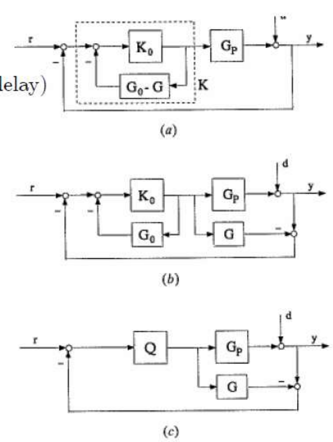
PID control.
Practical issues

Smith Predictor (NOT PID…)
PID Controller forms
Ziegler-Nichols tuning
Windup
Digital implementation

Smith Predictor

Actual plant: G_p
Model: G
Delay-free model: G_0
"Smith predictor": $G_0 - G$ (predicts y when G has delay)
Conventional feedback controller: K
K_0: designed for plant without delay

Example
$G = k\frac{e^{-\theta s}}{\tau s+1}$
Delay-free model: $G_0 = k\frac{1}{\tau s+1}$
$G_0 - G = \frac{k}{\tau s+1}(1 - e^{-\theta s})$
Then
$K = \frac{K_0}{1+K_0\frac{k}{\tau s+1}(1-e^{-\theta s})}$
which with $K_0 = \frac{1}{k}\frac{\tau s+1}{\tau_c s}$
(SIMC-PI for delay free G_0)
gives "Smith predictor controller"
$K = \frac{\tau s+1}{\tau_c s+1-e^{-\theta s}}$
(see also SIMC derivation)



a) Smith predictor control structure; (*b*) rearranged Smith predictor; (*c*) IMC structure.

SP looks good in theory. BUT: We have found that well-tuned PID (with $\tau_D = \theta/3$) is more robust and almost always better than Smith predictor controller ⇒ FORGET SP!

PID controller

"Ideal" form: $u(t) = u_0 + \underbrace{K_c[e(t) + \frac{1}{\tau_I}\int_0^t e(t)dt + \tau_D \frac{de(t)}{dt}]}_{\Delta u}$

- $e(t) = y_s - y_m(t)$
- P-part: MV (Δu) proportional to error
  - This is is main part of the controller!
  - Problem. Gives steady-state offset = 100%/(1+K_ck)
- I-part: To avoid offset, add contribution proportional to integrated error.
  - Integral keeps changing as long as e≠0
  - -> Will eventually make e=0 (no steady-state offset!)
- Possible D-part: Add contribution proportional to change in (derivative of) error
  - Can improve control for high-order (S-shaped) response, but sensitive to measurement noise

"Ideal" PID (parallel form)

The *parallel form* of the PID control algorithm (without a derivative filter) is given by

$$p(t) = \bar{p} + K_c\left[e(t) + \frac{1}{\tau_I}\int_0^t e(t^*)\,dt^* + \tau_D\frac{de(t)}{dt}\right] \quad (7\text{-}13)$$

The corresponding transfer function is

$$\frac{P'(s)}{E(s)} = K_c\left[1 + \frac{1}{\tau_I s} + \tau_D s\right] \quad (7\text{-}14)$$

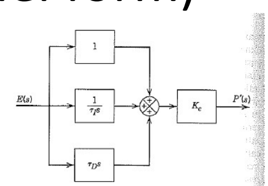


Figure 7.8 Block diagram of the parallel form of PID control (without a derivative filter).

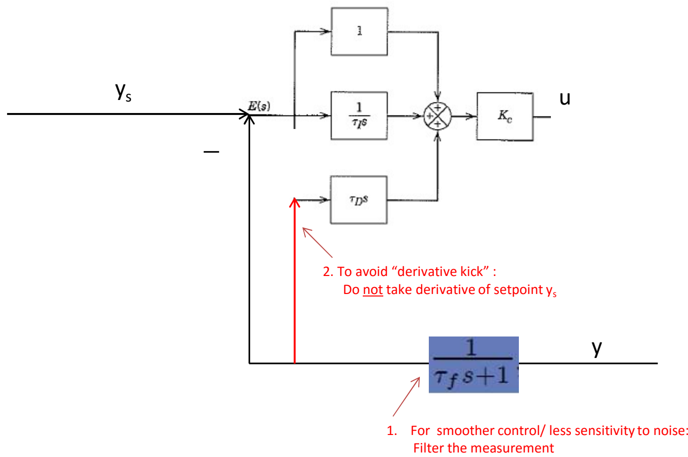
1. To get realizable controller and less sensitivity to measurement noise: Add filter on the measurement: $f(s) = \frac{1}{\tau_f s+1}$, where $\tau_f = \alpha\tau_d$. Typical: $\alpha = 0.1$
2. To avoid "derivative kick" do <u>not</u> take derivative of setpoint

   illustrate the elimination of derivative kick, consider the parallel form of PID control in Eq. 7-13. Replacing de/dt by $-dy_m/dt$ gives

$$p(t) = \bar{p} + K_c\left[e(t) + \frac{1}{\tau_I}\int_0^t e(t^*)\,dt^* - \tau_D\frac{dy_m(t)}{dt}\right] \quad (7\text{-}17)$$

