THIS 'SIMPLE' THREE-TERM CONTROLLER CAN HAVE MANY OPTIONAL PARAMETERS THAT GIVE IT RICH FUNCTIONALITY

The PID CONTROLLER: ALGORITHM AND IMPLEMENTATION

By Pasi Airikka

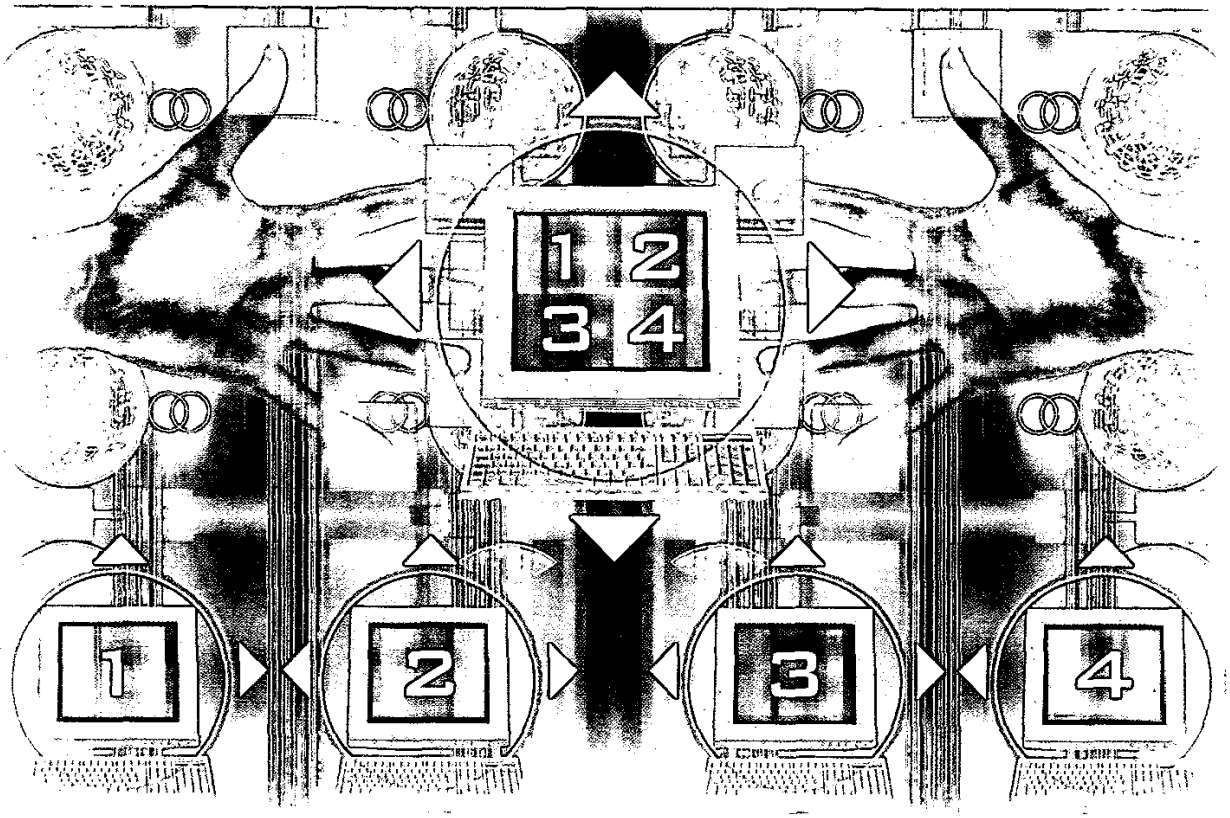
The PID controller (Proportional, Integral, Derivative) has been known for several decades in many fields of automatic control. It has had powerful applications and several modifications anywhere where automatic control has been applicable. In spite of its many modified structures and forms the basic idea has remained the same. The underlying working principle relies on feedback control and the PID controller is the most common embodiment of feedback control. It involves three different terms, each of which have a specific purpose. Proportional and integral terms were known in the 1930s but derivative control was not invented until the 1940s.

