Part 1

- Notation
- Introduction to advanced regulatory control (ARC)
  - The three main inventions of process control
- PID control
- Scaling of variables (0-100%)
- Feedforward control

Operation hiearchy

- **What is the difference between control and optimization?**
- **Control layers keep operation at setpoints**
  - $J = (y - y_s)^2$
  - Degrees of freedom may be valve positions (z) or setpoints to other loops ($CV2_s$ – cascade control)
- **Optimization layers minimize economic cost**
  - $J_\$ = p_F F + p_Q Q - p_P P$ [$/s]
  - Degrees of freedom may be setpoints ($y_s = CV1_s$) for control layer

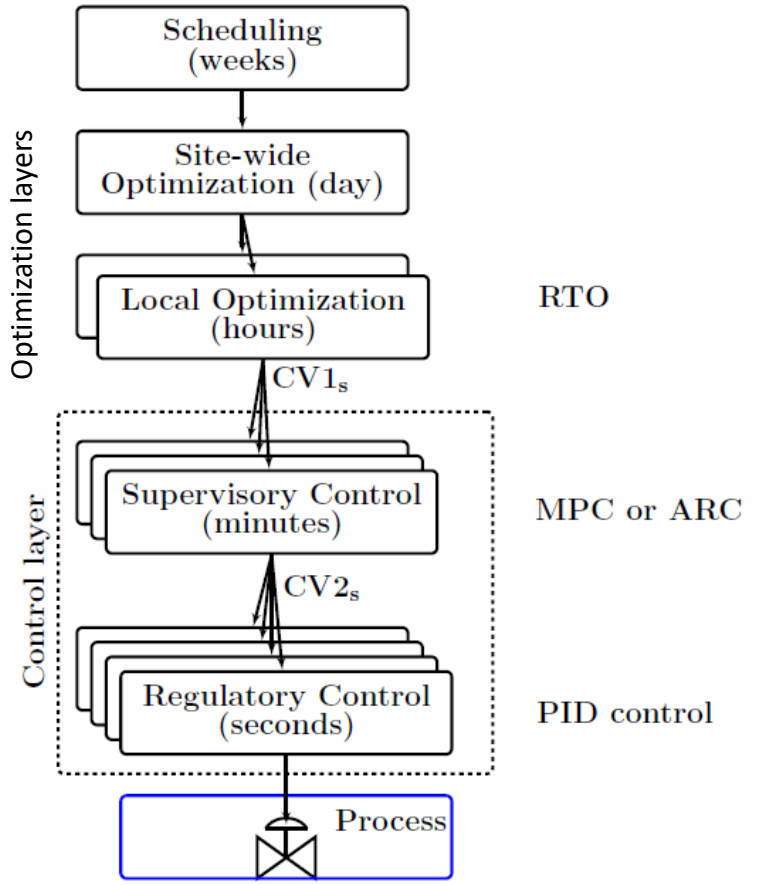


Figure 4: Decomposition of "overall control system" for optimal operation in typical process plant. This involves a vertical (hierarchical) decomposition Richalet et al. (1978) into decision layers based on time scale separation, and a horizontal decomposition into decentralized blocks/controllers, often based on physical distance. There is also feedback of measurements (y, w, CV1, CV2) (possibly estimates) from the process to the various layers and blocks but this is not shown in the figure. This paper considers the three lowest layers, with focus on the supervisory control layer.
CV1 = Economic controlled variables
CV2 = Regulatory/stabilizing controlled variables
RTO = Real-time optimization
MPC = Model predictive control
ARC = Advanced regulatory control
PID = Proportional-Integral-Derivative

«Advanced» control

- This is a relative term
- Usually used for anything than comes in addition to (or in top of) basic PID loops
- Main options
  - Standard «Advanced regulatory control» (ARC) elements
    - Cascade, ratio
    - Feedforward, split range control, selectors, VPC
    - This option is preferred if it gives acceptable performance and it’s not too complicated
  - Model predictive control (MPC)
    - Requires a lot more effort to implement
    - Can have large benefits «predictive» feedforward control (+ interactive process)

NOTATION and BLOCK DIAGRAMS

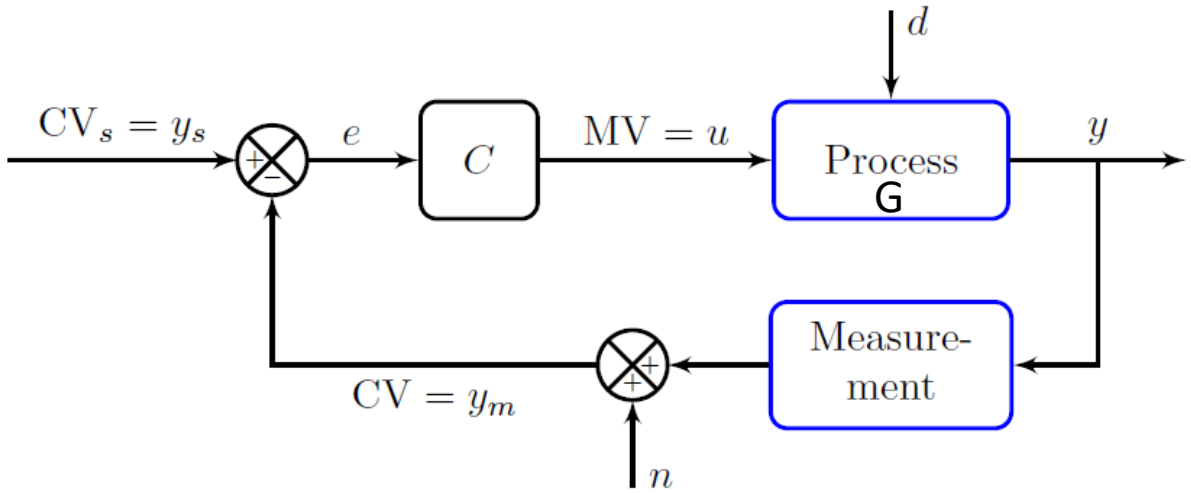


Figure 3: Block diagram of common "one degree-of-freedom" negative feedback control system.

u = process input (adjustable) = manipulated variable (MV) = controller output)
d = disturbance (non-adjustable)
y = process output (with setpoint y_s) = controlled variable (CV) (controller input)

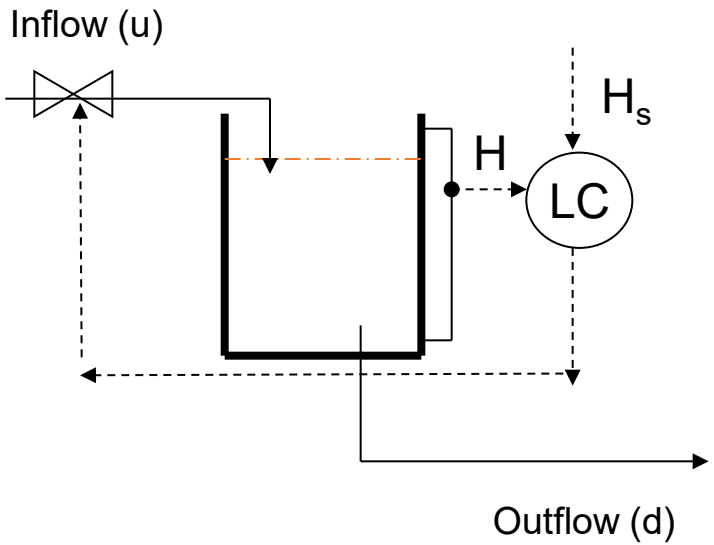
w = extra measured process variable (state x, y_2)

Block diagrams: All lines are signals (information)

Advanced regulatory control (ARC)

Uses Flowsheets rather than Block diagrams

Flowsheet (P&ID) of feedback control
Example: Level control (with given outflow)



Solid lines: mass flow (streams)

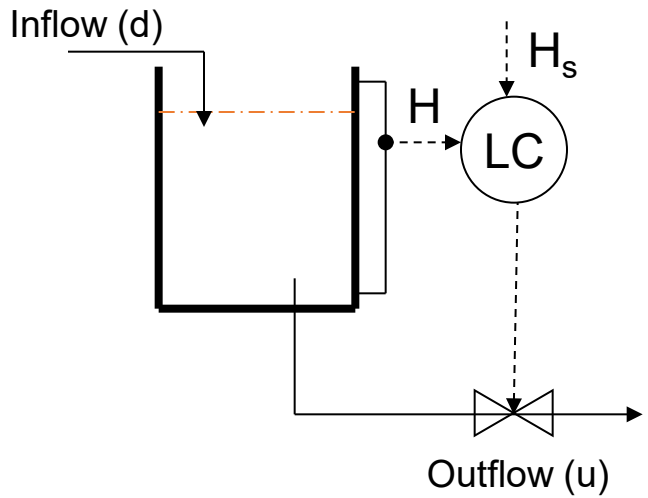
Dashed lines: signals (control)

CLASSIFICATION OF VARIABLES FOR CONTROL (MV, CV, DV):

INPUT (u, MV): INFLOW
OUTPUT (y, CV): LEVEL
DISTURBANCE (d, DV): OUTFLOW

MV = manipulated vartiable (input u)
CV = controlled variable (output y)
DV = disturbance variable (d)

Level control when inflow is given (alternative input/output-pairing)



CLASSIFICATION OF VARIABLES FOR CONTROL (MV, CV, DV):

INPUT (u, MV): OUTFLOW (Input for control!)
OUTPUT (y, CV): LEVEL
DISTURBANCE (d, DV): INFLOW

QUIZ
What are the three most important inventions of process control?

