

Standard Advanced control elements

- Each element links a subset of inputs with a subset of outputs
- Results in simple local tuning

First, there are some elements that are used to improve control for cases where simple feedback control is not sufficient:

- E1***. Cascade control²
- E2***. Ratio control
- E3***. Valve (input)³ position control (VPC) on extra MV to improve dynamic response.

Next, there are some control elements used for cases when we reach constraints:

- E4***. Selective (limit, override) control (for output switching)
- E5***. Split range control (for input switching)
- E6***. Separate controllers (with different setpoints) as an alternative to split range control (E5)
- E7***. VPC as an alternative to split range control (E5)

All the above seven elements have feedback control as a main feature and are usually based on PID controllers. Ratio control seems to be an exception, but the desired ratio setpoint is usually set by an outer feedback controller. There are also several features that may be added to the standard PID controller, including

- E8***. Anti-windup scheme for the integral mode
- E9***. Two-degrees of freedom features (e.g., no derivative action on setpoint, setpoint filter)
- E10**. Gain scheduling (Controller tunings change as a given function of the scheduling variable, e.g., a disturbance, process input, process output, setpoint or control error)

In addition, the following more general model-based elements are in common use:

- E11***. Feedforward control
- E12***. Decoupling elements (usually designed using feedforward thinking)
- E13**. Linearization elements
- E14***. Calculation blocks (including nonlinear feedforward and decoupling)
- E15**. Simple static estimators (also known as inferential elements or soft sensors)

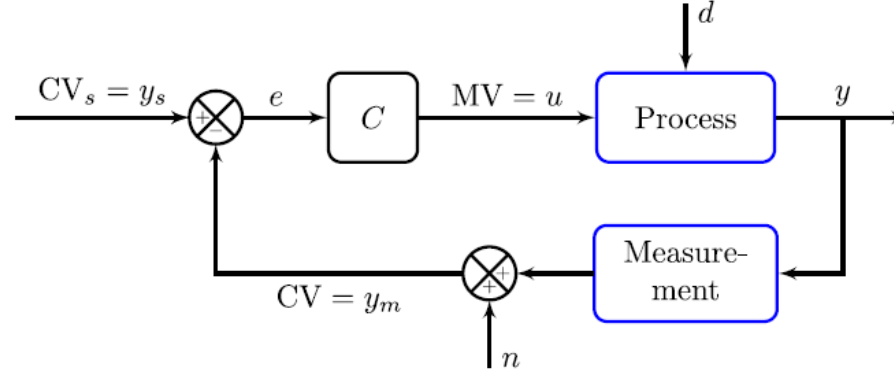
Finally, there are a number of simpler standard elements that may be used independently or as part of other elements, such as

- E16**. Simple nonlinear static elements (like multiplication, division, square root, dead zone, dead band, limiter (saturation element), on/off)
- E17***. Simple linear dynamic elements (like lead-lag filter, time delay, etc.)
- E18**. Standard logic elements

² The control elements with an asterisk * are discussed in more detail in this paper.

³ In this paper, Valve Position Control (VPC) refers to cases where the input (independent variable) is controlled to a given setpoint (“ideal resting value”) on a slow time scale. Thus, the term VPC is used for other inputs (actuator signals) than valve position, including pump power, compressor speed and flowrate, so a better term might have been Input Position Control.

Most basic element: Single-loop PID control (E0)



MV-CV Pairing. Two main pairing rules (supervisory layer*):

1. **“Pair-close rule”** : The MV should have a large, fast, and direct effect on the CV.
2. **“Input saturation rule”**: Pair a MV that may saturate with a CV that can be given up (when the MV saturates).*
 - Exception: Have extra MV so we use MV-MV switching (e.g., split range control)

Additional rule for interactive systems:

3. **“RGA-rule”**
 - Avoid pairing on negative steady-state RGA-element. Otherwise, the loop gain may change sign (for example, if the input saturates) and we get instability with integral action in the controller.

*For regulatory (stabilizing) control, we usually want to avoid using any MV that may saturate (so Rule 2 becomes: Avoid using a MV that may saturate), but for the supervisory layer this is not possible

Details on RGA-rule

Pairing rule 1 (page 450): *Prefer pairings such that the rearranged system, with the selected pairings along the diagonal, has an RGA matrix close to identity at frequencies around the closed-loop bandwidth.*

However, one should avoid pairings where the sign of the steady-state gain from u_j to y_i may change depending on the control of the other outputs, because this will yield instability with integral action in the loop. Thus, $g_{ij}(0)$ and $\hat{g}_{11}(0)$ should have the same sign, and we have:

Pairing rule 2 (page 450): *Avoid (if possible) pairing on negative steady-state RGA elements.*

The reader is referred to Section 10.6.4 (page 438) for derivation and further discussion of these pairing rules.

Most common “Advanced regulatory control” structures

Used when single-loop feedback control (PID) alone is not good enough

1. Cascade control (measure and control internal variable) **E1**
2. Feedforward control (measure disturbance, d) **E11**
 - (Very*) special case: ratio control **E2**
3. Extra MV dynamically: Valve position control **E3**
 - Also known as input resetting or midranging
4. Change in CV: Selectors (max, min) **E4**
5. Extra MV steady state: Split range control (+2 alternatives) **E5 (+E6, E7)**

All of these are extensively used in practice, but little academic work

*Ratio control is a «very» special case of feedforward because it requires no model for the output (property) y we want to control.

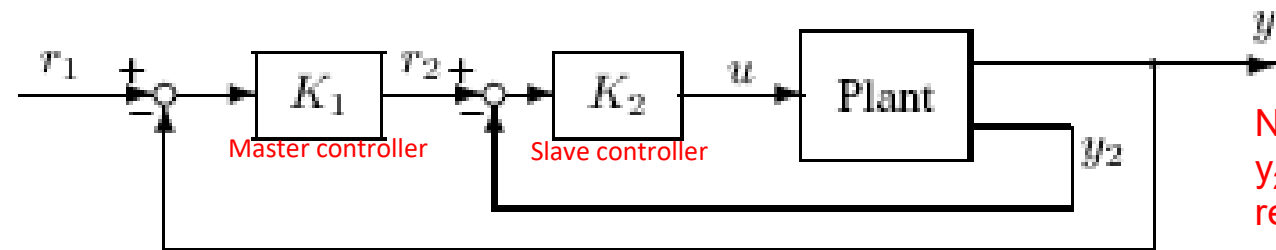
Standard advanced control elements

- E1-E18

E1. Cascade control

- 1 input u
- 1 (main) output y_1
- 1 extra measurement y_2
- **Key assumption: Control of y_2 indirectly makes it easier to control y_1**
- Solution: Primary controller (master) controls y_1 by setting setpoint $r_2 = y_2^{sp}$ to a fast secondary controller (slave) which manipulates u

General case (“parallel cascade”)



Not always helpful...
 y_2 must be closely related to y_1

(a) Extra measurements y_2 (conventional cascade control)

Special common case (“series cascade”)

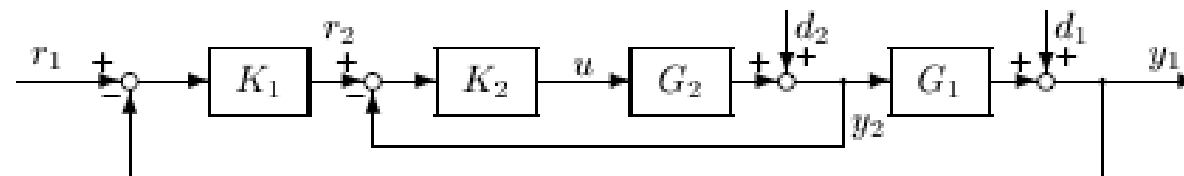
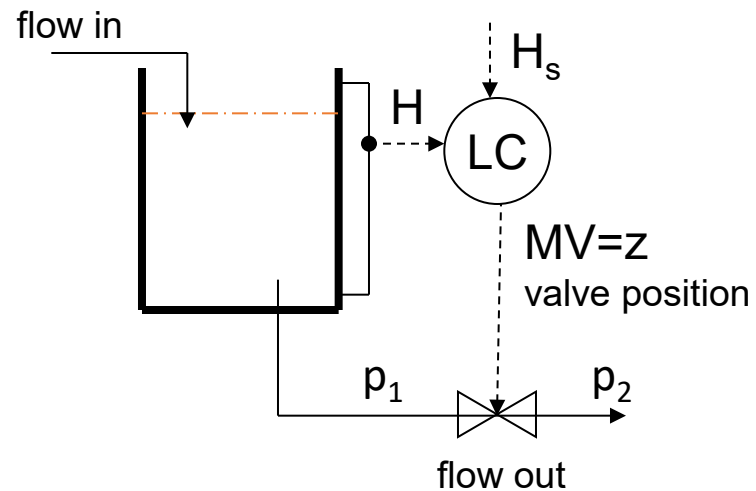


Figure 10.11: Common case of cascade control where the primary output y_1 depends directly on the extra measurement y_2

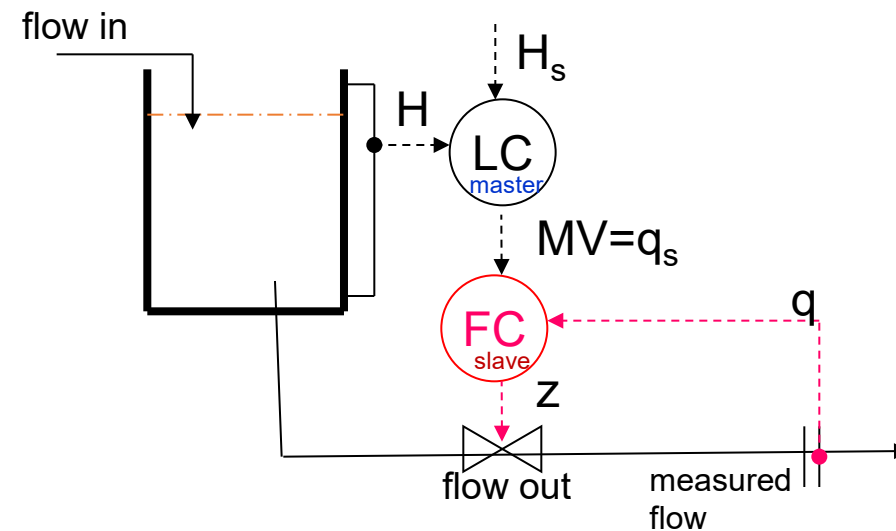
Example: Flow controller on valve (very common!)

- Helpful to reduce valve nonlinearity and provide local disturbance rejection ($d=p_1, p_2$)
 - y = level H in tank (or could be temperature etc.)
 - u = valve position (z)
 - y_2 = flowrate q through valve

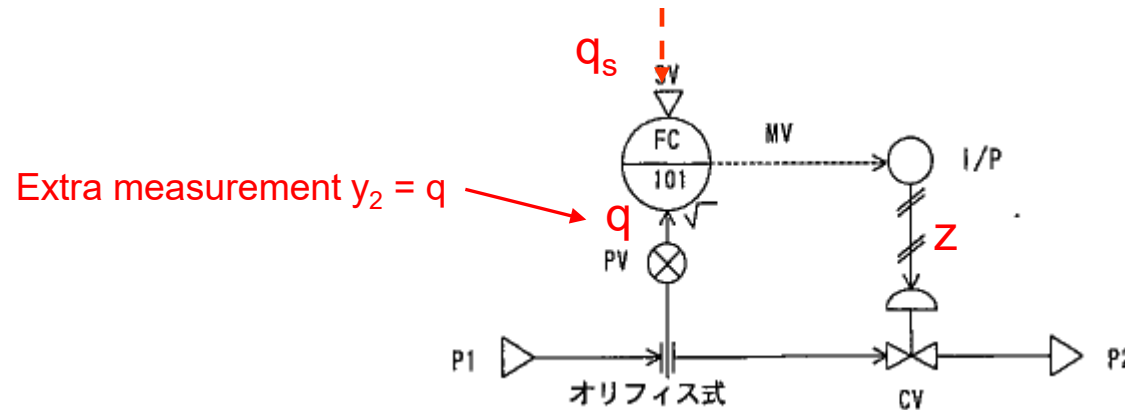
WITHOUT CASCADE



WITH CASCADE

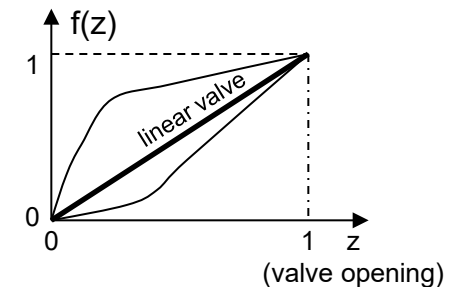


What are the benefits of adding a flow controller (inner cascade)?



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

1. Counteracts nonlinearity in valve, $f(z)$
 - With a fast flow controller we can assume $q = q_s$ (in spite of nonlinearity in the valve)
2. Eliminates effect of disturbances in p_1 and p_2 (FC reacts faster than outer level loop)



Block diagram flow controller

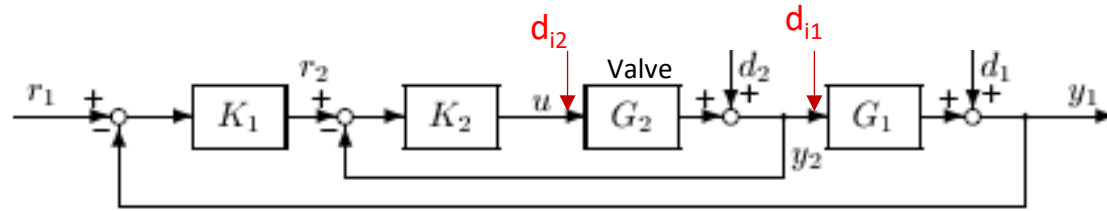
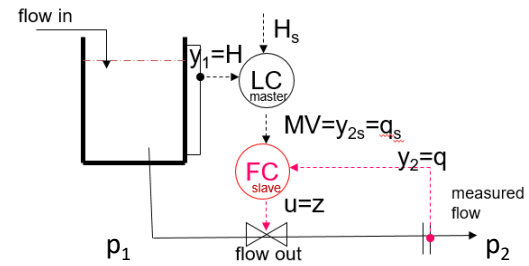


Figure 10.11: Common case of cascade control where the primary output y_1 depends directly on the extra measurement y_2

Example: Level control with slave flow controller:

$u = z$ (valve position, flow out)

$y_1 = H$

$y_2 = q$

$d'_1 = \text{flow in}$

$d_2 = p_1 - p_2$

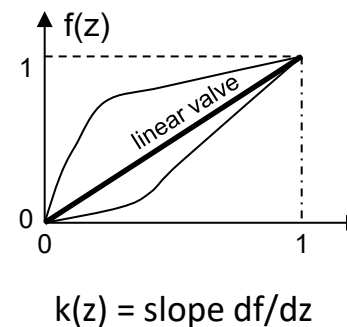
Transfer functions:

$G_2 = k(z)/(\tau s + 1)$ where $k(z) = dq/dz$ (nonlinear!)

$G_1 = -1/(As)$

$K_1 = \text{Level controller (master)}$

$K_2 = \text{Flow controller (slave)}$



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.

When use (series) cascade ?

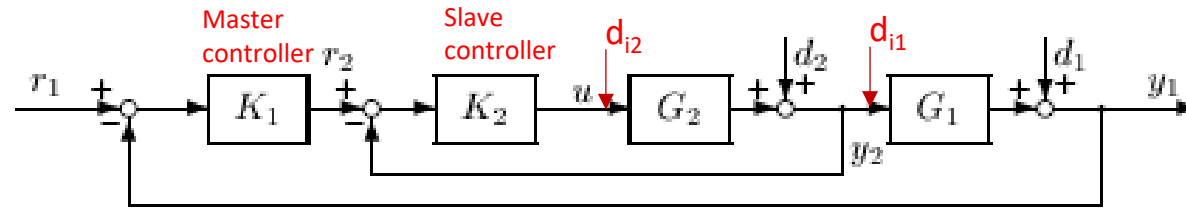


Figure 10.11: Common case of cascade control where the primary output y_1 depends directly on the extra measurement y_2

Use cascade control (with an extra secondary measurement y_2) when one or more of the following occur:

1. **Significant disturbances d_2 and d_{i2} inside slave loop** (and y_2 can be controlled faster than y_1)
2. **The plant G_2 is nonlinear or varies with time or is uncertain.**
3. **Measurement delay for y_1**
 - **Note:** In the flowsheet above, y_1 is the measured output, so any measurement delay is included in G_1
4. **Integrating dynamics** (including slow dynamics or unstable) in both G_1 and G_2 , (because without cascade a double integrating plant G_1G_2 is difficult to control)

Design / tuning

- First design K_2 ("fast loop") to deal with d_2 and d_{i2} (based on model G_2)
- Then, with K_2 closed, design K_1 to deal with d_1 and d_{i1} (based on model G_1T_2)

Transfer functions and tuning

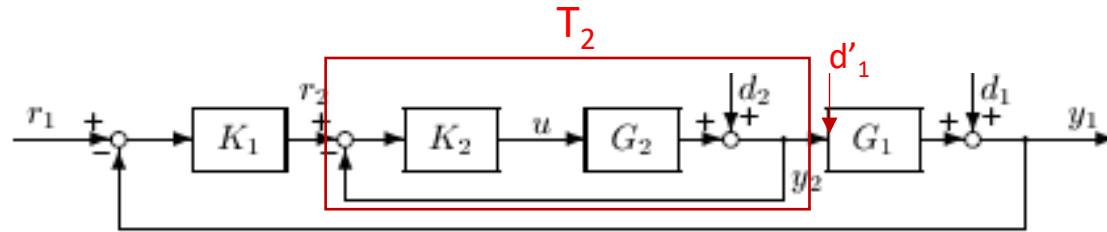


Figure 10.11: Common case of cascade control where the primary output y_1 depends directly on the extra measurement y_2

First tune fast inner controller K_2 (“slave”)

Design K_2 based on model G_2

Select τ_{c2} based on effective delay in G_2

Transfer function for inner loop (from y_{2s} to y_2): $T_2 = G_2 K_2 / (1 + G_2 K_2)$

Because of integral action, T_2 has loop gain = 1 for any G_2 .