In basic terms, the PID controller is a numerical recipe, an algorithm. The algorithm may be implemented, that is, programmed using any programming language supporting numerical computation. It can be written in Visual Basic, Fortran, Pascal, C or Java. Also, there are numerous platforms for the algorithm such as personal computers, distributed control systems (DCS), programmable logic controllers (PLC), and field devices or microchips enabling embedded solutions.

In spite of the PID controller syntax language and its platform or even application, there are some essential features that should be involved in the PID controller algorithm. The basic calculation covering arithmetic around three different terms is not enough. Other issues of automatic feedback control must also be taken care of. In addition to this, there are some potential characteristics that may be added to increase functionality and to improve applicability of the PID controller.

There are different types of PID controllers – such as ratio, cascade or split range controllers – and there are different algorithm types – such as position or incremental (velocity) algorithms. The position algorithms can be categorised into ISA, series and parallel algorithms. The most typical operation modes for PID controllers are automatic, manual, cascade and remote.

Some manufacturers hide their algorithms, but they



should be available so that users can properly tune the instrument. The PID controller is not a secret and it should not be treated as one, although it is very rare for the numerical robustness of the algorithm to be available at all.

PID IN BRIEF

PID control is based on feedback control (see fig 1 overleaf). The PID controllers receive the measured process variable and compares it to a given setpoint. According to this control error, the PID controllers calculate a correction to a control device such as a control valve. The control device causes a change in process input, resulting in a change in the process variable according to the process dynamics. Finally, after some time and corrections, the process variable is equal to the setpoint or is within a certain range of it.

Unfortunately, there are process disturbances that interfere with control. Load disturbances and sensor noise disrupt the process and cause process variability and process variable deviations from the setpoint. It is the PID controller tuning that determines how well the control loop performs under different process upsets such as setpoint changes and load disturbance changes.

Fig 2 illustrates a typical dynamic behaviour of control loop variables of an over-damped PID controlled process. The top plot shows setpoint and process variable as a function of time. There is a 10% setpoint change at time = 10 min (top) and load disturbance changes at time = 20 min (-5% permanent step) and time = 30 min (-5% rapid impulse). The load disturbance changes are illustrated in the bottom plot. The plot in the middle shows the control variable (PID controller output) reacting to the setpoint change and the process disturbance changes. The operation direction here is positive, that is, a positive change in control variable results in a positive change in process variable.

Fig 3 shows how PID controller's different terms (P, I and D control) work when facing setpoint or load disturbance changes. The control variable in figure 1 (middle) consists of proportional, integral and derivative control presented in fig 3. Proportional control immediately reacts to a non-zero control error between setpoint and process variable trying to make the error smaller by its contribution. The effect of proportional control is always dominant when the control error is large but it diminishes as soon as the error approaches zero.

Integral control is essential for automatic control to store the control level needed to provide a control error with zero mean. Integral control gradually removes the control error.

Derivative control reacts to a rate of change in the process variable. Therefore, when the process variable changes drastically (see fig 2, topmost), the derivative controller reacts strongly. The strong influence of derivative control is generally limited by a low-pass filter in conjunction with derivative control.