3. To avoid integral "windup" when input saturates (at max or min), use "anti" windup.
   Simplest: Stop integration while input saturates

Block diagram of practical "ideal" PID

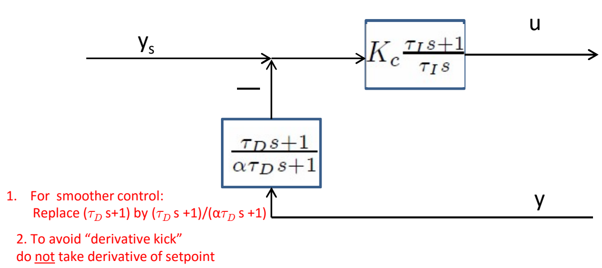


Series (cascade) PID

Commercial versions of the series-form controller have a derivative filter that is applied to either the derivative term, as in Eq. 7-12, or to the PD term, as in Eq. 7-15:

$$\frac{P'(s)}{E(s)} = K_c\left(\frac{\tau_I s + 1}{\tau_I s}\right)\left(\frac{\tau_D s + 1}{\alpha\tau_D s + 1}\right) \qquad (7\text{-}15)$$

Typical: α=0.1



Series to ideal form

Series (cascade) PID:

$$c(s) = K_c \frac{(\tau_I s + 1)(\tau_D s + 1)}{\tau_I s} = \frac{K_c}{\tau_I s}(\tau_I \tau_D s^2 + (\tau_I + \tau_D)s + 1)$$

The settings given in this paper (K_c, τ_I, τ_D) are for the series (cascade, "interacting") form PID controller in (1). To derive the corresponding settings for the ideal (parallel, "non-interacting") form PID controller

$$\text{Ideal PID}: \quad c'(s) = K_c'\left(1 + \frac{1}{\tau_I' s} + \tau_D' s\right) = \frac{K_c'}{\tau_I' s}\left(\tau_I' \tau_D' s^2 + \tau_I' s + 1\right) \tag{35}$$

we use the following translation formulas

$$K_c' = K_c\left(1 + \frac{\tau_D}{\tau_I}\right); \quad \tau_I' = \tau_I\left(1 + \frac{\tau_D}{\tau_I}\right); \quad \tau_D' = \frac{\tau_D}{1 + \frac{\tau_D}{\tau_I}} \tag{36}$$

Derivation: See exercise

Note: The reverse transformation (from ideal to series) is not always possible because the ideal controller may have complex zeros.