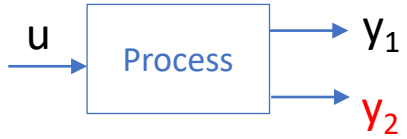
- Hint 1: According to Sigurd Skogestad
- Hint 2: All were in use around 1940

SOLUTION

1. PID controller, in particular, I-action
2. Cascade control
3. Ratio control (special case of feedforward which needs no explicit model)

Which one is the oldest?

Invention 2. Cascade control



y1=primary output (given setpoint)
y2=secondary output (adjustable setpoint)

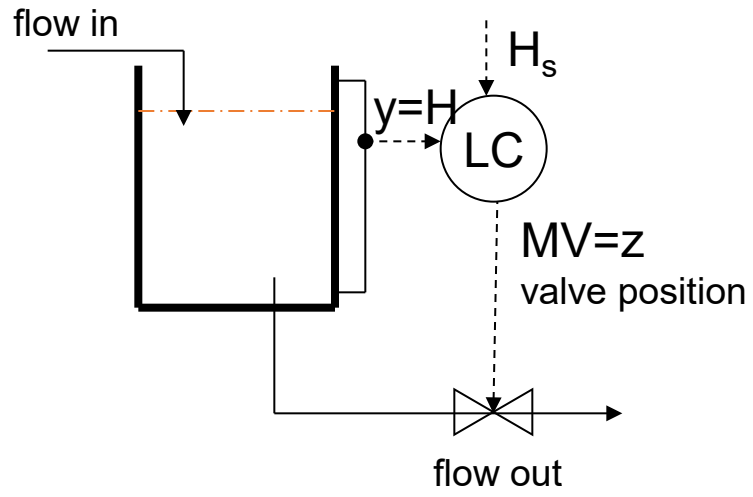
Idea: make use of extra "local" output measurement (y_2)

Implementation: Controller ("master") gives setpoint to another controller ("slave")

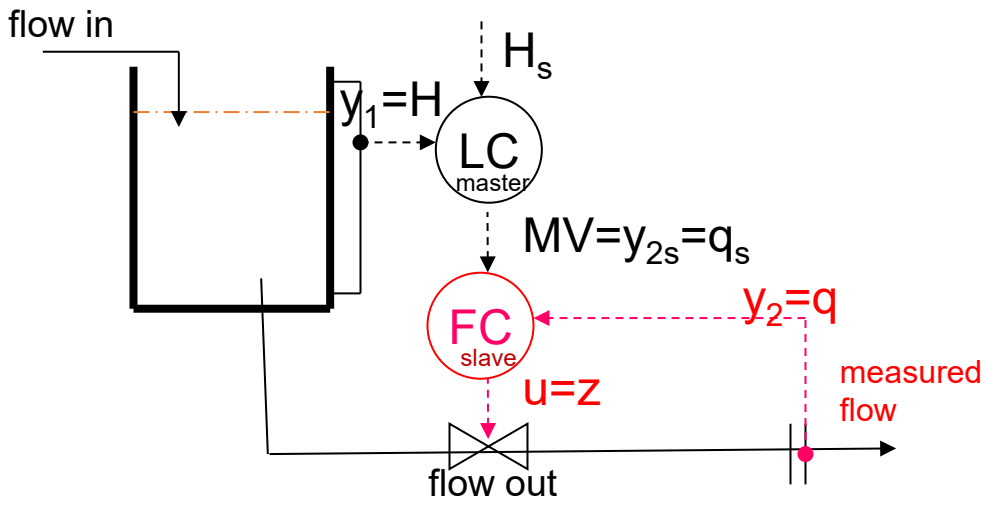
- Without cascade: "Master" controller directly adjusts u to control primary output y_1)
- With cascade: Local "slave" controller (fast) uses u to control "extra"/fast measurement (y_2). *
  "Master" (slow) controller adjusts setpoint y_{2s}.

- **Example: Flow controller on valve (very common!)**
  - y_1 = level H in tank
  - u = valve position (z)
  - y_2 = flowrate q through valve

WITHOUT CASCADE

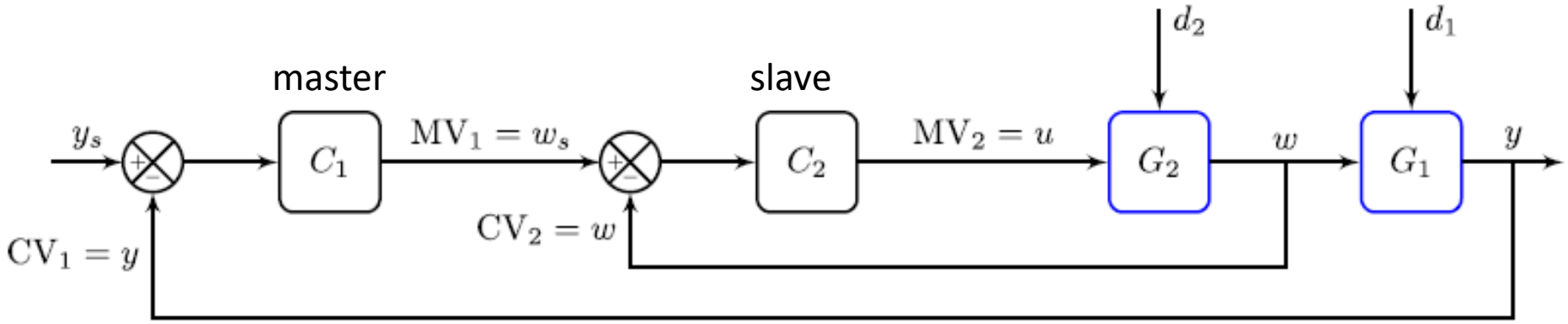


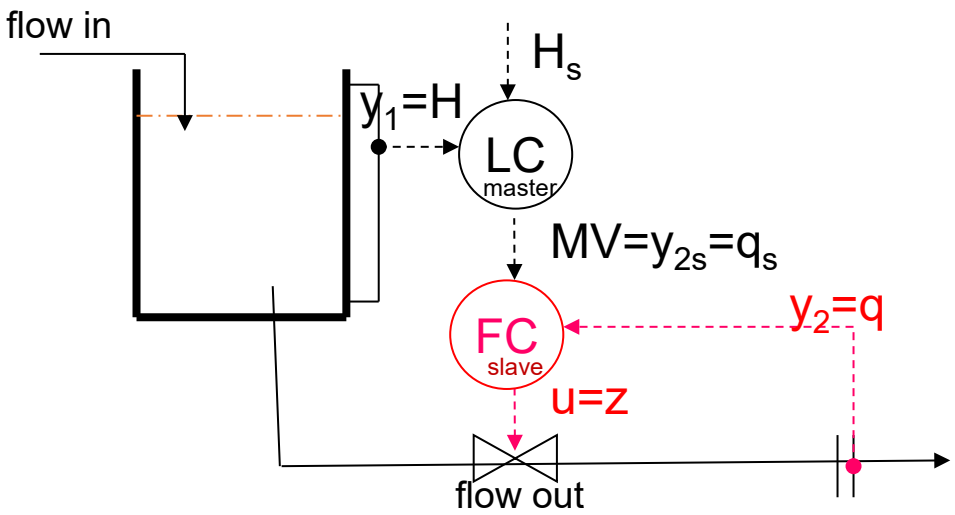
WITH CASCADE



*Comment: Another approach that uses extra measurements to improve control is «**Full state feedback**».