However, there are many other things to be included →

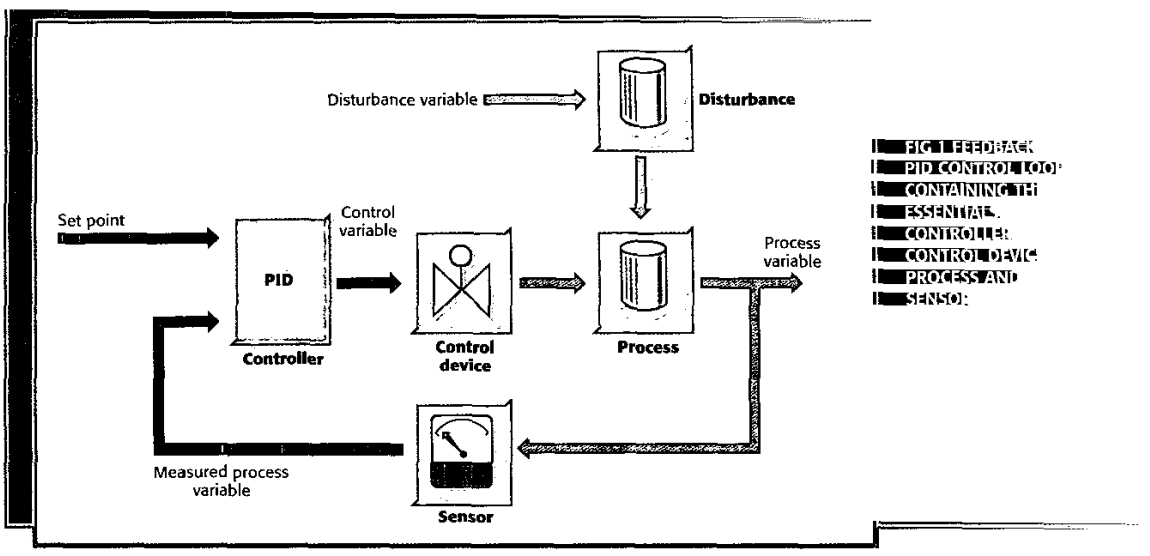


FIG 1 FEEDBACK PID CONTROL LOOP CONTAINING THE ESSENTIAL CONTROLLER, CONTROL DEVICE, PROCESS AND SENSOR.

in PID controller than just the basic functionality of P, I and D controllers.

SCALING AND LIMITS

The most fundamental feature lacking in text book versions of PID controllers is signal scaling. Every physical signal has a range with minimum and maximum values and a unit such as mA , %, deg or kPa. To avoid large systematic errors in PID control computation, all the input signals must be set according to their scales as well as the PID controller output signal. It may not be necessary to scale the signals for simulation purposes but for the real use, the PID controller can't operate in non-specified numerical ranges.

With the allowed signal scales there may be specific limit values for some signals such as setpoint and control variable. These limits restrict the allowed signal range even more. For example, the process variable, as well as a setpoint, may have a range from 0-600 degrees but the operation range under control is 100-400 degrees. Consequently, the operation range limits can be used for limiting the setpoint in PID controller. Sometimes there may also be a limit for the allowed rate of change that setpoint can be altered.

ANTI-WINDUP

Anti-windup is an absolutely necessary feature for preventing integral windup when using PI or PID control. Integral windup takes place whenever the control variable (PID controller output) is at its low or high limit, that is, limited and integral control is on. This results in a continuous integration of the control error, that is, infinite cumulative summing of control error. Integral anti-windup is needed to manipulate the integral part of the control variable so it does not exceed the control limit. This improves control performance significantly.

The best way to accomplish anti-windup is to apply dynamic back-calculation based on the feedback error between the control variable calculated by the PID controller and the real, saturated control variable (see fig 4). The feedback error is taken to the integral term calculation. The anti-windup parameter is either internally fixed in the PID controller or left as a tuning parameter for adjusting anti-windup dynamics.

Fig 5 illustrates the benefit of using anti-windup. There are two otherwise similar control loops but one has anti-windup (red) and the other does not have it (blue). There is a setpoint change from 90% to 98% at time = 10 min (top). The PID controller starts to eliminate the control error due to a setpoint change and thus increases its output (middle) until reaching the maximum 100% level. When there is no anti-windup (blue) the integral control continuously increases (bottom, blue) but with anti-windup there is a correction preventing the integral control from winding up. Thus, the obtained control performance (topmost) is better with anti-windup.

The difference is more visible at the load disturbance striking at time = 20 min. The disturbance is such that control immediately saturates at 100% level (middle) but when the disturbance loosens up, the difference can be seen. Due to integral windup, the PID controller with no anti-windup can't go lower from 100% level until the integral control has come down from where it was heading during the saturation time starting at time = 20 min. Therefore, there is a constant control error (topmost, blue) for a long time = 22 min whereas the PID controller with anti-windup is able to find the proper control level (middle, red) to remove the control error (top, red).