With SIMC we get: $T_2 \approx e^{-\theta_2 s} / (\tau_{c2} s + 1)$

Nonlinearity: Gain variations (in G_2) translate into variations in actual time constant τ_{c2} (see next page)

Then with slave closed, tune slower outer controller K_1 (“master”):

Design K_1 based on model $G_1' = T_2 * G_1$

Can often set $T_2 = 1$ if inner loop is fast!

- Alternatively, $T_2 \approx e^{-\theta_2 s} / (\tau_{c2} s + 1) \approx e^{-(\theta_2 + \tau_{c2})s}$
- Even more accurate: Use actual T_2 (normally not necessary)

Typical choice: $\tau_{c1} = \sigma \tau_{c2}$ where time scale separation $\sigma = 4$ to 10.

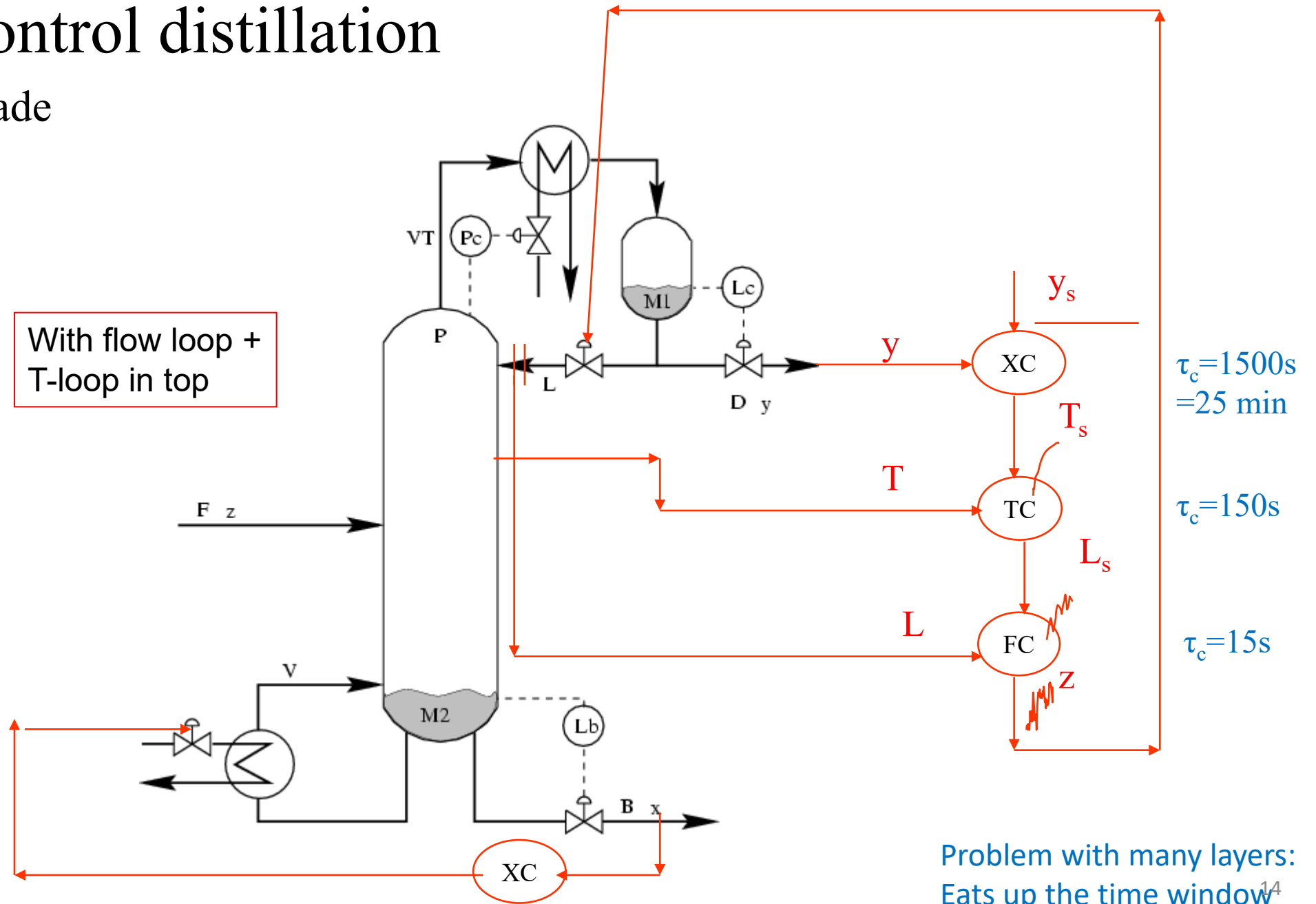
Time scale separation is needed for cascade control to work well

- 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.
- But normally 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 ($\sigma = 10$) is to avoid that control gets too slow, especially if we have many layers

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

Cascade control distillation

3 layers of cascade



Counteracting nonlinearity using cascade control

Example: Consider slave flow controller with $u = z$ (valve position) and $y_2 = q$ (flow)

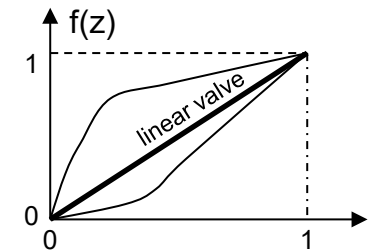
- Nonlinear valve with varying gain k_2 : $G_2(s) = k_2(z) / (\tau_2 s + 1)$
- Slave (flow) controller K_2 : PI-controller with gain K_{c2} and integral time $\tau_i = \tau_2$ (SIMC-rule).
Get

$$L_2 = K_2(s)G_2(s) = \frac{K_{c2}k_2}{\tau_2 s}$$

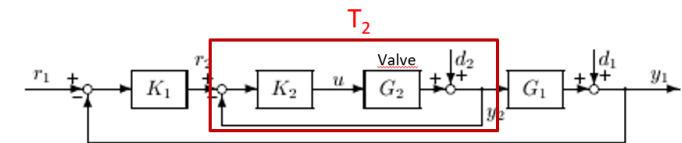
- With slave controller: Transfer function from y_{2s} to y_2 (as seen from outer loop):

$$T_2 = L_2 / (1 + L_2) = 1 / (\tau_{c2} s + 1), \text{ where } \tau_{c2} = \tau_2 / (k_2 K_{c2})$$

- **Important: Gain for T_2 is always 1 (independent of k_2) because of intergal action in the inner (slave) loop**
- **But: Gain variation in k_2 (inner loop) translates into variation in closed-loop time constant τ_{c2} . This may effect the master loop:**
 - The master controller K_1 is designed based on $G_1 T_2$.
 - A smaller process gain k_2 results in a larger τ_{c2} and thus a large effective delay, which mat be bad.
 - Recall $T_2 \approx e^{-\theta_2 s} / (\tau_{c2} s + 1) \approx e^{-(\theta_2 + \tau_{c2})s}$
- However, if the time scale separation σ is sufficiently large, the variations in τ_{c2} will not matter



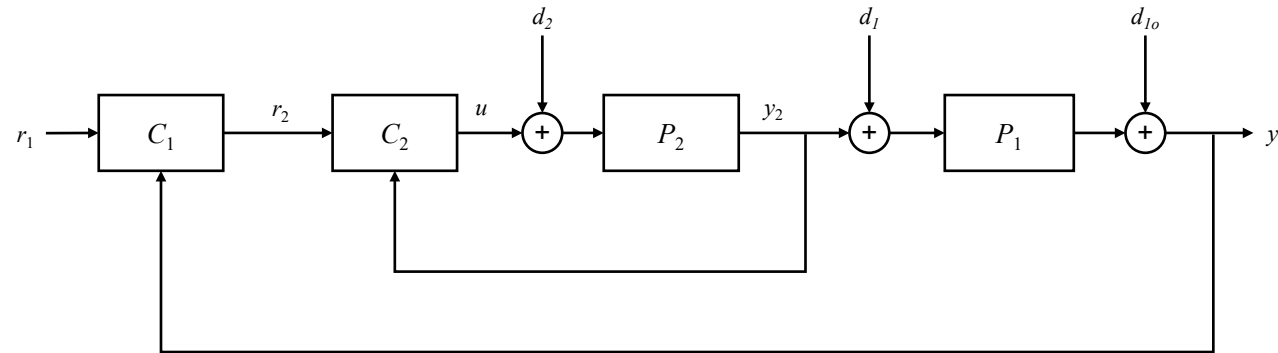
$$k_2(z) = \text{slope} = df/dz$$



$G_1 T_2 = \text{«Process»}$ for tuning master controller K_1

Cascade control block diagram

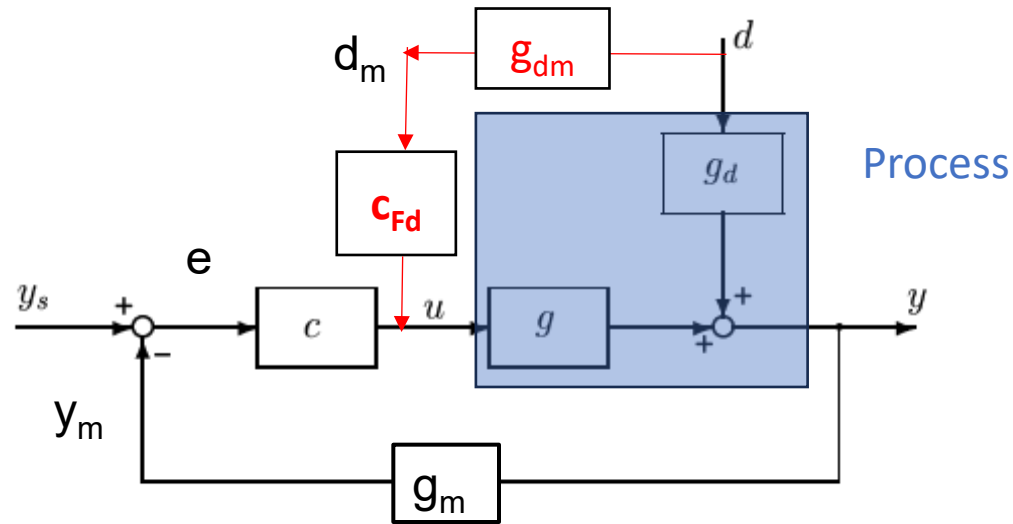
- Which disturbances motivate the use of cascade control?



Answer: d_2

A little on feedforward control (E11)

Feedforward control: Measure disturbance (d)



Block diagram of feedforward control

c = Feedback controller

c_{Fd} = Feedforward controller.

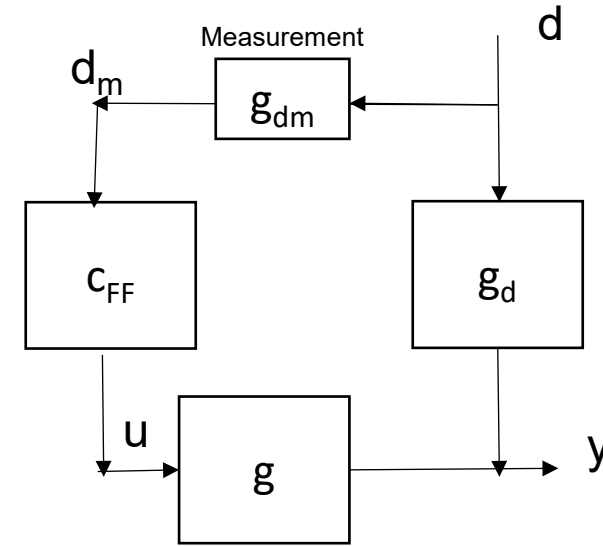
Ideal, inverts process g : $c_{Fd} = g^{-1} g_d g_{dm}^{-1}$

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

Details Feedforward control

- Model: $y = g u + g_d d$
- Measured disturbance: $d_m = g_{dm} d$
- Feedforward controller: $u = c_{FF} d_m$
- Get $y = (g c_{FF} g_{dm} + g_d) d$
- Ideal feedforward:
 - $y = 0 \Rightarrow c_{FF, ideal} = -g^{-1} g_d g_{dm}^{-1} = -\frac{g_d}{g_{dm} g}$
- In practice: $c_{FF}(s)$ must be realizable
 - Order pole polynomial \geq order zero polynomial
 - No prediction allowed (θ cannot be negative)
 - Must avoid that c_{FF} has too high gain to avoid (to avoid aggressive input changes)
- Common simplification: $c_{FF} = k$ (static gain)
- General. Approximate $c_{FF, ideal}$ as :

$$c_{FF}(s) = k \frac{(T_1 s + 1) \dots}{(\tau_1 s + 1)(\tau_2 s + 1) \dots} e^{-\theta s}$$



where we must have at least as many τ 's as T 's

Example feedforward

$$y = gu + g_{d1}d_1 + g_{d2}d_2$$

Feedforward control: $u = c_{FF}d_m$

Ideal feedforward controller: $c_{FF} = -\frac{g_d}{g_{dm}g}$

Example (assume perfect measurements, $g_{dm} = 1$):

$$g(s) = \frac{e^{-s}}{s(20s+1)}$$

$$g_{d1}(s) = \frac{1}{s}$$

$$g_{d2}(s) = \frac{e^{-s}}{s(20s+1)}$$

Disturbance 1:

Ideal: $c_{FF1} = -(20s + 1)e^s$ (has prediction + has more zeros than poles)

Actual: $c_{FF1} = -1 \cdot \frac{20s+1}{\tau s+1}$ where τ is tuning parameter

(smaller τ gives better control, but requires more input usage).

Comment: In the simulation we use $\tau = 2$ which is quite aggressive; $\tau = 20$ would give $c_{FF1} = -1$.

Disturbance 2:

Ideal: $c_{FF2} = -1$

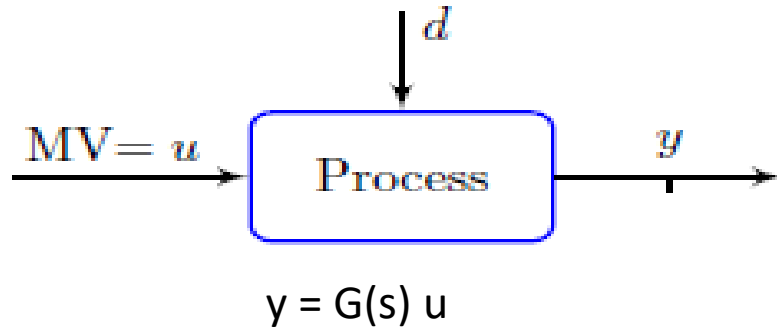
Actual: $c_{FF2} = -1$

«Chicken factor»

Comment: In practice, one often sets the feedforward gain about 80% of the theoretical, that is, $c_{FF2} = -0.8$. This is to avoid that the feedforward controller overreacts, which may confuse the operators. It also makes the feedforward action more robust.

What is best? Feedback or feedforward?

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



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

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

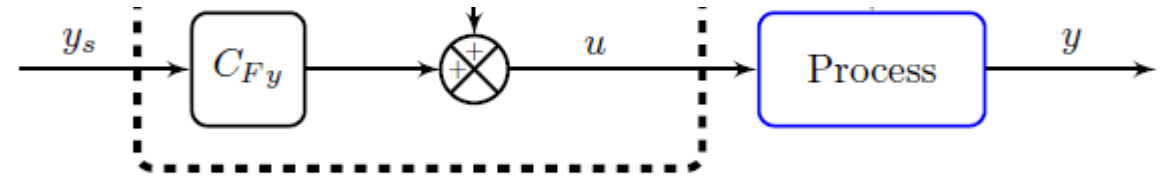


Figure A.42: Block diagram of feedforward control system with linear combination of feedforward from measured disturbance (d) and setpoint (y_s) (E14).