Block diagram of cascade control

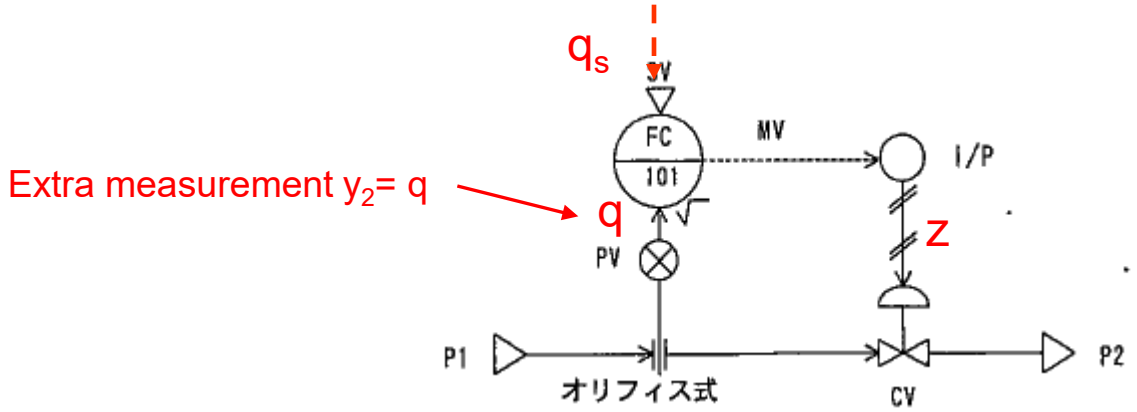




$y = y_1 = H$ (measured level)
$u = z$ (valve position)
$w = y_2 = q$ (measured outflow)
C_1 = LC (P-controller)
C_2 = FC (I-controller)
G_1 = k'/s (level is integrating)
G_2 = valve model (nonlinear, static)
d_1 = downstream pressure
d_2 = inflow

Cascade control

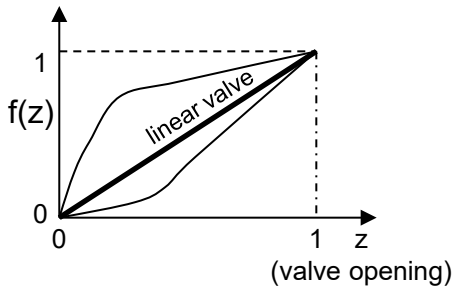
What are the benefits of adding a flow controller (slave=inner cascade)?



$$\text{Flow rate: } q = C_v f(z)\sqrt{\frac{p_1 - p_2}{\rho}} \quad [\text{m}^3/\text{s}]$$

1. Fast local control: Eliminates effect of disturbances in p_1 and p_2 (FC reacts faster than outer level loop)

2. Counteracts nonlinearity in valve, f(z)
   - With fast flow control we can assume $q = q_s$



Shinskey (1967)

The principal advantages of cascade control are these:

1. Disturbances arising within the secondary loop are corrected by the secondary controller before they can influence the primary variable.

2. Phase lag existing in the secondary part of the process is reduced measurably by the secondary loop. This improves the speed of response of the primary loop.

3. Gain variations in the secondary part of the process are overcome within its own loop.

4. The secondary loop permits an exact manipulation of the flow of mass or energy by the primary controller.

Time scale separation is needed for cascade control to work well

- Inner loop (slave) should be at least 4 times* faster than the outer loop (master)
  - This is to make the two loops (and tuning) independent.
  - Otherwise, the slave and master loops may start interacting
    - The fast slave loop is able to correct for local disturbances, but the outer loop does not «know» this and if it's too fast it may start «fighting» with the slave loop.
- Often recommend 10 times faster, $\sigma \equiv \frac{\tau_{c1}}{\tau_{c2}} = 10$.
  - A high σ is robust to gain variations (in both inner and outer loop)
  - The reason for the upper value (σ =10) is to avoid that control gets too slow, especially if we have many layers

* Shinskey (Controlling multivariable processes, ISA, 1981, p.12)

Time-scale separation

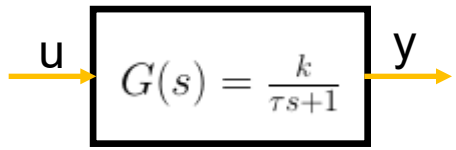
Response of linear first-order system (with time constant τ)

Standard form*: $\tau \frac{dy}{dt} = -y + ku,.$

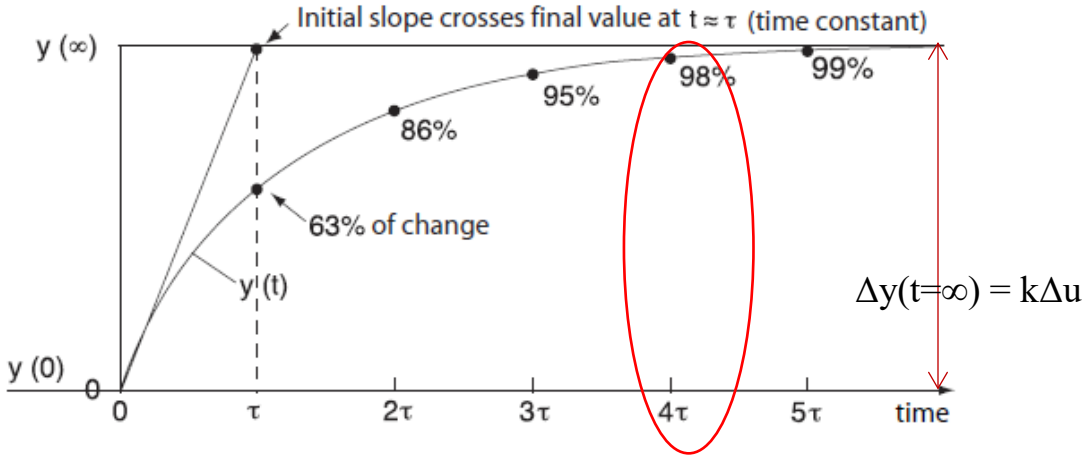
Initially at rest (steady state): $y(0) = y_0.$

Make step in u at $t = 0$: Δu

Solution: $y(t) = y_0 + \left(1 - e^{-t/\tau}\right) \underbrace{k\Delta u}_{\Delta y(t=\infty)}$



Block diagram with transfer function for first-order process



t/τ	$1 - e^{-t/\tau}$	Value	Comment
0	$1 - e^{0} =$	0	
0.1	$1 - e^{-0.1} =$	0.095	
0.5	$1 - e^{-0.5} =$	0.393	
1	$1 - e^{-1} =$	0.632	63% of change is reached after time $t = \tau$
2	$1 - e^{-2} =$	0.865	
3	$1 - e^{-3} =$	0.950	
4	$1 - e^{-4} =$	0.982	98% of change is reached after time $t = 4\tau$
5	$1 - e^{-5} =$	0.993	
∞	$1 - e^{-\infty} =$	1	

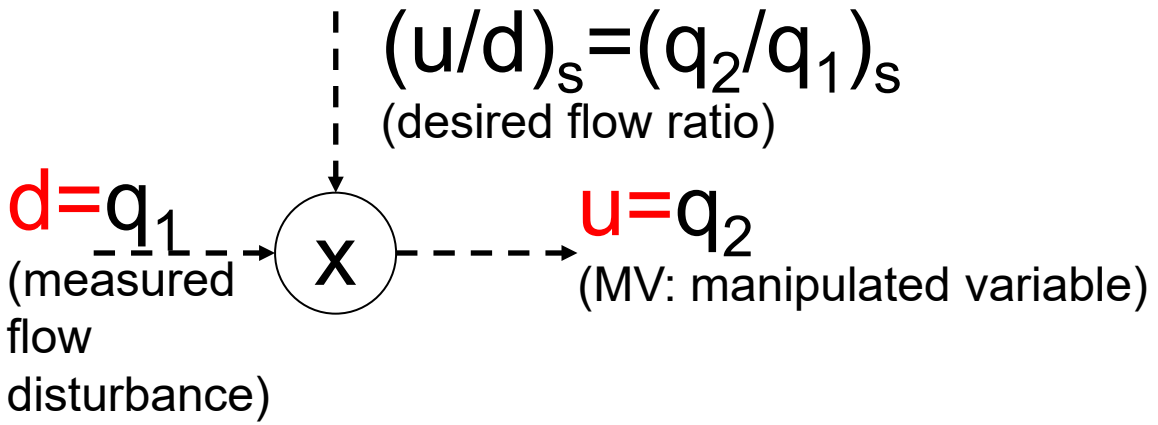
Convergence rate (of inner loop):

- 63% after 1τ
- 98% after 4τ (recommended lower limit)
- To be safe (process changes): 10 τ

Invention 3. Ratio control

Example: Process with two feeds $d=q_1$ and $u=q_2$, where ratio should be constant.

