t-1) + 8[y(t) - y(t-1)]$$

The response to a unit step in output $y(t)$ would then be

$$D(z^{-1}) = 8/(1 - 0.6z^{-1}) = \{8 + 4.8z^{-1} + 2.88z^{-2} + 1.73z^{-3} + \ldots\}$$

or

$$\{d(t)\} = \{8, 4.8, 2.88, 1.73, \ldots\}$$

The initial jump in the derivative action (for $K = 1$) is then 8 units compared with 20 units for an unfiltered algorithm.

There is no phase error with the bilinear transformation but there is a gain error, which implies a minimum useful value of γ. Inspection of Eqn (17) shows that the lagged value $d(t-1)$ has a coefficient of zero when $\gamma = 5$; this suggests that the maximum practicable value of h is about $T_d/5$.

The coding of Table 5 can be added to that of Table 4 to produce a non-interacting PID algorithm with a filtered derivative term. If an interacting PID is required, a value

Downloaded from tim.sagepub.com at Scientific library of Moscow State University on February 18, 2014

TABLE 5: Addition of a filtered derivative term

```
C   THIS PART INITIALISES THE COEFFICIENTS
    GAMMA = TD/H
    D1 = (0.2*GAMMA-1.0)/(0.2*GAMMA + 1.0)
    D2 = 2*GAMMA/(0.2*GAMMA + 1.0)
C
C   THIS PART EXECUTED EVER SAMPLE, AFTER THE PI
    CALCULATION
    IF (MANUAL) GOTO 20
    D = D1*D + D2*(YNEW – YOLD)
    OUTPUT = OUTPUT – GAINP*D
    IF (OUTPUT .GT. OUTHI) OUTPUT = OUTHI
    IF (OUTPUT .LT. OUTLO) OUTPUT = OUTLO
C
20  YOLD = YNEW
```

GAINP*D should simply be subtracted from OUTPUT in line 10 of Table 4. Note that clipping of the output is required again in case the derivative contribution drives OUTPUT past a limit.

5.2 Anti-aliasing filters

To avoid process and measurement noise of frequency greater than the Nyquist frequency ω_n being aliased by the sampling, an external analogue filter should be used before the sampler. If this is a simple first-order RC filter, the appropriate time constant is of the order of $1/\omega_n = h/\pi$. Though ideally a sharp-cutoff filter is required, a simple RC network is generally sufficient. With a passive network, R should not be more than 100 kΩ and C more than 1 μF, implying a maximum time constant of 0.1 s and thus a maximum sample interval of 0.3 s. To provide an adequate safety margin such that signals outside ω_n are sharply attenuated, a practicable maximum interval h is 0.1 s.

However, a sample interval of 0.1 s can be too short for some applications, particularly when using advanced digitally based algorithms. These may need to use an interval $h' = mh$ ($m = 1..600$) between control calculations and so need a correspondingly smaller value of ω_n. To reduce ω_n by a factor m can be achieved by digital filtering, where the filter is executed every minor sample but whose output is used only at every major (control) sample. Such a filter can be based on the development of Section 4.4, where Eqn (11) implies that a first-order digital filter with time constant $T = mh$ is:

$$y'(t) = \beta y'(t-1) + (1-\beta)y(t) \qquad \ldots(19)$$

Here $\beta = \exp(-h/T) \approx (2T-h)/(2T+h) = (2m-1)/(2m+1)$, and the delay which Eqn (11) places on $y(t)$ is ignored. The input to the filter is the measured $y(t)$ (after the anti-aliasing filter) and the filtered output is $y'(t)$. If a higher-order filter is required, several simple filters can be cascaded; this is particularly appropriate if m is large and round-off errors in Eqn (9) are to be avoided (β would be very close to 1.0).

6 Non-linear algorithms

There are certain circumstances in which non-linear algorithms can offer improved performance, though implicitly they cannot be analysed with the same degree of confidence as linear approaches.

6.1 'Spike' or logical filters

When an analogue multiplexer operates incorrectly by sampling the wrong channel, it induces a 'spike' on the data sequence of the intended channel. Though linear filtering reduces the effect somewhat, it can be important to detect the event and to stop the spike entering the control algorithm. This is done by having a 'window' which allows small changes of $y(t)$ through but clips large changes. If $y(t)$ is in fact changing rapidly, the window 'gap' should be allowed to expand according to the transient nature of the data.