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

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

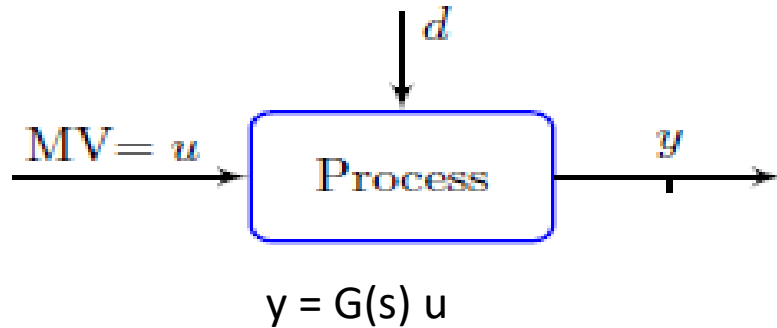
where we choose

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

The output response becomes as desired,

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

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



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

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

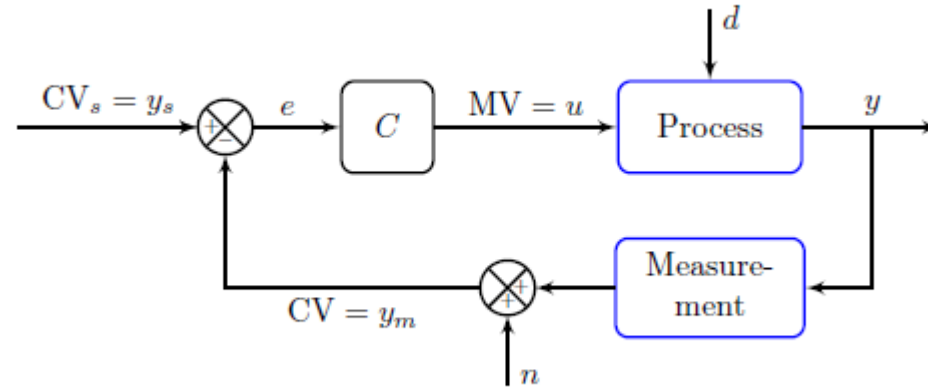


Figure 3: Block diagram of common “one degree-of-freedom” negative feedback control system.

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](#)):

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

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:

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

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 $k'=4.5$

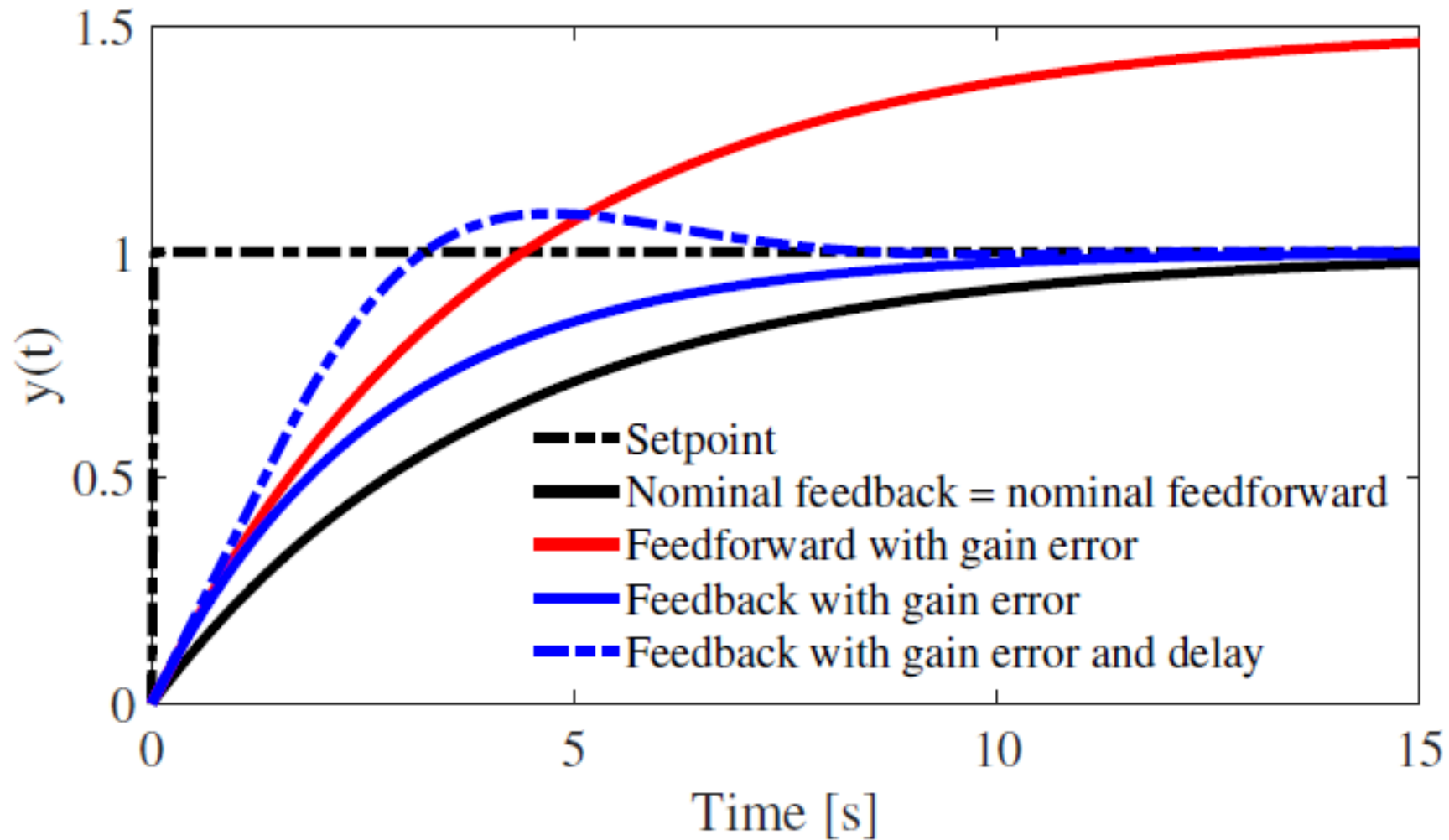


Figure B.43: Setpoint response for process (B.2) demonstrating the advantage of feedback control for handling model error.

Gain error (feedback and feedforward): From $k=3$ to $k'=4.5$
 Time delay (feedback): From $\theta = 0$ to $\theta = 1.5$

Combine: Two degrees-of-freedom control

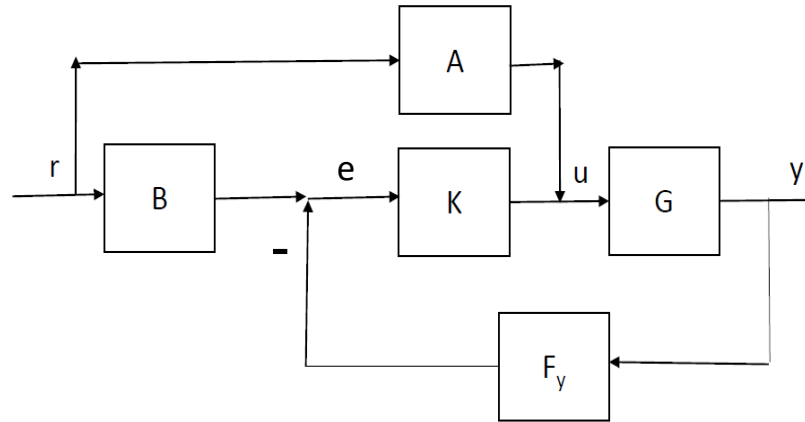


Fig. 3. Two degrees-of-freedom controller with feedforward controller A and prefilter B

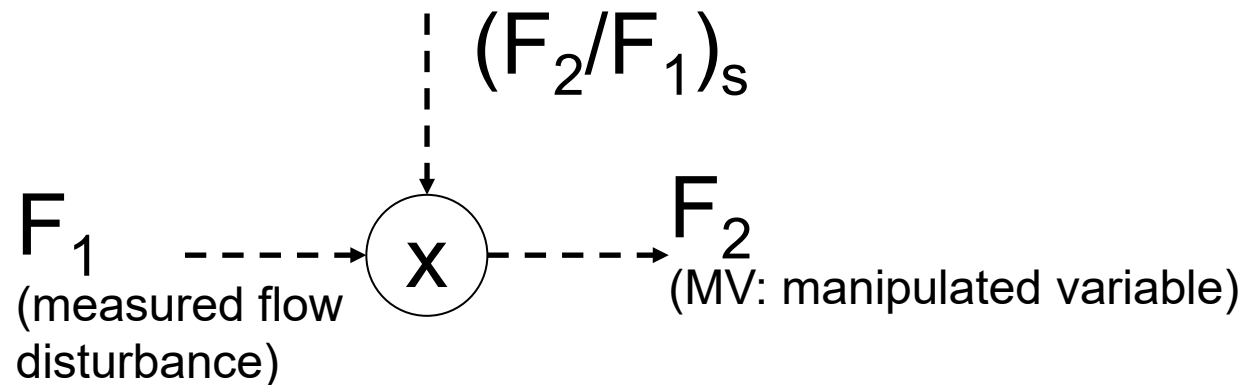
- Typically, the feedforward block is $A = G_-^{-1} F_r$, where G_- is the invertible part of G .
 - A typical choice for the prefilter is $F_r = \frac{1}{\tau_r s + 1}$
- **We want to choose B such that A and K can be designed independently!!**
 - Solution (Lang and Ham, 1955): Choose $B = F_y G A$ so that transfer function from r to e is zero (with perfect model)!
 - The feedback will then only take action if the feedforward is not working as expected (due to model error).
 - We must have $B(0) = I$ so that we will have no offset ($y = r$ at steady state) even with model error for G
- The feedback controller K can be designed for disturbance rejection and robustness, e.g., using SIMC rules.

E2. Ratio control

Special (and most common) case of feedforward

Example: Process with two feeds F_1 (d) and F_2 (u), where ratio should be constant.

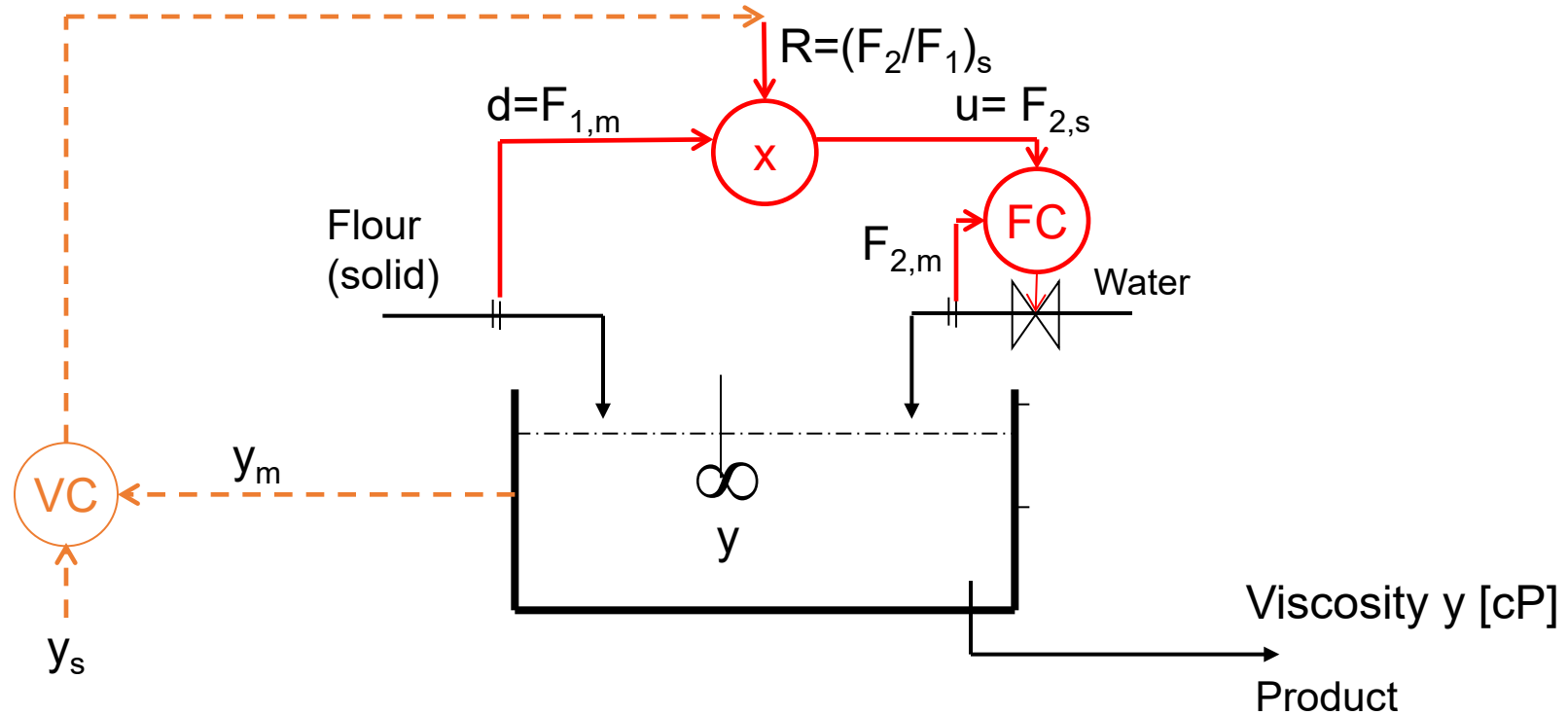
Use multiplication block:



Often $(F_2/F_1)_s$ is adjusted using feedback control in a cascade fashion.

EXAMPLE: CAKE BAKING MIXING PROCESS

RATIO CONTROL with outer feedback trim (to adjust ratio setpoint)



Ratio control

- Avoid divisions in implementation! (avoid divide by 0)
- Process control textbooks has some bad/strange suggestions, for example, division (bad) and “ratio stations” (complex):

Seborg:

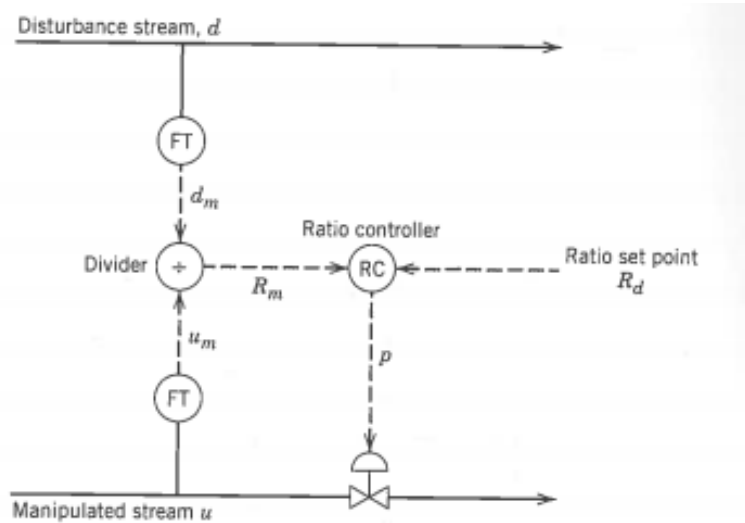


Figure 14.5 Ratio control, Method I.

Bad solution

Avoid divisions (divide by 0 if $u = 0$, for example, at startup)

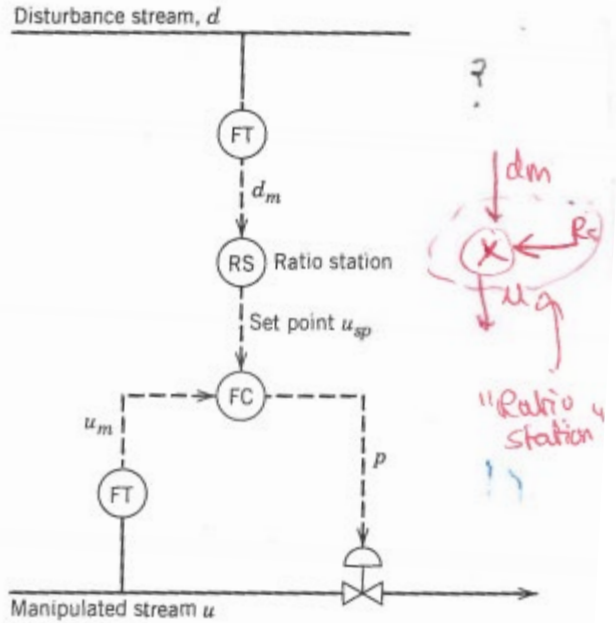


Figure 14.6 Ratio control, Method II.

