PID control.
Practical issues

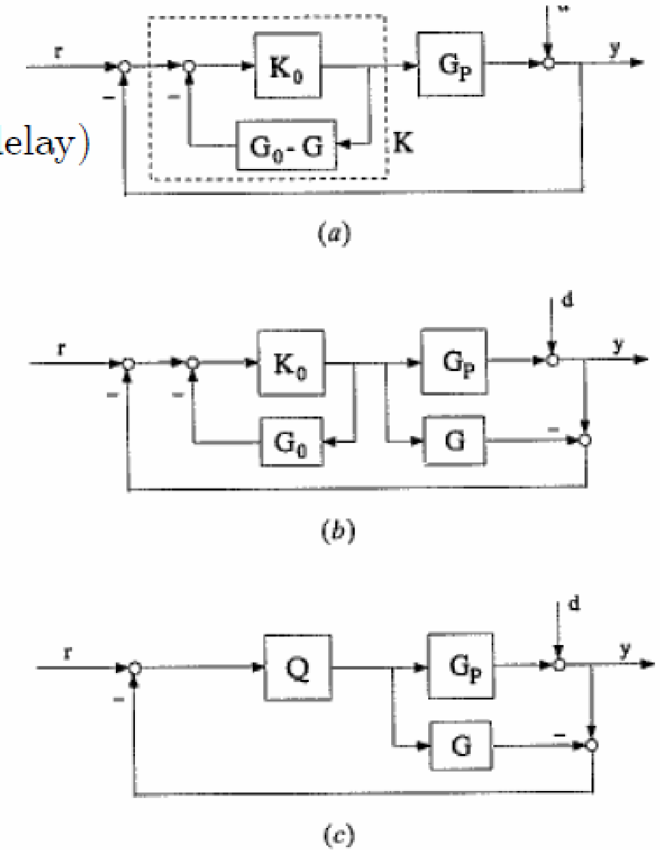
Smith Predictor (NOT PID…)
PID Controller forms
Ziegler-Nichols tuning
Windup
Digital implementation

Write on board

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: It's sensitive to time delay error AND 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!**

* **Chriss Grimholt and Sigurd Skogestad. "Should we forget the Smith Predictor?" (2018)**
In 3rd IFAC conference on Advances in PID control, Ghent, Belgium, 9-11 May 2018. *In IFAC papers Online (2018)*

PID controller

"Ideal/Standard" 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 usually the main part of the controller! (except for static process with no dynamics except delay)
  - Make sure K_c has the right sign! With negative feedback in the loop, Kc has the same sign as the process gain k.
  - Problem: Gives steady-state offset if used without I-action. Offset= 100%/(1+K_ck)

- I-part: To avoid offset, add contribution proportional to integrated error.
  - Note: Larger integral time τ_I gives <u>less</u> I-action (turn off by selecting tauI=9999)
  - Sometimes called "reset time"

    Physical interpretation: τ_I is essentially the time it takes to "reset" the bias (u_0).
  - Note: Integral term 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
  - Note: Larger derivative times more D-action (turn off by selecting taud=0).
  - Can improve control for high-order (S-shaped) response, but sensitive to measurement noise

Sign of the controller gain

- The most common error when tuning a controller is to use the wrong sign of the controller gain.
  - One may think that this is easily detected, but I have seen loops that have been oscillating for years because of the wrong sign (which results in positive rather than negative feedback control).
- The rule in a standard negative feedback implementation (with y_s-y as the controller input) is that the sign of the controller gain (K_c) and the process gain (k) should be the same. For example, recall the SIMC-rule: $K_c=(1/k)*(\tau/(\tau_c+\theta))$.

- But: Most commercial control systems only allow for positive controller gains and then instead distinguish between «direct» and «reverse» control action.
  - "Reverse acting" is used in the normal case when the process gain (k) is positive
    - because MV (u) should go down when CV (y) goes up (to get negative feedback), for example, when we use heat (u=Q) to control room temperature (y=T).
  - "Direct acting" is used when the process gain k is negative

    - Comment: This convention is common in process control, including most vendors such as Emerson, Honeywell, ABB, Yokogawa and also the Aspen/Hysys simulation software. Here is from the Aspen/Hysys manual:

There are two options for the Action of the controller, which are described in the table below:

Controller Action	Description
Direct	When the PV rises above the SP, the OP increases. When the PV falls below the SP, the OP decreases.
Reverse	When the PV rises above the SP, the OP decreases. When the PV falls below the SP, the OP increases.

Note common process control notation:
y = PV (process value)
y_s = SP
u = OP (output from controller)

    - BUT WARNING: Be careful and read the manual! Some people (maybe electrical engineers) use «direct» and «reverse» opposite!, e.g., wikipedia on PID control (2023): https://en.wikipedia.org/wiki/PID_controller

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…)

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

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

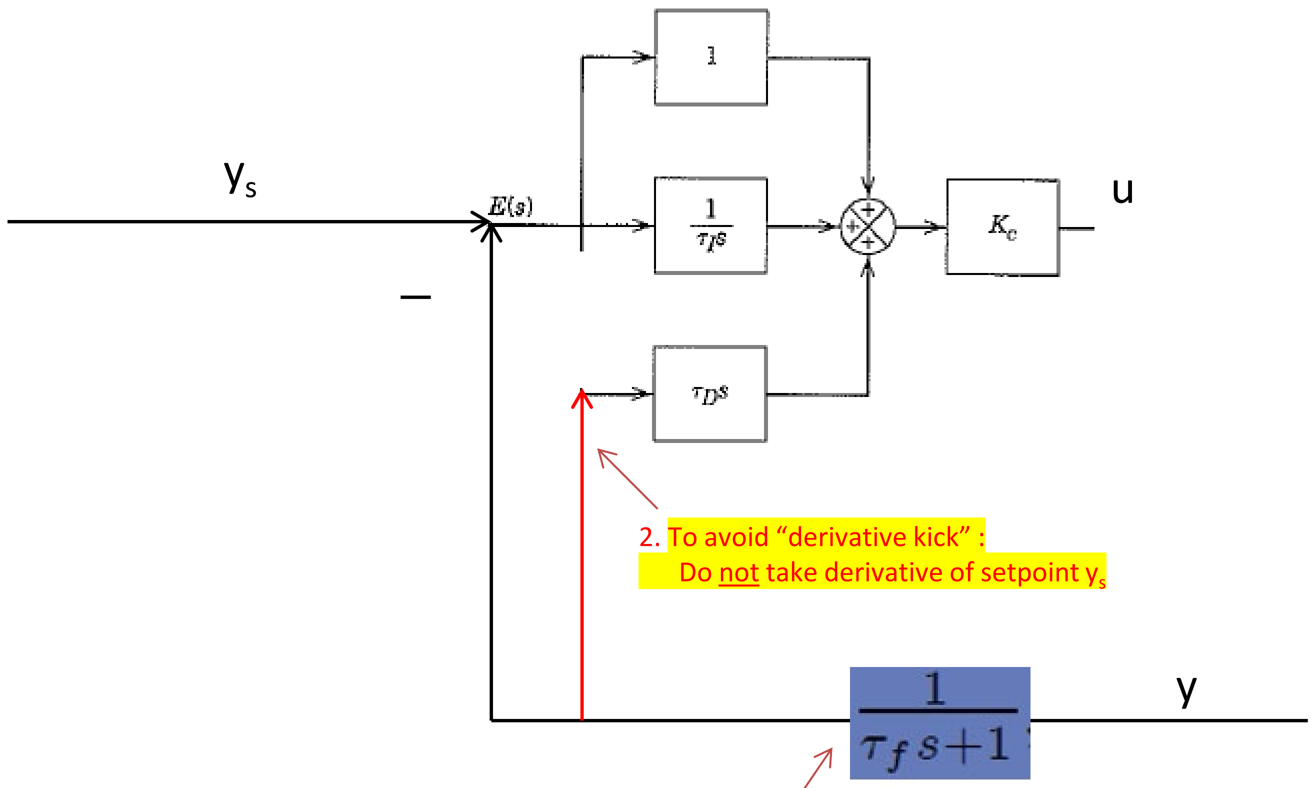
we use the following translation formulas