DERIVATIVE AND MEASUREMENT FILTERING

The derivative term as an ideal and anti-causal solution can't be implemented directly. Instead, it requires a low-pass filter which introduces a new parameter. This parameter is called either *derivative filtering time* or *derivative filtering constant*. It can be either a dimensionless parameter or a

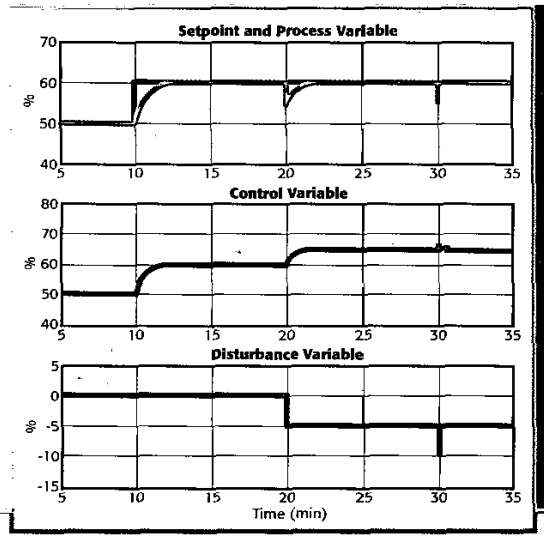


FIG 2 RESPONSES FOR PID CONTROLLED PROCESS. TOP: SETPOINT (BLUE) AND PROCESS VARIABLE (RED). MIDDLE: CONTROL VARIABLE. BOTTOM: LOAD DISTURBANCE

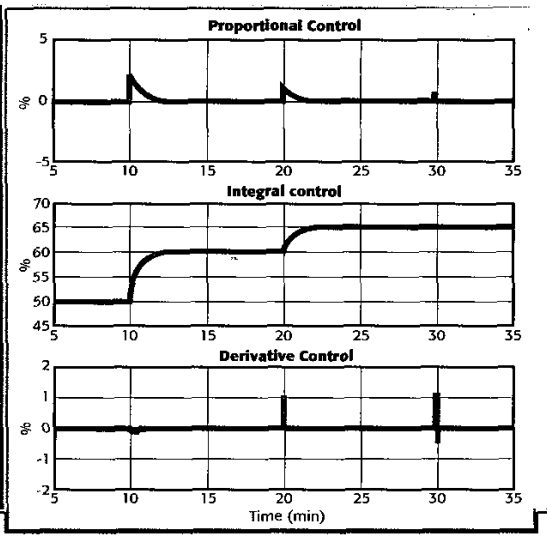


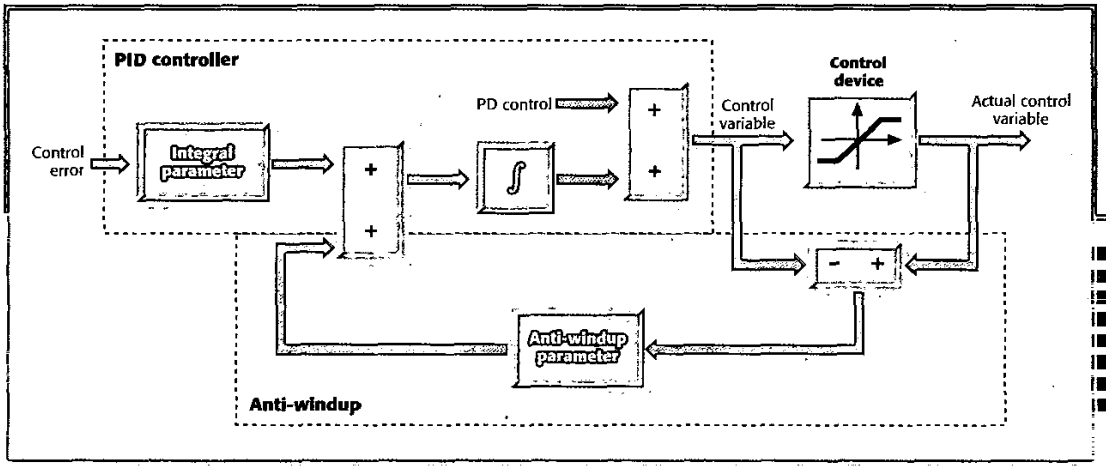
FIG 3 EFFECTS OF THE PID CONTROLLER'S THREE TERMS. TOP: PROPORTIONAL CONTROL. MIDDLE: INTEGRAL CONTROL. BOTTOM: DERIVATIVE CONTROL

parameter with a time horizon dimension. The parameter is either internally fixed in the algorithm or it is left as a tuning parameter for the user.