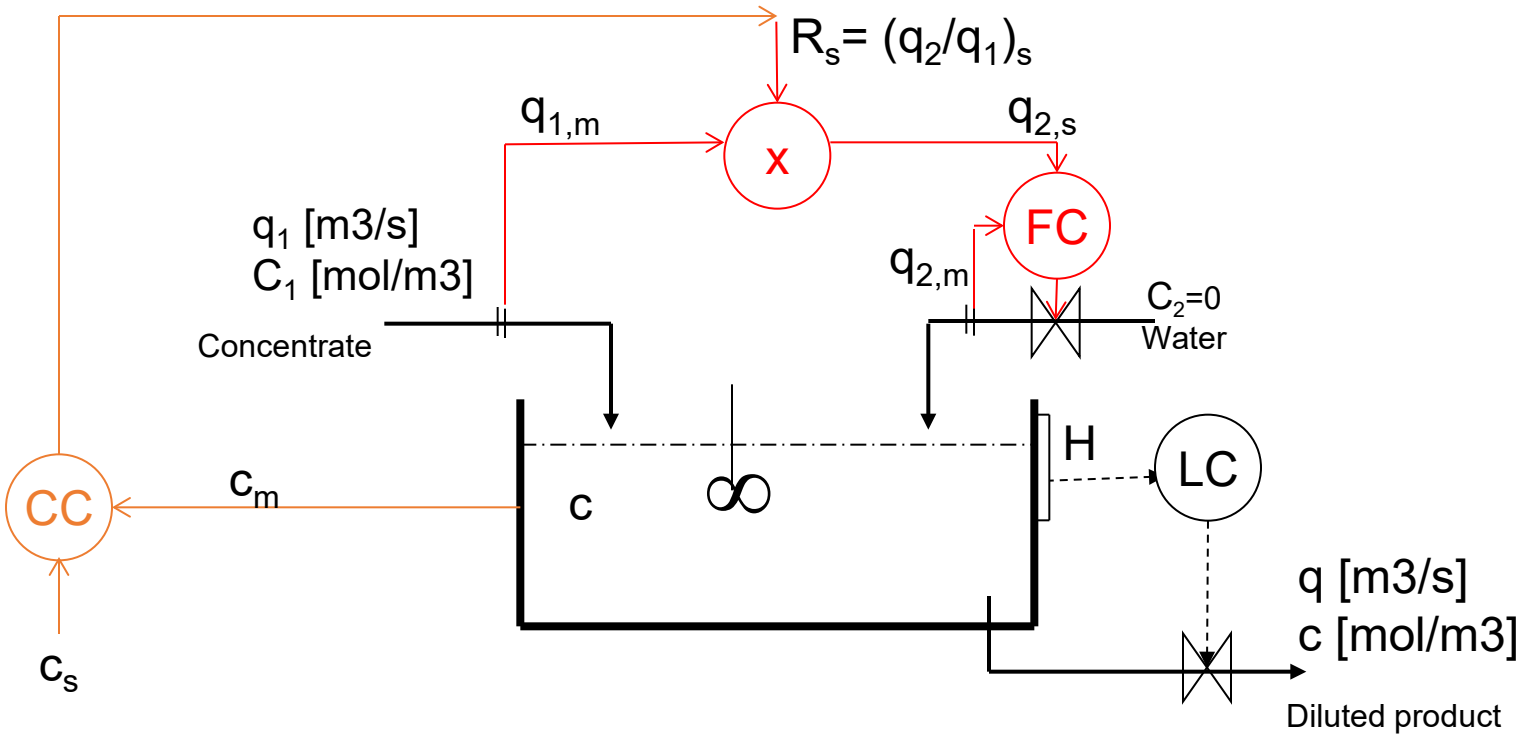
Use multiplication block (x):



Usually: Combine ratio (feedforward) with feedback

***Example cake baking**: Use recipe (ratio control = feedforward), but a good cook adjusts the ratio to get desired result (feedback)*

EXAMPLE: RATIO CONTROL FOR MIXING PROCESS



CC: outer "cascade" = feedback trim" (correction) of ratio setpoint

Later: Will see that this is a special case of **input transformation:**

Transformed input (as seen from feedback controller CC) is $v = (q_2/q_1)_s$

The three main inventions of process control can only indirectly and with effort be implemented with MPC

1. Integral action with MPC: Need to add artificial integrating disturbance in estimator
    - ARC: Just add an integrator in the controller (use PID)

2. Cascade control with MPC: Need model for how u and d affect y_1 and y_2.
    - ARC: Just need to know that control of y_2 indirectly improves control of y_1

3. Ratio control with MPC: Need model for how u and d affect property y
    - ARC: Just need the insight that it is good for control of y to keep the ratio R=u/d constant

Because of this, MPC should be on top of a regulatory control layer with the setpoints for y_2 and R as MVs.

PID control

$$u(t) = K_c e(t) + K_c \tau_D \frac{de(t)}{dt} + \underbrace{\overbrace{\frac{K_c}{\tau_I} \int_{t_0}^{t} e(t')dt'}^{u_I} + u_0}_{\text{bias}=b}$$

K_c = controller gain

τ_I = integral time [s, min]

τ_D = derivative time [s, min]

- «You need a PhD to tune a PID»

AMERICAN CONTROL CONFERENCE
San Diego, California
June 6-8, 1984

IMPLICATIONS OF INTERNAL MODEL CONTROL FOR PID CONTROLLERS

Manfred Morari
Sigurd Skogestad

Daniel F. Rivera

California Institute of Technology
Department of Chemical Engineering
Pasadena, California 91125

University of Wisconsin
Department of Chemical Engineering
Madison, Wisconsin 53706

252 *Ind. Eng. Chem. Process Des. Dev.* **1986**, *25*, 252-265

Internal Model Control. 4. PID Controller Design

Daniel E. Rivera, Manfred Morari,* and Sigurd Skogestad

Chemical Engineering, 206-41, California Institute of Technology, Pasadena, California 91125

For a large number of single input–single output (SISO) models typically used in the process industries, the Internal Model Control (IMC) design procedure is shown to lead to PID controllers, occasionally augmented with a first-order lag. These PID controllers have as their only tuning parameter the closed-loop time constant or, equivalently, the closed-loop bandwidth. On-line adjustments are therefore much simpler than for general PID controllers. As a special case, PI- and PID-tuning rules for systems modeled by a first-order lag with dead time are derived analytically. The superiority of these rules in terms of both closed-loop performance and robustness is demonstrated.

SIMC tuning rule = Generalized Lambda-tuning

Probably the best **simple** PID tuning rules in the world

Sigurd Skogestad*
Department of Chemical Engineering
Norwegian University of Science and Technology
N–7491 Trondheim Norway

Presented at AIChE Annual meeting, Reno, NV, USA, 04-09 Nov. 2001
Paper no. 276h
This version: November 19, 2001†

Abstract

The aim of this paper is to present analytic tuning rules which are as simple as possible and still result in a good closed-loop behavior. The starting point has been the IMC PID tuning rules of Rivera, Morari and Skogestad (1986) which have achieved widespread industrial acceptance. The integral term has been modified to improve disturbance rejection for integrating processes. Furthermore, rather than deriving separate rules for each transfer function model, we start by approximating the process by a first-order plus delay processes (using the "half method"), and then use a single tuning rule. This is much simpler and appears to give controller tunings with comparable performance. All the tunings are derived analytically and are thus very suitable for teaching.

1 Introduction

Hundreds, if not thousands, of papers have been written on tuning of PID controllers, and one must question the need for another one. The first justification is that PID controller is by far the most widely used control algorithm in the process industry, and that improvements in tuning of PID controllers will have a significant practical impact. The second justification is that the simple rules and insights presented in this paper may contribute to a significantly improved understanding into how the controller should be tuned.

ELSEVIER

Journal of Process Control 13 (2003) 291–309

JOURNAL OF PROCESS CONTROL

www.elsevier.com/locate/jprocont

Simple analytic rules for model reduction and PID controller tuning☆

Sigurd Skogestad*

Department of Chemical Engineering, Norwegian University of Science and Technology, N-7491 Trondheim, Norway

Received 18 December 2001; received in revised form 25 June 2002; accepted 11 July 2002

Abstract

The aim of this paper is to present analytic rules for PID controller tuning that are simple and still result in good closed-loop behavior. The starting point has been the IMC-PID tuning rules that have achieved widespread industrial acceptance. The rule for the integral term has been modified to improve disturbance rejection for integrating processes. Furthermore, rather than deriving separate rules for each transfer function model, there is a just a single tuning rule for a first-order or second-order time delay model. Simple analytic rules for model reduction are presented to obtain a model in this form, including the "half rule" for obtaining the effective time delay.

