10. Digital Process Control

10.1 Basic principle of digital control
10.2 Digital PID controllers
    10.2.1 A 2DOF continuous-time PID controller
    10.2.2 Discretisation of PID controllers
    10.2.3 Implementation and tuning
10.3 Sampling of time-delay models
    10.3.1 State-space model with a time delay
    10.3.2 Second-order model with real poles and zero
    10.3.3 First-order model
10.4 Controller synthesis using discrete-time model
    10.4.1 PID controller
    10.4.2 Time-delay compensation
    10.4.3 Dead-beat control
    10.4.4 Improving intersample behaviour
10.5 Example: comparison of various controllers
10. Digital Process Control

10.1 Basic principle of digital control

The block “Sample” receives the continuous-time signals \( y(t) \) and \( r(t) \), and discretises them to sequences of numbers, \( y(t_i) \) and \( r(t_i) \), defined at the time instants \( t_i, i = 1,2, ... \).

In practice, this is an A/D converter.

The block “Hold” receives the sequence of numbers \( u(t_i), i = 1,2, ..., \) from the control algorithm and sends out a piecewise continuous signal, \( u(t) = u(t_i), \ t_i \leq t < t_{i+1}. \)

In practice, this is a D/A converter.
10. Digital Process Control

10.2 Digital PID controllers

10.2.1 A 2DOF continuous-time PID controller

Many variants of continuous-time PID controllers were introduced in Section 7.1. A flexible form of the PID controller is

\[ u(t) = K_c \left( e_p(t) + \frac{1}{T_i} \int_0^t e(\tau) \, d\tau + T_d \frac{de_f(t)}{dt} \right) + u_0 \]  

(10.1)

\[ e_p(t) \equiv b r(t) - y(t), \quad e(t) \equiv r(t) - y(t), \quad e_f(t) \equiv c r(t) - y_f(t) \]  

(10.2)

\[ T_f \frac{dy_f(t)}{dt} + y_f(t) = y(t) \]  

(10.3)

Here \( b \) and \( c \) are setpoint weights yielding a 2DOF PID controller. The control law (10.1) can also be written

\[ u(t) = u_p(t) + u_i(t) + u_d(t) + u_0 \]  

(10.4)

\[ u_p(t) = K_c e_p(t) \]  

(10.5)

\[ u_i(t) = \frac{K_c}{T_i} \int_0^t e(\tau) \, d\tau \quad \Leftrightarrow \quad \frac{du_i(t)}{dt} = \frac{K_c}{T_i} e(t) \]  

(10.6)

\[ u_d(t) = K_c T_d \frac{de_f(t)}{dt} \]  

(10.7)
10.2.2 Discretisation of PID controllers

A discrete-time PID controller can be obtained by using a discrete-time approximation of the derivatives in (10.3), (10.6) and (10.7). There are many methods for discretization of a derivative.

**Forward-difference approximation**

According to the definition of a derivative,

\[
\frac{dx(t)}{dt} = \lim_{h \to 0} \frac{x(t+h) - x(t)}{h}
\]

(10.8)

A natural approximation of the derivative is then

\[
\frac{dx(t)}{dt} \approx \frac{x(t+h) - x(t)}{h}
\]

(10.9)

However, this is not a suitable approximation of the derivative term in (10.1) or (10.7), because it requires a future value at time \( t + h \) to be known when the control signal \( u(t) \) is calculated. It is possible to avoid this causality problem by including a derivative filter. However, even then the derivative part becomes unstable if

\[
T_f \leq h/2
\]

(10.10)

where \( h \) is the sampling interval.
### Backward-difference approximation

By a slight modification of (10.9), the derivative can be approximated by

\[
\frac{dx(t)}{dt} \approx \frac{x(t) - x(t-h)}{h}
\]  

(10.11)

This results in a stable controller for all non-negative values of \( T_d \) and \( T_f \).

The proportional part of the controller becomes (without approximation)

\[
u_p(t_k) = K_c e_p(t_k)
\]  

(10.12)

where \( t_k \) is the time instant of the \( k \)th sample.

Applying (10.11) to the integral part (10.6) gives

\[
\frac{u_i(t_k) - u_i(t_{k-1})}{h} = \frac{K_c}{T_i} e(t_k) \Rightarrow u_i(t_k) = u_i(t_{k-1}) + \frac{hK_c}{T_i} e(t_k)
\]  

(10.13)

Because the setpoint \( r(t_k) \) has not yet affected the integral at time \( t_k \) it appears that a better approximation than \( he(t_k) = h(r(t_k) - y(t_k)) \) for the increase of the integral between time instants \( t_{k-1} \) and \( t_k \) would be \( h(r(t_{k-1}) - y(t_k)) \). However, that would delay the effect of \( r(t_k) \) by one sample, which does not happen in a continuous-time controller. Therefore, \( e(t_k) \) is used in (10.13).
10.2.2 Discretisation of PID controllers

It is convenient to derive the **derivative part** via the Laplace domain. Combination of (10.3) and (10.7) in the Laplace domain yields

\[ T_F s U_d(s) + U_d(s) = K_c T_d s E_d(s) \]  \hspace{1cm} (10.14)

where \( E_d(s) \) is the Laplace transform of

\[ e_d(t) \equiv cr(t) - y(t) \]  \hspace{1cm} (10.15)

The inverse Laplace transform of (10.14) gives

\[ T_f \frac{du(t)}{dt} + u(t) = K_c T_d \frac{de_d(t)}{dt} \]  \hspace{1cm} (10.16)

Application of (10.11) to both sides of (10.16) at time \( t = t_k \) now yields

\[ T_f \frac{u_d(t_k) - u_d(t_{k-1})}{h} + u_d(t_k) = K_c T_d \frac{e_d(t_k) - e_d(t_{k-1})}{h} \]

or after rearrangement

\[ u_d(t_k) = \frac{T_f}{T_f + h} u_d(t_{k-1}) + \frac{K_c T_d}{T_f + h} (e_d(t_k) - e_d(t_{k-1})) \]  \hspace{1cm} (10.17)

The signals \( u_i(t_k) \) in (10.13) and \( u_d(t_k) \) in (10.17) are **calculated recursively** for \( k = 1, 2, \ldots \) and substituted into (10.4).
The forward- and backward-differences are asymmetrical in the sense that the difference \((x(t) - x(t - h))/h\) does not approximate the derivative in the middle of the range, i.e., at time \(t - 0.5h\), but at \(t - h\) (forward) or \(t\) (backward).

A symmetrical approximation is obtained by

\[
\frac{1}{2} \left( \frac{dx(t)}{dt} + \frac{dx(t-h)}{dt} \right) \approx \frac{x(t) - x(t-h)}{h}
\]  

(10.18)

In general, this approximation is better than the previous ones and for well chosen controller parameters, the resulting discrete-time PID controller approximates the continuous-time PID controller better than the previous ones.