This is complicated. What is RS?
 Ok if implemented as shown in red at right

Theory of Ratio control

- Assumes that «scaling property» holds (physical insight): Keep ratio R (between extensive variables) constant in order to keep property y constant
 - y : (any!) intensive variable
- Implementation
 - Feedforward: $R=u/d$
 - Decoupling: $R=u_1/u_2$
 - u,d : extensive variables
 - Setpoint for R may be found by «feedback trim»
- Don't really need a model (no inverse as in «normal» feedforward!)
- Scaling property holds for mixing and equilibrium processes
 - Ratio control is almost always used for mixing of reactants
 - Requires that all extensive variables are scaled by same amount
 - So does not hold for heat exchanger (since area A is constant) or non-equilibrium reactor (since volume V is constant)
 - So should not keep L/F constant for distillation column with saturated (max) heat input (V)

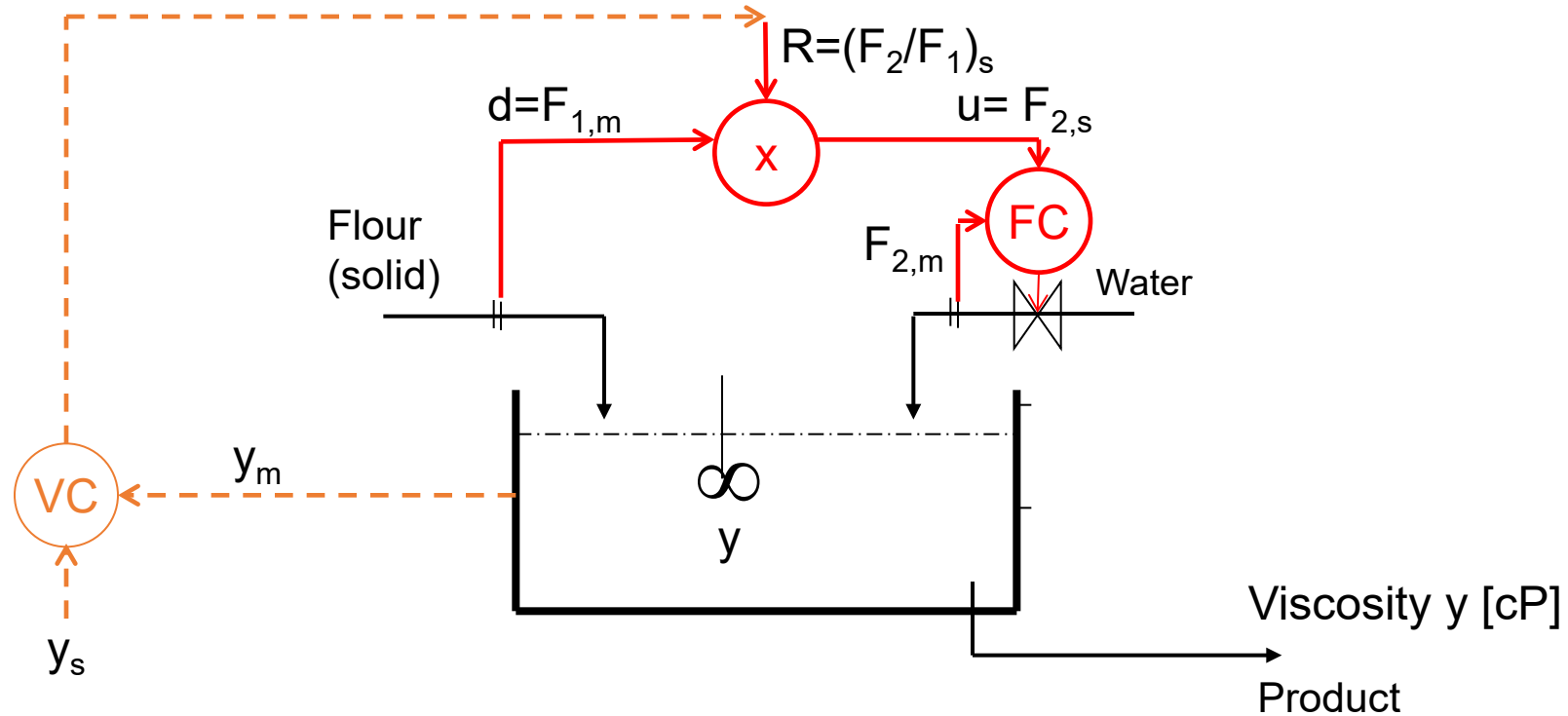
From paper by Skogestad (2023)

3.3.3. Theoretical basis for ratio control

Ratio control is most likely the oldest control approach (think of recipes for making food), but despite this, no theoretical basis for ratio control has been available until recently (Skogestad, 2023). Importantly, with ratio control, the controlled variable y is implicitly assumed to be an *intensive variable*, for example, a property variable like composition, density or viscosity, but it could also be temperature or pressure. On the other hand, the two variables included in the ratio R are implicitly assumed to be *extensive variables*.

Ratio control is more powerful than most people think, because its application only depends on a “scaling assumption” and does not require an explicit model for y . For a mixing process, the “scaling property” or “scaling assumption” says that if all extensive variables (flows) are increased proportionally (with a fixed ratio), then at steady state all mixture intensive variables y will remain constant (Skogestad, 1991). The scaling property (and thus the use of ratio control) applies to many process units, including mixers, equilibrium reactors, equilibrium flash and equilibrium distillation.

Implementation of ratio control: $u = R d$ (in unscaled/physical units)



LINEARITY OF RATIO CONTROL: This way of implementing ratio control makes it easy to tune the outer feedback loop (VC controller) because

- The gain from $MV = R_s$ to $CV=y$ (here viscosity) does not depend on the disturbance $d=F_1$.

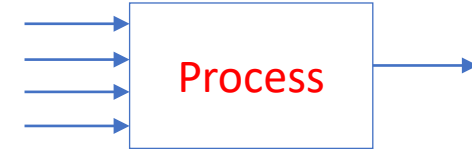
Proof: This is general for this way of implanting ratio control, because a given change in R has the same effect on the property y , independent of the total flow is.

Proof of constant gain for ideal mixing

- **Ideal mixing:** The property y depends linearly on composition c , $y = k_1 c_1 + k_2 c_2$, so we just need to show that the gain from $MV=R$ to composition c is independent of the total flow.
- **Proof.** The component balance gives: $CV=c=(c_1 F_1 + c_2 F_2)/(F_1+F_2)$
- We are here considering disturbances in F_1 , so assume that c_1 and c_2 are constant.
- We also assume that there is an outer loop so that c remains constant. From the component balance we see that $c=\text{constant}$ implies that as we change F_1 (disturbance) we will have that $R=F_2/F_1=\text{constant}$.
- **With no ratio control:** The gain from $MV=F_2$ to $CV=c$ is:
 - $K = (c_2-c_1)F_1/(F_1+F_2)^2 = (c_2-c_1)/[(1+R)(F_1+F_2)]$
 - With $R=\text{constant}$ (at steady state) we then have $K = \text{constant}/(F_1+F_2)$ so the gain K will change with operation, which will be a problem for the outer feedback controller (CC). Actually, we find that $K=\text{infinity}$ when $F=F_1+F_2$ goes to zero, so we may get instability in the outer feedback loop at low flowrates.
- **With ratio control:** The gain from $MV=R_s=(q_2/q_1)_s$ to $CV=c$ is:
 - $K_r = (c_2-c_1)F_1^2/(F_1+F_2)^2 = (c_2-c_1)/(1+R)^2$
 - With $R=\text{constant}$ (at steady state) we get $K_r = \text{constant}$ independent of the value of the disturbance (F_1)! So the outer loop always has the same gain and there no reason to be careful about the tunings.
- **With notmalized ratio control:** The gain from $MV=R_{s2}=(q_2/q_1+q_2)_s$ to $CV=c$ is:
 - $K_r = ??$ (remains to be done)
 - But it should be constant (so linear) with R_{s2} constant (according to the theory of transformed inputs, 2023)
- **Note:** An alternative to ratio control is “standard” feedforward control where $u = u_{FB} + u_{FF}$ (where FB is from the feedback controller CC and FF is from a feedforward controller from $d=F_1$.) In this case we get the problem with process gain variation for the feedback controller CC. So ratio control is the best!

Valve position control (VPC)

Have extra MV (input): One CV, many MVs



Two different cases of VPC:

- **E3.** Have extra dynamic MV
 - Both MVs are used all the time
- **E7.** Have extra static MV
 - MV-MV switching: Need several MVs to cover whole range at steady state
 - We want to use one MV at a time

E3. VPC for extra dynamic input

- u_2 = main input for steady-state control of CV (but u_2 is poor for directly controlling y
 - e.g. time delay or u_2 is on/off)
- u_1 = extra dynamic input for fast control of y



3.4. Input (valve) position control (VPC) to improve the dynamic response (E3)

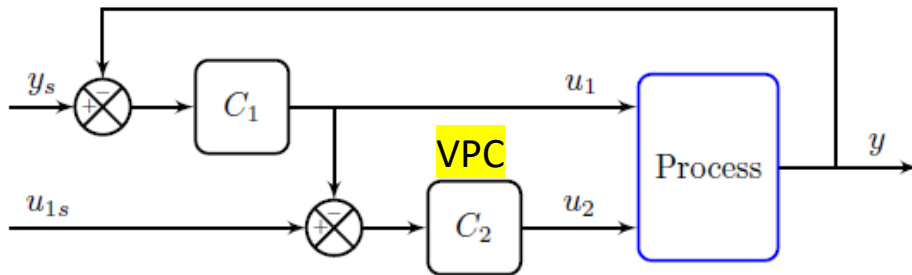


Figure 12: Valve (input) position control (VPC) for the case when an “extra” MV (u_1) is used to improve the dynamic response. A typical example is when u_1 is a small fast valve and u_2 is a large slower valve.

C_1 = fast controller for y using u_1 .

C_2 = slow valve position controller for u_1 using u_2 (always operating).

u_{1s} = steady-state resting value for u_1 (typically in mid range. e.g. 50%).

Example 1: Large (u_2) and small valve (u_1) (in parallel) for controlling total flowrate ($y=F$)

- The large valve (u_2) has a lot of stiction which gives oscillations if used alone for flow control. It could also be an on/off valve or pump (or even several).
- The small valve (u_1) has less stiction and gives good flow control, but it's too small to use alone

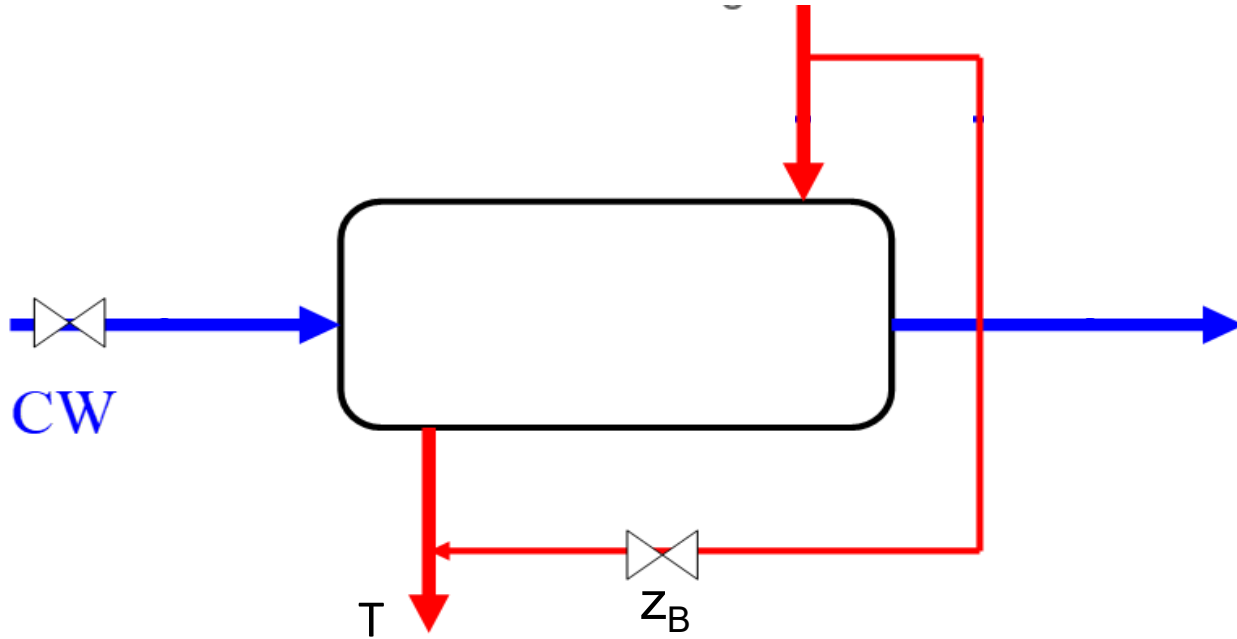
Example 2: Strong base (u_2) and weak base (u_1) for neutralizing acid (disturbance) to control $y=pH$

- Do pH change gradually (in two tanks) with the strong base (u_2) in the first tank and the weak base (u_1) in the last tank. u_1 controls the pH in the last tank (y)

Alternative term for dynamic VPC:

- Mid-ranging control (Sweden)

Example 3: Heat exchanger with bypass

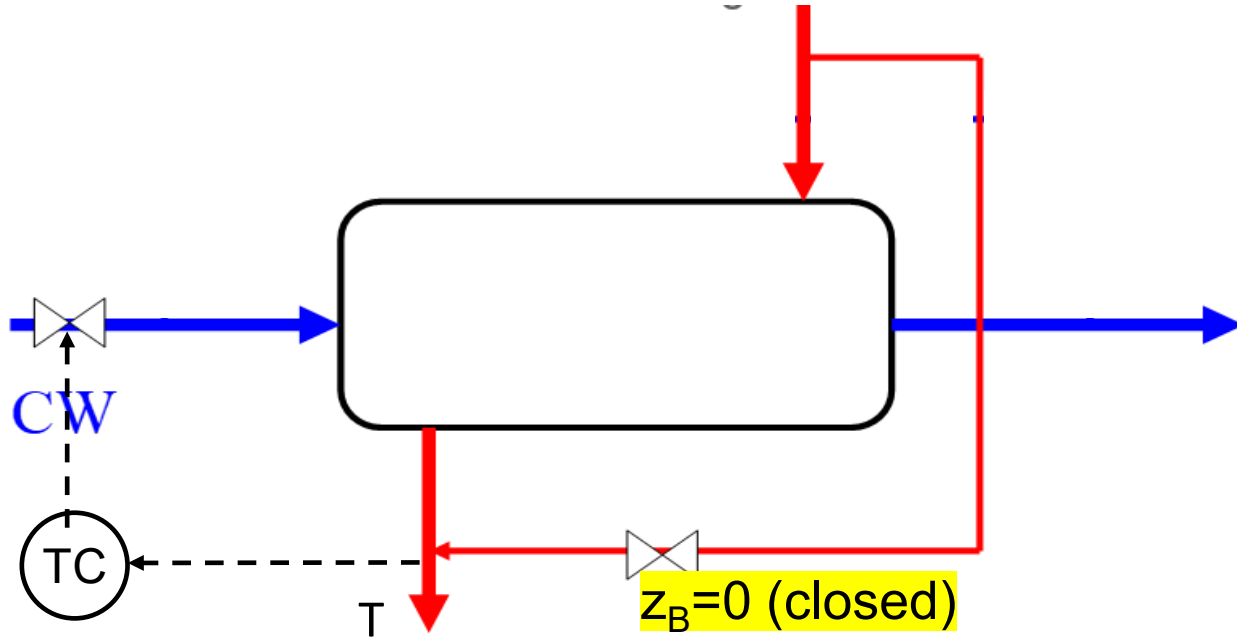


Want tight control of $y=T$.