Suppose that d_{min} is the minimum width of the window (which can readily be determined by inspecting operating records) and that $\delta(t)$ is its current width. Let $f(t)$ be the filter output and $S(t)$ be a count of the number of successive samples for which the output is clipped. Then a simple spike filter as discussed by Roberts (1975) follows the flow of Table 6.

For each time that the new data exceed the bounds, the gap is doubled, and when they are within the bounds the gap is halved until S reaches its minimum value of 0. An appropriate Fortran subroutine is given in Table 7.

A more comprehensive filter is given by Pehrson, (1968), which considers three components of the data: a real signal represented by a Gaussian coloured-noise process $\epsilon(t)$, a step variable $\xi(t)$ and a pulse variable $\zeta(t)$. The step and pulse variables are equal to 0 if no step or pulse is present in the data. The total signal $y(t)$ is then given by

$$y(t) = \epsilon(t) + \xi(t) + \zeta(t)$$

The objective of the filter is to follow the steps, remove the pulses and to smooth (predict) the Gaussian component. The Fortran program given appears to work effectively on practical data taken from a paper-making process.

TABLE 6: A simple 'spike' filter
(Initialise the count S to 0)

$\delta(t) \leftarrow y(t) - f(t-1)$	[detect the change in y]	
$d(t) \leftarrow 2^{S(t)}$	[open gap according to count]	
If $\lvert\delta(t)\rvert > d(t) \Rightarrow$	$f(t) \leftarrow f(t-1) + d(t)\,\mathrm{sgn}\,\delta(t)$	
	$S(t) \leftarrow S(t-1) + 1$	[count up]
If $\lvert\delta(t)\rvert \leqslant d(t) \Rightarrow$	$f(t) \leftarrow y(t)$	[allow through]
	$S(t) \leftarrow S(t-1) - 1 \geqslant 0$	[count down to 0]

TABLE 7: Fortran code of the spike filter

```
    SUBROUTINE SPIKE (YNEW, F, DMIN, IS)
C   YNEW is the new measured value
C   F is the filter output (at entry the previous output)
C   DMIN is the minimum gap allowable
C   IS is the current count of bound-hits
    D = DMIN*2**IS                  | CURRENT GAP
    DELTA = YNEW – F                | JUMP IN OUTPUT
    IF (ABS(DELTA) .LE. D) GOTO 10  | INSIDE WINDOW?
    F = F + D*SGN(DELTA)            | STEP A BIT
    IS = IS + 1                     | AND COUNT
    GOTO 20
10  F = Y                           | OK, LET THROUGH
    IS = IS – 1                     | COUNT DOWN
    IF (IS .LE. 0) IS = 0           | UNTIL 0
20  RETURN
```


Downloaded from tim.sagepub.com at Scientific library of Moscow State University on February 18, 2014

6.2 A non-linear control algorithm

Sometimes the effect of noise can be reduced by using a dead zone on the output, such that no control is transmitted if its change over the sample is less than a certain amount. Alternatively, a dead zone on $y(t)$ (which occurs naturally with quantisation) will cut down some of the effect of noise whose amplitude is less than the gap. It has been suggested that such improvements can also be achieved by a nonlinear controller in which the proportional gain is varied according to the magnitude of the error

ie, $K_p = K_0[1 + K_1|e(t)|]$

The argument for this controller is that if $|e|$ is 'small' the gain of the controller will be small so that noise does not affect the loop too much when it is near the setpoint. When $|e|$ is 'large' then the controller gain will be large, causing vigorous control action. Such a controller might be useful in cases in which the plant gain itself varies inversely with amplitude, or in which the measured variable is corrupted by so much noise that derivative action is worthless.

6.3 A non-linear filter with phase-lead

Derivative action provides phase lead for tight closed-loop operation, but amplifies noise. A non-linear algorithm which provides phase lead without amplifying high frequencies is shown in Fig 8.