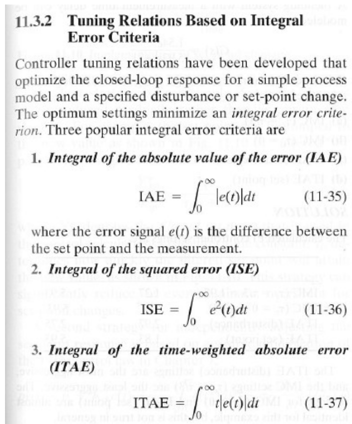
Table 7.1 Common PID Controllers

Controller Type	Other Names Used	Controller Equation	Transfer Function
Parallel	Ideal, additive, ISA form	$p(t) = \bar{p} + K_c\left(e(t) + \frac{1}{\tau_I}\int_0^t e(t^*)\,dt^* + \tau_D \frac{de(t)}{dt}\right)$	$\frac{P'(s)}{E(s)} = K_c\left(1 + \frac{1}{\tau_I s} + \tau_D s\right)$
Parallel with derivative filter	Ideal, realizable, ISA standard	*See Exercise 7.10(a)*	$\frac{P'(s)}{E(s)} = K_c\left(1 + \frac{1}{\tau_I s} + \frac{\tau_D s}{\alpha \tau_D s + 1}\right)$
Series	Multiplicative, interacting	*See Exercise 7.11*	$\frac{P'(s)}{E(s)} = K_c\left(\frac{\tau_I s + 1}{\tau_I s}\right)(\tau_D s + 1)$
Series with derivative filter	Physically realizable	*See Exercise 7.10(b)*	$\frac{P'(s)}{E(s)} = K_c\left(\frac{\tau_I s + 1}{\tau_I s}\right)\left(\frac{\tau_D s + 1}{\alpha \tau_D s + 1}\right)$
Expanded	Noninteracting	$p(t) = \bar{p} + K_c e(t) + K_I \int_0^t e(t^*)\,dt^* + K_D \frac{de(t)}{dt}$	$\frac{P'(s)}{E(s)} = K_c + \frac{K_I}{s} + K_D s$
Parallel, with proportional and derivative weighting	Ideal β, γ controller	$p(t) = \bar{p} + K_c\left(e_P(t) + \frac{1}{\tau_I}\int_0^t e(t^*)\,dt^* + \tau_D \frac{de_D(t)}{dt}\right)$ where $e_P(t) = \beta y_{sp}(t) - y_m(t)$, $e(t) = y_{sp}(t) - y_m(t)$, $e_D(t) = \gamma y_{sp}(t) - y_m(t)$	$P'(s) = K_c\left(E_P(s) + \frac{1}{\tau_I s}E(s) + \tau_D s E_D(s)\right)$ where $E_P(s) = \beta Y_{sp}(s) - Y_m(s)$, $E(s) = Y_{sp}(s) - Y_m(s)$, $E_D(s) = \gamma Y_{sp}(s) - Y_m(s)$

+ many more (see manual for your control system…)

Optimal PID settings

- Can find optimal settings using optimization
- SIMC-rules are close to IAE-optimal for combined setpoints and disturbances*



11.3.2 Tuning Relations Based on Integral Error Criteria

Controller tuning relations have been developed that optimize the closed-loop response for a simple process model and a specified disturbance or set-point change. The optimum settings minimize an *integral error criterion*. Three popular integral error criteria are

1. *Integral of the absolute value of the error (IAE)*

$$\text{IAE} = \int_0^\infty |e(t)|dt \qquad (11\text{-}35)$$

where the error signal $e(t)$ is the difference between the set point and the measurement.

2. *Integral of the squared error (ISE)*

$$\text{ISE} = \int_0^\infty e^2(t)dt \qquad (11\text{-}36)$$

3. *Integral of the time-weighted absolute error (ITAE)*

$$\text{ITAE} = \int_0^\infty t|e(t)|dt \qquad (11\text{-}37)$$

*Chriss Grimholt and Sigurd Skogestad. "Optimal PI-Control and Verification of the SIMC Tuning Rule". Proceedings *IFAC conference on Advances in PID control* (PID'12), Brescia, Italy, 28-30 March 2012.

Methods for *online* tuning of PID controllers

I. Trial and error

II. Ziegler Nichols
   - Oscillating P-control
   - Relay method to get oscillations

III. Closed-loop response with P-control
   - Shams method

On-line tuning: Avoids an open-loop experiment, like a step input change.
Advantage on-line: Process is always "under control"
In practice: Both "open-loop" and "closed-loop" (online) methods are used

Tuning of your PID controller

I. "Trial & error" approach (online)

(a) P-part: Increase controller gain (K_c) until the process starts oscillating or the input saturates

(b) Decrease the gain (~ factor 2)

(c) I-part: Reduce the integral time (τ_I) until the process starts oscillating

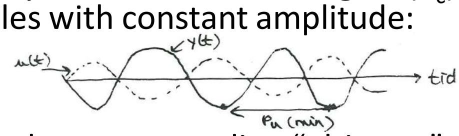
(d) Increase a bit (~ factor 2)

(e) Possible D-part: Increase τ_D and see if there is any improvement

Very common approach,
BUT: Time consuming and does not give good tunings: NOT recommended

II. Ziegler-Nichols closed-loop method (1942)

- P-control only: Increase controller gain (K_c) until the process cycles with constant amplitude:



- Write down the corresponding "ultimate" period (P_u) and controller gain (K_u).
- Based on this "process information" obtain PID settings:

Table 11.4 Controller Settings based on the Continuous Cycling Method

Ziegler-Nichols	K_c	τ_I	τ_D
P	$0.5K_{cu}$	—	—
PI	$0.45K_{cu}$	$P_u/1.2$	—
PID (ideal)	$0.6K_{cu}$	$P_u/2$	$P_u/8$
Tyreus-Luyben[†]	K_c	τ_I	τ_D
PI	$0.31K_{cu}$	$2.2P_u$	—
PID	$0.45K_{cu}$	$2.2P_u$	$P_u/6.3$

[†] Luyben and Luyben (1997).

ZN is often a bit aggressive.

TL-modification is smoother (smaller K_c and larger τ_I).