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

Derivation: See exercise 6. Problem 3

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

Practical “Ideal” PID



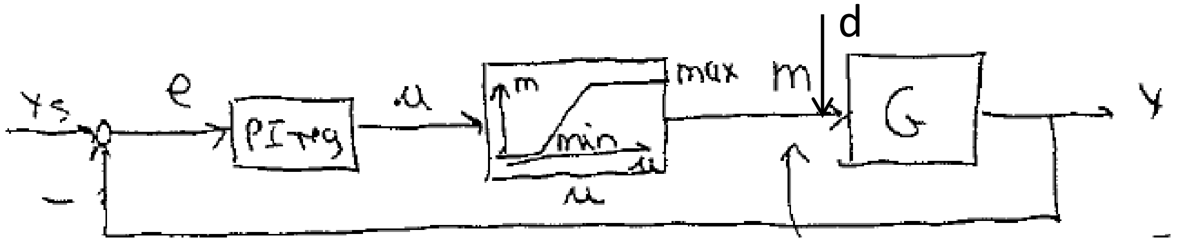
3. Also need to avoid “windup” of the integral action if u saturates (at umin or umax) so that $e = y_s - y \neq 0$ at steady state. See next slide

2. To avoid “derivative kick” :
   Do not take derivative of setpoint y_s

1. For smoother control/ less sensitivity to noise: Filter the measurement.
   Must require $\tau_F \leq 0.5\, \tau_c$
   Typical 1/10 of closed-loop time constant : $\tau_F = \alpha\tau_c,\ \alpha \approx 0.1$

Integral windup

- Problem: Input saturates so e(t) does not go to zero.
- Integrator "winds up" u(t) when actual input has saturated



Actual input is m=$\tilde{u}$.
m=u if no saturation

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

Anti-windup

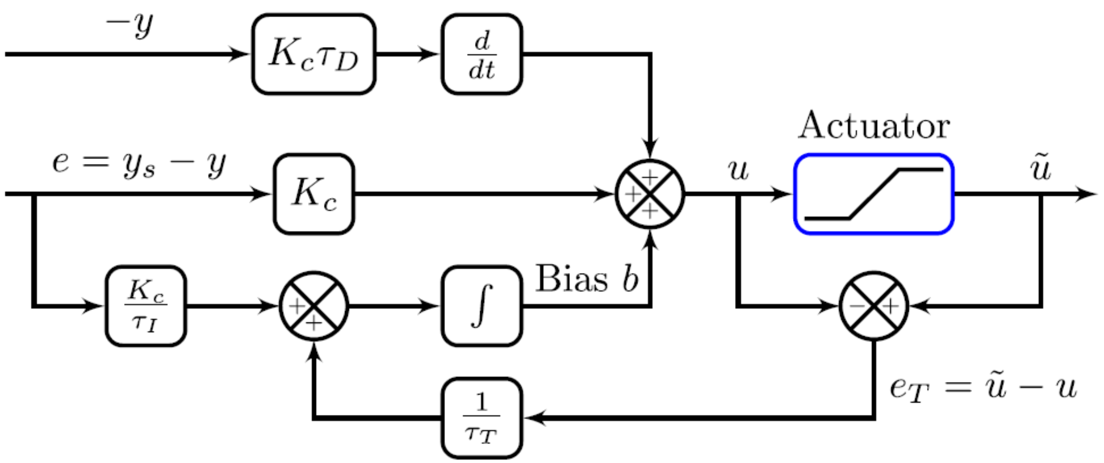
Many approaches to avoid windup

1. **Simplest: Limit u (=output from the controller) to be within specified bounds (by updating u_0)**
   - *For example, with Sigurd’s discrete controller (later)*
2. **Better: Make integrator track true input using feedback correction *(see Example, Exercise and Lab)***
3. Use discrete controller in *velocity form*
   - *BUT requires I-action*
4. Stop integration (e.g. set $\text{¿}_I=9999$) when saturation in input occurs *(requires logic)*

Approaches 1 or 2 are recommended

Anti-windup with tracking (approach 2)

without D-action on the setpoint



The idea is to «back-calculate» a correction so that u' tracks u. That is, we want to avoid «wind-up» of the error

$e_T = \tilde{u} - u$

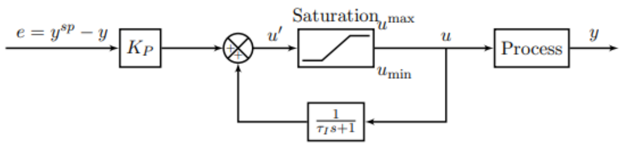
where

u = desired input = output from the controller

$\tilde{u}$ = m = actual input.

$\tilde{u}$ could be different from u for many reasons:

1. Saturation (u is valve position, as shown in Fig. 3)
2. Selector (so another controller determines u)
3. Cascade control (e.g., caused by saturation in the inner loop)
   - In this case $u=y_{2s}$ and $\tilde{u} = y_2$