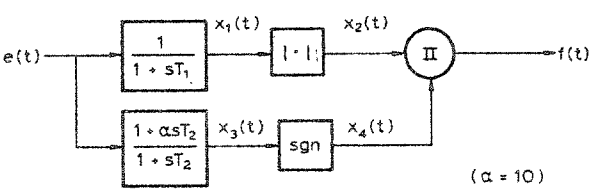


Fig 8 A non-linear phase-lead filter

The function of the simple lag with time constant T_1 is to provide attenuation at high frequencies, but at the same time its output $x_1(t)$ lags the input $e(t)$. When $\alpha = 10$ the second linear transfer function provides a phase lead θ_2 in $x_3(t)$, but at the same time amplifies high frequencies, as described earlier. The signum element produces a square wave $x_4(t)$ of unit amplitude and whose fundamental component is $4 \sin(\omega t + \theta_2)/\pi$, if $e(t) = Y \sin \omega t$. This component has the phase lead of the second linear element, but and amplitude independent of ω. The output $x_1(t)$ is rectified to produce $x_2(t)$ and finally $f(t)$ is the product $x_2(t)x_4(t) = |x_1(t)| \operatorname{sgn}(x_3(t))$.

The filter output will contain many harmonic components, but note that it will be passing into a PI algorithm and thence into a plant, both of which will attenuate these harmonics. Therefore, using the ideas of describing functions, the behaviour of the filter can be considered in terms of the fundamental frequency component. To first order this is seen to have an amplitude dependent on $x_1(t)$ and a phase lead dependent on $x_3(t)$, as required.

To discretise the algorithm the usual $s \rightarrow z$ transforms can be used, or the Z-transforms of the linear elements can be obtained directly:

Element 1

$$\frac{(1-\beta)z^{-1}}{1-\beta z^{-1}}; \quad \text{where} \quad \beta = \exp(-h/T_1)$$

Element 2

$$\frac{\alpha + (1-\alpha-\gamma)z^{-1}}{1-\gamma z^{-1}}; \quad \text{where} \quad \gamma = \exp(-h/T_2)$$

This gives the digital algorithm of Table 8.

TABLE 8: A non-linear phase-advance filter

$x_1(t) \leftarrow \beta x_1(t-1) + (1-\beta)e(t)$	{attenuator}
$x_3(t) \leftarrow \gamma x_3(t-1) + \alpha e(t) + (1-\alpha-\gamma)e(t-1)$	{phase-lead}
$f(t) \leftarrow \lvert x_1(t) \rvert \operatorname{sgn}[x_3(t)]$	

There have not, as yet, been reports of applications of this algorithm. Note that it needs zero-mean data for the signum element to work correctly – that is why $e(t)$ rather than $y(t)$ is suggested as input to the filter.

7 Quantisation

Quantisation errors are introduced in three ways: via the A–D converter, the D–A converter and from round-off in the calculations. The calculation errors can be eliminated to a large extent by using floating-point working at the cost of extra computation time and storage requirements. Quantisation in the A–D converter generally affects the differential action (noise) whereas D–A converter quantisation and round-off error affect integral action (offset).

7.1 The effect on integral action

Consider the elementary integral calculation:

$$I(t) = I(t-1) + he(t)/T_i \qquad \ldots(20)$$

for which the corresponding component of control is $KI(t)$. The objective of integral action is to keep ramping the control if an offset ($e \neq 0$) exists; this can only be achieved if the computed quantity he/T_i is non-zero after round-off. If ϵ is the quantisation level (using fixed-point arithmetic), then an error $|e| = T_i\epsilon/h$ will not induce integration in Eqn (20) and so will remain as an offset. Note that, for any given ϵ, $|e|$ increases as T_i increases and as h reduces. Usually when using fixed-point arithmetic a scale is chosen by specifying a minimum value of T_i, and the quantity h/T_i is stored as a single coefficient. Hence $h/(T_i)_{\min}$ could be assigned the value 1.0 to give the widest possible range of T_i that the computer can cope with. The maximum T_i that can be stored in this way corresponds to 1 bit, ie, a value $\epsilon = h/(T_i)_{\max}$, implying $(T_i)_{\max} = (T_i)_{\min}/\epsilon$. With this optimum assignment of coefficients, the maximum sustained error $|e|$ that could occur is $(T_i)_{\max}\epsilon/h = (T_i)_{\min}/h$.