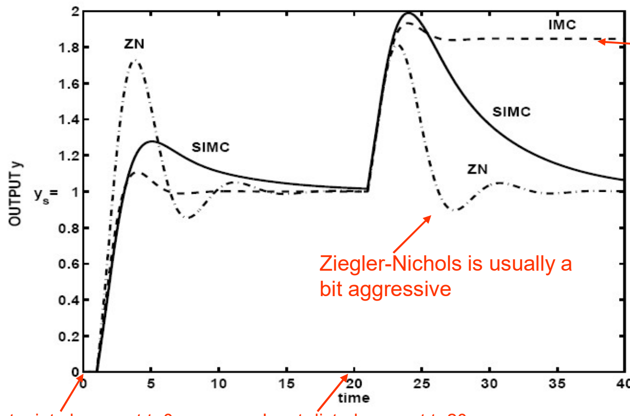
TIGHT CONTROL

Example. Integrating process with delay=1. G(s) = e^{-s}/s.

Model: k'=1, θ=1, $\tau_1=\infty$

SIMC-tunings with τ_c with =θ=1:

$$K_c = \frac{1}{k'}\frac{1}{\tau_c+\theta} = 1 \cdot \frac{1}{1+1} = 0.5$$
$$\tau_I = \min(\tau_1, 4(\tau_c+\theta)) = \min(\infty, 8) = 8$$



IMC has $\tau_I=\infty$

Ziegler-Nichols is usually a bit aggressive

Setpoint change at t=0c

Input disturbance at t=20

TIGHT CONTROL

EXAMPLE: Process from Astrom et al. (1998)

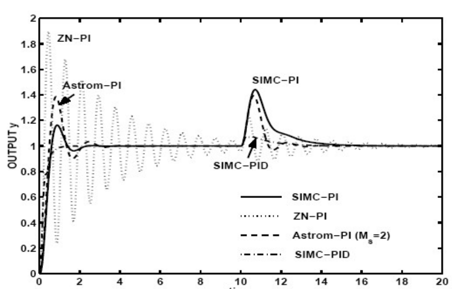


Figure 3: Load disturbance of magnitude 2 occurs at $t = 10$.

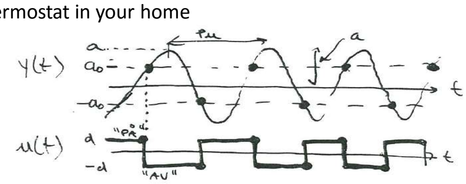
$$g_0(s) = \frac{1}{(s+1)(0.2s+1)(0.04s+1)(0.008s+1)}$$

1. Approximate as first-order model with k=1, τ_1 = 1+0.1=1.1, θ=0.1+0.04+0.008 = 0.148
   Get SIMC PI-tunings (τ_c=θ): K_c = 1 · 1.1/(2· 0.148) = 3.71, τ_I=min(1.1,8· 0.148) = 1.1

2. Approximate as second-order model with k=1, τ_1 = 1, τ_2=0.2+0.02=0.22, θ=0.02+0.008 = 0.028
   Get SIMC PI**D**-tunings (τ_c=θ): K_c = 1 · 1/(2· 0.028) = 17.9, τ_I=min(1,8· 0.028) = 0.224, τ_D=0.22

Åstrøm relay method (1984): Alternative approach to obtain cycling (and K_u+P_u)

- Avoids operating at limit to instability
- Use ON/OFF controller (=relay) were input u(t) varies ± d (around nominal)
- Switch when output y(t) reaches ± a_0 (deadband) (around setpoint)
- Example: Thermostat in your home



- From this obtain P_u and

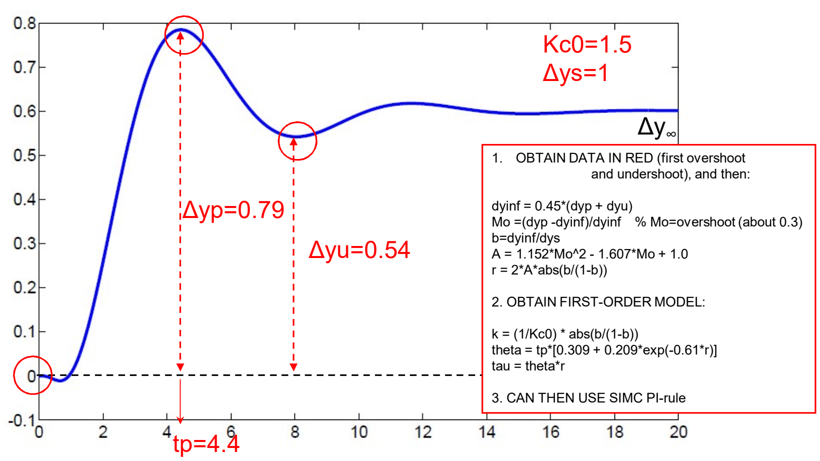
$$K_u = \frac{4}{¼}\frac{d}{a}$$

d: amplitude u(t) (set by user)

a: amplitude y(t) (from experiment)

Alternative to Ziegler-Nichols closed-loop experiment that avoids cycling.