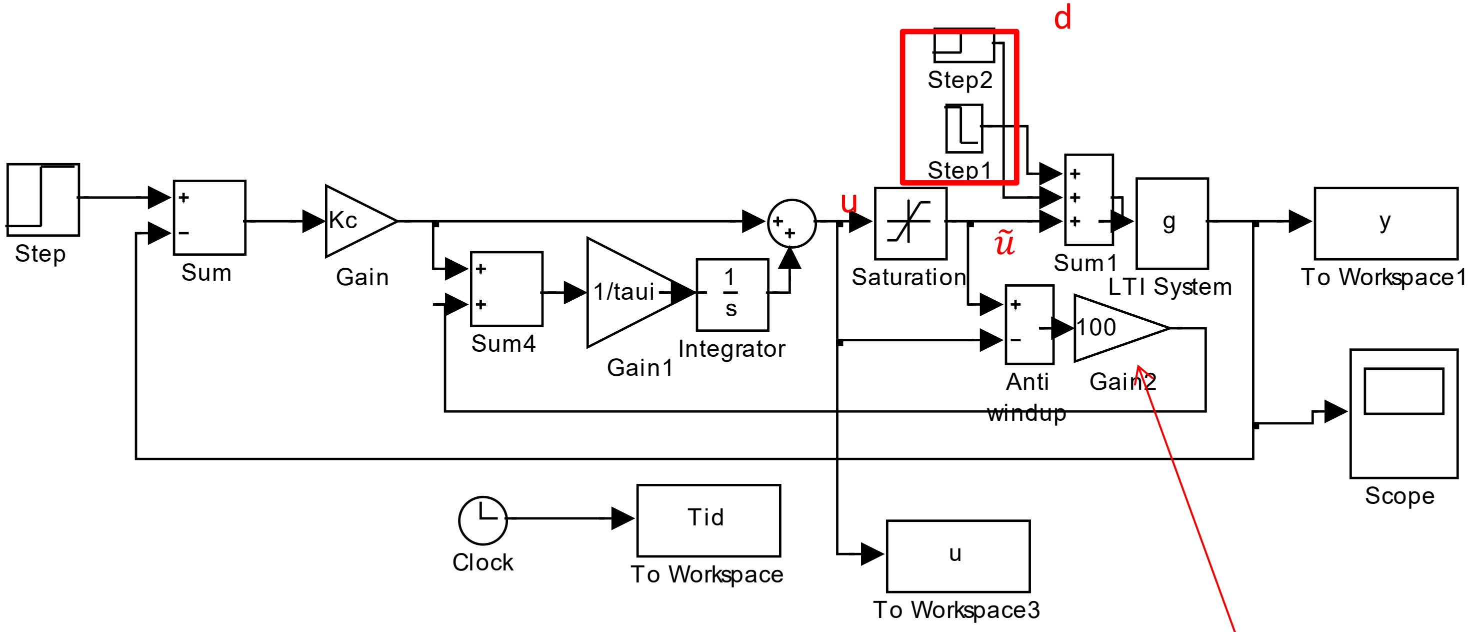


Simpler common «external reset» implementation where $\tau_T = \tau_I$. See Exercise 11 (Problem 3)

Choice of tracking time: A common choice is $\tau_T = \tau_I$ (= integral time) or equivalently $K_T \equiv \frac{\tau_I}{\tau_T} = 1$.

- Note: for with $\tau_T = \tau$, a simple positive feedback implementation of the I-action («external reset») can be used (see Exercise 11). This implementation is very common in commercial systems (ABB, etc.).
- However, the above implementation is recommended because it gives an extra tuning pararmeter. Note that the best the tracking time may differ for each of the three cases (saturation, selector, cascade control)
- How does it work? At steady-state the input to the intergrator is zero, and we have
  - (Kc/tauI) e + (1/tauT) eT = 0
    -> eT= $\tilde{u}$ -u = (tauT/tauI)*(Kc*e) (at steady state)
  - Here Kc*e is the contribution from the P-action, so with the choice tauT=tauiI, the P-action will activate u (go out of saturation) if e «jumps» to 0, so just as y crosses its setpoint ysp. This may be a resasonable choice.
- Choosing tauT smaller will activatate u earlier, which may be an advantage, for example, if we want to avoid that y overshoots its setpoint. On the other hand, this may make the «anti-windup» a bit nervous. For example, it may make the input u switch uncessesary out of saturation.
- **Example electric heater**. In the summer, the heater is off (umin) and y=T > ys=Ts=22C. If it gets cold, then with a small value of tauT (less than tauI), the P-action will turn on the heater before y=T reaches its setpoint (22C), which may be good if we don't like it cold. However, it may be a danger that the heat is turned on unecessary (although it will only be for a short time as the integral action will turn it off again).

Example anti-windup (Approach 2)



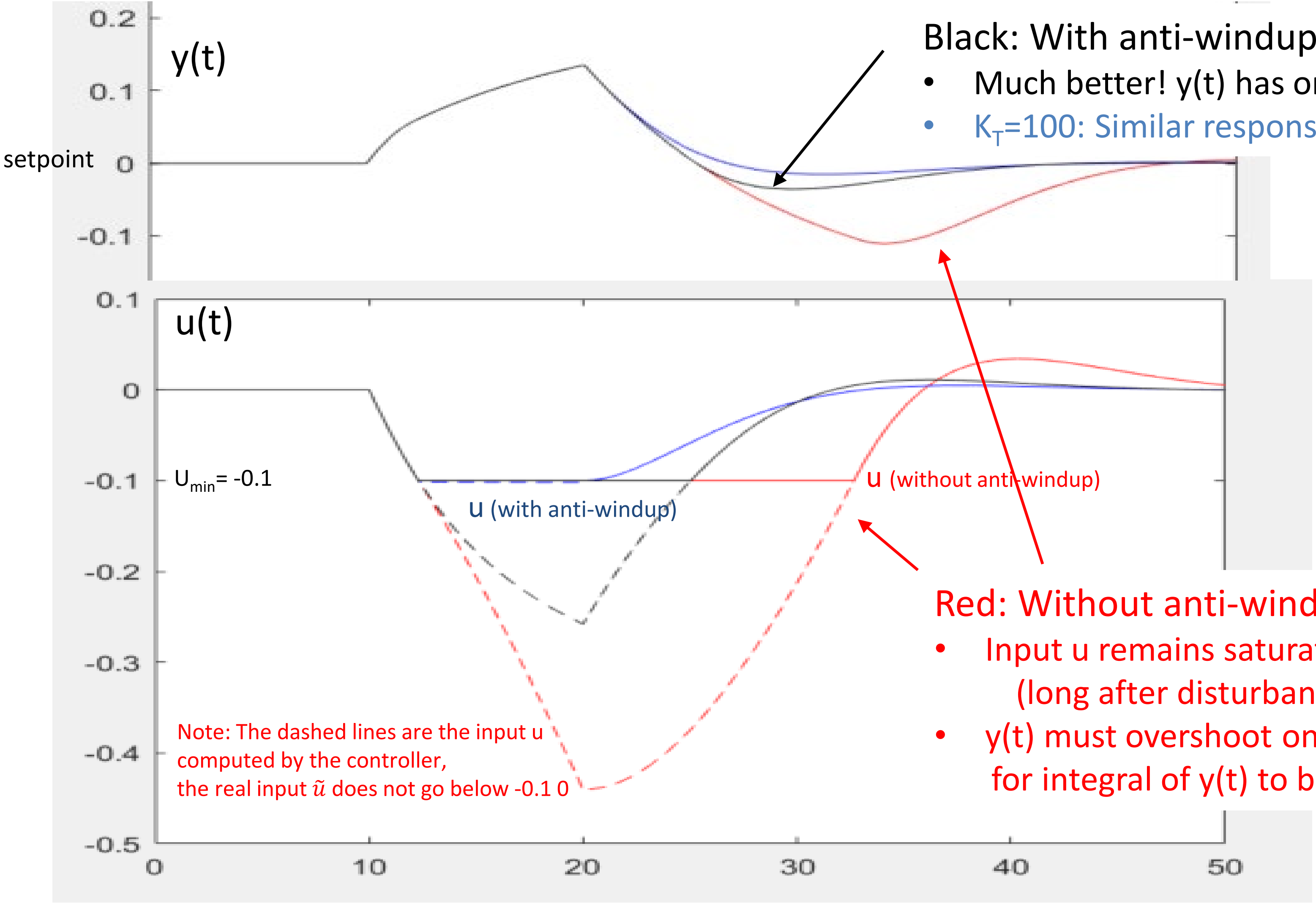
Approach 2:
Feedback correction which makes u' track u.