Supposing that in the worst case an offset of 1% = 0.01 is allowable, then the above implies that $0.01 = (T_i)_{\min}/h$. If h is chosen to be 0.1 s, then $(T_i)_{\min} = 0.001$ s (a slightly ludicrous figure) and $(T_i)_{\max} = 0.001/\epsilon$ s. Consider now the use of 16-bit working (surely enough for control calculations?) in which ϵ is 1 part in 32767 (allowing $e(t)$ to have a range of -1.0 to 1.0). The maximum T_i is then 32.767 s $\approx$ 0.5 min. This is unacceptably small for sluggish plant: either the maximum allowable sustained offset or the sample interval would have to be increased to get a large value of $(T_i)_{\max}$.

Bristol (1977) discusses examples in which a different scaling induces offsets as large as 100% for large T_i, and

Downloaded from tim.sagepub.com at Scientific library of Moscow State University on February 18, 2014

makes recommendations for avoiding these difficulties. For example, double-precision working (or indeed floating-point calculation) can be used, but he finds that simply using a random number in the least-significant part of the calculation produces the same results. This can be readily achieved by adding *dither* to Eqn (20): simply add a random least-significant part of the calculation.

An important practical case arises in which even floating-point working is insufficient: an incremental algorithm using a stepping motor. Here the truncation amounts to sending no pulse even though the desired control increment is non-zero. The integral component of the algorithm of Fig 2 produces a desired *change* of control signal (or output *rate*)

$$u_1(t) = K' e(t)/T_i' = Ke(t)/T_i$$

where

$$T_i' = N/r_{max}$$

If the sample interval is h, a number of steps given by $hu_1(t)$ will be requested. Suppose that an error E is chosen such that for the minimum value of T_i the maximum stepping rate is generated. E might be as much as 1.0 (full scale), but would more typically be 0.1. Then $r_{max} = EK'/(T_i')_{min}$. The smallest number of steps that can be requested in the sample is 1, corresponding to the smallest offset e_{min} which can be detected, giving

$$1 = hK' e_{min}/T_i' = h(T_i')_{min} r_{max} e_{min}/T_i' E$$

Hence if δ is the smallest allowable offset, the corresponding range of T_i is given by $(T_i')_{max}/(T_i')_{min} = hr_{max}\delta/E$. For example, if $r_{max} = 500$ step/s, $h = 0.1$ s, $\delta = 0.01$, then for $E = 0.1$ this gives a range of only 5 : 1 in allowable integration times.

A solution to this problem is to use *internal* integration to accumulate fractional components of $u_1(t)$ in a variable $u'(t)$, as shown in Table 9.

TABLE 9: Avoiding quantisation errors in stepping-motor control

$u'(t) \leftarrow u'(t-1) + hu_1(t)$	[accumulate control demands]
$r_d \leftarrow \text{Int}[u'(t)]$	[find no of full steps]
$u'(t) \leftarrow u'(t) - r_d$	[leave a fractional remainder]
If $r_d > hr_{max} \Rightarrow r_d \leftarrow hr_{max}$	[clip the desired signal]
if $r_d < -hr_{max} \Rightarrow r_d \leftarrow -hr_{max}$	[up or down]

Note in particular the subtle way integral desaturation is implemented; it does *not* (unlike previous positional algorithms) use the actual value of r_d transmitted. This is because the saturation limits of the final element would not, in general, be known: r_{max} might be transmitted but might not cause corresponding valve movement.

7.2 The effect on derivative action

The contribution of an elementary differential term to the control action is $T_d[y(t-1) - y(t)]/h$. The value of T_d depends on the time constants of the plant, but may be of the order of several minutes for slow processes such as thermal systems. Suppose $T_d = 10$ min and $h = 0.1$ s, then