Keywords: Process control; Feedback control; IMC; PI-control; Integrating process; Time delay

1. Introduction

Although the proportional-integral-derivative (PID) controller has only three parameters, it is not easy, without a systematic procedure, to find good values (settings) for them. In fact, a visit to a process plant will usually show that a large number of the PID controllers are poorly tuned. The tuning rules presented in this paper have developed mainly as a result of teaching this material, where there are several objectives:

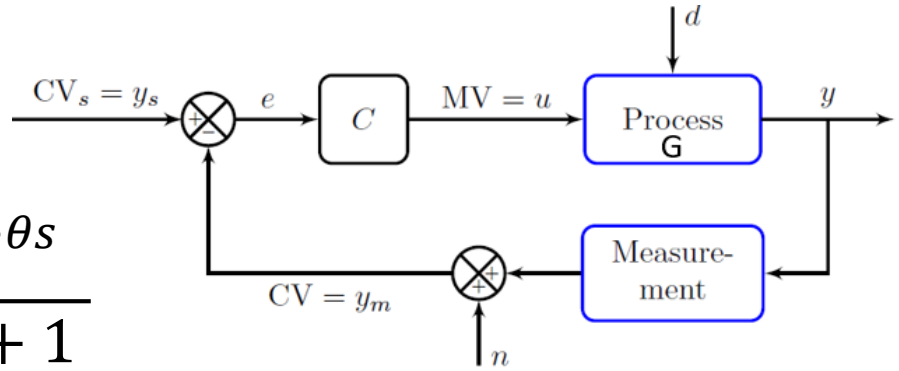
1. The tuning rules should be well motivated, and preferably model-based and analytically derived.
2. They should be simple and easy to memorize.
3. They should work well on a wide range of processes.

Step 2. Derive model-based controller settings. PI-settings result if we start from a first-order model, whereas PID-settings result from a second-order model.

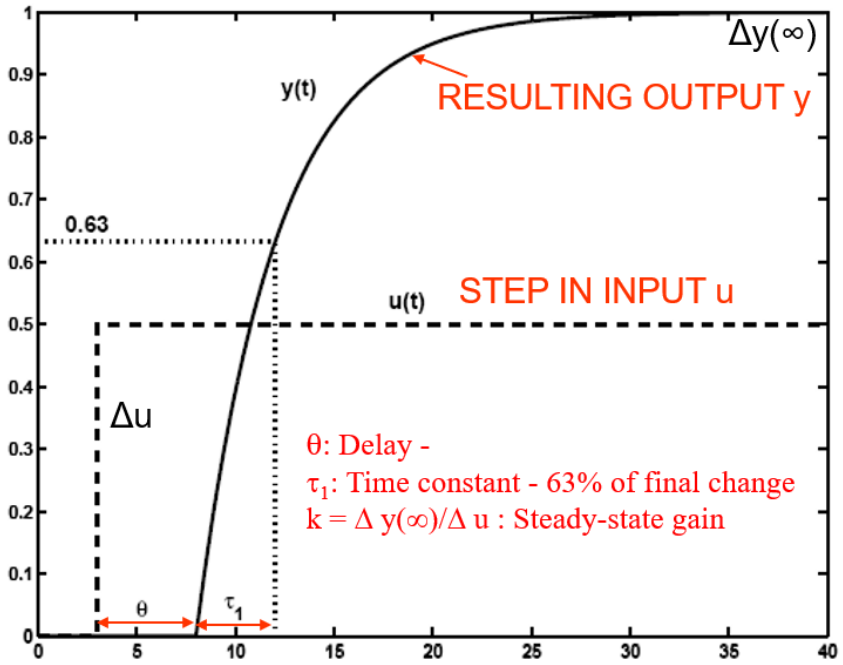
There has been previous work along these lines, including the classical paper by Ziegler amd Nichols [1], the IMC PID-tuning paper by Rivera et al. [2], and the closely related direct synthesis tuning rules in the book by Smith and Corripio [3]. The Ziegler–Nichols settings result in a very good disturbance response for integrating processes, but are otherwise known to result in rather aggressive settings [4,5], and also give poor performance for processes with a dominant delay. On the other hand, the analytically derived IMC-settings in [2] are known to result in a poor disturbance response for integrating processes (e.g., [6,7]), but are robust and

Main idea of SIMC-tuning (and Lambda-tuning)

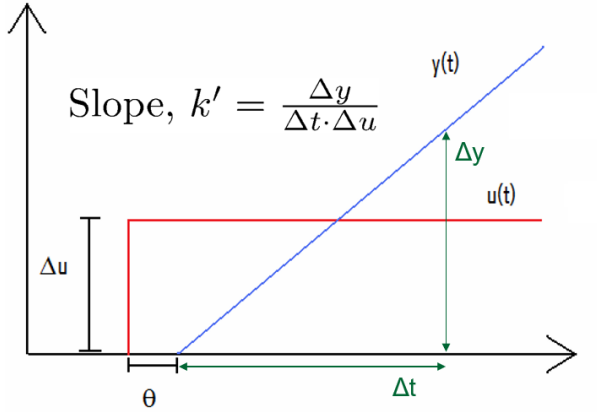
$$G(s) = k\frac{e^{-\theta s}}{\tau s + 1}$$



- τ_c = desired closed-loop time constant = λ
- The ideal «loop shape» for setpoints is GC=1/τ_cs (integrator).
  - Gives setpoint response $\frac{y}{y_s} = \frac{GC}{1+GC} = \frac{1}{\tau_c s+1}$
- Find C by algebra



Step response integrating process



Anti windup is needed for I-action. Recommended «tracking» implementation:

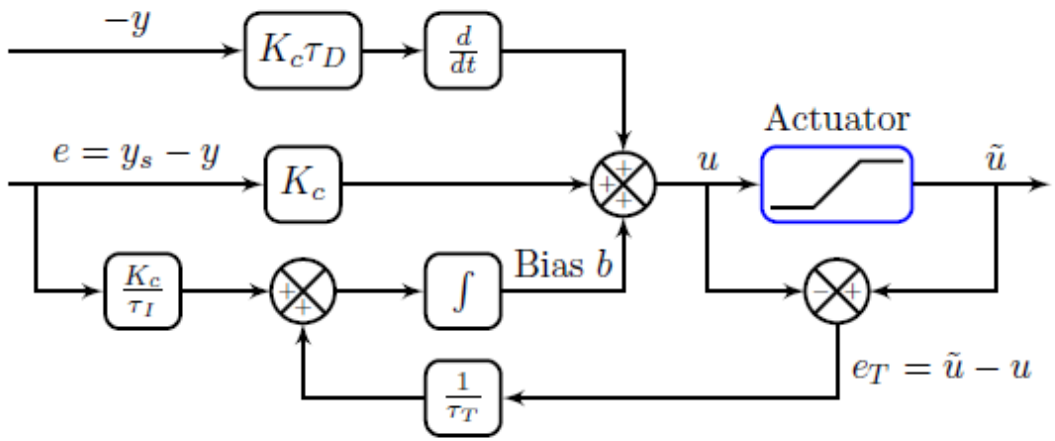
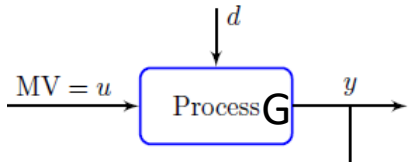


Figure 7: Recommended PID-controller implementation with anti-windup using tracking of the actual controller output ($\tilde{u}$), and without D-action on the setpoint. (Åström & Hägglund, 1988).
$\int$ = integral = $\frac{1}{s}$ in Laplace domain
$\frac{d}{dt}$ = derivative = s in Laplace domain
K_c = controller gain
τ_I = integral time [s, min]
τ_D = derivative time [s, min]
τ_T = tracking time constant for anti-windup [s, min]

τ_T = tracking time (tuning parameter for anti windup)
Common choice: $\tau_T = \tau_I$ (integral time) – same as «external reset» - which is implemented in industrial systems

SIMC PID tuning rule



$$G(s) = k\frac{e^{-\theta s}}{\tau s + 1}$$

k'=k/τ_1

$$K_c = \frac{1}{k'}\frac{1}{\tau_c + \theta}$$

$$\tau_I = \min(\tau_1, 4(\tau_c + \theta))$$

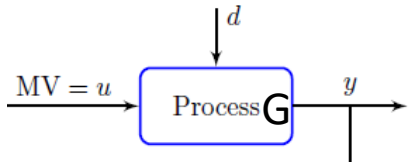
$$\tau_D = \tau_2$$

Only one tuning parameter:
Closed-loop time constant:

$$\tau_c \geq \theta$$

(gives Gain Margin>3)

SIMC PID tuning rule



$$G(s) = k\frac{e^{-\theta s}}{\tau s + 1}$$

k'=k/τ_1

$$K_c = \frac{1}{k'}\frac{1}{\tau_c + \theta}$$

$$\tau_I = \min(\tau_1, 4(\tau_c + \theta))$$

$$\tau_D = \tau_2$$

Only one tuning parameter:
Closed-loop time constant:

$$\tau_c \geq \theta$$

(gives Gain Margin>3)

⇒The «ideal» controller for an integrating process ($\tau = \infty$) is a P-controller

- But need integral action for disturbances, $\tau_I = 4(\tau_c + \theta)$
- Level controller

⇒The «ideal» controller for a delay process (τ=0) is a pure I-controller.