- Gain2= K_T = taui/tauT = tracking constant
- Often K_T= 1 is recommended (corresponds to have tracking time = integral time)
- If K_T is too high the P-action may make the system go out of saturation prematurely
- No anti-windup: Set K_T = 0.

g= 2/(10*s+1)
PI: Kc=1.25, taui=4

Input saturation: umax=0.1, umin=-0.1
Disturbance (d): Pulse from 0 to 0.2 and back to 0 at t=10

File: tunepidantiwindup.mdl



Black: With anti-windup (K_T=1)

- Much better! y(t) has only small undershoot
- K_T=100: Similar response (but recommend K_T=1)

Red: Without anti-windup (K_T=0):

- Input u remains saturated to t=32
  (long after disturbance is gone at t=20).
- y(t) must overshoot on other side
  for integral of y(t) to become zero.

t=10: Disturbance starts
t=20: Disturbance ends

1. Black = with anti-windup (K_T=1)
2. Blue = with anti-windup (K_T=100)
3. Red = without anti-windup (K_T=0)

Anti-windup with cascade control

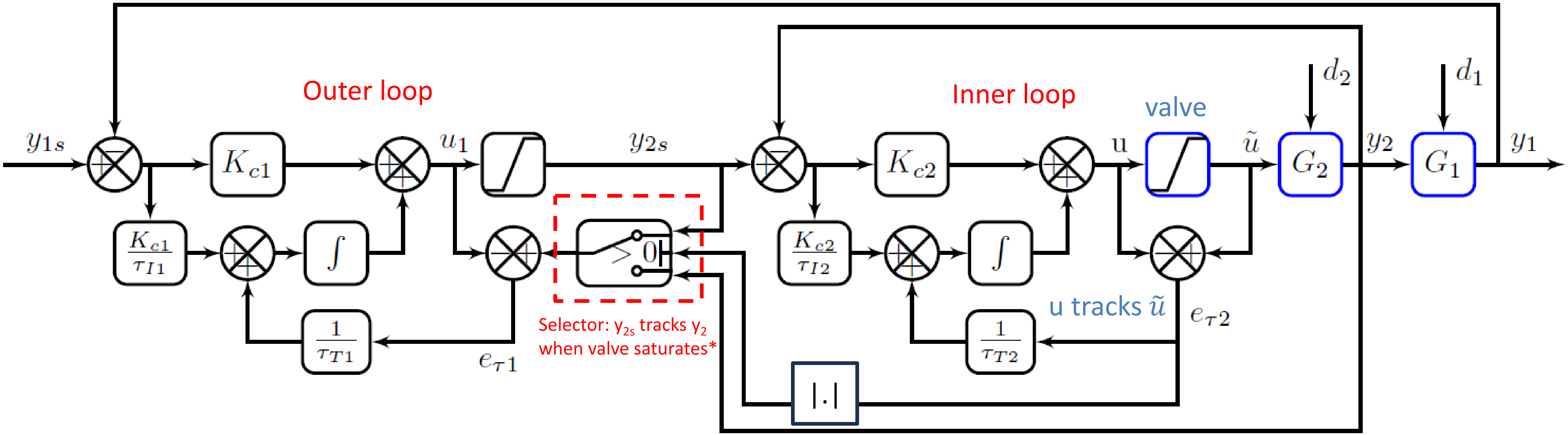


Figure 25: Cascade control with anti windup using the industrial switching approach (Leal et al., 2021).

* The selector makes sure we use anti windup in the outer loop only when the inner loop (u) is saturating, and not just because the inner loop is a little slow.

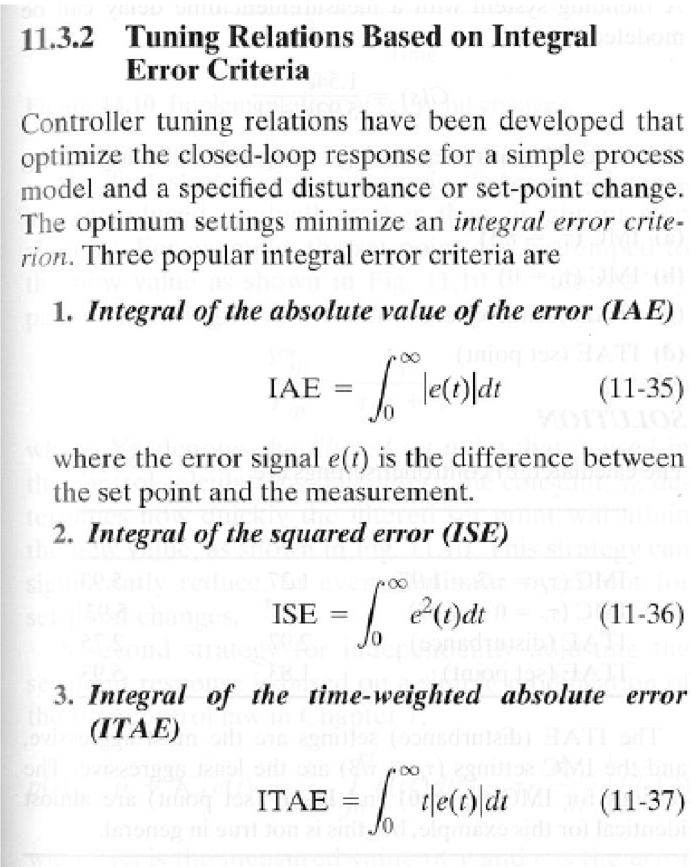
Bumpless transfer

- We want a "soft" transition when the controller is switched between "manual" and "auto"
  - or back from auto to manual
  - or when controller is retuned
- Simple solution: reset bias u_0 as you switch, so that u(t) = $u_{manual}(t)$.

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

Optimal PID settings

- Can find optimal settings using optimization
- SIMC-rules are close to IAE-optimal for combined setpoints and disturbances (with given robustness in terms of M_s)*



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)***

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

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

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

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

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

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

*Chriss Grimholt and Sigurd Skogestad, ''Optimal PI and PID control of first-order plus delay processes and evaluation of the original and improved SIMC rules'' , Published in: *J. Process Control*, vol. 70 (2018), 36-46.