However, if the filtering time constant is so small that (10.10) applies, the PID controller will cause intersample ripple, also called ringing. As a result of this, the inputs and the outputs will oscillate between sampling points, which is a very undesirable feature.

Thus, Tustin’s approximation is not suitable (without some modification) if no derivative filter is used or if the sampling interval is “too large”.
Trapezoidal approximation

In the forward- and backward-difference approximations, the signal to be approximated is assumed to be constant in the sampling interval.

It appears better to assume that a changing signal changes linearly in the sampling interval. This assumption yields the trapezoidal approximation.

The figure to the left illustrates the forward-difference, the backward-difference and the trapezoidal approximation of a signal $y$.

If the signal is a control error $e = r - y$, the signal and the approximation is discontinuous when a setpoint change is made, as shown to the right.
Using the trapezoidal approximation, (10.6) gives for the integral part

\[ u_i(t_k) = \frac{hK_c}{T_i} \sum_{j=1}^{k} 0.5(e(t_{j-1}) + e(t_j)) \]  

(10.19)

which can be rewritten as

\[ u_i(t_k) = u_i(t_{k-1}) + \frac{hK_c}{2T_i} (e(t_k) + e(t_{k-1})) \]  

(10.20)

Also here, \( r(t_k) \) has not affected the true integral at time \( t_k \), but use of \( r(t_{k-1}) - y(t_k) \) instead of \( e(t_k) \) would again delay the control effect of \( r(t_k) \) by one sample, which does not happen in a continuous-time controller. Therefore, \( e(t_k) \) is used in (10.20). This is also consistent with formal use of the trapezoidal approximation when the fact that \( r(t_k) \) is constant in the sampling interval is neglected.

Again, the derivative part is most easily derived via the Laplace domain. From (10.7) it follows that \( \frac{de_f}{dt} \) is needed. Because the setpoint is constant in the sampling intervals, \( \frac{de_f}{dt} = -\frac{dy_f}{dt} \). From the Laplace transform of (10.3) it then follows that

\[ sE_f(s) = \frac{s}{T_f s + 1} E_d(s) \]  

(10.21)
10.2.2 Discretisation of PID controllers

According to the trapezoidal approximation, \( e_d(t) \) changes linearly between sampling points. The Laplace transform of these changes up to time \( t = t_k \) is (note that \( j \) is an integer, not the imaginary unit \( j = \sqrt{-1} \))

\[
E_d(s) = \frac{e_d(t_0)}{s} + \frac{e_d(t_1) - e_d(t_0)}{hs^2} + \sum_{j=1}^{k-1} \frac{e_d(t_{j+1}) - 2e_d(t_j) + e_d(t_{j-1})}{hs^2} e^{-jhs}
\]  

(10.22)

Substitution into (10.21) and taking the inverse Laplace transform yields

\[
\frac{d e_f(t_k)}{dt} = \frac{e_d(t_0)}{T_f} e^{-t_k/T_f} + \frac{e_d(t_1) - e_d(t_0)}{h} \left( 1 - e^{-(t_k-jh)/T_f} \right) + \sum_{j=1}^{k-1} \frac{e_d(t_{j+1}) - 2e_d(t_j) + e_d(t_{j-1})}{h} \left( 1 - e^{-(t_k-jh)/T_f} \right)
\]

Combination with the corresponding expression for \( d e_f(t_{k-1})/dt \) yields

\[
\frac{d y_f(t_k)}{dt} = \frac{d y_f(t_{k-1})}{dt} e^{-h/T_f} + \frac{e_d(t_k) - e_d(t_{k-1})}{h} \left( 1 - e^{-h/T_f} \right) \]

(10.23)

Combination with (10.7) finally gives

\[
u_d(t_k) = e^{-h/T_f} u_d(t_{k-1}) + \frac{K_c T_d (1 - e^{-h/T_f})}{h} \left( e_d(t_k) - e_d(t_{k-1}) \right)
\]

(10.24)
### 10.2.3 Implementation and tuning

#### Unification of methods

The control laws according to the backward-difference and the trapezoidal approximation can both be expressed in the form

\[
\begin{align*}
u_i(t_k) &= u_i(t_{k-1}) + b_{i1} e(t_k) + b_{i2} e(t_{k-1}) \\u_d(t_k) &= a_d u_d(t_{k-1}) + b_d (e_d(t_k) - e_d(t_{k-1}))
\end{align*}
\]

where \( u_i(t_0) = u_d(t_0) = 0 \) can be used as starting values for the iterative calculation. The control errors \( e(t_k) \) and \( e_d(t_k) \) are defined in (10.2b) and (10.15), respectively, and the parameters are given below.

<table>
<thead>
<tr>
<th>Method</th>
<th>( b_{i1} )</th>
<th>( b_{i2} )</th>
<th>( a_d )</th>
<th>( b_d )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backward</td>
<td>( \frac{K_c h}{T_i} )</td>
<td>0</td>
<td>( \frac{T_f}{T_f + h} )</td>
<td>( \frac{K_c T_d}{T_f + h} )</td>
</tr>
<tr>
<td>Trapezoid</td>
<td>( \frac{K_c h}{2T_i} )</td>
<td>( \frac{K_c h}{2T_i} )</td>
<td>( e^{-h/T_f} )</td>
<td>( \frac{K_c T_d}{h} \left( 1 - e^{-h/T_f} \right) )</td>
</tr>
</tbody>
</table>

In general, the trapezoidal approximation is more accurate than the backward-difference approximation, but both tend to be good choices.
10.2 Digital PID controllers

Preventing integrator windup

If the control signal reaches a process constraint (e.g., a control valve fully open) when there is a control error, and the controller contains standard integral action, the absolute value of the integral will continue to increase as long as the sign of the control error is unchanged. This will result in integrator windup, which is illustrated in the figure.

Here a setpoint change is made at \( t = 0 \) (upper). The control signal \( u \) immediately reaches its maximum value 0.1 (middle). The integral \( u_i \) continues to increase as long as \( y < r \) (lower).

The control signal will remain at the maximum value until the integral decreases sufficiently even though \( y > r \). After this, the control signal might reach its minimum value and oscillations follow.
Integrator windup can be prevented in many ways. One way is to adjust the value of the integrator so that the calculated control signal with the adjusted integrator exactly reaches the constraint. This is achieved with

$$u_i(t) = \frac{K_c}{T_i} \int_0^t e(\tau)d\tau + \frac{1}{T_t} \int_0^t e_s(\tau)d\tau$$

(10.27)

where $$e_s(t) \equiv \min\left(u_{\text{max}}, \max(u_{\text{min}}, u(t))\right) - u(t)$$

(10.28)

This function has the property that $$e_s = 0$$ if $$u_{\text{min}} \leq u \leq u_{\text{max}}$$, otherwise $$e_s = u_{\text{max/min}} - u$$.

$$T_t$$ is the tracking time constant, which should be chosen as $$T_d < T_t < T_i$$.