III. Shams' method: Closed-loop setpoint response with P-controller with about 20-40% overshoot



Example 2: Get k=0.99, theta =1.68, tau=3.03

Ref: Shamssuzzoha and Skogestad (JPC, 2010)
+ modification by C. Grimholt (Project, NTNU, 2010; see also PID-book 2012)

Example E2 (Further continued) We want to derive PI- and PID-settings for the process

$$g_0(s) = \frac{(-0.3s+1)(0.08s+1)}{(2s+1)(1s+1)(0.4s+1)(0.2s+1)(0.05s+1)^3}$$

using the SIMC tuning rules with the "default" recommendation $\tau_c = \theta$. From the closed-loop setpoint response, we obtained in a previous example a first-order model with parameters $k = 0.994, \theta = 1.67, \tau_1 = 3.00$ (5.10). The resulting SIMC PI-settings with $\tau_c = \theta = 1.67$ are

$$\mathrm{PI_{cl}}: \quad K_c = 0.904, \qquad \tau_I = 3.$$

From the full-order model $g_0(s)$ and the half rule, we obtained in a previous example a first-order model with parameters $k = 1, \theta = 1.47, \tau_1 = 2.5$. The resulting SIMC PI-settings with $\tau_c = \theta = 1.47$ are

$$\mathrm{PI_{half\text{-}rule}}: \quad K_c = 0.850, \qquad \tau_I = 2.5.$$

From the full-order model $g_0(s)$ and the half rule, we obtained a second-order model with parameters $k = 1, \theta = 0.77, \tau_1 = 2, \tau_2 = 1.2$. The resulting SIMC PID-settings with $\tau_c = \theta = 0.77$ are

$$\text{Series PID}: \quad K_c = 1.299, \qquad \tau_I = 2, \qquad \tau_D = 1.2.$$

The corresponding settings with the more common ideal (parallel form) PID controller are obtained by computing $f = 1 + \tau_D/\tau_I = 1.60$, and we have

$$\text{Ideal PID}: \quad K_c' = K_c f = 1.69, \qquad \tau_I' = \tau_I f = 3.2, \qquad \tau_D' = \tau_D/f = 0.75. \tag{5.30}$$

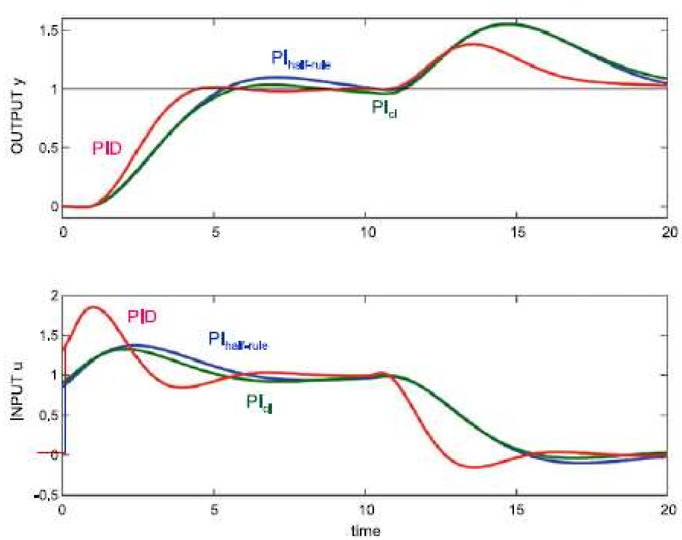
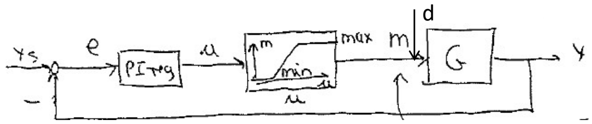


Fig. 5.6 Closed-loop responses for process E2 using SIMC PI- and PID-tunings with $\tau_c = \theta$. Setpoint change at $t = 0$ and input (load) disturbance at $t = 10$. For the PID controller, D-action is only on the feedback signal, i.e., not on the setpoint y_s

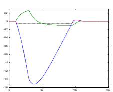
Integral windup

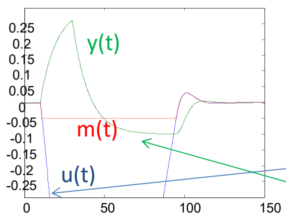
- Problem: Integrator "winds up" u(t) when actual input has saturated