Methods for *online* tuning of PID controllers

I. Trial and error

II. Ziegler Nichols (see Exercise 8, Problem 1)

   – Oscillating P-control

   – Relay method to get oscillations

III. Closed-loop response with P-control

   – Shams method (see Exercise 8, Problem 1)

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

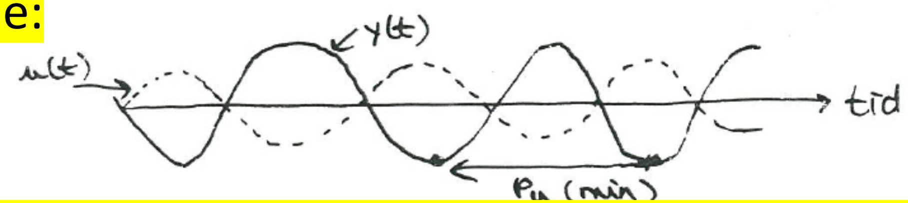
(d) Increase a bit (~ factor 2)

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

[†] Luyben and Luyben (1997).

PID is for ideal form

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

Main problems ZN:

1. Too aggressive (and has no tuning parameter)
2. Two pieces of information (Pu, Ku) is too little to capture all processes. Because of this ZN works poorly on static (delay-dominant) processes (the same applies to TL-modification)

Example PI. Integrating process with delay=1, $G(s) = \frac{e^{-s}}{s}$.

Process model: $k' = 1, \theta = 1$, $(\tau_1 = \infty)$.

SIMC-tunings with $\tau_c = \theta = 1$ (“tight tuning”):

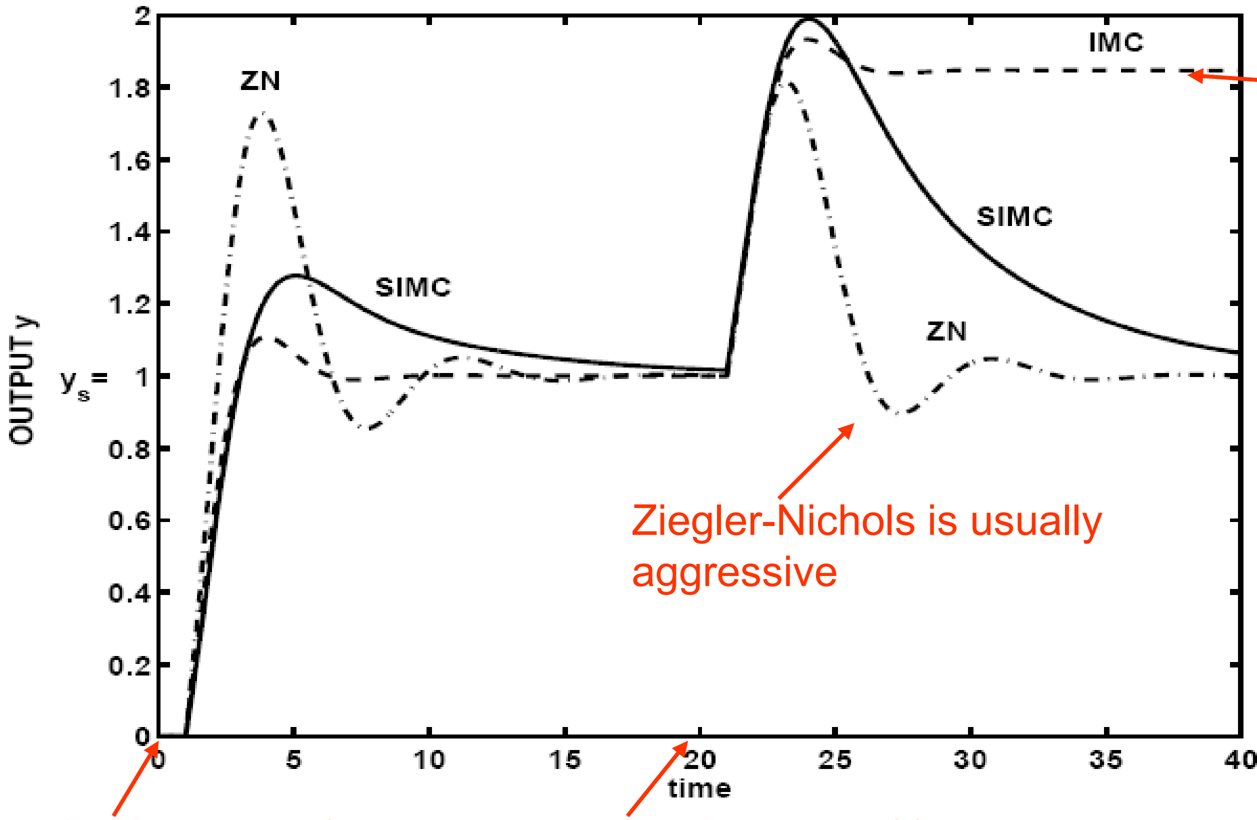
$$K_c = \frac{1}{k'}\frac{1}{\tau_c+\theta} = 1 \cdot \frac{1}{1+1} = 0.5$$
$$\tau_I = \min(\tau_1, 4(\tau_c + \theta)) = \min(\infty, 8) = 8$$

Ziegler-Nichols:

Experiment: $P_u = 4, K_u = 1.57$ (=π/2)

PI-control: $K_c = 0.45\ K_u = 0.71$

$$\tau_I = \frac{P_u}{1.2} = 3.33$$



IMC has $\tau_I = \infty$

Ziegler-Nichols is usually aggressive

Setpoint change at t=0c

Input disturbance at t=20

EXAMPLE: Process from Astrom et al. (1998)

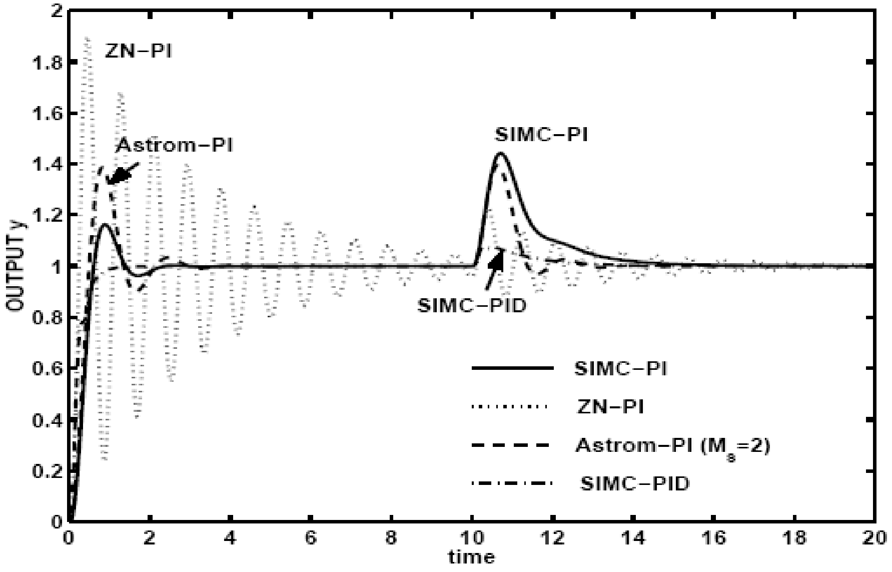


Figure 3: Load disturbance of magnitude 2 occurs at $t = 10$.