- So use small integral time
- Optimal for pure delay process: $\tau_I = \theta/3$
- Flow controller

- Intermediate cases, select $\tau_I = \tau$ (Lambda–tuning)

Anti-windup …more details..

Normal PID:

$$u(t) = K_c e(t) + K_c \tau_D \frac{de(t)}{dt} + \underbrace{\overbrace{\frac{K_c}{\tau_I}\int_{t_0}^{t} e(t')dt'}^{u_I} + u_0}_{\text{bias}=b} \qquad \text{(C.1)}$$

Appendix C.6.1. Simple anti-windup schemes

Many industrial anti-windup schemes exist. The simplest is to limit u in (C.1) to be within specified bounds (by updating u_0), or to limit the bias $b = u_0 + u_I$ to be within specified bounds (also by updating u_0). These two options have the advantage that one does not need a measurement of the actual applied input value ($\tilde{u}$), and for most loops these simple anti-windup approaches suffice (Smith, 2010) (page 21).

Appendix C.6.2. Anti-windup using external reset

A better and also common anti-windup scheme is "external reset" (e.g., Wade (2004) Smith (2010)) which originates from Shinskey. This scheme is found in most industrial control systems and it uses the "trick" of realizing

Appendix C.6.3. Recommended: Anti-windup with tracking

The "external reset" solution is a special case of the further improved "tracking" scheme in Figure 7 which is recommended by Åström & Hägglund (1988). The tracking scheme (sometimes referred to as the "back-calculation" scheme (Åström & Hägglund, 2006)) has a very useful additional design parameter, namely the tracking time constant τ_T, which tells how fast the controller output u tracks the actual applied value $\tilde{u}$. This makes it possible to handle more general cases in a good way, e.g., switching of CVs. In the simpler "external reset" scheme, the tracking time is "by design" equal to the integral time ($\tau_T = \tau_I$) (Åström & Hägglund, 1988).

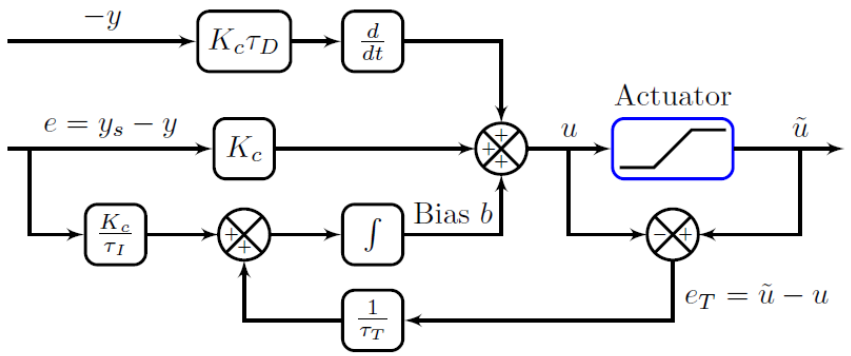


Figure 7: Recommended PID-controller implementation with anti-windup using tracking of the actual controller output ($\tilde{u}$), and without D-action on the setpoint. (Åström & Hägglund, 1988).

With AW:

$$u(t) = K_c e(t) + K_c \tau_D \frac{de(t)}{dt} + \underbrace{\overbrace{\int_{\hat{t}=t_0}^{t} \left(\frac{K_c}{\tau_I} e(\hat{t}) + \frac{1}{\tau_T} e_T(\hat{t})\right) d\hat{t}}^{u_I(t)} + u_0}_{\text{bias}=b} \qquad \text{(C.14)}$$

to choose the tracking time equal to the integral time ($\tau_T = \tau_I$). With this value, we get at steady state that the output from the integral part (u_I) is such that the bias b is equal to the constraint value, $b = u_{lim}$. To derive this, note that with

MV-MV switching

Anti-windup with cascade control

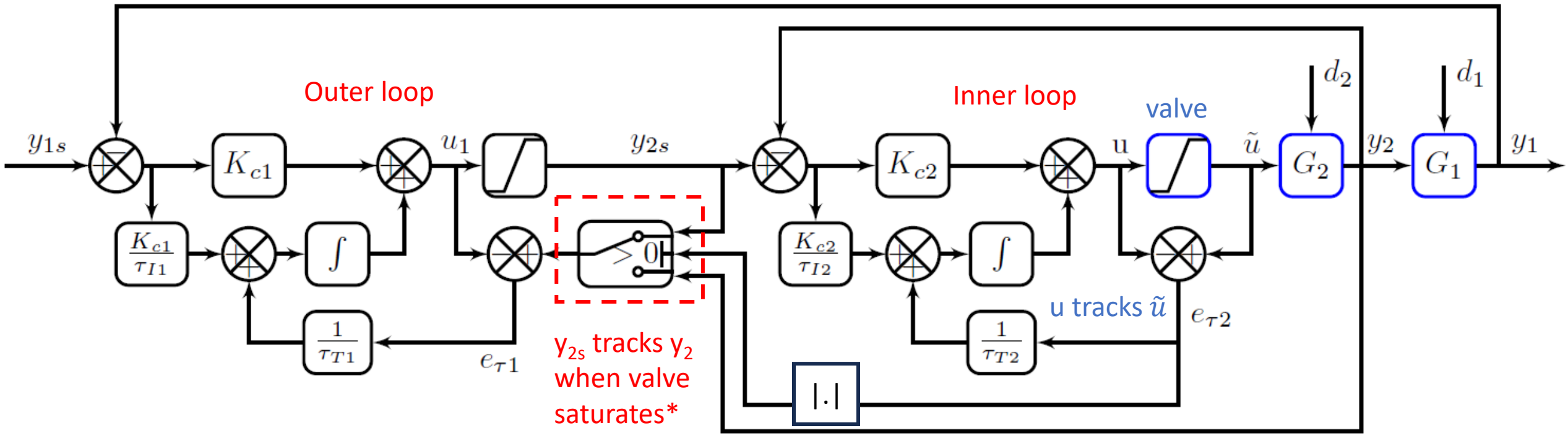


Figure 25: Cascade control with anti windup using the industrial switching approach (Leal et al., 2021).

* Normally, it's opposite: y_2 is tracking y_{2s}.