Actual input is m.
m=u if no saturation

$$u(t) = u_0 + K_c e(t) + \underbrace{\frac{K_c}{\tau_I}\int_0^t e(t)dt}_{\text{Keeps changing when } e(t)\neq 0}$$



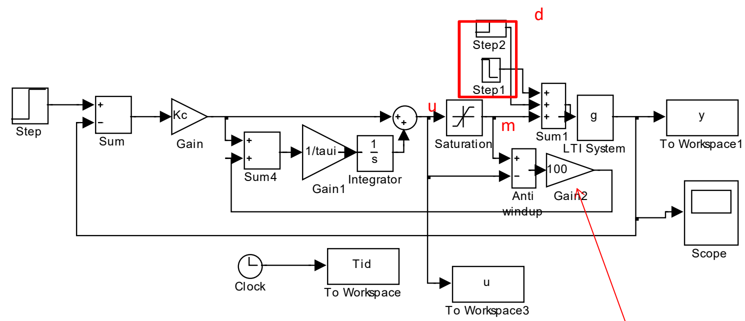


0. t=10: Disturbance d starts
1. t≈12: Reach saturation in m (actual input)
2. oopss! Integration makes u (desired input) "wind up" …
3. …so when disturbance ends (at t=30)
y(t) has to overshoot on the negative side to bring u(t) back.

Anti-windup

- Approaches to avoid windup
  1. Stop integration (e.g. set τ_I=9999) when saturation in input occurs *(requires logic)*
  2. Force integrator to follow true input using high-gain feedback correction *(see Example)*
  3. Use discrete controller in *velocity form*

Example anti-windup



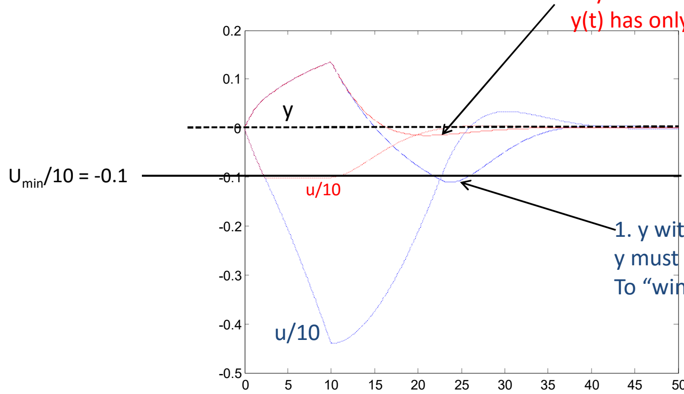
Approach 2:
High-gain feedback correction which makes u follow m.
Gain should be 1 or larger.
(Without anti-windup:
Set this gain to zero)

g= 0.2/(10*s+1)
tauc=1: Kc=12.5, taui=4

Input: max=1, min=-1
Disturbance: Pulse from 0 to 2 and back to 0 at t=10

File: tunepidantiwindup.mdl

2. y with anti-windup: Much better!
y(t) has only small overshoot



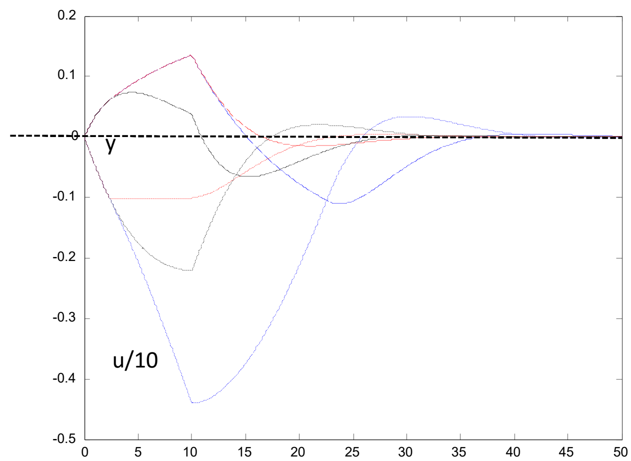
1. y without anti-windup:
y must overshoot on other side
To "wind input u back"

t=0: Disturbance starts
t=10: Disturbance ends

1. Blue = without anti-windup
2. Red = with anti-windup

y = output process
u = output controller (the real process input cannot go below u/10=-0.1)
m = sat(u) = input process

Comparison with no constraint (black line)



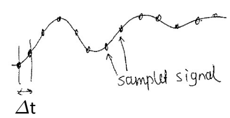
Red = with anti-windup
Blue = without
Black = no constraint