Although the derivative control is an anticipator and capable of linear prediction it has several weak points. The most severe problem with derivative control is that it amplifies noise in process variable.

Fig 6 shows a case where derivative control is turned on and off and derivation has been changed. There are two periods longer than 5 minutes when derivative control has been on (bottom). During the first period the derivative time is twice as large as during the second period. By looking at the process variable (top), there seems to be no difference at all. The process variable is noisy due to sensor noise and there is no difference if derivative control is on or off. However, the control variable makes the difference (middle). Process variable noise is amplified when derivative control is on and it can be clearly seen. Although the noise doesn't fortunately propagate from control to process variable, the noisy control variable will put excessive wear on controlling devices.

Derivative filtering applies only on derivative control but measurement filtering can be applied to process variables before entering the PID algorithm. The measurement filter is typically a low-pass filter which attenuates high frequencies, that is, measurement noise from the measured process variable. The measurement filter is often characterised by a single filter value indicating its bandwidth. →



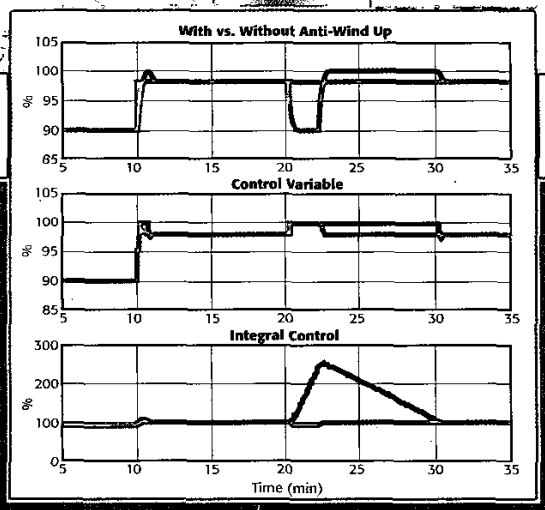


FIG 5 RESPONSES FOR A PID CONTROLLED PROCESS WITH (RED) AND WITHOUT (BLUE) ANTI-WINDUP. TOPMOST: PROCESS VARIABLES. MIDDLE: CONTROL VARIABLES. BOTTOMT: INTEGRAL CONTROL

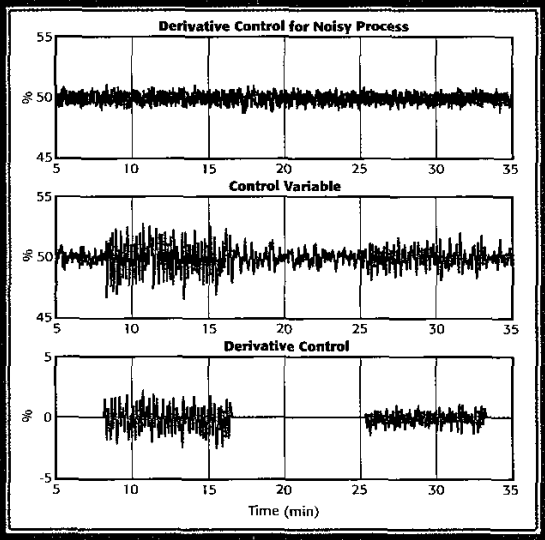


FIG 6 RESPONSES FOR PID CONTROLLED NOISY PROCESS. TOP: PROCESS VARIABLE. MIDDLE: CONTROL VARIABLE. DOWN: DERIVATIVE CONTROL

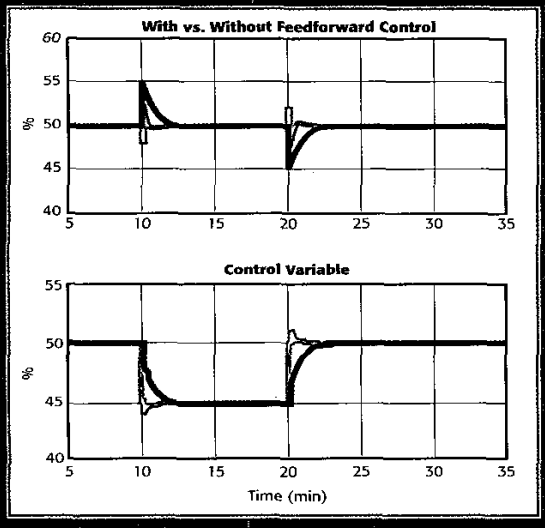


FIG 7 RESPONSES FOR PID CONTROLLED PROCESS WITH (RED) AND WITHOUT (BLUE) FEEDFORWARD CONTROL UPPER: PROCESS VARIABLES LOWER: CONTROL VARIABLES