ZN-PI: Close to unstable
ZN-PID: unstable

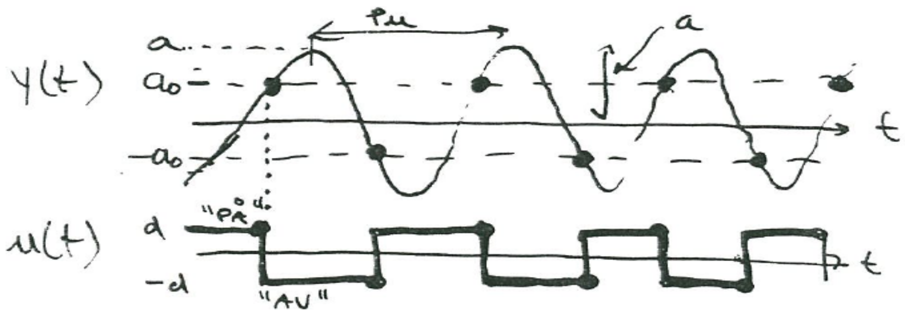
$$g_0(s) = \frac{1}{(s+1)(0.2s+1)(0.04s+1)(0.008s+1)}$$

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)

- 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; can use a_0=0)
- Example: Thermostat in your home



- From this obtain P_u and

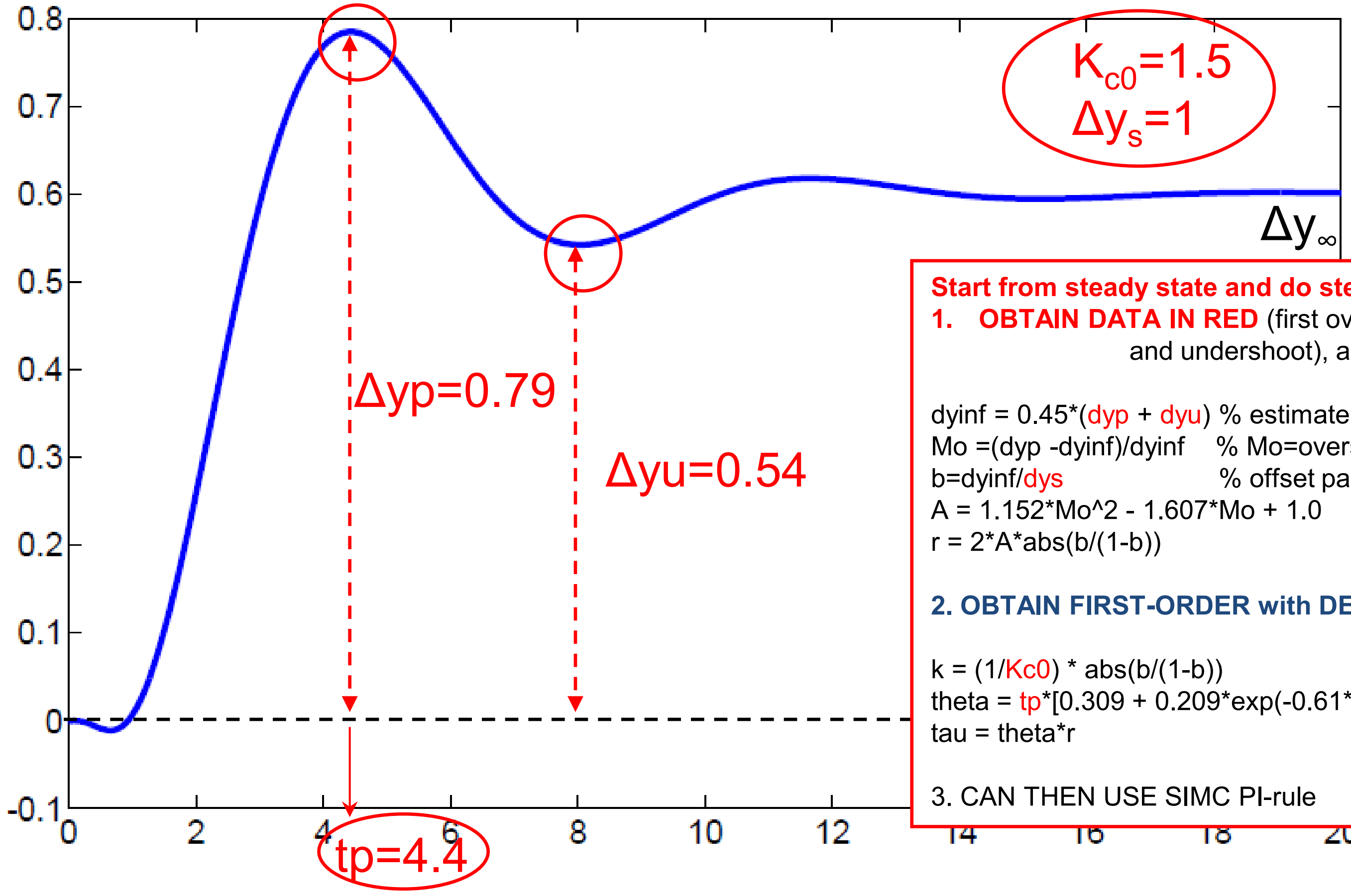
$$K_u = \frac{4d}{\pi a}$$

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

a: amplitude y(t) (from experiment)

Alternative to Ziegler-Nichols closed-loop experiment: Obtains more information and avoids cycling.

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



Start from steady state and do step P-response

1. **OBTAIN DATA IN RED** (first overshoot and undershoot), and then:

dinf = 0.45*(dyp + dyu) % estimate dyinf (so don't wait)
Mo =(dyp -dyinf)/dyinf % Mo=overshoot (about 0.3)
b=dyinf/dys % offset parameter
A = 1.152*Mo^2 - 1.607*Mo + 1.0
r = 2*A*abs(b/(1-b))

2. OBTAIN FIRST-ORDER with DELAY MODEL:

k = (1/Kc0) * abs(b/(1-b))
theta = tp*[0.309 + 0.209*exp(-0.61*r)]
tau = theta*r

3. CAN THEN USE SIMC PI-rule

Example 2: Get k=0.99, theta =1.67, tau=3.0

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

See Exercise 8!