Bumpless transfer

- We want a “soft” transition when the controller is switched between “manual” (1) and “auto” (2)
  - or back from auto to manual
  - or when controller is retuned
- Simple solution: reset u_0 as you switch, so that $u_1(t) = u_2(t)$.

$$u(t) = u_0 + \underbrace{K_c[e(t) + \frac{1}{\tau_I}\int_0^t e(t)dt + \tau_D\frac{de(t)}{dt}]}_{\Delta u}$$

Effect of sampling



- All real controllers are digital, based on sampling
- Δt = sampling time (typical 1 sec. in process control, but could be MUCH faster)
- Max sampling time (Shannon): $\Delta t < \tau_c/2$, but preferably much smaller (τ_c = closed-loop response time)
- With continuous methods: Approximate sampling time as effective delay $\theta \approx \Delta t/2$
- Strange things can happen if Δt is too large:

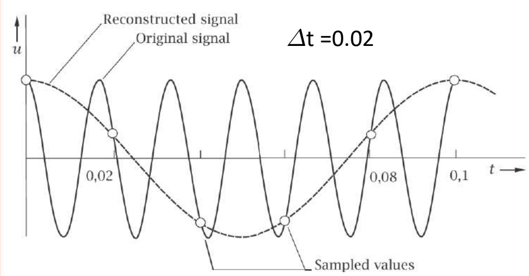


Figure 6-8: Falsification due to undersampling

Figure 6-8 illustrates in a particularly drastic manner the consequences of a violation of the Shannon theorem. A sinusoidal original signal with a frequency of 60 Hz is sampled with a frequency of 50 Hz, although the sampling frequency should be higher than 120 Hz. The consequence

"How to program a PID controller in 5 minutes." (In addition you should filter the measurement)

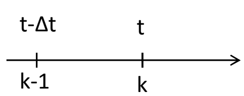
Discrete (digital) implementation (practical in computer) of PID controller

Continuous (not possible in computer):

$$u(t) = \underbrace{u_0 + \frac{K_c}{\tau_I}\int_0^t e(t)dt}_{\bar{u}(t)} + K_c e(t) + K_c \tau_D \frac{de(t)}{dt}$$

e(t) = y_s – y but use -dy/dt to avoid differentiation of setpoint
y = filtered measurement

where $\bar{u}(t)$ = "bias" term with integral action included



Digital PID implementation:

$$\boxed{\begin{aligned} \bar{u}_k &= \bar{u}(t) \approx \bar{u}_{k-1} + \frac{K_c}{\tau_I} e_k \Delta t \\ u_k &= \bar{u}_k + K_c e_k - K_c \tau_D \frac{y_k - y_{k-1}}{\Delta t} \end{aligned}}$$

(Backward Euler)

This is Sigurd's recommendation = "Alt. 3" (see next page)

To avoid windup and to get bumpless transfer:
Adjust bias $\bar{u}_k$ so that u_k = actual input
(Comment: Can "clamp" $\bar{u}_k$ as we enter saturation (= stop integration), but may then take a little longer to get out of saturation, because e must return to value it has as we entered saturation.)

Comparison with book: Digital implementation of PID controllers

Alt. 1 (position form)

$$p(t) = \bar{p} + K_c\left[e(t) + \frac{1}{\tau_I}\int_0^t e(t^*)\,dt^* + \tau_D\frac{de(t)}{dt}\right] \quad (7\text{-}13)$$

Note: p = output from controller

Finite difference approximation:

$$\int_0^t e(t^*)\,dt^* \approx \sum_{j=1}^{k} e_j\Delta t \quad (7\text{-}24)$$

$$\frac{de}{dt} \approx \frac{e_k - e_{k-1}}{\Delta t} \quad (7\text{-}25)$$

where

Δt = the sampling period (the time between successive measurements of the controlled variable)

e_k = error at the kth sampling instant for $k = 1, 2, \ldots$

Substituting Eqs. 7-24 and 7-25 into (7-13) gives the *position form*,

$$\boxed{p_k = \bar{p} + K_c\left[e_k + \frac{\Delta t}{\tau_I}\sum_{j=1}^{k} e_j + \frac{\tau_D}{\Delta t}(e_k - e_{k-1})\right] \quad (7\text{-}26)}$$ Alt. 1

e_k = present sampled value = e(t)

e_{k-1} = previous sample = e(t-Δt)

e_{k-2} = e(t-2Δt)

Alt. 3 (Sigurd's with bias as extra state, better than Alt. 1 and Alt. 2)