- $u_1=z_B$ (bypass)
- $u_2=CW$

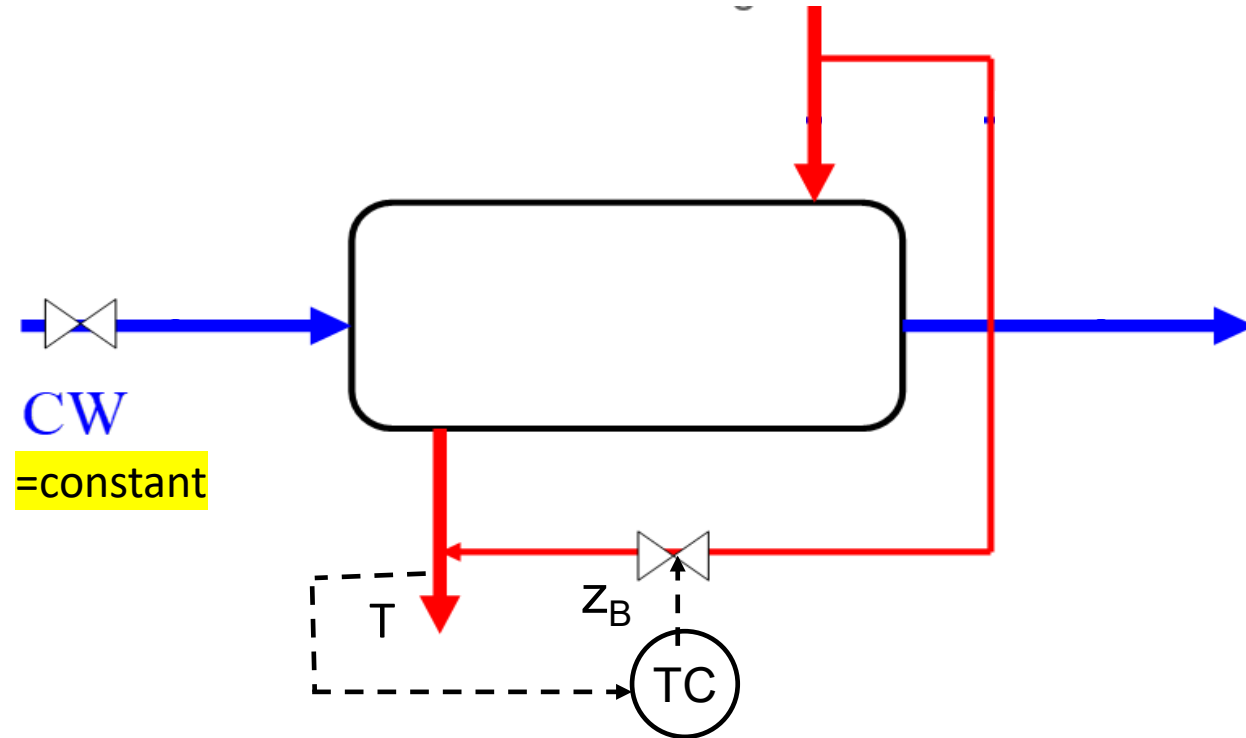
Proposed control structure?

Attempt 1. Use u_2 =cooling water: TOO SLOW



Attempt 2. Use $u_1 = z_B = \text{bypass}$. SATURATES

(at $z_B = 0 = \text{closed}$ if CW too small)

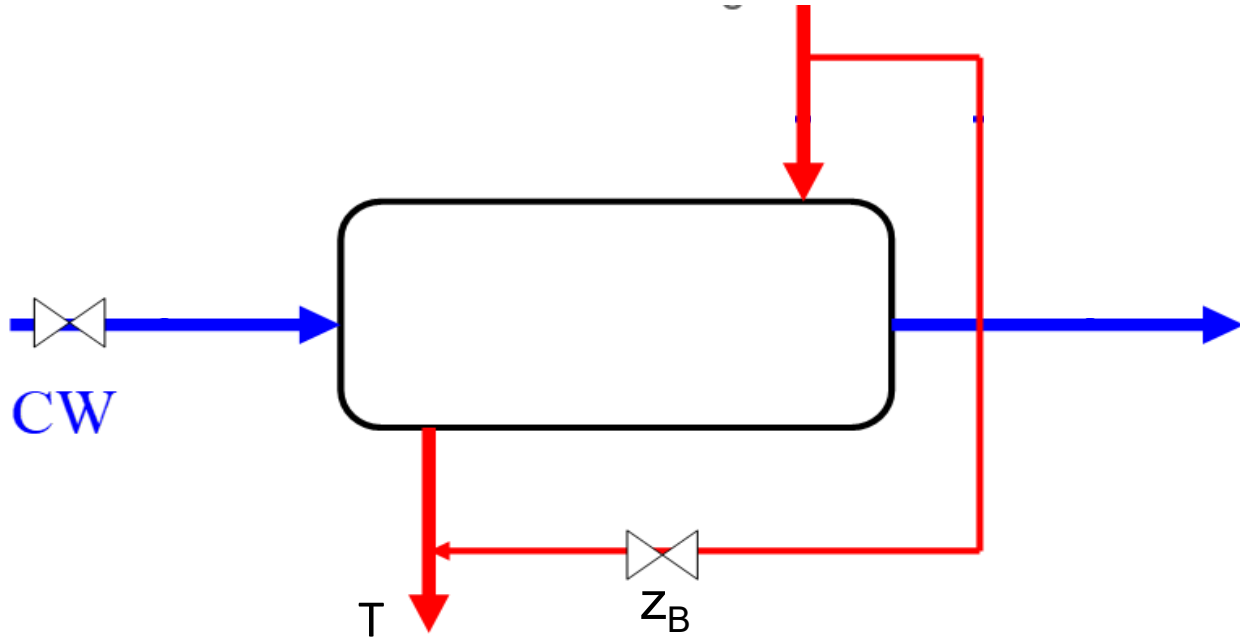


Advantage: Very fast response (no delay)

Problem: z_B is too small to cover whole range

+ not optimal to fix at large bypass (waste of CW)

What about VPC?

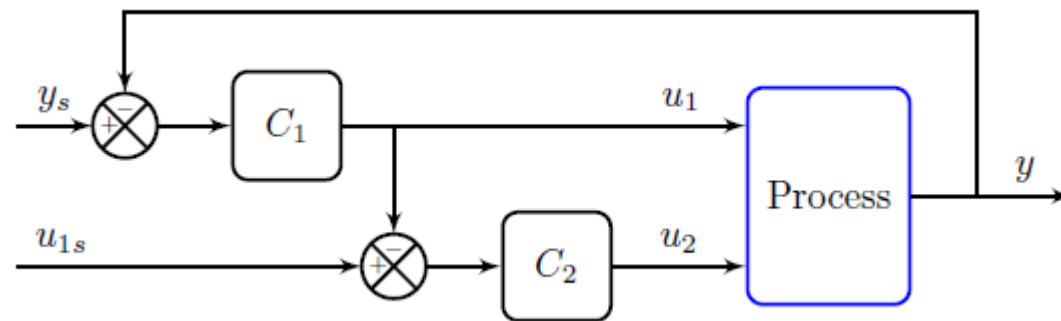


Want tight control of $y=T$.

- $u_1 = Z_B$
- $u_2 = CW$

Proposed control structure?

- Main control: $u_2 = CW$
- Fast control: $u_1 = Z_B$



Comment on heat exchanger example

- The above example assumes that the flows on the two sides are «**balanced**» (mc_p for cooling water (CW) and hot flow (H) are not too different) such that both the bypass flow (u_1) and CW flow (u_2) have an effect on T (CV)
- There are two «**unbalanced**» cases, which us when we have «pinch» in the heat exchanger ends:
 - If CW flow is small, then T_{outCW} will always approach T_{inH} , so from a total energy balance, the bypass will have almost zero *steady-state* effect on T.
 - If CW flow is large, then T_{outH} (before bypass mixing point) will always approach T_{inCW} , so CW will have almost zero effect on T (*both* steady state and dynamically)
- This illustrates that heat exchanger may behave very nonlinearly, and a good control structure for one heat exchanger case, may not work well for another case

Alternative to VPC: Parallel control

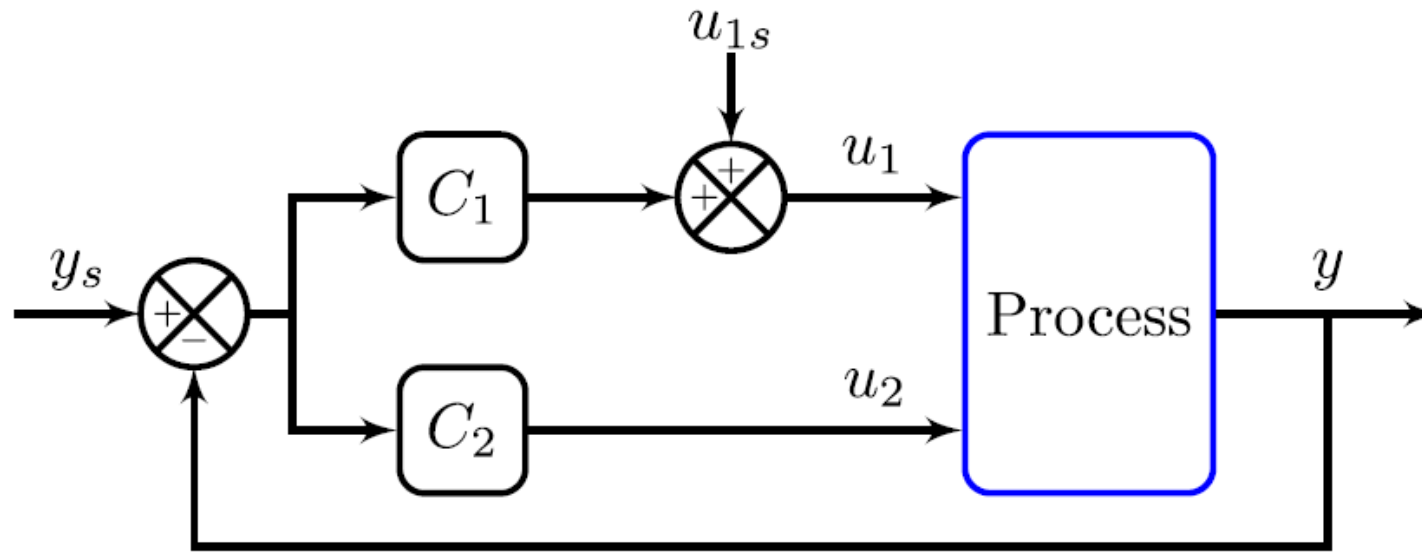


Fig. 13. Parallel control to improve dynamic response – as an alternative to the VPC solution in Fig. 12.

The “extra” MV (u_1) is used to improve the dynamic response, but at steady-state it is reset to u_{1s} . The loop with C_2 has more integral action and wins a steady state.

The advantage with valve position control compared to parallel control is that the two controllers in Figure 12 can be tuned independently (but C_1 must be tuned first) and that both controllers can have integral action. On the other hand, with some tuning effort, it may be easier to get good control performance for y with parallel control.

VPC with one MV: Stabilizing control with resetting of MV

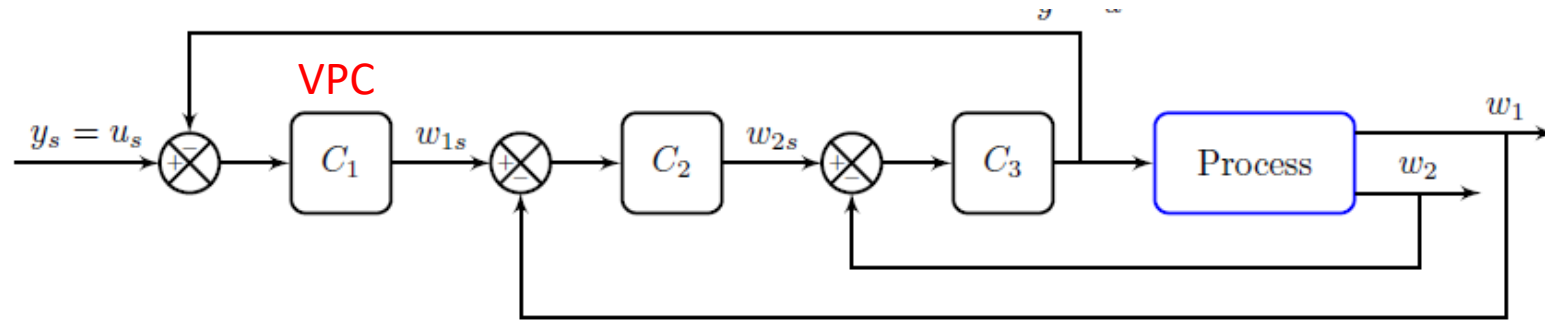
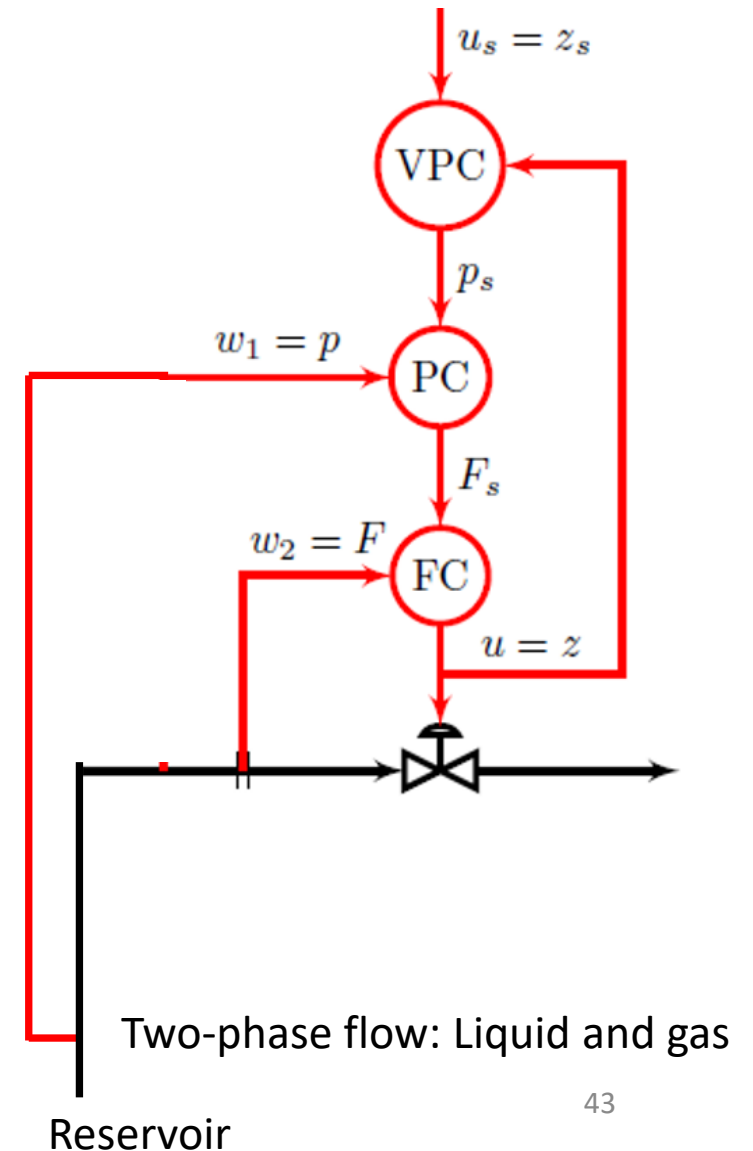


Figure 14: Stabilizing control of variable w_1 combined with valve position control (VPC) for u (=valve position) and inner flow controller ($w_2 = F$). It corresponds to the flowsheet in Figure 15 with $w_1 = p$ (pressure), $C_1 =$ outer VPC (slow), $C_2 =$ stabilizing controller (fast), $C_3 =$ inner flow controller (very fast). Note that the process variables (w_1, w_2) have no fixed setpoint, so they are “floating”.

Note: u is both an MV and a CV



Example: Anti-slug control

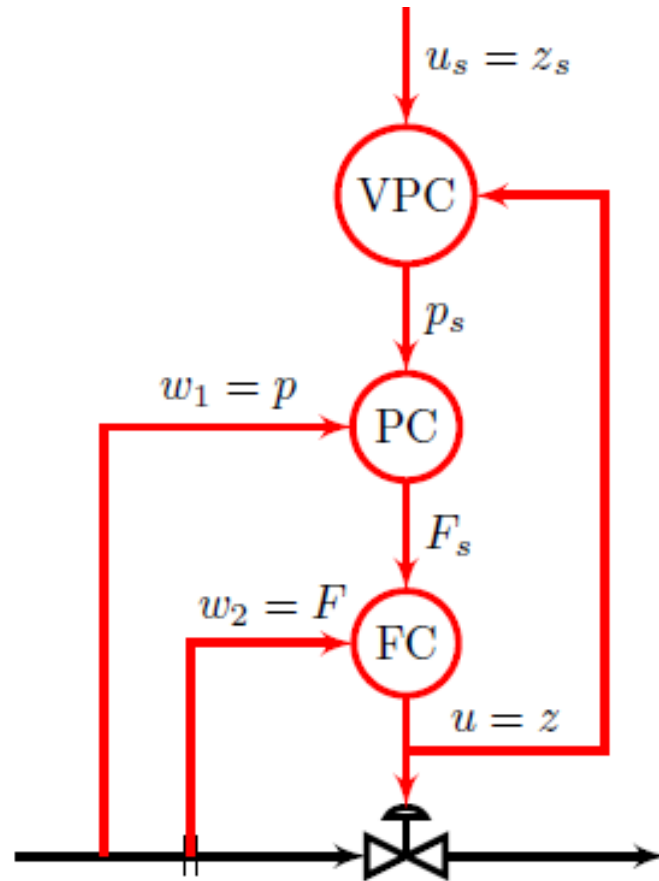


Figure 15: Anti-slug control where the pressure controller (PC) is used to stabilize a desired non-slugging flow regime. The inner flow controller (FC) (fast) provides linearization and disturbance rejection. The outer valve position controller (VPC) (slow) resets the valve position to its desired steady-state setpoint ($u_s = z_s$). It corresponds to the block diagram in Figure 14.