Filters for setpoints and measurements

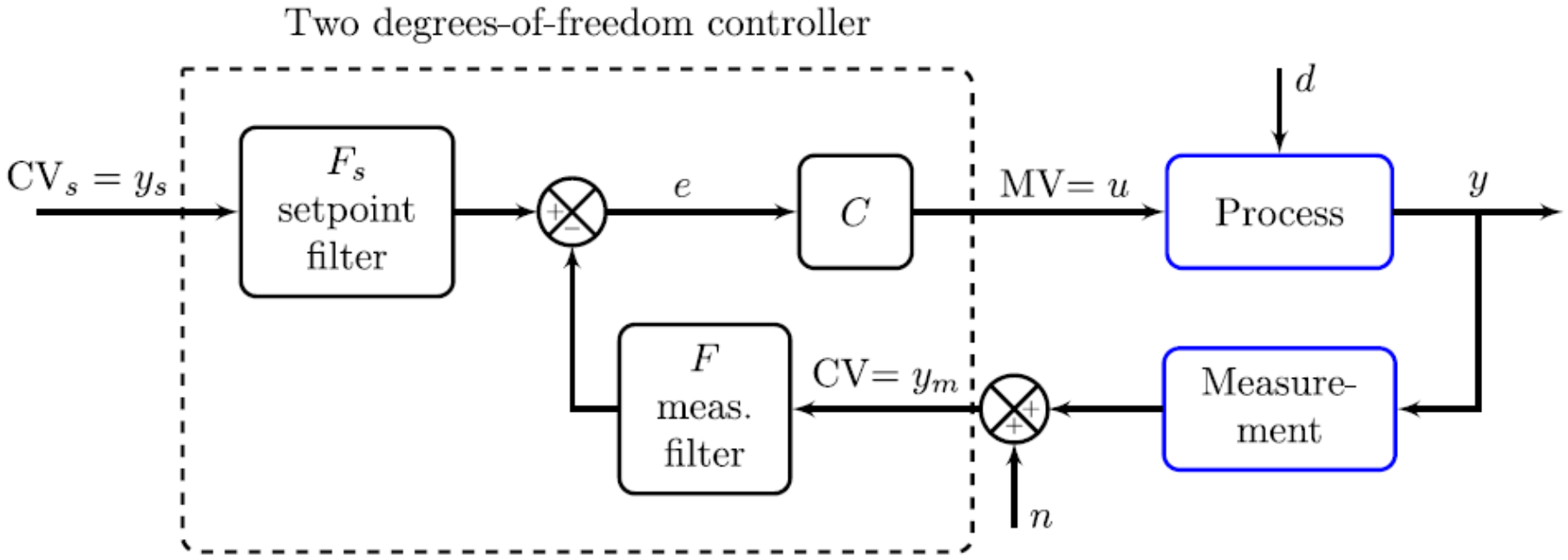


Fig. A.41. Two degrees-of-freedom control system with setpoint filter F_s and measurement filter F. All blocks are possibly nonlinear.

- Measurement filter F. Typical use first-order $F=1/(\tau_F s+1)$.
  - Filter time constant, $\tau_F \leq \frac{\tau_c}{2}$ (preferably much smaller to avoid introducing effective delay in loop)
- Setpoint filter F_s. Typical use lead-lag.
  - β-factor on P-action for setpoints corresponds to: $F_s(s) = \frac{\beta\tau_I s+1}{\tau_I s+1}$

Scaling of variables

Scale input u and output y to be 0-100%

- u = original physical units, u' = scaled units (0-100%)

$$u = \frac{u' - u'_{min}}{u'_{max} - u'_{min}} \cdot 100\%$$

Advantages:

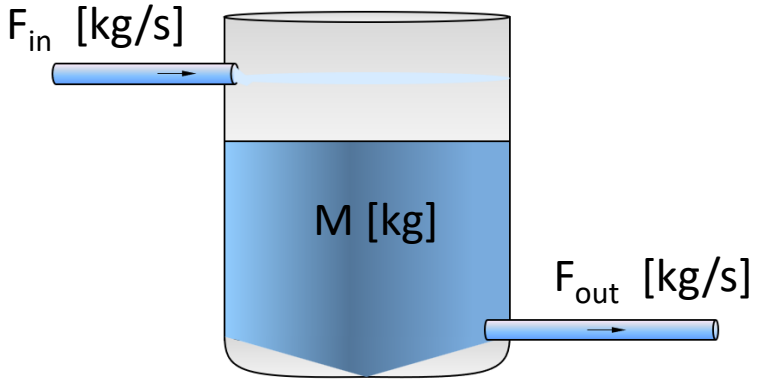
- Controller gain is dimensionless
- Can control actuator (valve) to max and min position
  - u=0%: Valve closed
  - u=100%: Valve fully open

Example level control (often poorly tuned!)

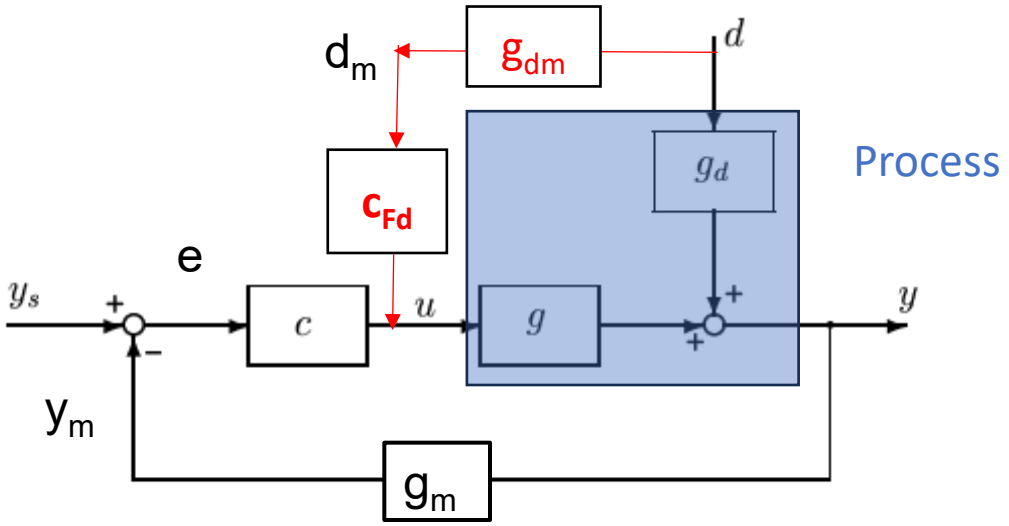
- Model: $dM/dt = F_{in} - F_{out}$ [kg/s]
- Scaling: $u = F/F_{max} \cdot 100\%$

  $y = (M/M_{max}) \cdot 100\%$
- Scaled model: $dy/dt = k' (u_{in} - u_{out})$,

  $k' = 1/\tau$, $\tau = M_{max}/F_{max}$ = tank residence time
- P-control. Averaging level control, $K_c = 2$ %/%
- Add I-action, but avoid slow cycling for integratring process (SIMC): $K_c \tau_I \geq 4 \tau$



Feedforward control: Measure disturbance (d)



c = Feedback controller

c_{Fd} = Feedforward controller

Get: $y = (g\boldsymbol{c_{Fd}}\,\boldsymbol{g_{dm}} + g_d)\,d$

Ideal y=0 ⇒ $\boldsymbol{c_{Fd} = -g^{-1} g_d\, g_{dm}^{-1}}$ (invert process model)

Usually: Add feedforward when feedback alone is not good enough, for example, because of measurement delay in g_m

Main problem with feedforward:
Sensitive to model error

Main problem feedforward: Sensitive to model error

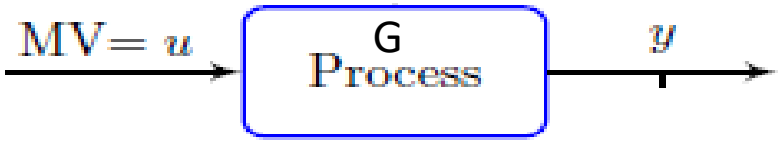
- "If process gain increases by more than a factor 2, then ideal feedforward control is worse than no control"
- Why? Overcompensate in wrong direction
  - Proof: $y = gu + g_d\, d$ where $u = c_{FF}\, g_{dm}\, d$
  - Response with feedforward controller:
    $y = (g\, c_{FF}\, g_{dm} + g_d)\, d$
  - Ideal: Use $c_{FF,ideal} = -\, g_d/g\, g_{dm}$. Gives $y = (-g_d + g_d)\, d = 0\, d$
  - But note that g is $c_{FF,ideal}$ is a model
  - Real: If the real process gain (g) has increased by a factor x then
    $y = (-x g_d + g_d)\, d = (-x+1)\, g_d\, d$
    For $x>2$: $|-x+1|>1$ (worse than no control)….

Examples. Krister

- Tuning flow controller
- Feed forward
- QUIZ: Alternative to feed forward

Discussion: What is best? Feedback or feedforward?

Example: Feedback vs. feedforward for setpoint control of uncertain process



y = G(s) u

$$G(s) = \frac{k}{\tau s + 1}, \quad k = 3,\ \tau = 6 \qquad \text{(B.2)}$$

Desired response : $y = \frac{1}{\tau_c s + 1} y_s = \frac{1}{4s + 1} y_s$

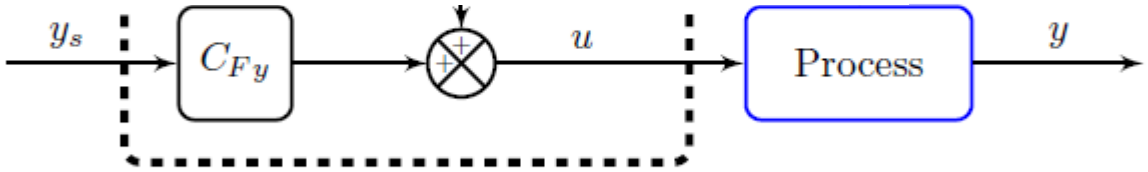


Figure A.42: Block diagram of feedforward control system with linear combination of feedforward from measured disturbance (d) and setpoint (y_s) (E14).