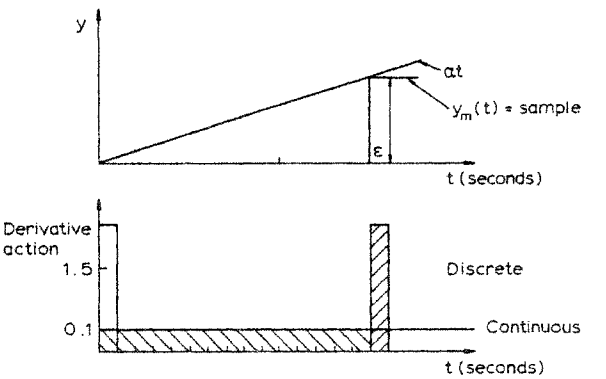


Fig 9 Quantisation effect in derivative action

$T_d/h = 6000$. Now a 12-bit A–D converter has a quantisation of 1 part in 4096, so if y changes by 1 quantisation interval then the derivative term will suddenly jump by about 1.5 units. Consider a slowly ramping output signal $y(t) = \alpha t$ as shown in Fig 9. For a plant requiring a derivative time of 10 min, α might itself be of the order of 10% in 10 min or 0.01/min, with a corresponding *continuous* derivative action of $T_d d(\alpha t)/dt = \alpha T_d = 0.1$ – perfectly acceptable value. However, the ramp will not pass through a quantisation interval of 1/4096 until 60/4096 x 0.01 = 1.46 s have elapsed, which is approximately 15 sample intervals when $h = 0.1$ s.

Note that the shaded areas of the graphs in Fig 9 are the same, and that instead of the differential action being a constant of 0.1, it produces spikes at wide intervals. Filtering of the derivative term (not 'spike' filtering which is for larger spikes on $y_m(t)$ itself) will go some way towards improving the situation, but clearly the most appropriate action is to choose a larger value of h. At least 1 s would be required in the above example to reduce the severe spiking, but possibly 30 s would be better still.

7.3 The choice of sample interval

There is an understandable desire to have as short a sample interval as possible in a digital controller. Partly this is because straightforward methods can then be used to discretise well understood analogue controllers, but also because if a large sample interval is used, a sudden disturbance might affect the process for some time before counteracting control action is seen to be taken. Similarly, an operator might have to wait before seeing any action after the imposition of a setpoint change. However, the examples above show the dangers of having too short a sample interval, and Section 4 indicates that the alternative bilinear transformation gives significantly better results for longer sample intervals with little extra complexity.

In most cases rapid sampling gives perfectly acceptable results, particularly if PI algorithms and floating-point arithmetic are used. However, if derivative action is needed and long integral and derivative times are required to match the plant time constants, then rapid sampling leads to considerable difficulties. It has been found in most cases that there is little point in having h such that there are more than 10–20 samples during the 95% risetime in the response to a step, or for a plant with deadtime to have more than 4 or 5 samples during that time. Common recommendations are that the sample intervals for typical processes should be as shown in Table 10.

Downloaded from tim.sagepub.com at Scientific library of Moscow State University on February 18, 2014

Table 10: Sample intervals for typical loops

Flow loop	1 s
Level loop	5 s
Temperature loop	30 s–10 min

These are only a guide, and depend on the characteristics of the process. For example, electromechanical systems would have to be sampled at a faster rate. In any event, better performance will always be obtained by using the bilinear discretisation to obtain the digital algorithm.

8 Tuning three-term controllers

Given an analytical or a well identified plant model, the best way to tune a PID controller is to use parameter optimisation methods as discussed by Isermann (1981). However, in most cases it is not economical to derive a mathematical model for noncritical loops, so an experimental procedure is necessary. A step-by-step procedure can be employed, iterating over the PID parameters until a satisfactory response is obtained, or all three parameters can be obtained on the basis of a single test, using so-called tuning rules. The test can either be open or closed loop; in the open-loop case a simple approximate model is generated, while with a closed-loop trial data are obtained by putting the loop into oscillation.

8.1 The step-test method

The open-loop procedure is to inject a step input to the manipulated variable and from the response to obtain the parameters of a first-order model

$$y(s) = \exp(-sT_2)\frac{K_p}{1+sT_1}u(s) \qquad \ldots(21)$$

The plant model has gain K_p, time constant T_1 and deadtime T_2; the values are derived by the graphical procedure of Fig 10.

Given the values of K_p, T_1 and T_2, the method of Pemberton, as critically evaluated by Hang (1979) recommends the controller parameters of

$$K = 2T_1/3K_pT_2; \quad T_i = T_1; \quad T_d = T_1/4 \qquad \ldots(22)$$

Note that with an interactive algorithm is used the settings of Eqn (8) as generated by Fig 4 would be:

$$K = T_1/3K_pT_2; \quad T_i = T_d = T_1/2$$

It is possible in many cases to derive a second-order plant model from a step response; Hang concludes that the corresponding tuning rule is more sensitive to parameter error and may not lead to better closed-loop performance.