A reasonable value is $$T_t = \sqrt{T_i T_d}$$. If $$T_d = 0$$, $$T_t = 0.1T_i$$ can be used.

The figure illustrates how this anti-windup works. (Note different time scale.)
Discrete-time control

For discrete-time control, (10.27) can be discretised similarly to (10.6). Because

\[
\frac{du_i(t)}{dt} = \frac{K_c}{T_i} e(t) + \frac{1}{T_t} e_s(t)
\]  \hspace{1cm} (10.30)

the backward-difference approximation yields

\[
u_i(t_k) = u_i(t_{k-1}) + \frac{hK_c}{T_i} e(t_k) + \frac{h}{T_t} e_s(t_k)
\]  \hspace{1cm} (10.31)

and the trapezoidal approximation yields

\[
u_i(t_k) = u_i(t_{k-1}) + \frac{hK_c}{2T_i} (e(t_k) + e(t_{k-1}))
\]
\[
+ \frac{h}{2T_t} (e_s(t_k) + e_s(t_{k-1}))
\]  \hspace{1cm} (10.32)
10.2 Digital PID controllers

**Incremental control algorithms**

The control algorithms where the control signal is calculated by (10.4) are called *positional algorithms*.

Sometimes it is more convenient to calculate the control signal as an addition to its previous value, i.e.,

\[ u(t_k) = u(t_{k-1}) + \Delta u_p(t_k) + \Delta u_i(t_k) + \Delta u_d(t_k) \]  \hspace{1cm} (10.33)

where

\[ \Delta u(t_k) \equiv u(t_k) - u(t_{k-1}) \]  \hspace{1cm} (10.34)

Such an algorithm is called an *incremental algorithm*. It should only be used with integral action.

The given algorithms for the integral part are essentially in incremental form, but the other parts have to be determined by (10.34). The result is

\[ \Delta u_p(t_k) = K_c (e_p(t_k) - e_p(t_{k-1})) \]  \hspace{1cm} (10.35)

\[ \Delta u_i(t_k) = b_{i1} e(t_k) + b_{i2} e(t_{k-1}) \]  \hspace{1cm} (10.36)

\[ \Delta u_d(t_k) = a_d \Delta u_d(t_{k-1}) \]
\[ + b_d (e_d(t_k) - 2e_d(t_{k-1}) + e_d(t_{k-2})) \]  \hspace{1cm} (10.37)

If anti-windup is desired, (10.36) should be modified as (10.31) or (10.32).
Choice of sampling interval

There are no firm rules on how to choose the sampling interval. A long sampling interval (low sampling rate) has many drawbacks such as

- longer apparent time delay $h/2$
- the aliasing effect — no frequency above the Nyquist frequency $\pi/h$ can be detected

An unnecessarily short sampling interval (high sampling rate) means more measurements, calculations and control actions with almost no new information at sampling instants.

The following are some guidelines on the choice of sampling rate:

- Use 10 times the rate suggested by Shannon’s sampling theorem.
- Use 4–10 samples per rise time of the closed-loop system.
- Use 15–45 samples per period of oscillation of the closed-loop system.
- The sampling frequency should be 10–30 times the bandwidth.
- Choose $\omega_g h = 0.15 \ldots 0.5$, where $\omega_g$ is the gain crossover frequency, i.e., the frequency where the amplitude ratio of the open-loop system is 1.

Note that we have not defined all above terms in this course.
### Tuning

Because the controller parameters in (10.12), (10.25) and (10.26) are functions of the parameters of a continuous-time PID controller, any tuning method for continuous-time PID controllers can be used.

Because the sampling increases the average time delay by half a sampling interval, it is advisable to increase the time delay of the system by $h/2$ before calculating the continuous-time controller parameters.

In the derivation of (10.17) and (10.24), it was not taken into account that the setpoint changes stepwise, and only at sampling instants. This kind of setpoint changes can be handled exactly. The correct handling is achieved if the derivative setpoint weight $\tilde{c}$ in the continuous case is changed to

- $c = \frac{h\tilde{c}}{T_f(1-e^{-h/T_f})}$ for the trapezoidal approximation

- $c = \frac{(T_f+h)\tilde{c}}{T_f}$ for the backward-difference approximation (this is approximate)
10.2 Digital PID controllers

Implementation in SIMULINK

Mathwork’s SIMULINK software provides two discrete PID controller blocks, one of which is a 2DOF controller. The type of discretization can be chosen, e.g., backward Euler or trapezoidal.

The controller parameters are

- $P = K_c$
- $I = K_c/T_i$
- $D = K_cT_d$
- $T_s = h$ (not adjustable here)

If the trapezoid approximation is used both for the integrator and the filter

- $N = \frac{2\left(1-e^{-h/Tf}\right)}{h\left(1+e^{-h/Tf}\right)}$

If the backward approximation is used

- $N = 1/T_f$

SIMULINK’s derivative implementation requires $N > 0$!
10. Digital Process Control

10.3 Sampling of time-delay models

10.3.1 State-space model with a time delay

Sampling

A continuous-time system with the same time delay $\theta$ for all inputs has the state-space representation

$$\dot{x}(t) = Ax(t) + Bu(t - \theta)$$  \hspace{2cm} (10.38)

If the system is sampled with the sampling interval $h$, the solution is

$$x(t_{k+1}) = e^{Ah}x(t_k) + e^{A(t_{k+1} - t_k)} \int_{t_k}^{t_{k+1}} e^{-A\tau}Bu(\tau - \theta) d\tau$$  \hspace{2cm} (10.39)

where $t_k$ and $t_{k+1}$ are two adjacent sampling points, $t_{k+1} = t_k + h$.

Assume that the time delay is smaller than the sampling interval, i.e., $0 \leq \theta < h$. If the input $u(t)$ is constant in the sampling interval $t_k \leq t < t_{k+1}$, $u(t - \theta)$ is not constant over the same interval if $\theta > 0$; the value of $u(t - \theta)$ changes from $u(t_{k-1})$ to $u(t_k)$ at the time $t = t_k + \theta$.

