3.4 Discretizing a PID controller

3.4.1 Computer based control loop

Figure 3.2 shows a control loop where controller is implemented in a computer. The computer registers the process measurement signal via an AD converter (from analog to digital). The AD converter produces a numerical value which represents the measurement. As indicated in the block diagram this value may also be scaled, for example from volts to percent. The resulting digital signal, $y(t_k)$, is used in the control function, which is in the form of a computer algorithm or program calculating the value of the control signal, $u(t_k)$.

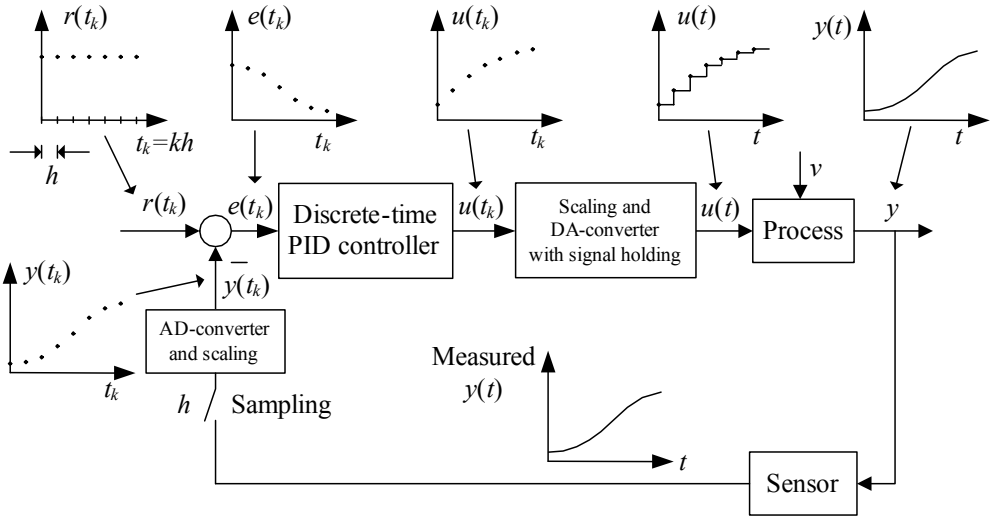

Figure 3.2: Control loop where the controller function is implemented in a computer

The control signal is scaled, for example from percent to milliamperes, and sent to the DA converter (from digital to analog) where it is held constant during the present time step. Consequently the control signal becomes a staircase signal. The time step or the sampling interval, h [s], is usually small compared to the time constant of the actuator (e.g. a valve) so the actuator does not feel the staircase form of the control signal. A typical value of h in commercial controllers is 0.1 s.

3.4.2 Development of discrete-time PID controller

The starting point of deriving the discrete-time PID controller is the continuous-time PID (proportional + integral + derivate) controller:

$$u(t) = K_p e(t) + \frac{K_p}{T_i}\int_0^t e\, d\tau + K_p T_d \dot{e}_f(t) \tag{3.19}$$

where u is the controller output (the control variable), e is the control error:

$$e(t) = r(t) - y(t) \tag{3.20}$$

where r is the reference or setpoint, and y is the process measurement. e_f is the filtered control error. It is the output of the following lowpass filter:

$$e_f(s) = \frac{1}{T_f s + 1} e(s) \tag{3.21}$$

where T_f is the filter time-constant, which is typically selected as

$$T_f = aT_d \tag{3.22}$$

where typically $a = 0.1$.

We will now derive a discrete-time formula for $u(t_k)$, the value of the control signal for the present time step. The discretization can be performed in a number of ways. Probably the simplest way is as follows: Differentiating both sides of (3.19) gives[2]

$$\dot{u}(t) = K_p \dot{e}(t) + \frac{K_p}{T_i} e(t) + K_p T_d \ddot{e}_f(t) \tag{3.23}$$

Applying the Backward differentiation method (3.2) to $\dot{u}$, $\dot{e}$, and $\ddot{e}_f$ gives

$$\frac{u(t_k) - u(t_{k-1})}{h} = K_p \frac{e(t_k) - e(t_{k-1})}{h} + \frac{K_p}{T_i} e(t_k) + K_p T_d \frac{\dot{e}_f(t_k) - \dot{e}_f(t_{k-1})}{h} \tag{3.24}$$

Applying the Backward differentiation method on $\dot{e}_f(t_k)$ and $\dot{e}_f(t_{k-1})$ in (3.24) gives

$$\begin{aligned} \frac{u(t_k) - u(t_{k-1})}{h} &= K_p \frac{e(t_k) - e(t_{k-1})}{h} && (3.25)\\ &\quad + \frac{K_p}{T_i} e(t_k) && (3.26)\\ &\quad + K_p T_d \frac{\frac{e_f(t_k) - e_f(t_{k-1})}{h} - \frac{e_f(t_{k-1}) - e_f(t_{k-2})}{h}}{h} && (3.27) \end{aligned}$$

[2] The time derivative of an integral is the integrand.

Solving for $u(t_k)$ finally gives the discrete-time PID controller:

$$u(t_k) = u(t_{k-1}) + K_p\left[e(t_k) - e(t_{k-1})\right] + \frac{K_p h}{T_i}e(t_k) \tag{3.28}$$

$$+\frac{K_p T_d}{h}\left[e_f(t_k) - 2e_f(t_{k-1}) + e_f(t_{k-2})\right] \tag{3.29}$$

The discrete version of the filter (3.21) can be derived as described in Section 3.3.