VPC to reset input u

- If the underlying process is unstable, then the instability will result in an inverse response when attempting to reset u .
- Proof: $G(s)$ has unstable pole at $s=p$.
 - Then transfer function KS from u_d (at input) to u has unstable zero at $s=p$.
 - This is because $G(p)=\text{infinity}$ so $S(p)=(I+G(p)K(p))^{-1} = 0$.

Example: Stabilize bicycle

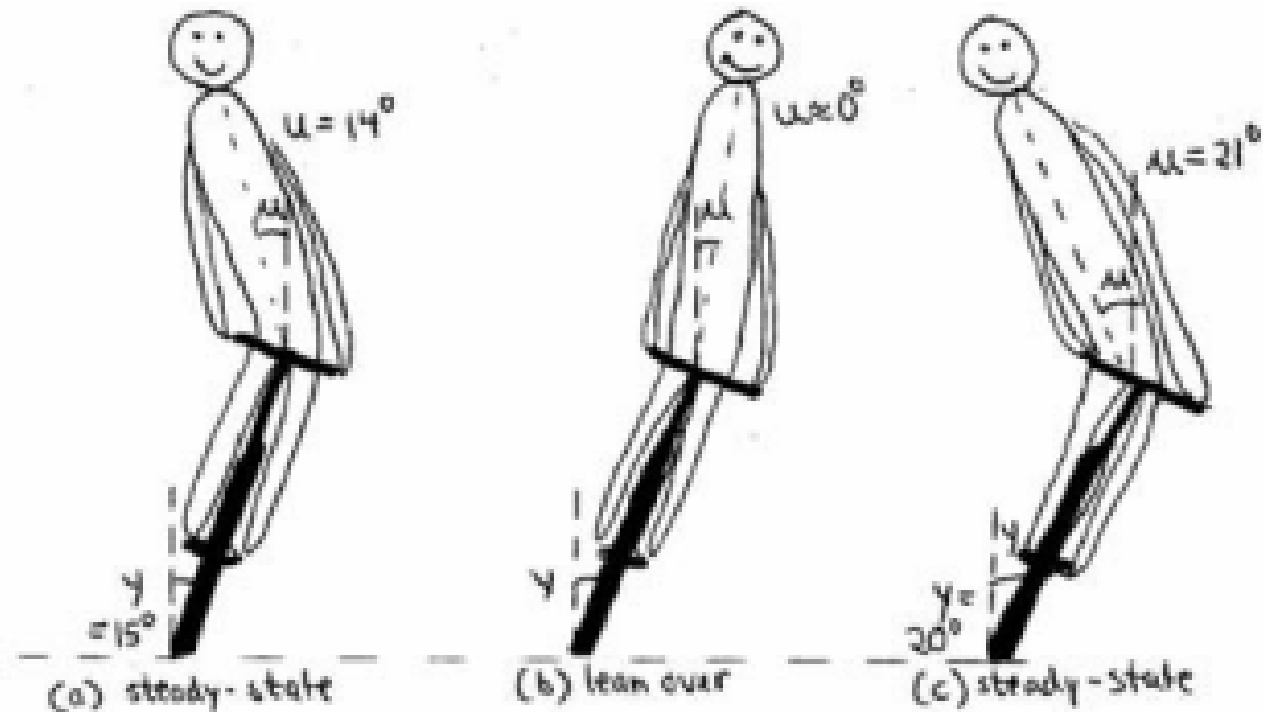


Fig. 2. Inverse response for a bicycle caused by an underlying instability

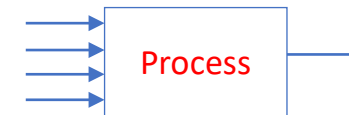
Consider Figure 2 where the aim is to tilt the bike from an initial angle $y = 15^\circ$ (Fig. 2a) using your body (u) to an angle $y = 20^\circ$ (Fig. 2c). Because of the inverse response, you first have to tilt your body in the direction of the tilt to start the movement (Fig. 2b). Eventually, you will have to move your body back to restore balance. This inverse response will be slower the greater the angle y , changing the angle while keeping balanced gets progressively slower as the tilting angle is increased.

Comment: Another example is a motorcycle where tilting is required for making turns.

Constraint switching

(because it is optimal at steady state)

- CV-CV switching
 - Control one CV at a time
- MV-MV switching
 - Use one MV at a time
- MV-CV switching
 - MV saturates so must give up CV
 - Two alternatives:
 - Simple («do nothing»). If we followed input saturation rule
 - Complex (repairing of loops). Need to combine MV-MV and CV-CV switching



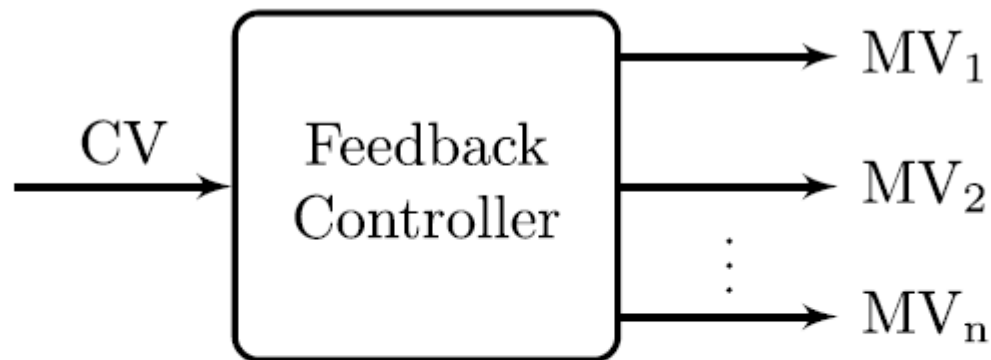


Fig. 5. MV-MV switching is used when we have multiple MVs to control one CV, but only one MV should be used at a time. The block “feedback controller” usually consists of several elements, for example, a controller and a split range block.

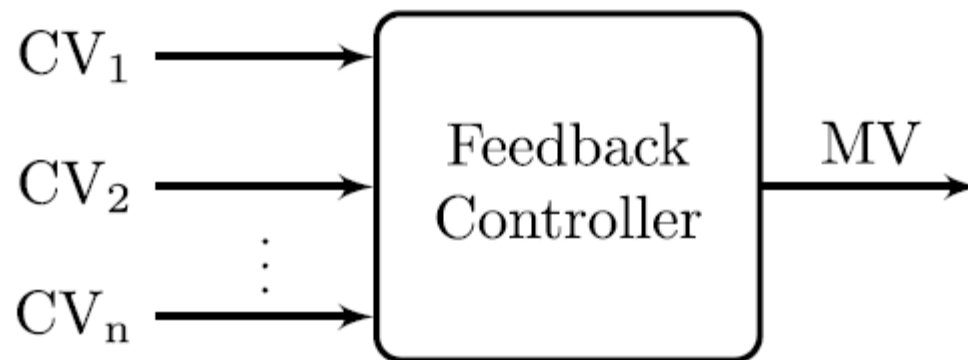
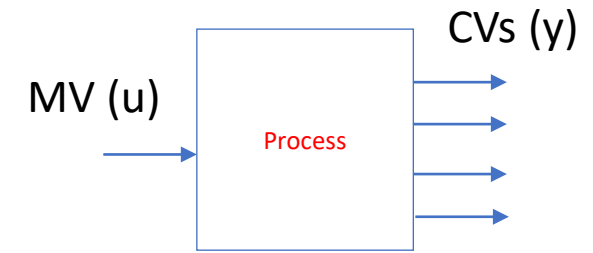


Fig. 6. CV-CV switching is used when we have one MV to control multiple CVs, but the MV should control only one CV at a time. The block “feedback controller” usually consists of several elements, typically several PID-controllers and a selector.

CV-CV switching

- Always use selector

CV-CV switching



- Only one input (MV) controls many outputs (CVs)
 - Typically caused by change in active constraint
 - Example: Control car speed (y_1) - but give up if too small distance (y_2) to car in front.
- Always use selector

Example adaptive cruise control: CV-CV switch followed by MV-MV switch

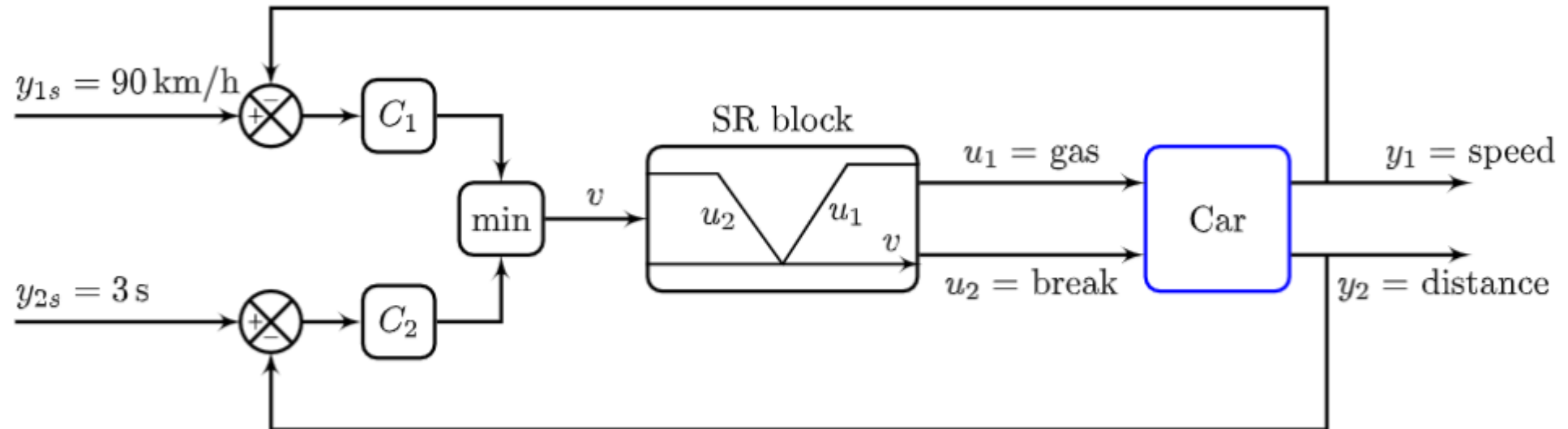
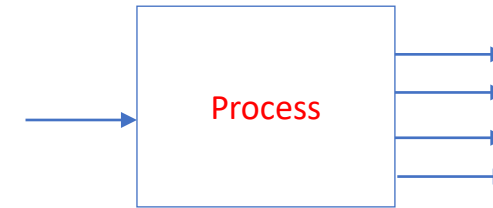


Fig. 31. Adaptive cruise control with selector and split range control.

Note: This is not Complex MV-CV switching, because then the order would be opposite.

E4. Selector (for CV-CV switching*)

- Many CVs paired with one MV.
- But only one CV controlled at a time.
- Use: Max or Min selector



Note: Selectors are logic blocks



- Sometimes called “override”
 - But this term may be misleading, because the switching is usually the optimal thing to do**
- Selector is generally on MV (compare output from many controllers)

*Only option for CV-CV switching. Well, not quite true: Selectors may be implemented in other ways, for example, using «if-then»-logic.

** I prefer to use the term «override» for undesirable temporary (dynamic) switches, for example, to avoid overflowing a tank dynamically. Otherwise, it's CV-CV switching

Implementation selector



Alt. I (General). Several controllers (different CVs)

- Selector on MVs*
- Must have anti windup for C_1 and C_2 !

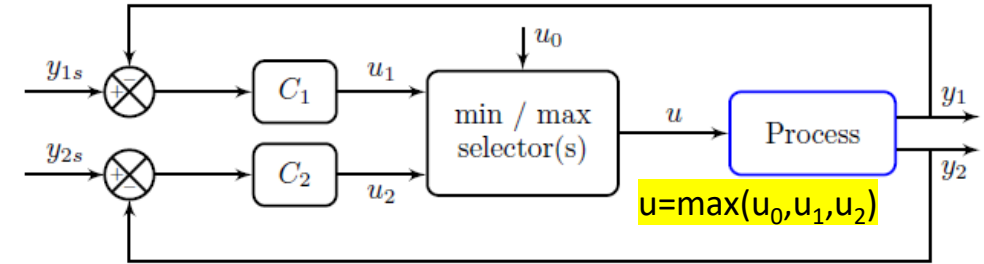


Figure 17: CV-CV switching with selector on MV (input u).

Alt. II (Less general) Controllers in cascade

- Selector on CV setpoint
- Good alternative if CVs (y_1 and y_2) are related so that cascade is good
- In this case: Selector may be replaced by saturation element (with y_{2s} as the max or min)

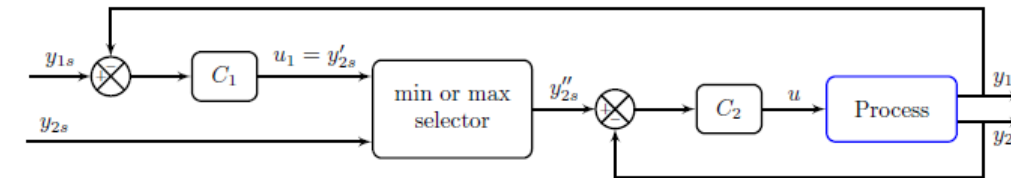
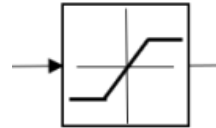
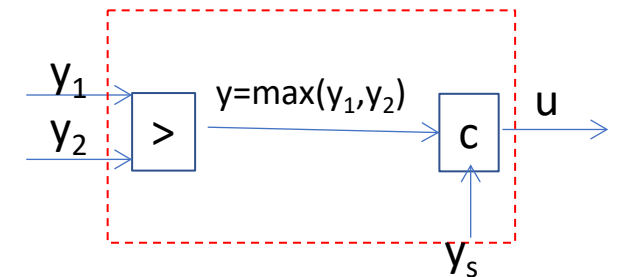


Figure 19: Alternative cascade CV-CV switching implementation with selector on the setpoint. In many cases, u_{1s} and u_{2s} are constraint limits.

Alt. III (For special case where all CVs have same bound). One controller

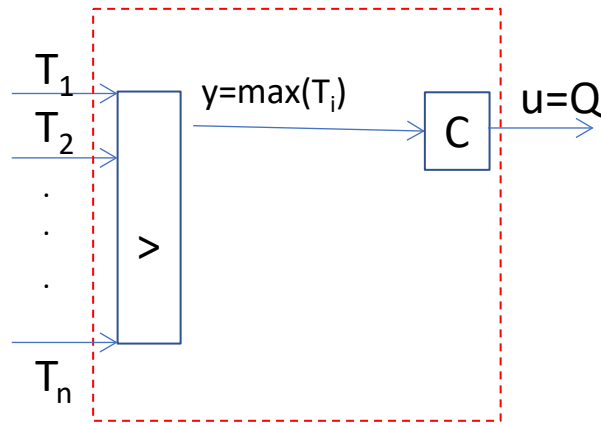
- Selector is on CVs (Auctioneering)
- Also assumes that dynamics from u to y_1 and y_2 are similar; otherwise use Alt.I
- Example: Control hot-spot in reactor or furnace.



*It may seem surprising that the selection is on the MV for a CV-CV switch, but this turns out to be the most general and most effective.

Example Alt. III

- Hot-spot control in reactor or furnace



- Comment: Could use General Alternative I (many controllers) for hot-spot control, with each temperature controller (c_1, c_2, \dots) computing the heat input ($u_1 = Q_1, u_2 = Q_2, \dots$) and then select $u = \min(u_1, u_2, \dots)$, but it is more complicated.