Because of this, it is useful to split the integral into two parts so that the input is constant in each part.
10.3.1 State-space model with a time delay

Splitting the integral in the indicated way yields

\[
x(t_{k+1}) = e^{Ah} x(t_k) + \left( e^{Ah} \int_{t_k}^{t_{k+\theta}} e^{-A\tau} d\tau \right) B u(t_{k-1}) \\
+ \left( e^{Ah} \int_{t_k+\theta}^{t_{k+1}} e^{-A\tau} d\tau \right) B u(t_k)
\] (10.40)

The solution can be written

\[
x(t_{k+1}) = F x(t_k) + G_0 u(t_k) + G_1 u(t_{k-1})
\] (10.41)

where

\[
F = e^{Ah}, \quad G_0 = \left( \int_0^{h-\theta} e^{A\tau} d\tau \right) B, \quad G_1 = e^{A(h-\theta)} \left( \int_0^\theta e^{A\tau} d\tau \right) B
\] (10.42)

The integrals can be calculated numerically according to

\[
S_t \equiv \int_0^t e^{A\tau} d\tau = \left( I + \frac{1}{2!} A t + \frac{1}{3!} A^2 t^2 + \frac{1}{4!} A^3 t^3 + \cdots \right) t
\] (10.43)

from which

\[
F = I + AS_h, \quad G_0 = S_{h-\theta} B, \quad G_1 = S_\theta B
\] (10.44)

If the time delay \( L \) is such that

\[
L = Nh + \theta, \quad N \text{ is a non-negative integer}
\] (10.45)

\( u(t_k) \) is replaced by \( u(t_{k-N}) \) and \( u(t_{k-1}) \) is replaced by \( u(t_{k-1-N}) \).
**Pulse transfer operator**

Generally, a state-space model with a time delay \( L = Nh + \theta, 0 \leq \theta < h \), for all inputs has the form

\[
\begin{align*}
\mathbf{x}(t_{k+1}) &= \mathbf{F}\mathbf{x}(t_k) + \mathbf{G}_0\mathbf{u}(t_{k-N}) + \mathbf{G}_1\mathbf{u}(t_{k-1-N}) \\
\mathbf{y}(t_k) &= \mathbf{C}\mathbf{x}(t_k) + \mathbf{D}_0\mathbf{u}(t_{k-N}) + \mathbf{D}_1\mathbf{u}(t_{k-1-N})
\end{align*}
\] (10.46)

The *shift operator* \( q \) operating on a signal \( f(t_k) \) is defined

\[
q f(t_k) = f(t_{k+1}) \iff q^{-1} f(t_{k+1}) = f(t_k)
\] (10.47)

Application of the shift operator to (10.46) yields

\[
\begin{align*}
q\mathbf{x}(t_k) &= \mathbf{F}\mathbf{x}(t_k) + \mathbf{G}_0 q^{-N}\mathbf{u}(t_k) + \mathbf{G}_1 q^{-1-N}\mathbf{u}(t_k) \\
\mathbf{y}(t_k) &= \mathbf{C}\mathbf{x}(t_k) + \mathbf{D}_0 q^{-N}\mathbf{u}(t_k) + \mathbf{D}_1 q^{-1-N}\mathbf{u}(t_k)
\end{align*}
\] (10.48)

from which

\[
\begin{align*}
\mathbf{x}(t_k) &= (\mathbf{qI} - \mathbf{F})^{-1}(\mathbf{G}_0 + \mathbf{G}_1 q^{-1})q^{-N}\mathbf{u}(t_k) \\
\mathbf{y}(t_k) &= [\mathbf{C}(\mathbf{qI} - \mathbf{F})^{-1}(\mathbf{G}_0 + \mathbf{G}_1 q^{-1}) + \mathbf{D}_0 + \mathbf{D}_1 q^{-1}]q^{-N}\mathbf{u}(t_k)
\end{align*}
\] (10.49)

Since the pulse transfer operator is defined by \( \mathbf{y}(t_k) = \mathbf{H}(q^{-1})\mathbf{u}(t_k) \), the sampled state-space model has the pulse transfer operator

\[
\mathbf{H}(q^{-1}) = [\mathbf{C}(\mathbf{qI} - \mathbf{F})^{-1}(\mathbf{G}_0 + \mathbf{G}_1 q^{-1}) + \mathbf{D}_0 + \mathbf{D}_1 q^{-1}]q^{-N}
\] (10.50)
10.3.2 Second-order model with real poles and zero

Diagonal state-space form

A second-order time-delay system with two distinct real poles and a possible zero has the transfer function

\[ G(s) = \frac{K(T_3s+1)e^{-Ls}}{(T_1s+1)(T_2s+1)} = \left( \frac{k_1}{s-\lambda_1} - \frac{k_2}{s-\lambda_2} \right) e^{-Ls} \] (10.51)

where

\[ \lambda_1 = -1/T_1, \quad \lambda_2 = -1/T_2 \] (10.52a)

\[ k_1 = \frac{K(T_1-T_3)}{T_1(T_1-T_2)}, \quad k_2 = \frac{K(T_3-T_2)}{T_2(T_1-T_2)} \] (10.52b)

The system can be written on diagonal state-space form as

\[ \dot{x}(t) = \Lambda x(t) + bu(t - L) \]
\[ y(t) = c^T x(t) \] (10.53)

\[ \Lambda = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}, \quad b = \begin{bmatrix} k_1 \\ k_2 \end{bmatrix}, \quad c = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \] (10.54)
10.3 Sampling of time-delay models

**Sampling**

Because $\Lambda$ is diagonal, sampling and calculation of $e^{Ah}$ is easy. With $L = Nh + \theta$, the result is

$$x(t_{k+1}) = Fx(t_k) + G_0u(t_k) + G_1u(t_{k-1})$$

$$y(t_k) = c^T x(t_k)$$

where

$$F = e^{Ah} = \begin{bmatrix} e^{\lambda_1 h} & 0 \\ 0 & e^{\lambda_2 h} \end{bmatrix}$$

(10.56)

$$G_0 = \left( \int_{0}^{h-\theta} e^{\Lambda \tau} d\tau \right) b = \begin{bmatrix} \lambda_1^{-1}(e^{\lambda_1(h-\theta)} - 1) & 0 \\ 0 & \lambda_2^{-1}(e^{\lambda_2(h-\theta)} - 1) \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \end{bmatrix}$$

$$= \begin{bmatrix} k_1 \lambda_1^{-1}(e^{\lambda_1(h-\theta)} - 1) \\ k_2 \lambda_2^{-1}(e^{\lambda_2(h-\theta)} - 1) \end{bmatrix}$$

(10.57)

$$G_1 = e^{\Lambda(h-\theta)} \left( \int_{0}^{\theta} e^{\Lambda \tau} d\tau \right) b$$

$$= \begin{bmatrix} e^{\lambda_1(h-\theta)} & 0 \\ 0 & e^{\lambda_2(h-\theta)} \end{bmatrix} \begin{bmatrix} \lambda_1^{-1}(e^{\lambda_1 \theta} - 1) & 0 \\ 0 & \lambda_2^{-1}(e^{\lambda_2 \theta} - 1) \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \end{bmatrix}$$

$$= \begin{bmatrix} k_1 \lambda_1^{-1} e^{\lambda_1 h}(1 - e^{-\lambda_1 \theta}) \\ k_2 \lambda_2^{-1} e^{\lambda_2 h}(1 - e^{-\lambda_2 \theta}) \end{bmatrix}$$