FEEDFORWARD CONTROL

Feedforward control is a good way to compensate for load disturbances that are measured. Feedforward control enables the attenuation of the disturbance at the time it is about to affect the process variable. Feedforward control only requires that the disturbance be measured and taken to the controller at every control period and that the signal path from the disturbance to the process variable is equal or longer than the path from the control variable to the process variable. Feedforward control is characterised by a single tuning parameter that is a function of process and disturbance static behaviour.

Fig 7 shows how much better a PID controlled loop works with feedforward control. In this example there are two load disturbances, at 10 min and 20 min. In both cases, the PID controller without feedforward control begins to move its controlled variable (blue line, bottom diagram) immediately. However the same PID controller with feedforward (red line) reacts much sooner and stronger to compensate for the detected disturbance. Thus, the process variable response with feedforward control (upper diagram, red) is clearly better than without it (upper, blue).

TRACKING

Sometimes the controller output should be able to follow an external signal instead of operating according to its received setpoint. In this case, the control variable is not calculated according to the algorithm but it is simply replaced by the external signal (bypass) or it is calculated using back-calculation so that the control variable is soon equal to the external tracking signal. Tracking based on back-calculation also provides a smooth transition from tracking mode to automatic control.

BUMPLESS MODE AND PARAMETER TRANSFER

In manual control, the automatic control variable is bypassed by a control variable set by the operator. At the time the PID controller is set back in automatic, these control variables most likely differ from each other. This difference may cause a bump in control variable behaviour. This undesired behaviour can be avoided by updating the automatic control variable even when the PID controller is set in manual. The method is called *bumpless mode transfer*. Similarly, bumpless parameter transfer guarantees the harmless transition in control variable computing when PID controller tuning parameters are changed.

SETPOINT WEIGHTING

Setpoint weighting means setting a weight for the setpoint in proportional control. By setting a weight, the setpoint response can be designed separately from the disturbance response. The PID controller can be first designed (tuned) for obtaining good disturbance attenuation and, after that, the setpoint weight is set to obtain a desired setpoint response. Therefore, setpoint weighting provides another degree of freedom for PID controller design. Setpoint weighting is characterised by a scalar value from 0 to 1.

PREDICTIVE CONTROL FACILITY

It is possible to expand the PI controller's functionality into a predictive PI controller (PPI) by adding predictive control into it. This increases the controller's applicability for deadtime dominated systems (see C&CE issue Aug/Sep 2003, pp16-19). Predictive control is characterised by two tuning parameters: prediction gain and deadtime estimate.

NUMERICAL ROBUSTNESS

Numerical robustness of any algorithm is essential. Similarly, PID controllers should be coded in order to preserve their numerical robustness. Here, numerical robustness means that the algorithm maintains its functionality and characteristics even when facing different numerical inputs and parameter combinations. For some PID controller discretisation methods, the resulting PID controller can suffer from robustness failures when facing sufficiently small tuning parameters.

PID CONTROLLER AS A FUNCTION BLOCK

The PID controller functionality can be described as in fig 8. The PID controller has input signals that can be categorised into an array of incoming signals, of parameters, of scale parameters, of limits and of binary selection parameters. There are also mode and internal state variables. The functions of the PID controller are filtering, signal scaling, signal limiting, setpoint weighting, PID controller with integral anti-windup, predictive control, feedforward control, and tracking and mode handling. The output signals are the control variable, warnings of exceeded limits, auxiliary output signals and state variables.