Furnace control with safety constraint (Alt. I)

Input (MV)

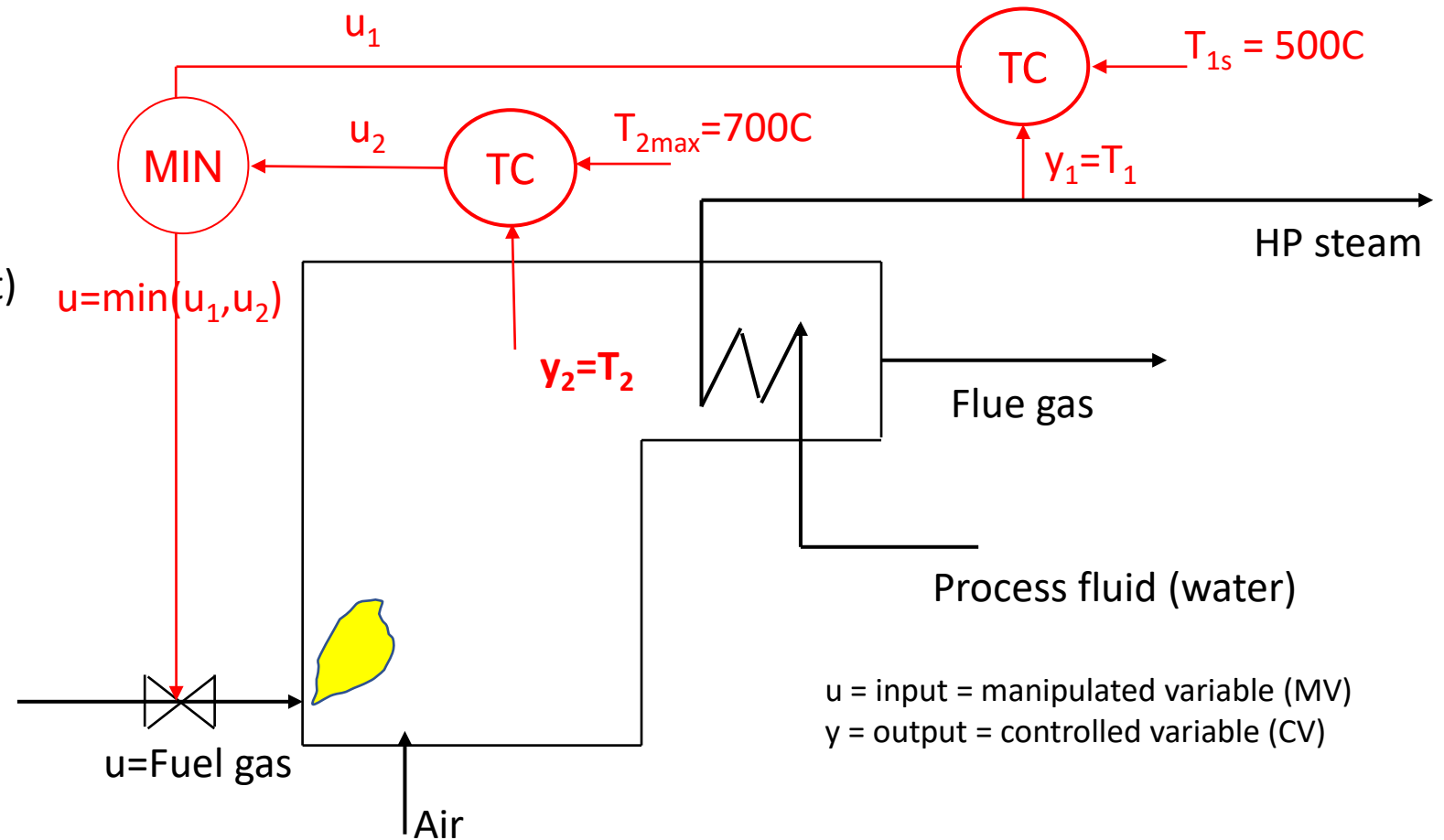
u = Fuel gas flowrate

Output (CV)

y_1 = process temperature T_1
(desired setpoint or max constraint)

y_2 = furnace temperature T_2
(max constraint)

Rule: Use *min-selector* for constraints that are satisfied with a small input

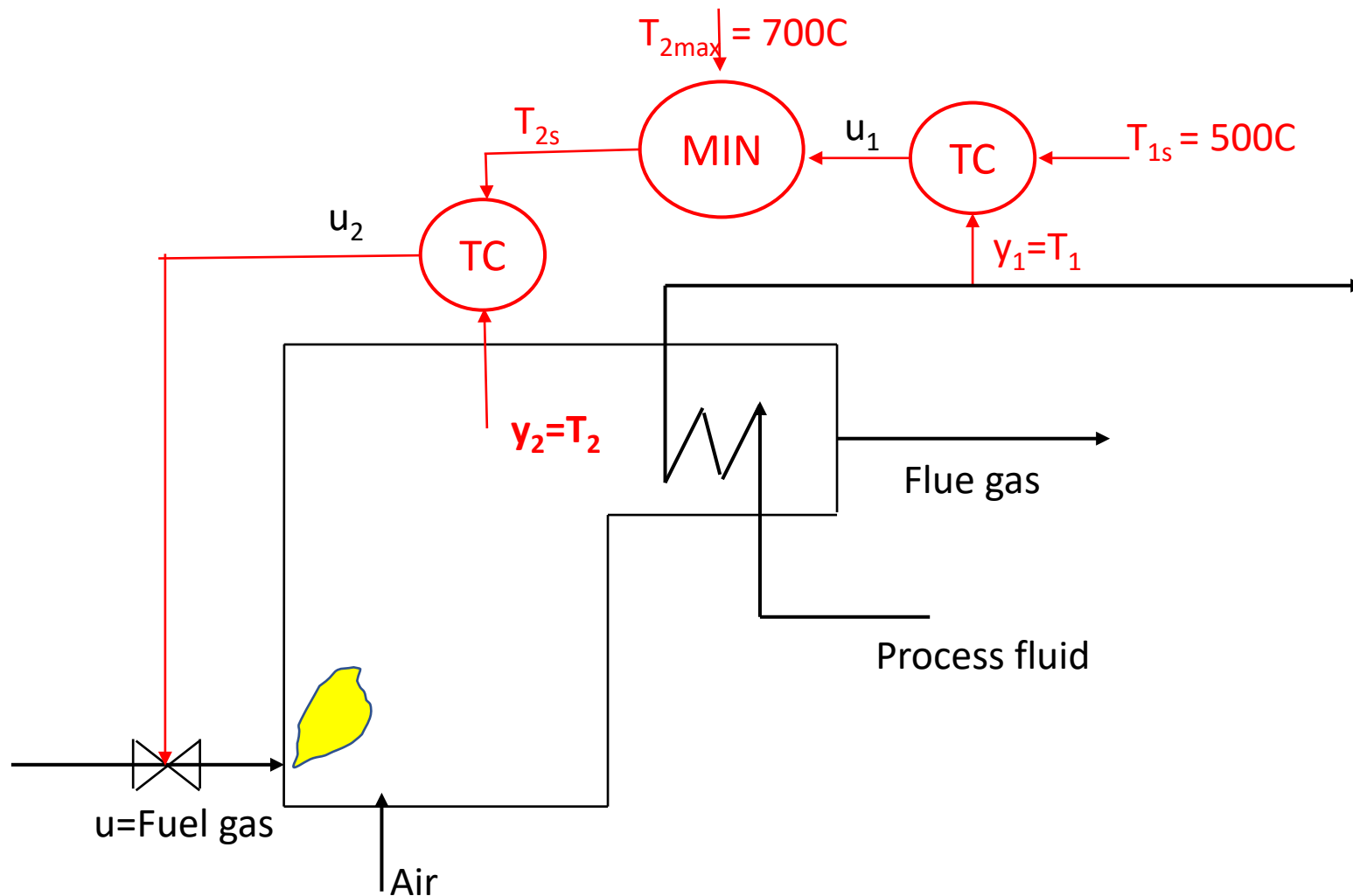


Furnace control with cascade (Alt. II, selector on CV-sp)

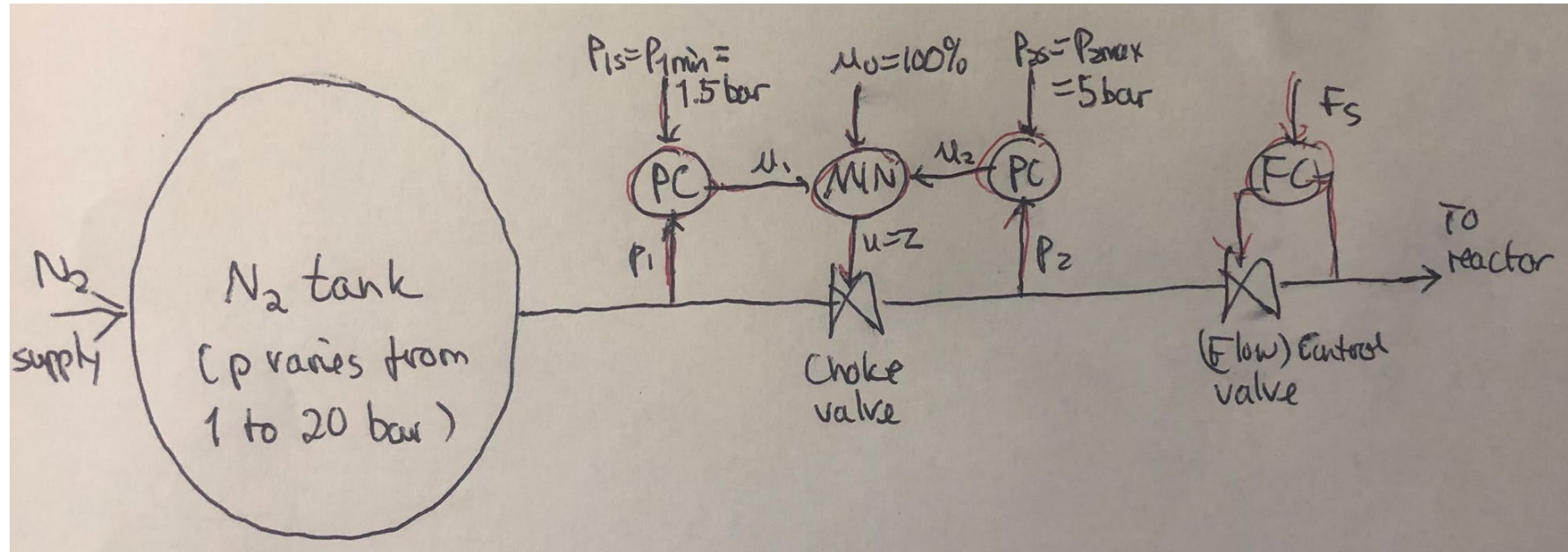
Comparison

The cascade solution is less general but it may be better in this case.

Why better? Inner T2-loop is fast and always active and may improve control of T1.



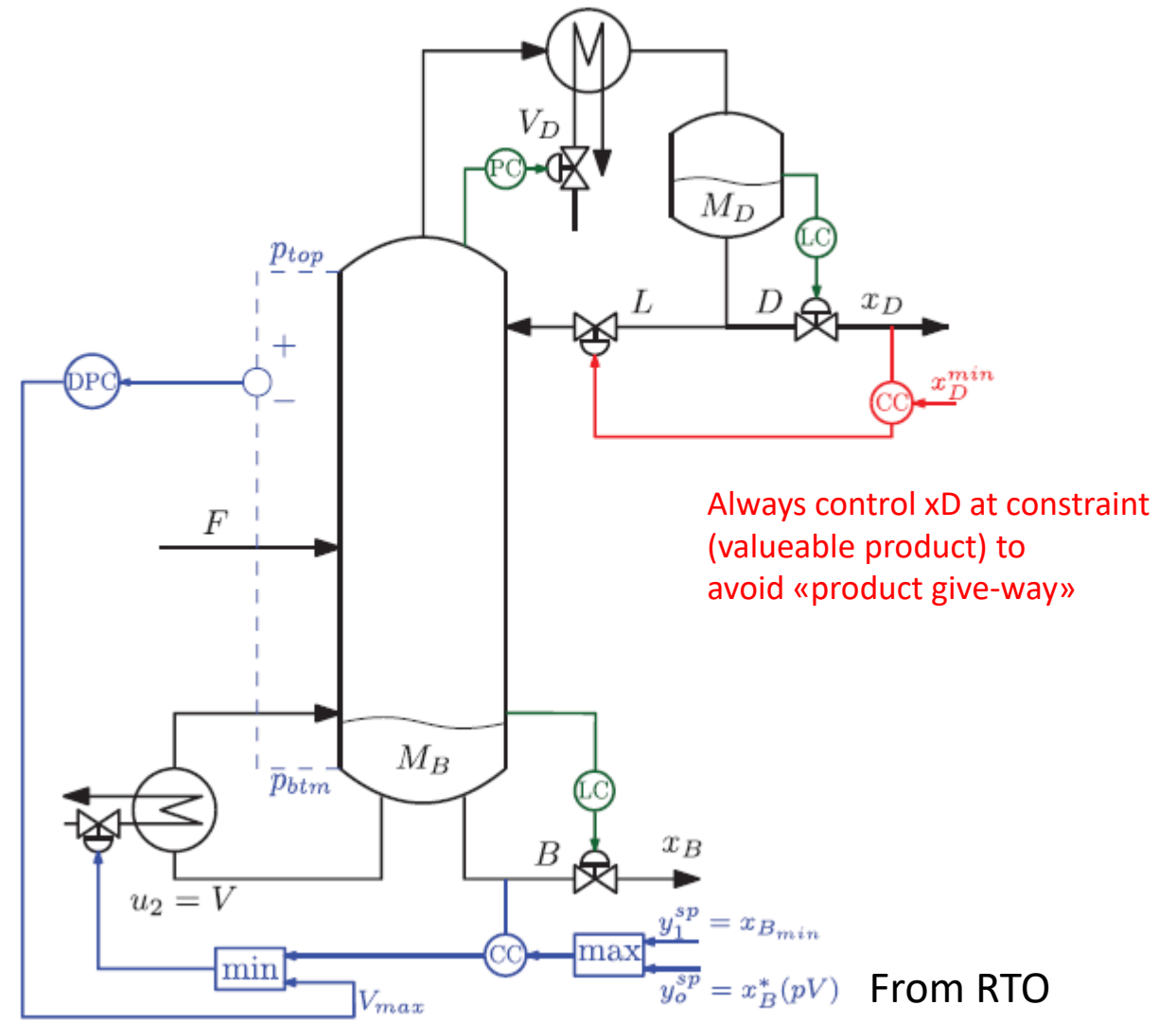
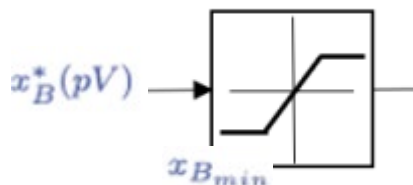
Example Alt. I (Choke valve)



- The choke valve is **normally** fully open to avoid unnecessary pressure drop (**normally**, we may have $p_1=4.5 \text{ bar}$, $p_2=4 \text{ bar}$)
- But may close choke valve to avoid too low p_1 (if N₂ supply stops) or too high p_2 (if p_1 is high).
- **Here Alt. II (cascade) should not be used** (Why? Cascade control makes it necessary to have one loop slow, which makes little sense since control of p_1 does not improve control of p_2 (or vice-versa).
- Comment 1: Strictly speaking, the input $u_0=100\%$ to the min-block is not needed, since the valve has a “built-in” min-selector at fully open (see later). But including it is not wrong - and it shows more clearly that we **normally** want $u_0=100\%$.
- Comment 2: There may be quite a lot of interaction between the pressure and flow control. The best solution is probably to make one fast and one slow. In this case, it seems most reasonable to make the pressure control fast. But if fast flow control is needed, we may do it opposite.
- Comment 3: I here use the term “choke valve” when the aim is to regulate pressure; and “control valve” when it is to regulate flow (but other people may not agree on this)

Distillation example

Note: A selector where one input is a constant (like the max-block for x_B) may be replaced by a saturation element



Always control x_D at constraint (valuable product) to avoid «product give-away»

Avoid flooding using constraint on DP (Alt. I: selector in input)

May overpurity bottom to get more of the valuable product (Alt. II: Selector on setpoint)

From RTO

Design of selector structure

Rule 1 (max or min selector)

- Use max-selector for constraints that are satisfied with a large input
- Use min-selector for constraints that are satisfied with a small input

Rule 2 (order of max and min selectors):

- If need both max and min selector: Potential infeasibility (conflict)
- Order does not matter if problem is feasible
- If infeasible: Put highest priority constraint at the end

“Systematic design of active constraint switching using selectors.” Dinesh Krishnamoorthy , Sigurd Skogestad. [Computers & Chemical Engineering, Volume 143](#), (2020)

“Advanced control using decomposition and simple elements”. Sigurd Skogestad. Annual Reviews in Control, Volume 56, 100903 (2023)

Rule 2 (order of selectors)

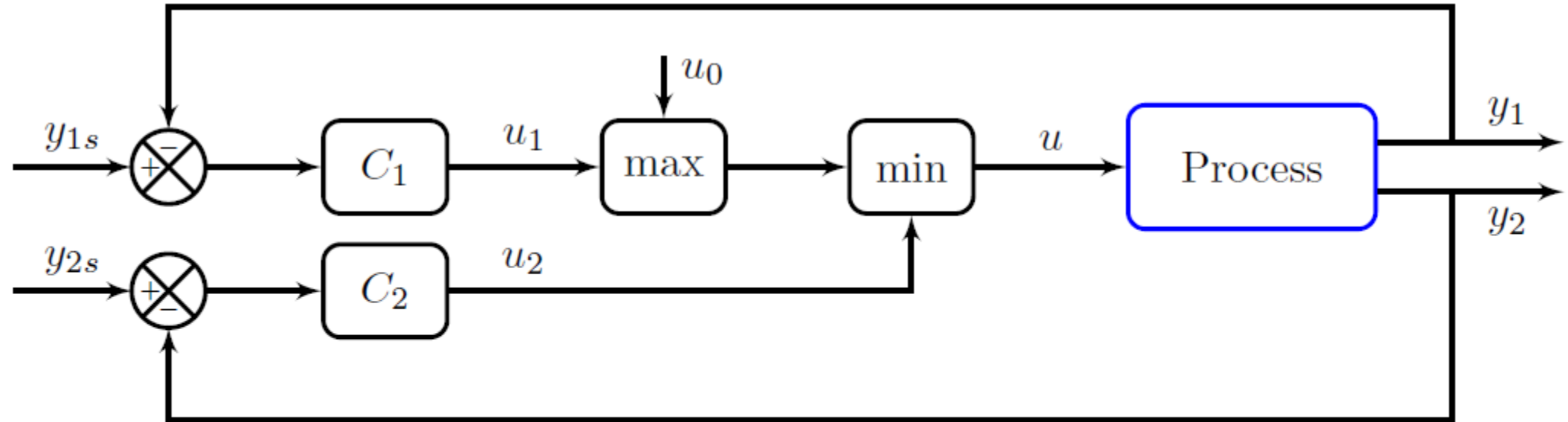


Figure 18: CV-CV switching for case with possibly conflicting constraints. In this case, constraint y_{1s} requires a max-selector and y_{2s} requires a min-selector. The selector block corresponding to the most important constraint (here y_{2s}) should be at the end (Rule 2).