The discrete-time PID controller algorithm (3.28) is denoted the *absolute* or *positional* algorithm. Automation devices typically implements the *incremental* or *velocity* algorithm. because it has some benefits. The incremental algorithm is based on splitting the calculation of the control value into two steps:

1. First the incremental control value $\Delta u(t_k)$ is calculated.
2. Then the total or absolute control value is calculated with $u(t_k) = u(t_{k-1}) + \Delta u(t_{k-1})$.

Thus, the incremental PID algorithm is

$$\Delta u(t_k) = K_p\left[e(t_k) - e(t_{k-1})\right] + \frac{K_p h}{T_i}e(t_k) \tag{3.30}$$

$$+\frac{K_p T_d}{h}\left[e_f(t_k) - 2e_f(t_{k-1}) + e_f(t_{k-2})\right] \tag{3.31}$$

$$u(t_k) = u(t_{k-1}) + \Delta u(t_k) \tag{3.32}$$

The summation (3.32) implements the (numerical) integral action of the PID controller.

The incremental PID control function is particularly useful if the actuator is controlled by an incremental signal. A step-motor is such an actuator. The motor itself implements the numerical integration (3.32). It is (only) $\Delta u(t_k)$ that is sent to the motor.

3.4.3 Some practical features of the PID controller

A practical PID controller must have certain features to be functional:

- **Integrator anti windup**: Large excitations of the control system, typically large disturbances or large setpoint changes, may cause the

control signal to reach its maximum or minimum limits with the control error being different from zero. The summation in (3.32), which is actually a numerical integration, will then cause u to increase (or descrease) steadily – this is denoted *integral windup* – so that u may get a very high (or low) value. When the excitations are back to normal values, it may take a very long time before the large value of u is integrated back to a normal value (i.e. within 0 – 100%), causing the process output to deviate largely from the setpoint.

Preventing the windup is (not surprisingly) denoted *anti windup*, and it can realized as follows:

1. Calculate an intermediate value of the control variable $u(t_k)$ according to (3.32), but do not send this value to the DA (Digital-to-Analog) converter.
2. Check if this intermediate value is greater than the maximum value $u_{\max}$ (typically 100%) or less than the minimum value $u_{\min}$ (typically 0%). If it exceeds one of these limits, set $\Delta u(t_k)$ in (3.32) to zero.
3. Write $u(t_k)$ to the DA converter.

- **Bumpless transfer:** Suppose the controller is switched from automatic to manual mode, or from manual to automatic mode (this will happen during maintenance, for example). The transfer between modes should be bumpless, ideally. Bumpless transfer can be realized as follows:

  – *Bumpless transfer from automatic to manual mode:* In manual mode it is the manual (or nominal) control signal u_0 that controls the process. We assume here that the control signal $u(t_k)$ in (3.32) has a proper value, say u_{good}, so that the control error is small, immediately before the switch to manual mode. To implement bumpless transfer, set u_0 equal to u_{good} at the switching moment.

  – *Bumpless transfer from manual to automatic mode:* While the controller is in manual mode, $\Delta u(t_k)$ in (3.32) is set (forced) to zero, and both $u(t_k)$ and $u(t_{k-1})$ are set to u_0, the manual control value (which may be changed during the manual mode, of course). After the switching from manual to automatic mode allow $\Delta u(t_k)$ to be calculated according to (3.30) (hence, do not force $\Delta u(t_k)$ to be zero any longer).

3.4.4 Selecting the sampling time of the control system

The DA converter (digital to analog) which is always between the discrete-time control function and the continuous-time process to be controlled, implements holding of the calculated control signal during the time-step (sampling interval). This holding implies that the control signal is *time delayed* approximately $h/2$, see Figure 3.3. The delay influences the

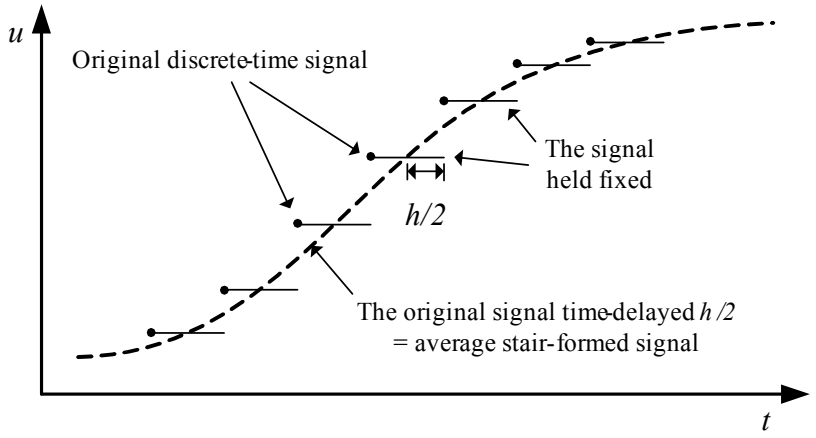

Figure 3.3: The DA-converter holds the calculated control signal throyghout the sampling interval, thereby introducing an approximate time-delay of $h/2$.

stability of the control loop. Suppose we have tuned a continuous-time PID controller, and apply these PID parameters on a discrete-time PID controller. Then the control loop will get reduced stability because of the approximate delay of $h/2$. As a rule of thumb (this can be confirmed from a frequency response based analysis), the stability reduction is small and tolerable if the time delay is less than *one tenth of the response-time* of the control system as it would have been with a continuous-time controller or a controller having very small sampling time:

$$\frac{h}{2} \leq \frac{T_r}{10} \tag{3.33}$$

which gives

$$h \leq \frac{T_r}{5} \tag{3.34}$$

The response time is here the 63% rise time which can be read off from the setpoint step response. For a system the having dominating time constant T, the response-time is approximately equal to this time constant.