(10.58)
10.3 Sampling of time-delay models

**Pulse transfer operator**

According to (10.50), the pulse transfer operator is

\[ H(q^{-1}) = c^T(qI - F)^{-1}(G_0 + G_1q^{-1})q^{-N} \]  

(10.59)

Substitution of (10.57–58) into (10.59) gives a pulse transfer operator of the form

\[ H(q^{-1}) = \frac{b_1+b_2q^{-1}+b_3q^{-2}}{1+a_1q^{-1}+a_2q^{-2}} q^{-1-N} \]  

(10.60)

Using the parameters

\[ b_{11} = k_1 \lambda_1^{-1} \left( e^{\lambda_1 (h-\theta)} - 1 \right), \quad b_{12} = k_2 \lambda_2^{-1} \left( e^{\lambda_2 (h-\theta)} - 1 \right) \]

\[ b_{21} = k_1 \lambda_1^{-1} e^{\lambda_1 h} \left( 1 - e^{-\lambda_1 \theta} \right), \quad b_{22} = k_2 \lambda_2^{-1} e^{\lambda_2 h} \left( 1 - e^{-\lambda_2 \theta} \right) \]

(10.61)

the parameters of (10.60) are given by

\[ a_1 = -(e^{\lambda_1 h} + e^{\lambda_2 h}), \quad a_2 = e^{(\lambda_1+\lambda_2)h}, \quad b_1 = b_{11} + b_{12} \]

\[ b_2 = -(e^{\lambda_2 h} b_{11} + e^{\lambda_1 h} b_{12}) + b_{21} + b_{22} \]

\[ b_3 = -(e^{\lambda_2 h} b_{21} + e^{\lambda_1 h} b_{22}) \]

(10.62)
10.3.2 Second-order model

**The case $\theta = 0$**

If the time delay is an integer multiple the sampling interval, i.e., $L = Nh, \theta = 0$, the expressions for the second-order pulse transfer operator are simplified. In this case

$$H(q^{-1}) = \frac{b_1 + b_2 q^{-1}}{1 + a_1 q^{-1} + a_2 q^{-2}} q^{-1-N}$$  \hspace{1cm} (10.63)

In (10.61), $b_{21} = b_{22} = 0$ and

$$b_{11} = k_1 \lambda_1^{-1}(e^{\lambda_1 h} - 1), \quad b_{12} = k_2 \lambda_2^{-1}(e^{\lambda_2 h} - 1)$$  \hspace{1cm} (10.64)

The parameters of (10.63) are then given by

$$a_1 = -(e^{\lambda_1 h} + e^{\lambda_2 h}), \quad a_2 = e^{(\lambda_1 + \lambda_2)h}$$

$$b_1 = b_{11} + b_{12}, \quad b_2 = -(e^{\lambda_2 h} b_{11} + e^{\lambda_1 h} b_{12})$$  \hspace{1cm} (10.65)

**Exercise 10.3.2**

Determine the pulse transfer operator $H(q^{-1})$ for the system

$$G(s) = \frac{Ke^{-Ls}}{(T_1s+1)(T_2s+1)}$$

when $K = 1$, $L = 1$ min, $T_1 = 1$ min, $T_2 = 0.5$ min, $h = 0.1$ min.
10.3.3 First-order model

For a strictly proper first-order system with a time delay, the transfer function is

\[ G(s) = \frac{Ke^{-Ls}}{Ts+1} \]  
(10.66)

The sampling result and the pulse transfer operator can be directly obtained from the solution for the second-order system using

\[ k_1 = \frac{K}{T}, \quad \lambda_1 = -\frac{1}{T}, \quad \lambda_2 = -\infty \]

For the general time-delay case (i.e., arbitrary \( \theta \)), (10.61) and (10.62) yield the pulse transfer operator

\[ H(q^{-1}) = \frac{b_1 + b_2 q^{-1}}{1 + a_1 q^{-1}} q^{-1-N} \]  
(10.67)

with

\[ a_1 = -e^{-\theta/T}, \quad b_1 = K(1 + a_1 e^{\theta/T}), \quad b_2 = Ka_1(1 - e^{\theta/T}) \]  
(10.68)
10.4 Controller synthesis using discrete-time model

10.4.1 PID controller

Loop-transfer specification

If the controller and the system have the pulse transfer operators $H_c(q^{-1})$ and $H(q^{-1})$, respectively, the loop transfer operator is

$$H_\ell(q^{-1}) = H(q^{-1})H_c(q^{-1})$$  \hspace{1cm} (10.69)

resulting in the closed-loop pulse transfer operator

$$H_r(q^{-1}) = \frac{H_\ell(q^{-1})}{1+H_\ell(q^{-1})}$$  \hspace{1cm} (10.70)

For a system with the time delay $L = Nh + \theta$, $0 \leq \theta < h$, the loop transfer

$$H_\ell(q^{-1}) = \frac{q^{-1}-N}{(2N+1)(1-q^{-1})}$$  \hspace{1cm} (10.71)

gives a closed-loop system having approximately a 4% overshoot and a rise time equal to 3 ... 4 time delays. Such a control performance can be considered good.
10.4 Controller synthesis using discrete model

First-order model with $L = Nh$

According to (10.67–68), a first-order system with a time delay being an integer multiple of the sampling interval has the pulse transfer operator

$$H(q^{-1}) = \frac{b_1}{1+a_1q^{-1}} q^{-1-N} \quad (10.72)$$

The loop transfer operator (10.71) is achieved with the controller

$$H_c(q^{-1}) = H(q^{-1})^{-1} H_{\ell}(q^{-1})$$

$$= \frac{1+a_1q^{-1}}{b_1q^{-1-N}} \cdot \frac{q^{-1-N}}{(2N+1)(1-q^{-1})} = \frac{1}{(2N+1)b_1} \cdot \frac{1+a_1q^{-1}}{1-q^{-1}} \quad (10.73)$$

$$= \frac{1}{(2N+1)b_1} \cdot \frac{1+a_1q^{-1}}{1-q^{-1}} \quad (10.74)$$

Since $H_c(q^{-1}) = U(q^{-1})/E(q^{-1})$, this is a controller implementing the incremental control law

$$u(t_k) = u(t_{k-1}) + \frac{1}{(2N+1)b_1} (e(t_k) + a_1 e(t_{k-1})) \quad (10.75)$$

where $e(t_k) = r(t_k) - y(t_k)$ is the standard control error. Comparison with (10.33–37) shows that this is a PI controller with

$$b_{i1} + K_c = \frac{1}{(2N+1)b_1} \quad b_{i2} - K_c = \frac{a_1}{(2N+1)b_1}$$
10.4.1 PID controller

If the trapezoidal approximation is used, this yields a controller with

\[
K_c = \frac{1-a_1}{2(2N+1)b_1}, \quad b_{i1} = b_{i2} = \frac{1+a_1}{2(2N+1)b_1} \Rightarrow T_i = \frac{h}{2} \cdot \frac{1-a_1}{1+a_1}
\]  

(10.76)

**Implementation**

The controller can be implemented in various ways using **SIMULINK**.