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

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

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

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

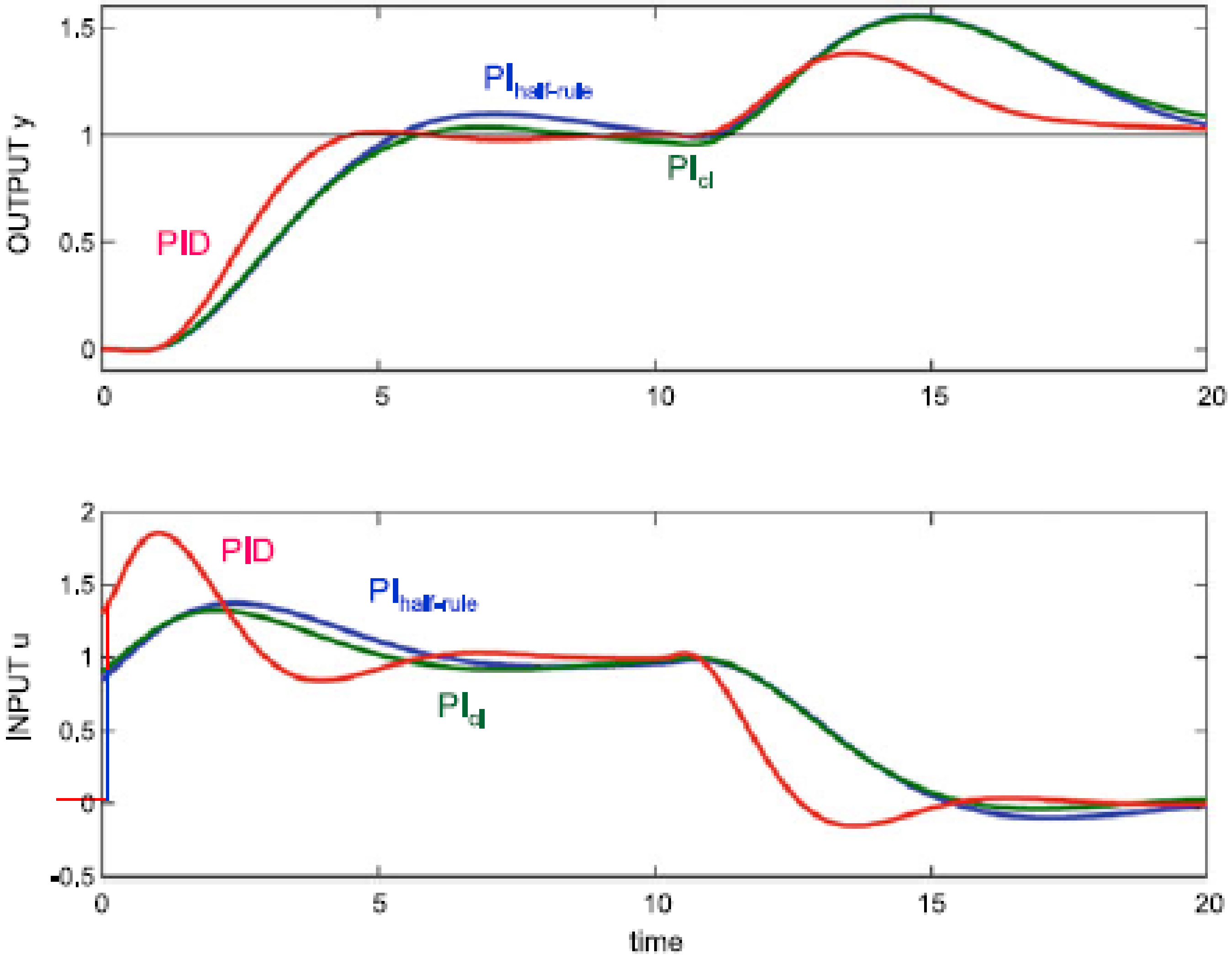
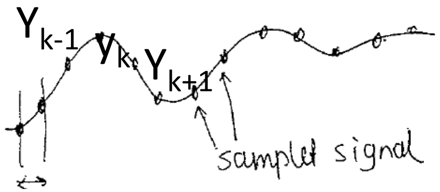


Fig. 5.6 Closed-loop responses for process E2 using SIMC PI- and PID-tunings with $\tau_c = \theta$. Setpoint change at $t = 0$ and input (load) disturbance at $t = 10$. For the PID controller, D-action is only on the feedback signal, i.e., not on the setpoint y_s

Effect of sampling



k=present time

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

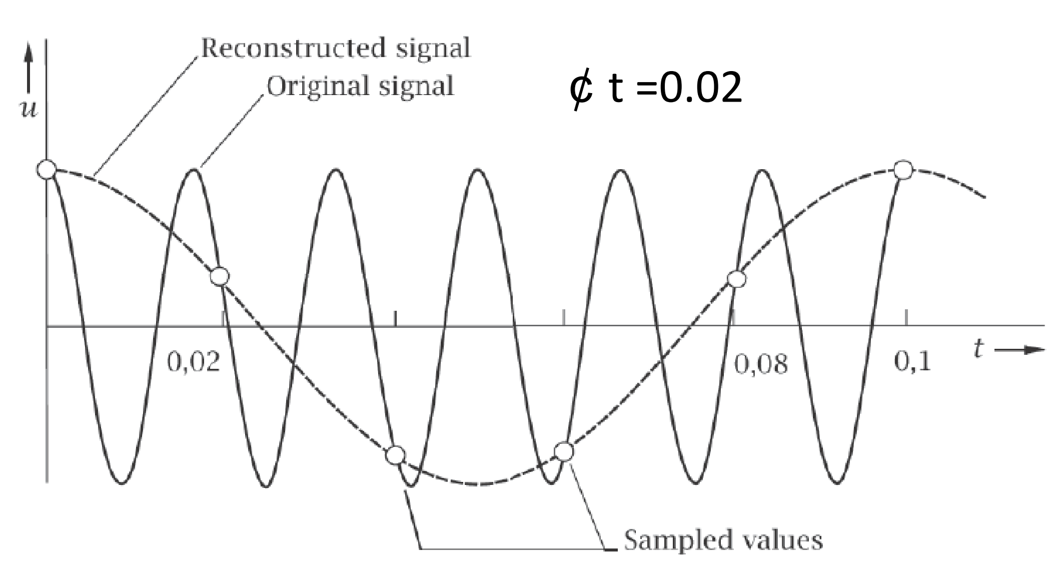


Figure 6-8: Falsification due to undersampling

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

Digital (discrete) implementation of first-order filter of measurement*

$$y_m \longrightarrow \boxed{\frac{1}{\tau_F s+1}} \longrightarrow y$$

Tuning: Select ¿$_F$ = 0.1 ¿$_c$

Continuous s-domain

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

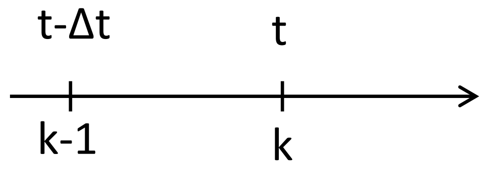
$$\tau_F s y(s) + y(s) = y_m(s)$$

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

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

$$\tau_F = 0 \Rightarrow \alpha = 1 \text{ (no filtering)}$$
$$\tau_F = \Delta t \Rightarrow \alpha = 0.5$$
$$\tau_F = 9\Delta t \Rightarrow \alpha = 0.1$$

But: τ_F should be selected independent of Δt
Typical: $\tau_F = 0.1\ \tau_c$ (normally much larger than Δt).