Alternative to Alt. 1: Store integral action in the bias $\bar{p}_k$

$$p_k = \bar{p}_k + K_c[e_k + \tfrac{\tau_D}{\Delta t}(e_k - e_{k-1})]$$

where we update ("reset") the bias: $\bar{p}_k = \bar{p}_{k-1} + K_c\frac{\Delta t}{\tau_I}e_k$

To avoid windup and to get bumpless transfer:

Adjust bias $\bar{p}_k$ so that p_k = actual input

Alt. 2 (velocity form)

In the *velocity form*, the change in controller output is calculated. The velocity form can be derived by writing Eq. 7-26 for the $(k-1)$ sampling instant:

$$p_{k-1} = \bar{p} + K_c\left[e_{k-1} + \frac{\Delta t}{\tau_I}\sum_{j=1}^{k-1} e_j + \frac{\tau_D}{\Delta t}(e_{k-1} - e_{k-2})\right] \quad (7\text{-}27)$$

Note that the summation still begins at $j = 1$, because it is assumed that the process is at the desired steady state for $j \le 0$, and thus $e_j = 0$ for $j \le 0$. Subtracting Eq. 7-27 from (7-26) gives the velocity form of the digital PID algorithm:

Velocity form

$$\boxed{\Delta p_k = p_k - p_{k-1} = K_c\left[(e_k - e_{k-1}) + \frac{\Delta t}{\tau_I}e_k + \frac{\tau_D}{\Delta t}(e_k - 2e_{k-1} + e_{k-2})\right] \quad (7\text{-}28)}$$ Alt. 2

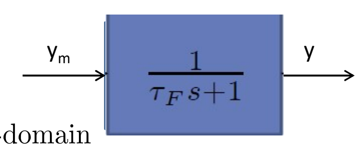
The velocity form has three advantages over the position form:

1. It inherently contains antireset windup, because the summation of errors is not explicitly calculated.
2. This output is expressed in a form, Δp_k, that can be utilized directly by some final control elements, such as a control valve driven by a pulsed stepping motor.
3. For the velocity algorithm, transferring the controller from manual to automatic model does not require any initialization of the output ($\bar{p}$ in =Bumpless transfer

A ? minor disadvantage of the velocity form is that the integral mode *must* be included. When the set point is constant, it cancels out in both the proportional and derivative error terms. Consequently, if the integral mode were omitted, the process response to a disturbance would tend to drift away from the set point.

? This is a major disadvantage of Alt. 2

Digital implementation of first-order filter of measurement*



Tuning: Select $\tau_F < \tau_c/10$

Continuous s-domain

$$y(s) = \frac{1}{\tau_F s+1}y_m(s) \Rightarrow$$

$$\tau_F s y(s) + y(s) = y_m(s)$$

Continuous time domain

$$\tau_F\frac{dy(t)}{dt} + y(t) = y_m(t)$$

Discrete (digital) approximations :

$$\frac{\tau_F}{\Delta t}(y_k - y_{k-1}) + y_k = y_{m,k}$$

Rearrange

$$y_k = \alpha y_{m,k} + (1-\alpha)y_{k-1}$$

where

$$\alpha = \frac{1}{1+\tau_F/\Delta t}$$

$\tau_F = 0 \Rightarrow \alpha = 1$ (no filtering)

$\tau_F = \Delta t \Rightarrow \alpha = 0.5$

$\tau_F = 9\Delta t \Rightarrow \alpha = 0.1$

*Equivalent to exponential moving average of time series data

Block diagram symbols

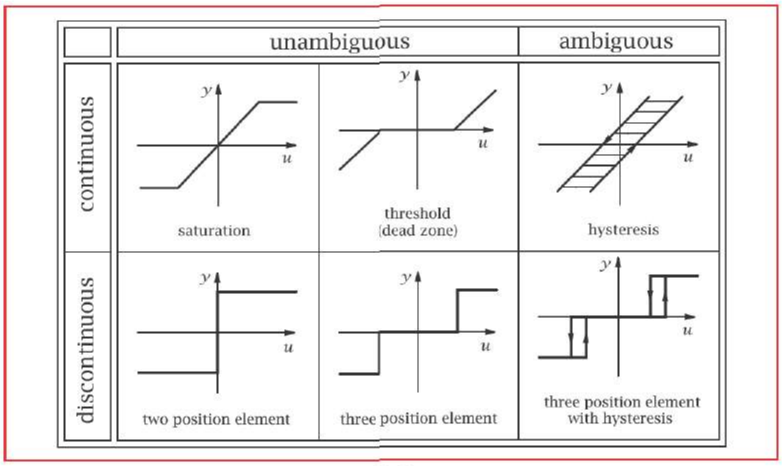


Figure 9-1: Types of characteristic curves