- One way is to implement the **positional controller** as explained in Section 10.2.3. As this is a PI controller, \( D = 0 \) should then be used.

- Another way is to implement the incremental form using the **Discrete Zero-Pole** block. The variable \( z \) used in the block definition is obtained by replacing the operator \( q \) by \( z \). Since

\[
H_c(q^{-1}) = \frac{1}{(2N+1)b_1} \cdot \frac{1+a_1 q^{-1}}{1-q^{-1}} \Leftrightarrow H_c(z) = \frac{1}{(2N+1)b_1} \cdot \frac{z+a_1}{z-1}
\]

the pole is +1, the zero is \(-a_1\) and the gain is \(1/(2N + 1)b_1\) for this block.

- The **Discrete Transfer Fcn** block can also be used. In this block, the numerator and denominator polynomials are defined. In this case, the numerator parameters are \([1 \ a_1]\) and the denominator parameters are \([1 \ -1]\).
### Second-order model with $L = Nh$

A second-order system with a time delay equal to an integer multiple of the sampling interval has the pulse transfer operator (10.63). Substitution into (10.73) yields

$$H_c(q^{-1}) = \frac{1+a_1q^{-1}+a_2q^{-2}}{(b_1+b_2q^{-1})q^{-1-N}} \cdot \frac{q^{-1-N}}{(2N+1)(1-q^{-1})}$$

$$= \frac{1}{(2N+1)b_1} \cdot \frac{1+a_1q^{-1}+a_2q^{-2}}{(1-q^{-1})(1+dq^{-1})} \quad (10.77)$$

where

$$d = \frac{b_2}{b_1} \quad (10.78)$$

This is a controller implementing the incremental control law

$$u(t_k) = u(t_{k-1}) + \frac{1}{(2N+1)b_1} \left( e(t_k) + a_1 e(t_{k-1}) + a_2 e(t_{k-2}) \right)$$

$$-d(u(t_{k-1}) - u(t_{k-2})) \quad (10.79)$$

Comparison with (10.33–37) shows that this is a **PID controller with a derivative filter**.
10.4.1 PID controller

**Implementation**

Using SIMULINK there are again various implementation options

- Easiest is to use the Discrete Transfer Fcn block. In this case the numerator parameters are \([1 \ a_1 \ a_2]\), the denominator parameters are \([1 \ (d-1) \ -d]\), and the gain is \(1/(2N+1)b_1\).

- It is also possible to determine the controller parameters \(K_c, T_i, T_d\) and \(T_f\) and to use the Discrete PID Controller block as explained in Section 10.2.3. If the trapezoidal approximation is used

\[
a_d = d \Rightarrow T_f = -h/\ln d \quad (10.80a)
\]

\[
T_i = h \left( \frac{1-a_2}{1+a_1+a_2} - \frac{1-d}{2(1+d)} \right) \quad (10.80b)
\]

\[
K_c = \frac{T_i}{h} \cdot \frac{1+a_1+a_2}{(1+d)(2N+1)b_1} \quad (10.80c)
\]

\[
T_d = h \frac{T_i}{T_i} \left( \frac{1}{1+a_1+a_2} - \frac{1}{2(1+d)} \right) \quad (10.80d)
\]
10.4.2 Time-delay compensation

**Closed-loop specification**

A more direct way of specifying the desired control performance is to specify the desired closed-loop performance instead of the loop transfer. Eq. (10.69) and (10.70) yield the closed-loop pulse transfer operator

$$H_r(q^{-1}) = \frac{H(q^{-1})H_c(q^{-1})}{1+H(q^{-1})H_c(q^{-1})}$$  \hspace{1cm} (10.81)

where $H(q^{-1})$ and $H_c(q^{-1})$ are the pulse transfer operators of the process and the controller. From (10.81) the controller needed to achieve $H_r(q^{-1})$ can be derived as done in “direct synthesis”, i.e.,

$$H_c(q^{-1}) = \frac{1}{H(q^{-1})} \cdot \frac{H_r(q^{-1})}{1-H_r(q^{-1})}$$  \hspace{1cm} (10.82)

The closed-loop system cannot have a smaller time delay than the open loop system. Assume that the open-loop system has the time delay $L = Nh + \theta$, $0 \leq \theta < h$. According to (10.67), a first-order closed-loop system with this time delay and the gain $K = 1$, which is desired for setpoint tracking, has a pulse transfer operator of the form
10.4.2 Time-delay compensation

\[ H_r(q^{-1}) = \frac{1-\beta+(\beta-\alpha)q^{-1}}{1-\alpha q^{-1}} q^{-1-N} \]  

(10.83)

where \( \alpha = e^{-h/T_r}, \beta = \alpha e^{\theta/T_r}, \) and \( T_r \) is the desired closed-loop time constant. Note that \( \beta = \alpha \) if \( \theta = 0 \), i.e., if the time delay is an integer multiple of the sampling interval, \( L = Nh \). This choice of \( H_r(q^{-1}) \) yields

\[ \frac{H_r(q^{-1})}{1-H_r(q^{-1})} = \frac{1-\beta+(\beta-\alpha)q^{-1}}{1-\alpha q^{-1} - [1-\beta+(\beta-\alpha)q^{-1}]q^{-1-N}} q^{-1-N} \]  

(10.84)

**First-order time-delay system**

A first-order time-delay system has the pulse transfer operator given in (10.67). Substitution of (10.67) and (10.84) into (10.82) gives

\[ H_c(q^{-1}) = \frac{1+a_1 q^{-1}}{b_1+b_2 q^{-1}} \cdot \frac{1-\beta+(\beta-\alpha)q^{-1}}{1-\alpha q^{-1} - [1-\beta+(\beta-\alpha)q^{-1}]q^{-1-N}} \]  

(10.85)

\[ = \frac{c_0 (1+a_1 q^{-1})(1+d_2 q^{-1})}{(1+d_1 q^{-1})[1-\alpha q^{-1}-(1-\beta)(1+d_2 q^{-1})q^{-1-N}]} \]  

(10.86)

where

\[ c_0 = \frac{1-\beta}{b_1}, \quad d_1 = \frac{b_2}{b_1}, \quad d_2 = \frac{\beta-\alpha}{1-\beta} \]  

(10.87)
Eq. (10.86) gives a control law in difference form as
\[ u(t_k) = c_0 [e(t_k) + (a_1 + d_2)e(t_{k-1}) + a_1 d_2 e(t_{k-2})] 
+ (\alpha - d_1)u(t_{k-1}) + \alpha d_1 u(t_{k-2}) 
+ (1 - \beta)[u(t_{k-1-N}) + (d_1 + d_2)u(t_{k-2-N}) + d_1 d_2 u(t_{k-3-N})] \]  

(10.88)

This is a PID-like controller (because of the \( e \) terms) with delayed control signals to compensate for the time delay.