Feedforward solution. We use feedforward from the setpoint (Fig. A.42):

$$u = C_{Fy}(s) y_s$$

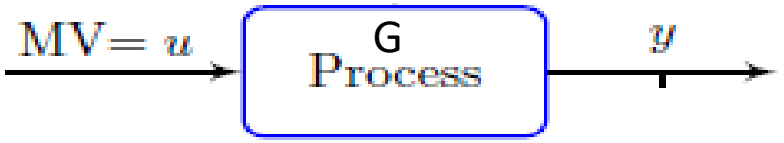
where we choose

$$C_{Fy}(s) = \frac{1}{\tau_c s + 1} G(s)^{-1} = \frac{1}{k} \frac{\tau s + 1}{\tau_c s + 1} = \frac{1}{3} \frac{6s + 1}{4s + 1} \qquad \text{(B.3)}$$

The output response becomes as desired,

$$y = \frac{1}{4s + 1} y_s \qquad \text{(B.4)}$$

Example: Feedback vs. feedforward for setpoint control of uncertain process



y = G(s) u

$$G(s) = \frac{k}{\tau s + 1}, \quad k = 3,\ \tau = 6 \qquad \text{(B.2)}$$

Desired response : $y = \frac{1}{\tau_c s + 1} y_s = \frac{1}{4s + 1} y_s$

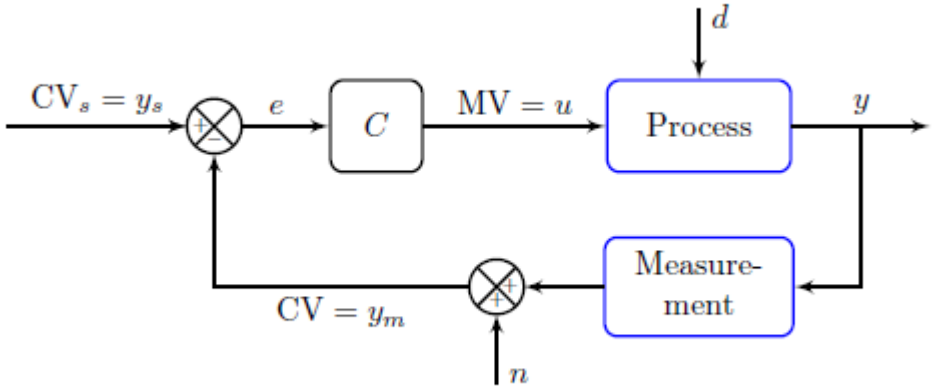


Figure 3: Block diagram of common "one degree-of-freedom" negative feedback control system.

Feedback solution. We use a one degree-of-freedom feedback controller (Fig. 3) acting on the error signal $e = y_s - y$:

$$u = C(s)(y_s - y)$$

We choose a PI-controller with $K_c = 0.5$ and $\tau_I = \tau = 6$ (using the SIMC PI-rule with $\tau_c = 4$, see Appendix C.2):

$$C(s) = K_c \left(1 + \frac{1}{\tau_I s}\right) = 0.5 \frac{6s + 1}{6s} \qquad \text{(B.5)}$$

Note that we have selected $\tau_I = \tau = 6$, which implies that the zero dynamics in the PI-controller C, cancel the pole dynamics of the process G. The closed-loop response becomes as desired:

$$y = \frac{1}{\tau_c s + 1} y_s = \frac{1}{4s + 1} y_s \qquad \text{(B.6)}$$

Proof. $y = T(s) y_s$ where $T = L/(1 + L)$ and $L = GC = kK_c/(\tau_I s) = 0.25/s$. So $T = \frac{0.25/s}{1 + 0.25/s} = \frac{1}{4s+1}$.

Thus, we have two fundamentally different solutions that give the same nominal response, both in terms of the process input $u(t)$ (not shown) and the process output $y(t)$ (black solid curve in Fig. B.43).

- But what happens if the process changes?
  - Consider a gain change so that the model is wrong
    - Process gain from k=3 to k'=4.5

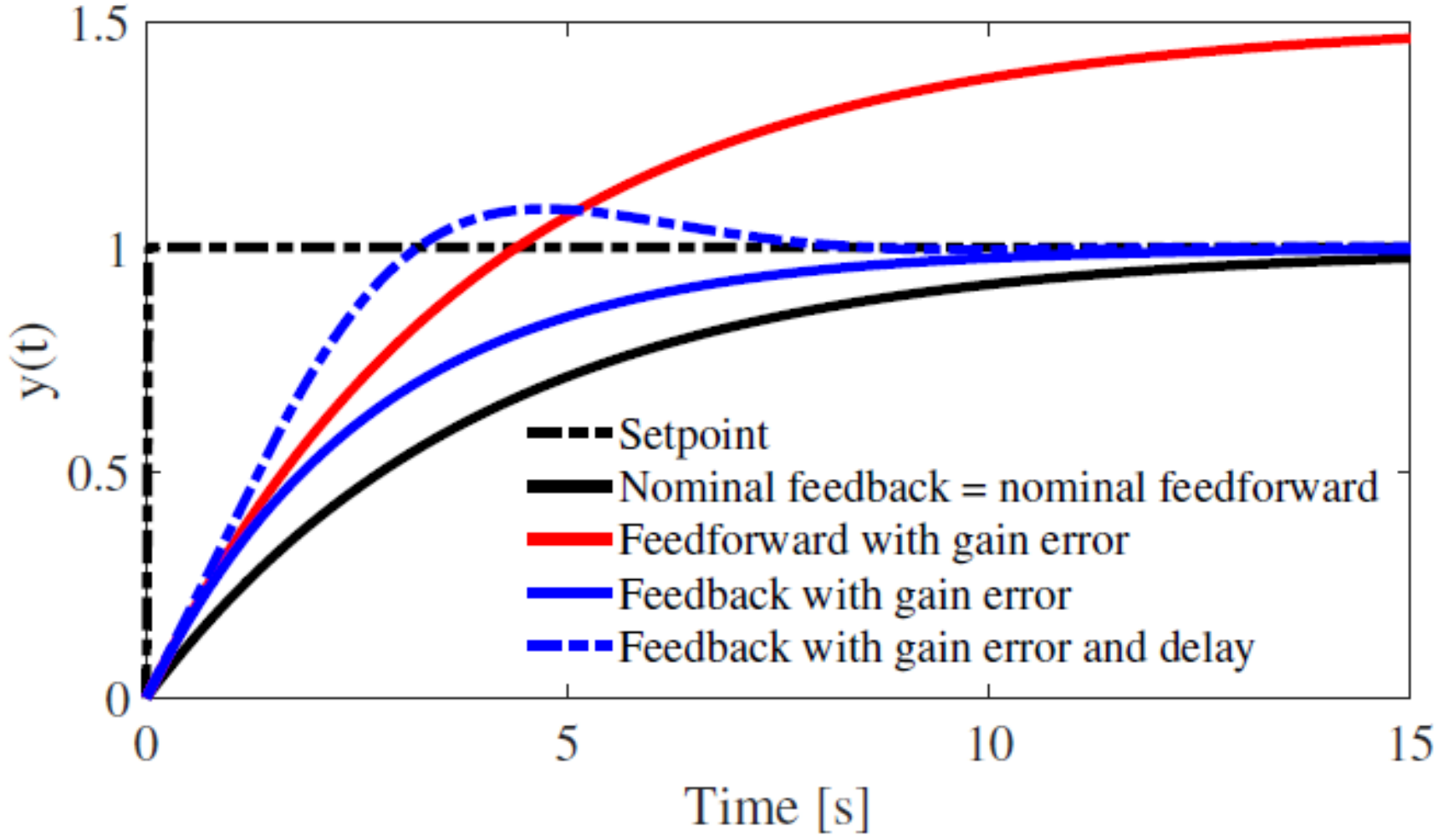


Figure B.43: Setpoint response for process (B.2) demonstrating the advantage of feedback control for handling model error.

Gain error (feedback and feedforward): From k=3 to k'=4.5

Time delay (feedback): From $\theta = 0$ *to* $\theta = 1.5$