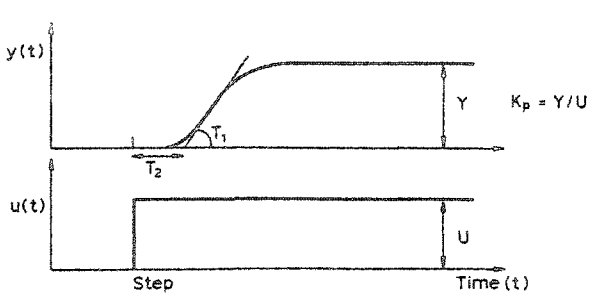


Fig 10 Open-loop step test on a process

8.2 The closed-loop method

This is a classical method inspired by Ziegler and Nichols (1942). The idea is to control the process under proportional action only and to increase K until the loop oscillates. The value K_u of the gain and the period T_u of the oscillation are noted; clearly, such a procedure can be used only on uncomplaining loops. The recommended values of the controller parameters are then given as in Table 11, in which the second set of columns correspond to the use of an interactive PID algorithm.

TABLE 11: Ziegler–Nichols' tuning rule

	K	T_i	T_d	K	T_i	T_d
P only	$0.5K_u$	–	–	$0.5K_u$	–	–
P + I	$0.45K_u$	$T_u/1.2$	–	$0.45K_u$	$T_u/1.2$	–
P + I + D	$K_u/1.7$	$T_u/2$	$T_u/8$	$K_u/3.4$	$T_u/4$	$T_u/4$

Note that, as in the open-loop, the suggested zeros of the PID algorithm are coincident (ie, equal values in the interactive implementation) at $-4/T_u$ (cf $-2/T_1$ in the open-loop method).

Though Hang finds these to be an excellent set of rules, his evaluation uses differential action in the forward path which is not generally acceptable. Many find that the Ziegler–Nichols rules tend to give rather too much overshoot to a step in setpoint demand, and Astrom (1982) recommends the rules of Table 12, which are based on the closed loop having a specified phase margin ψ; the larger the phase margin the less that overshoot is likely.

TABLE 12: Astrom's phase-margin rule

ψ	K	T_i	T_d
30°	$0.87K_u$	$0.55T_u$	$0.14T_u$
45°	$0.71K_u$	$0.77T_u$	$0.30T_u$
60°	$0.50K_u$	$1.29T_u$	$0.30T_u$

An appropriate sample interval h in these cases is about $0.01T_u$ to $0.05T_u$. Note that to increase the phase margin and hence to reduce overshoot, K is reduced and both T_i and T_d are increased.

The Ziegler–Nichols method is best explained by considering the simplest open-loop plant which will oscillate under proportional-only control, having an open-loop transfer function $1/(1+s)^3$. The root locus of this plant under increasing proportional control K is shown in Fig 11(i).

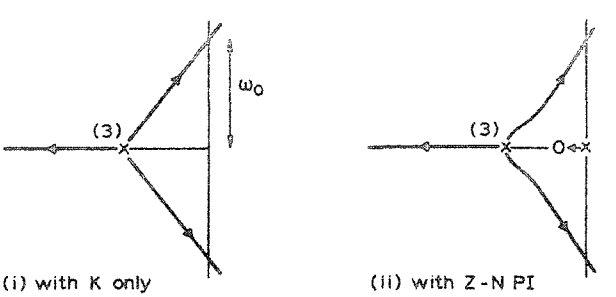


Fig 11 Root locus for a third-order plant. i. with K only; ii. with Z–N PI

Downloaded from tim.sagepub.com at Scientific library of Moscow State University on February 18, 2014

The value ω_0 is the natural frequency in radian/s, which from the diagram is $\sqrt{3}$. Hence $T_u = 2\pi/\omega_0 \approx 3.62$ s. The Ziegler–Nichols PI rule would place a zero at $-1.2/T_u = -0.33$, which together with the pole at the origin has only a marginal effect on the root locus, as shown in Fig 11(ii). The gain at any given point on a root locus is given by the product of its distances to the open-loop poles, divided by the product of the distances to the zeros. Hence reducing the gain by a factor of 0.45 means that the closed-loop pole has moved about 75% of the way to the imaginary axis, which implies that the damping factor of the closed loop is only about 0.17. In this example (possibly a worst case), the Ziegler–Nichols PI rule certainly suggests too high a gain, though the PID values are much better. In the PID case the two closed-loop zeros are specified at $-2/T_u = -0.55$. This produces a significantly improved root locus in which the closed loop is stable for all K. Hence, derivative action is a useful feature provided that it can be tuned properly.