If the time delay is an integer multiple of the sampling interval, i.e., if \( \theta = 0 \), then \( b_2 = 0, \ d_1 = d_2 = 0, \ \beta = \alpha \), and the control law reduces to the PI-like control law
\[ u(t_k) = c_0 [e(t_k) + a_1 e(t_{k-1})] 
+ \alpha u(t_{k-1}) + (1 - \alpha)u(t_{k-1-N}) \]  

(10.89)

The discrete-time control laws (10.88), (10.89) and (10.91), with explicit compensation for the time delay, are known as Dahlin controllers (or Dahlin-Higham’s algorithms). Because quite old information is used in the form of delayed control signals, they can be expected to be sensitive to model errors, especially concerning the time delay.
Second-order time-delay system with $L = Nh$

A second-order time-delay system, where the time delay is an integer multiple of the time delay, has the pulse transfer operator (10.63). Substitution of (10.63) and (10.84) into (10.82) yields

$$H_c(q^{-1}) = \frac{1+a_1q^{-1}+a_2q^{-2}}{b_1+b_2q^{-1}} \cdot \frac{1-\alpha}{1-\alpha q^{-1}-(1-\alpha)q^{-1-N}}$$ (10.90)

This gives the PID-like control law

$$u(t_k) = c_0[e(t_k) + a_1 e(t_{k-1}) + a_2 e(t_{k-2})]$$

$$+ (\alpha - d_1)u(t_{k-1}) + \alpha d_1 u(t_{k-2})$$

$$+ (1 - \alpha)[u(t_{k-1-N}) + d_1 u(t_{k-2-N})]$$ (10.91)

with $c_0$ and $d_1$ as given in (10.87).

Note that the control law (10.91) for a second-order system with $L = Nh$ is less complicated than the control law (10.88) for a first-order system with $L > Nh$.

**Implementation**

In **SIMULINK**, the control laws (10.88), (10.89) and (10.91) can be implemented by a **Discrete Transfer Fcn** block. Note that the denominator polynomial will have many zeroes for the missing inputs up to $u(t_{k-1-N})$. 

---

Closed-loop specification
10.4.3 Dead-beat control

In the controller synthesis according to Dahlin-Higham’s method, the time constant \( T_r \) of a desired first-order closed-loop system is specified. How short can \( T_r \) be? The theoretical minimum is \( T_r = 0 \), of course. This results in the closed-loop transfer operator as a pure time delay

\[
H_r(q^{-1}) = q^{-1-N} \tag{10.92}
\]

Substitution into (10.82) yields the controller as

\[
H_c(q^{-1}) = \frac{1}{H(q^{-1})} \cdot \frac{q^{-1-N}}{1-q^{-1-N}} \tag{10.93}
\]

where \( q^{-1-N} \) can be cancelled against the same operator in the pulse transfer operator \( H(q^{-1}) \) of the model.

This kind of controller is called a dead-beat controller. There is no corresponding continuous-time controller since \( T_r = 0 \) would require an infinite controller gain in the continuous-time case.
10.4 Controller synthesis using discrete model

First-order time-delay model

A first-order time-delay system has the pulse transfer operator given in (10.67). Substitution of (10.67) into (10.93) gives

\[ H_c(q^{-1}) = \frac{1+a_1q^{-1}}{b_1+b_2q^{-1}} \cdot \frac{1}{1-q^{-1}-N} = \frac{b_1^{-1}(1+a_1q^{-1})}{(1+d_1q^{-1})(1-q^{-1}-N)} \] (10.94)

where \( d_1 \) is as defined in (10.87). The control law in difference form is

\[ u(t_k) = b_1^{-1}[e(t_k) + a_1 e(t_{k-1})] + u(t_{k-1-N}) - d_1[u(t_{k-1}) - u(t_{k-2-N})] \] (10.95)

This is PI-like control law where the dominating old control signal is \( u(t_{k-1-N}) \) instead of \( u(t_{k-1}) \). Note that \( d_1 = 0 \) if \( L = Nh \) (i.e., \( \theta = 0 \)).

Second-order time-delay system with \( L = Nh \)

For a second-order time-delay system with \( L = Nh \), the control law becomes

\[ u(t_k) = b_1^{-1}[e(t_k) + a_1 e(t_{k-1}) + a_2 e(t_{k-2})] + u(t_{k-1-N}) - d_1[u(t_{k-1}) - u(t_{k-2-N})] \] (10.96)

which is a PID-like controller.
### 10.4.4 Improving intersample behaviour

#### Intersample ripple

A possible problem with discrete-time control of a continuous-time system is the behaviour between the sample points, as illustrated by the figure.

- If only the sample points of the output $y$ are considered, the step response looks like the response of a first-order system.
- The true continuous-time value of $y$ oscillates; this is called **intersample ripple** (or *ringing*).
- The input signal oscillates at sample points (and in continuous time).

![Intersample ripple diagram](image)

○ = sample point, full line = continuous-time value [SEMD]
10.4.4 Improving intersample behaviour

In particular, intersample ripple can occur when a discrete-time controller has been designed based on the inverse of a discrete-time (sampled) model.

- When a continuous-time model is sampled using standard techniques, the transfer function of the sampled model often receives a zero close to $-1$ (but slightly $> -1$) even if the continuous-time model has no zero;
  
  - e.g. $G(s) = \frac{Ke^{-Ls}}{(T_1s+1)(T_2s+1)} \Rightarrow H(q^{-1}) = \frac{b_0 + b_1q^{-1}}{1 + a_1q^{-1} + a_2q^{-2}} q^{-1-N}, \frac{b_1}{b_0} < \approx 1$.

- If the controller design is based on the inverse of $H(q^{-1})$, $b_0 + b_1q^{-1}$ becomes a pole of the controller, i.e., a controller pole close to $-1$, which causes intersample ripple.
10.4 Controller synthesis using discrete model

### Elimination of intersample ripple

There are basically two ways to eliminate intersample ripple.