Fig 9 is an example of a PID controller function block with its input and output parameters. The arrays in fig 8 are ungrouped here into individual parameters with a corresponding input/output pin. The continuous-time signal pins are denoted by green and the parameters are in blue. The scale parameters are in grey and the limit values are in red. The binary selection parameters are in black and mode in cyan. As you can see, there is much more than just the setpoint, the process variable, the control variable and the PID controller tuning parameters. And who said that the PID controller is just a simple algorithm with a few parameters? ■

The author Pasi Airikka is a Senior Research Engineer with Metso Automation in Tampere, Finland. He may be reached at pasiairikka@hotmail.com

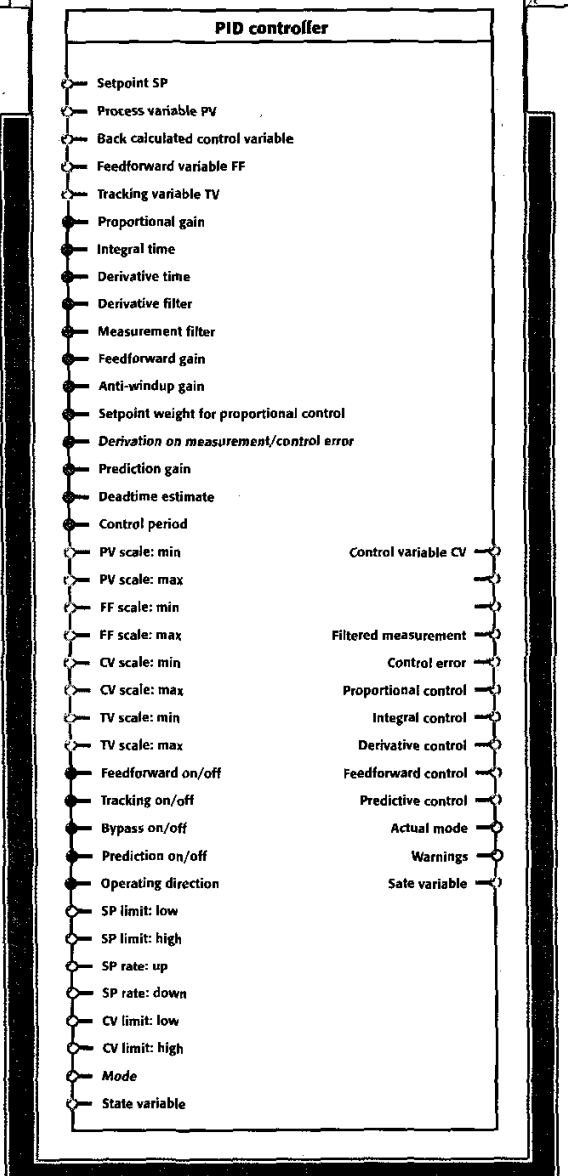


FIG 9 EXAMPLE OF A PID CONTROLLER BLOCK WITH ITS INPUT AND OUTPUT PARAMETERS

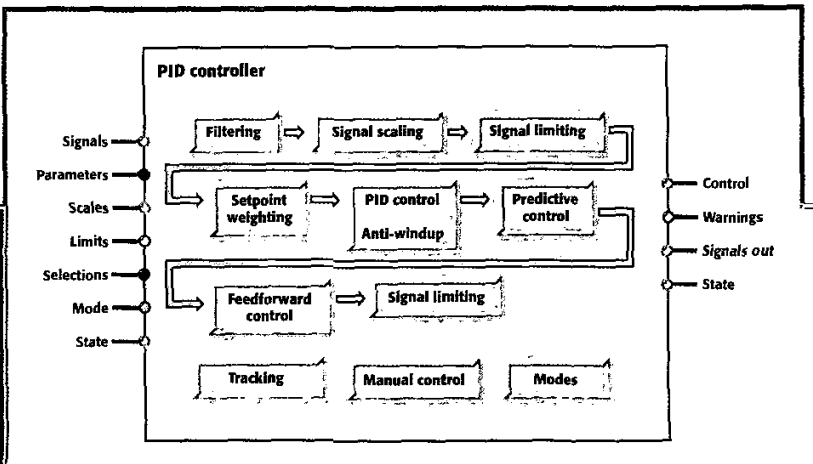


FIG 8 PID CONTROLLER INPUTS, OUTPUTS AND STRUCTURE