Comment. «Normal» moving average (not as good) for last 10 measurements is

$$y_k = \frac{1}{10}\sum_{n=0}^{9} y_{m,k-n}$$

*Equivalent to "exponentially moving average" of time series data. See Exercise 8, Problem 2

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

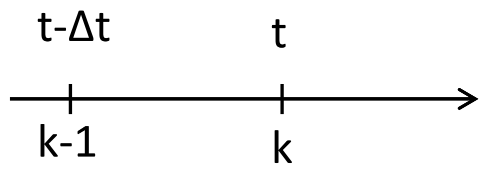
Discrete (digital) implementation (practical in computer) of PID controller

Continuous (not possible in computer): e(t) = $y_s - y$, y = filtered measurement

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

Normally: Use -dy/dt rather than de/dt to avoid differentiation of setpoint

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 get bumpless transfer between manual and auto):
After implementing u_k, adjust the bias $\bar{u}_k$ (which becomes $\bar{u}_{k-1}$ at the next sample point) so that $u_k = m$ = actual input.
With PI-control this gives $\bar{u}_{k-1} = m - K_c e_{k-1}$.
Comment: This implementation has the problem that the controller may go prematurely out of saturation if e_k crosses 0 for a short time (e.g. because of measurement noise) or because of derivative action. To avoid this we may instead require that $\bar{u}_{k-1}$ should not exceed u_{max} or u_{min}, where we may set these limits to be larger than the true limits for m. For example, if m is limited to be within 0 and 1 we may set u_{max}=1.5 and $u_{min} = -0.5$. There will then be some windup, but not too much, and we avoid that we prematurely go out of 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*,

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

Alt. 1

e_k = present sampled value = e(t)

e_{k-1} = previous sample = e(t-Δt)

e_{k-2} = e(t-2Δt)

Alt. 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

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

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

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

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

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 only exceeds limits by small amount

Block diagram symbols

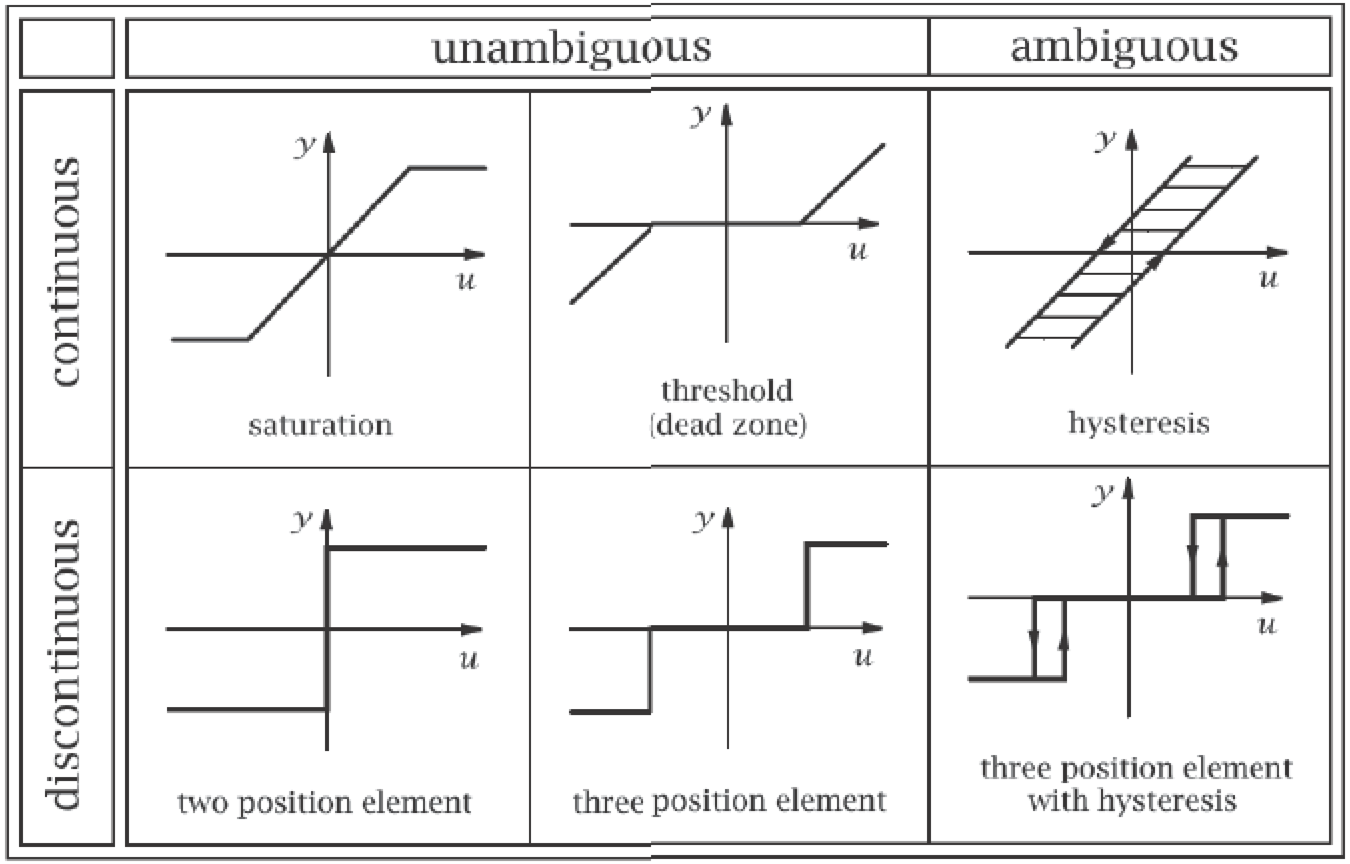
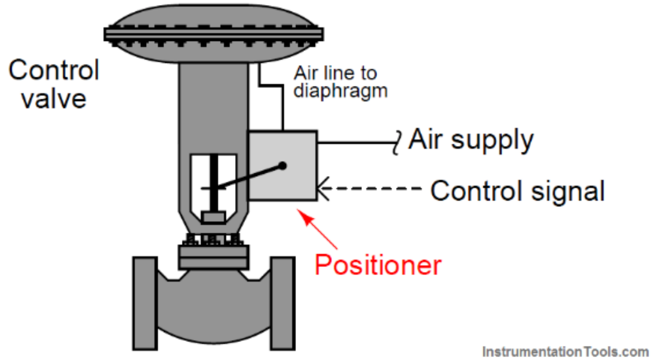


Figure 9-1: Types of characteristic curves

Should I buy a valve positioner?

- Usually not, if the valve is for automatic control
    - But it may be difficult to avoid because most vendors include them
- If you can measure the flow, then a slave **flow controller** eliminates the need to buy a valve positioner.
- Also, the valve positioner is often slow (and tunings cannot be changed) and it then adds an effective delay which may make feedback control difficult



Conclusion PID

- Use SIMC-tunings for Kc, taui, taud
  - Tuning parameter tauc
  - Note that taud is for cascade PID form
- Add filter on noisy measurements
  - To avoid «nervous» MV (= controller output)
  - First-order filter 1/(tauf*s+1).
    - Typical tauf=0.1*tauc
    - Overall controller (cascade form) is then: C(s) = $K_c \frac{\tau_I s+1}{\tau_I s} \frac{\tau_d s+1}{\tau_F s+1}$
    - Usually taud=0 and often tauf=0.
- Add anti-windup
  - Recommend «input tracking».
    - Tracking constant K_T. Typical K_T=1
- May in some cases add filter on setpoint (2-DOF control)
  - Less general approach: Use a different P-gain K_{cs} for the setpoint