The Ziegler–Nichols procedure can be automated, and one approach is described by Astrom (1982). One critical feature is that the closed loop may not in fact oscillate under high gain; sometimes an extra integrator needs to be introduced so that oscillations are generated, and the tuning rule then needs appropriate modification. If oscillations are undesirable for process reasons, other methods such as the proposal of Gawthrop (1982) should be used, in which a plant model is implicitly identified online using normal operating data.

9 Conclusions

This report has investigated PID algorithms and how they can be implemented in a digital computer. General comments on the procedures are

- Implementation in feedback form produces a more elegant algorithm.
- The ordinary and interactive algorithms have different parameters; the interactive algorithm might be easier to use but restricts the effective differential action and the compensating zeros.
- The bilinear transform should be used for discretisation.
- Better performance is attainable with derivative action, provided that there is appropriate filtering.
- Quantisation can cause difficulties, particularly for small sample intervals and when using an incremental algorithm.
- The sample interval in most loops can well be extended, provided that an accurate discretisation is used.
- If the anti-aliasing analogue filter does not allow a large value of h, digital filtering at each minor sample should be used to reduce the effective Nyquist frequency (and also to remove plant noise).

The report also outlines various linear and non-linear filters which could be useful in certain cases.

10 Acknowledgement

I would like to thank P. S. Tuffs for many useful discussions during the production of this report.

11 References

Astrom, K. J. 1982. *Ziegler–Nichols auto-tuners,* Report LUDFD2/(TFRT-3167)/01–025/(1982), Lund Institute of Technology.

Bristol, E. H. 1977. 'Designing and programming control algorithms for DDC systems', *Control Engineering,* 24–26.

Buckley, P. S. 1964. *Techniques of process control,* Wiley.

Chin, K. C., Corripio, A. B. and Smith, C. L. 1973. *Digital control algorithms* (2 parts), Instruments and Control Systems, Pittsburgh.

Franklin, G. F. and Powell, J. D. 1980. *Digital control of dynamic systems,* Addison–Wesley.

Gawthrop, P. J. 1982. 'Self-tuning PI and PID controllers', *IEEE Conf. on Applications of Adaptive and Multivariable Control,* Hull.

Hang, C. C., Tan, K. K. and Ong, S. L. 1979. 'A comparative study of controller tuning formulae', ISA Annual Conference.

Harriott, P. 1964. *Process control,* McGraw–Hill.

Isermann, R. 1981. *Digital control systems,* Springer–Verlag.

Moroney, P., Willsky, A. S. and Houpt, P. K. 1980. 'The digital implementation of control compensators: the coefficient wordlength issue', *IEEE Trans. Autom. Control,* **AC-25**, (4), 621–630.

Pehrson, B. 1968. *A non-linear digital filter for industrial measurements, Technical paper TP 18.185,* IBM Systems Development Division, Sweden.

Ragazzini, J. R. and Franklin, G. F. 1958. *Sampled-data control systems,* McGraw–Hill.

Roberts, P. D. 1975. 'Control actions and algorithms', *IEE vacation school on Industrial Digital Control Systems.*

Shinskey, F. G. 1979. *Process control systems,* McGraw–Hill.

Thomas, H. W., Sandoz, D. J. and Thomson, M. 1983. 'New desaturation strategy for digital PID controllers', *Proc IEE, Pt.D,* **130**, (4), 188–192.

Ziegler, J. G. and Nichols, N. B. 1942. 'Optimum settings for automatic controllers', *Trans ASME,* 64, 759–768.

Downloaded from tim.sagepub.com at Scientific library of Moscow State University on February 18, 2014