**Dahlin’s modification**

According to Dahlin’s modification, the denominator $b_1 + b_2 q^{-1}$ of the model used in the controller synthesis is replaced by $b_1 + b_2$. Instead of the model (10.63), for example,

$$H(q^{-1}) = \frac{b_1 + b_2}{1 + a_1 q^{-1} + a_2 q^{-2}} q^{-1-N}$$

(10.97)

is used. The figure illustrates the control performance with such a modification. There is

- no intersample ripple
- no (extensive) control signal oscillations
10.4.4 Improving intersample behaviour

**Vogel-Edgar modification**

Vogel and Edgar have suggested that $b_1 + b_2 q^{-1}$ be included in the numerator of the desired closed-loop model. Instead of, for example, the desired first-order model

$$H_r(q^{-1}) = \frac{1 - \alpha}{1 - \alpha q^{-1}}$$

the controller synthesis is done to obtain the closed-loop transfer operator

$$H_r(q^{-1}) = \frac{(1 - \alpha)(b_1 + b_2 q^{-1})}{1 - \alpha q^{-1}} q^{-1-N}$$

(10.99)

The figure shows the control performance with this modification. There is

- no intersample ripple
- no control signal oscillations
- smoother but slightly slower response than with Dahlin’s modification
10. Digital Process Control

10.5 Example: comparison of various controllers

10.5.1 Example system

In Exercise 10.3.2 the pulse transfer operator was determined for the system

\[ G(s) = \frac{Ke^{-Ls}}{(T_1s+1)(T_2s+1)} \]  \hspace{1cm} (10.100)

with \( K = 1 \), \( L = 1 \) min, \( T_1 = 1 \) min, \( T_2 = 0.5 \) min, \( h = 0.1 \) min. The result was

\[ H(q^{-1}) = \frac{b_1 + b_2 q^{-1}}{1 + a_1 q^{-1} + a_2 q^{-2}} q^{-1-N} \]  \hspace{1cm} (10.101)

with

\[ a_1 = -1.7236, \ a_2 = 0.7408, \ b_1 = 0.009056, \ b_2 = 0.008194, \ N = 10. \]

Four different discrete-time controllers are to be compared:

- a discretized PID controller tuned by Ziegler-Nichols’s method
- a discrete PID controller tuned for small overshoot (4%)
- a Dahlin controller with \( T_r = 0.5 \) min and wrong model
- a Dahlin controller with \( T_r = 0.5 \) min and correct model
10.5.2 Discretized PID with Ziegler-Nichols tuning

The system (10.100) has the phase shift

$$\varphi = -L \omega - \arctan T_1 \omega - \arctan T_2 \omega$$

$$= -1 \omega - \arctan 1 \omega - \arctan 0.5 \omega$$

which for the phase shift $\varphi = -\pi$ yields the critical frequency $\omega_c = 1.5094$ rad/min. The amplitude ratio at this frequency is

$$|G(\omega_c)| = \frac{K}{\sqrt{1+(T_1 \omega_c)^2}\sqrt{1+(T_2 \omega_c)^2}} = \frac{1}{\sqrt{1+1.5094^2\sqrt{1+0.25\cdot1.5094^2}}} = 0.4408$$

from which the ultimate gain $K_{c,\text{max}} = 1/0.4408 \approx 2.27$. According to Ziegler-Nichols’ recommendations for a PID controller, this yields $K_c = 0.6K_{c,\text{max}} = 1.36$, $T_i = \pi/\omega_c = 2.08$ min, $T_d = 0.25T_i = 0.52$ min.

Combination of (10.33–37) and the parameters for the backward-difference approximation gives for an ideal PID controller

$$u(t_k) = K_c \left[ \left( 1 + \frac{h}{T_i} + \frac{T_d}{h} \right) e(t_k) - \left( 1 - \frac{2T_d}{h} \right) e(t_{k-1}) + \frac{T_d}{h} e(t_{k-2}) \right] + u(t_{k-1})$$

which here results in

$$u(t_k) = 8.51e(t_k) - 15.5e(t_{k-1}) + 7.08e(t_{k-2}) + u(t_{k-1})$$
10.5.3 **Discrete PID tuned for small overshoot**

The pulse transfer operator (10.101) yields the controller (10.79), i.e.,

\[
u(t_k) = \frac{1}{(2N+1)b_1} [e(t_k) + a_1 e(t_{k-1}) + a_2 e(t_{k-2})] + u(t_{k-1})
\]

\[+ d[u(t_{k-1}) - u(t_{k-2})]\]

where \(d = b_2 / b_1\). Numerically, this results in

\[
u(t_k) = 5.26e(t_k) - 9.06e(t_{k-1}) + 3.90e(t_{k-2}) + u(t_{k-1})
\]

\[ - 0.905[u(t_{k-1}) - u(t_{k-2})]\]
10. Digital Process Control

10.5 Example: comparison of controllers

10.5.4 Dahlin controller and wrong model

Assume that

\[ G(s) = \frac{Ke^{-Ls}}{Ts+1} \]

with \( K = 1 \), \( L = 1 \) min, and \( T = 1.5 \) min is thought to be the correct model. The corresponding pulse transfer operator is then

\[ H(q^{-1}) = \frac{b_1}{1+a_1q^{-1}}q^{-1-N} \]

with \( a_1 = e^{-h/T} \approx 0.9355 \), \( b_1 = K(1 + a_1) \approx 0.06449 \), \( N = 10 \). According to (10.89), the control law is

\[
\begin{align*}
    u(t_k) &= \frac{1-\alpha}{b_1} [e(t_k) + a_1e(t_{k-1})] + u(t_{k-1}) \\
    &\quad + (1 - \alpha)[u(t_{k-1}) - u(t_{k-1-N})]
\end{align*}
\]

With \( \alpha = e^{-h/T_r} = e^{-0.1/0.5} \approx 0.8187 \), this gives

\[
\begin{align*}
    u(t_k) &= 2.81e(t_k) - 2.63e(t_{k-1}) + u(t_{k-1}) \\
    &\quad - 0.181[u(t_{k-1}) - u(t_{k-11})]
\end{align*}
\]
10.5.5 Dahlin controller and correct model

With the correct model, the Dahlin controller is given by (10.91):

\[ u(t_k) = c_0 [e(t_k) + a_1 e(t_{k-1}) + a_2 e(t_{k-2})] + (\alpha - d_1)u(t_{k-1}) + \alpha d_1 u(t_{k-2}) + (1 - \alpha)[u(t_{k-1-N}) + d_1 u(t_{k-2-N})] \]

where \( c_0 = (1 - \alpha)/b_1 \) and \( d_1 = b_2/b_1 \). \( \alpha = e^{-h/T_r} \approx 0.8187 \) gives

\[ u(t_k) = 20.0 e(t_k) - 34.5 e(t_{k-1}) + 14.8 e(t_{k-2}) - 0.086 u(t_{k-1}) + 0.741 u(t_{k-2}) - 0.181 u(t_{k-11}) + 0.164 u(t_{k-12}) \]

10.5.6 Simulations

- Ziegler-Nichols PID (—)
- Small overshoot PID (— —)
- Dahlin controller with wrong model (— ⋅ —)
- Dahlin controller with correct model (⋅⋅⋅⋅)