To understand the logic with selectors in series, start reading from the first selector. In this case, this is the max-selector: The constraint on y_1 is satisfied by a large value for u which requires a max-selector (Rule 1). u_0 is the desired input for cases when no constraints are encountered, but if y_1 reaches its constraint y_{1s} , then one gives up u_0 . Next comes the min-selector: The constraint on y_2 is satisfied by a small value for u which requires a min-selector (Rule 1). If y_2 reaches its constraint y_{2s} , then one gives up controlling all previous variables (u_0 and y_1) since this selector is at the end (Rule 2). However, note that there is also a "hidden" max- and min- selector (Rule 3) at the end because of the possible saturation of u , so if the MV (input) saturates, then all variables (u_0, y_1, y_2) will be given up.

Example. Maximize flow with pressure constraints

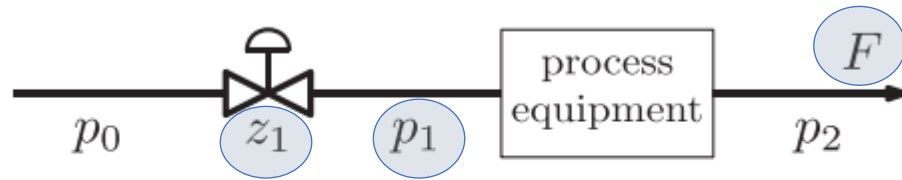


Fig. 6. Example 2: Flow through a pipe with one MV ($u = z_1$).

Input $u = z_1$

Want to maximize flow, $J = -F$:

Optimization problem is:

$$\max_{z_1} F$$

s.t.

$$\begin{aligned} F &\leq F_{max} \\ p_1 &\leq p_{1,max} \\ p_1 &\geq p_{1,min} \\ z_1 &\leq z_{1,max} \end{aligned}$$

Satisfied by

Small u

Small u

Large u

-

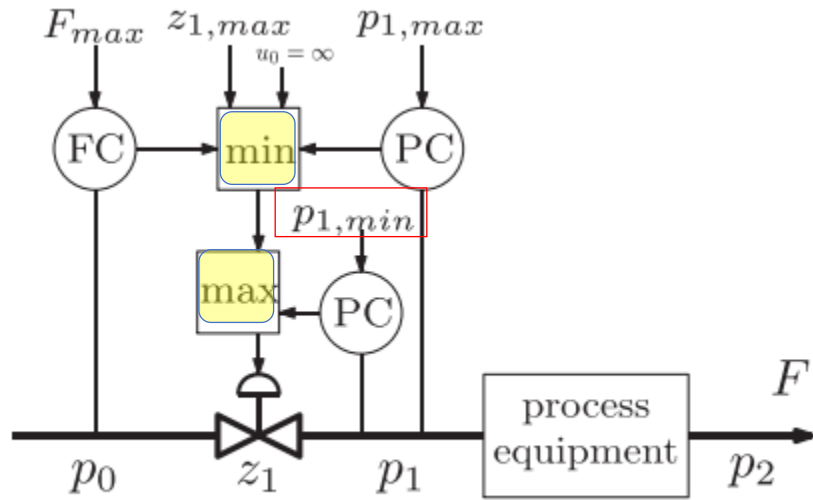
(15)

Possible conflict

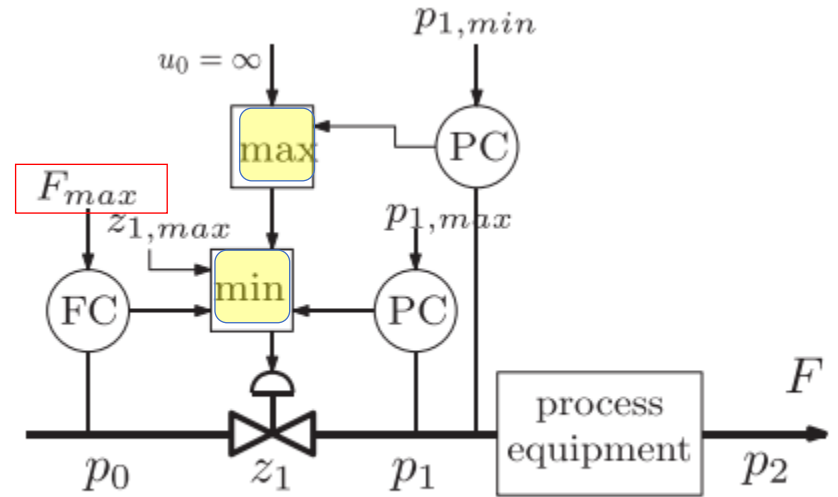
where $F_{max} = 10$ kg/s, $z_{1,max} = 1$, $p_{1,max} = 2.5$ bar, and $p_{1,min} = 1.5$ bar. Note that there are both max and min- constraints on p_1 . De-

The two p_1 -constraints are not conflicting, because they are on the same variable. However the F_{max} -constraint and p_{1min} -constraint may be conflicting: Must choose which is most important.

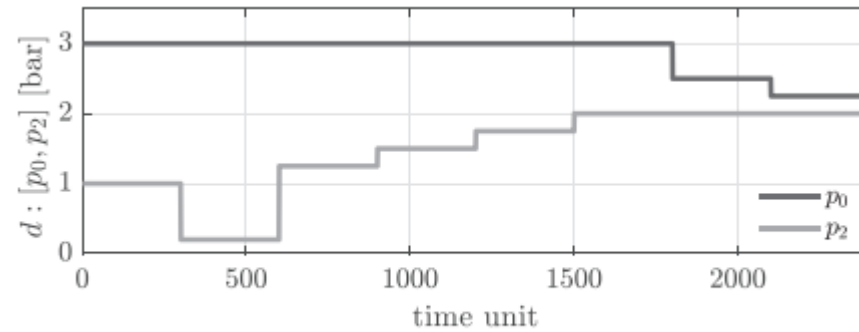
CV-CV switching



(a)



(b)

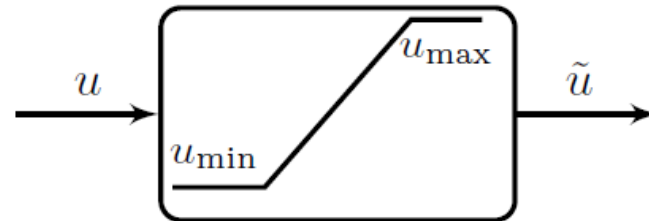


Disturbances in p_0 and p_2 (unmeasured)

Valves have “built-in” selectors

Rule 3 (a bit opposite of what you may guess)

- A closed valve ($u_{\min}=0$) gives a “built-in” max-selector (to avoid negative flow)
- An open valve ($u_{\max}=1$) gives a “built-in” min-selector
 - So: Not necessary to add these as selector blocks (but it will not be wrong).
 - The “built-in” selectors are never conflicting because cannot have closed and open at the same time
 - Another way to see this is to note that a valve works as a saturation element

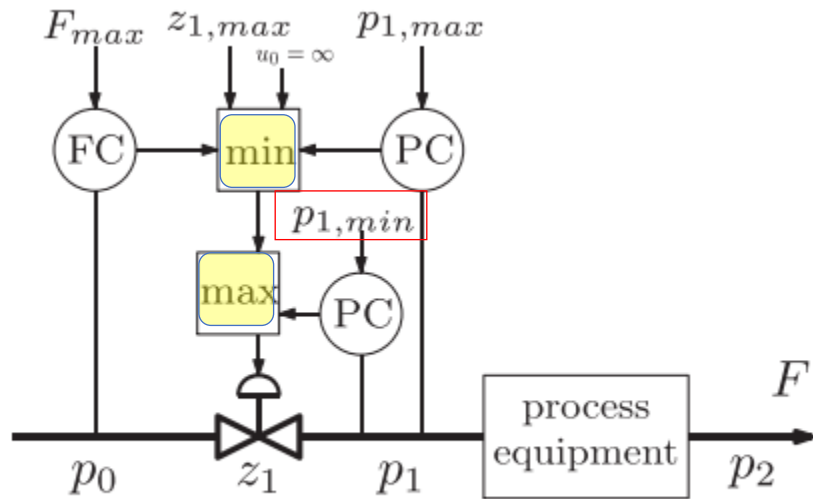


Saturation element may be implemented in three other ways (equivalent because never conflict)

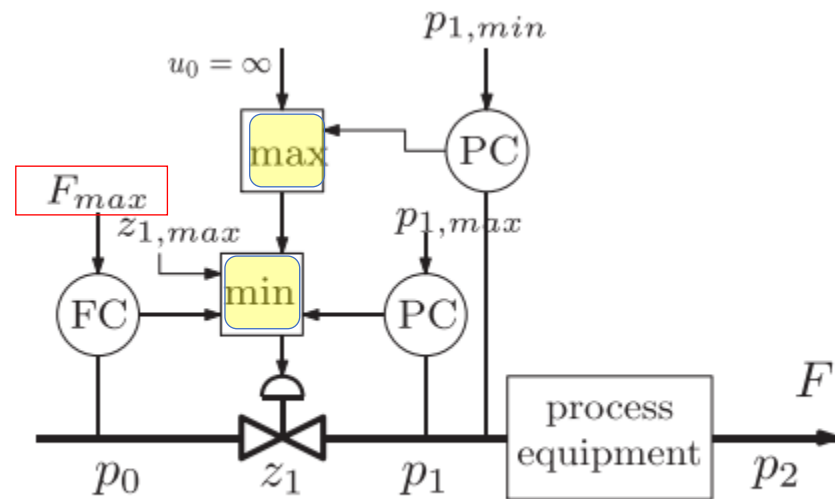
1. Min-selector followed by max-selector
2. Max-selector followed by min-selector
3. Mid-selector

$$\tilde{u} = \max(u_{\min}, \min(u_{\max}, u)) = \min(u_{\max}, \max(u_{\min}, u)) = \text{mid}(u_{\min}, u, u_{\max})$$

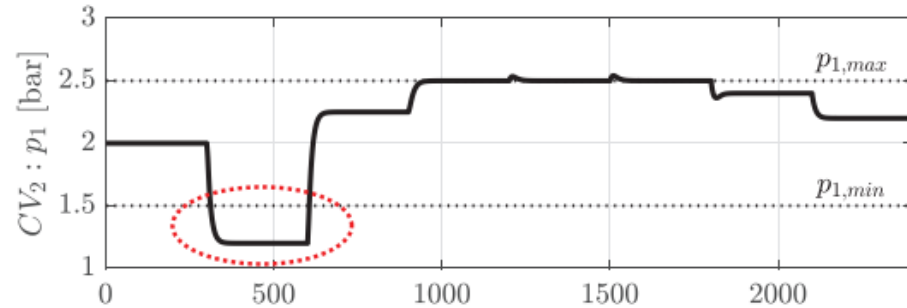
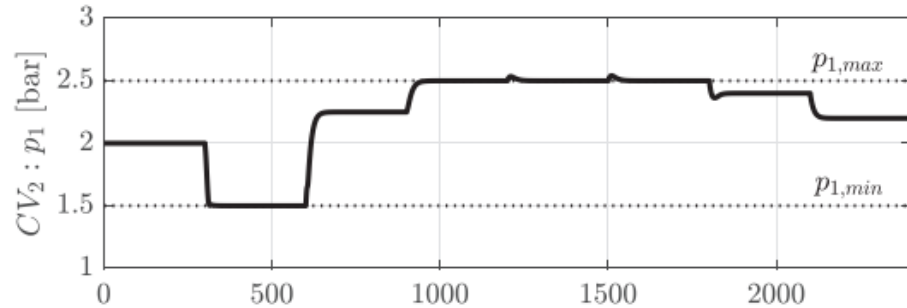
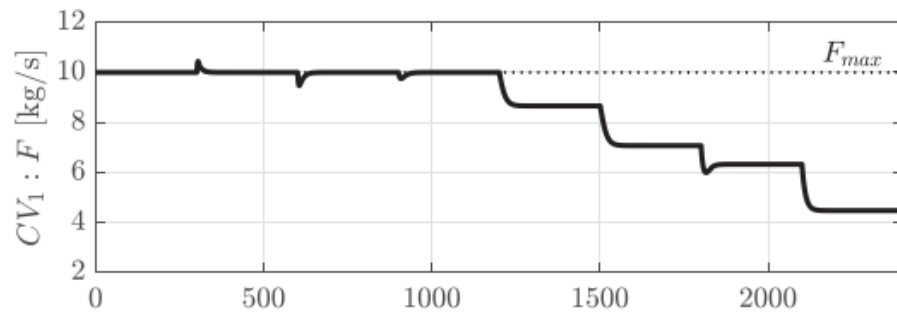
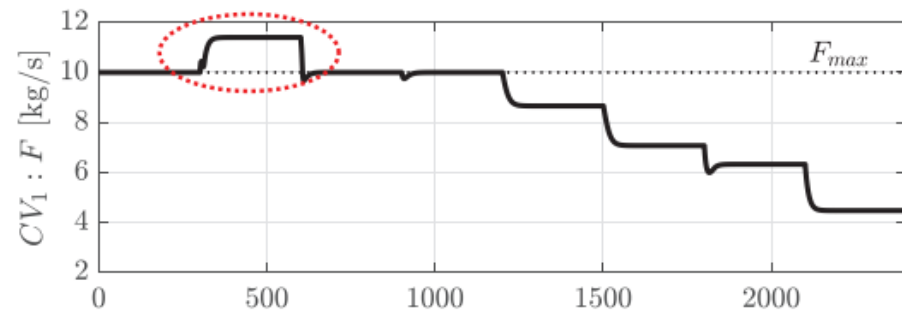
CV-CV switching



(a)



(b)

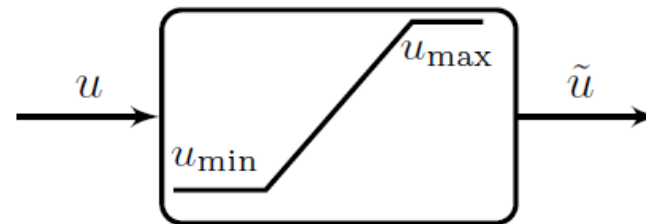


t > 1800: u = zmax = 1

Valves have “built-in” selectors

Rule 3 (maybe a bit opposite of what you may guess)

- A closed valve ($u_{\min}=0$) gives a “built-in” max-selector (to avoid negative flow)
- An open valve ($u_{\max}=1$) gives a “built-in” min-selector
 - So: Not necessary to add these as selector blocks (but it will not be wrong).
 - The “built-in” selectors are never conflicting because cannot have closed and open at the same time
 - Another way to see this is to note that a valve works as a saturation element



Saturation element may be implemented in three ways (equivalent because never conflict)

1. Min-selector followed by max-selector
2. Max-selector followed by min-selector
3. Mid-selector

$$\tilde{u} = \max(u_{\min}, \min(u_{\max}, u)) = \min(u_{\max}, \max(u_{\min}, u)) = \text{mid}(u_{\min}, u, u_{\max})$$

Quiz. Is this OK?

Cristina: I am looking at a control solution using selectors for keeping the pressure within constraints, while maximizing the valve opening. See figure This is for the valve before a steam turbine. This should work OK, right? Of course, if pmax is reached while the valve is fully open, the turbine bypass will have to open.

Answer:

Rule 1. Yes, the rule is to use a max-selector for a constraint which is satisfied with a large input. And since the pressure is measured upstream, the pressure will get lower if we increase the valve opening, making it easier to satisfy the pmax-constraint. So yes, this is OK.

Rule 1. Similar for the min-block with pmin.

Rule 2. Since you have two constraints on the same variable, you cannot have infeasibility so the order of the min. and max-blocks doesn't matter for pmin and pmax.

Rule 2. Yes, the desired value $u_0 = z_{max}$ should always enter the first block.

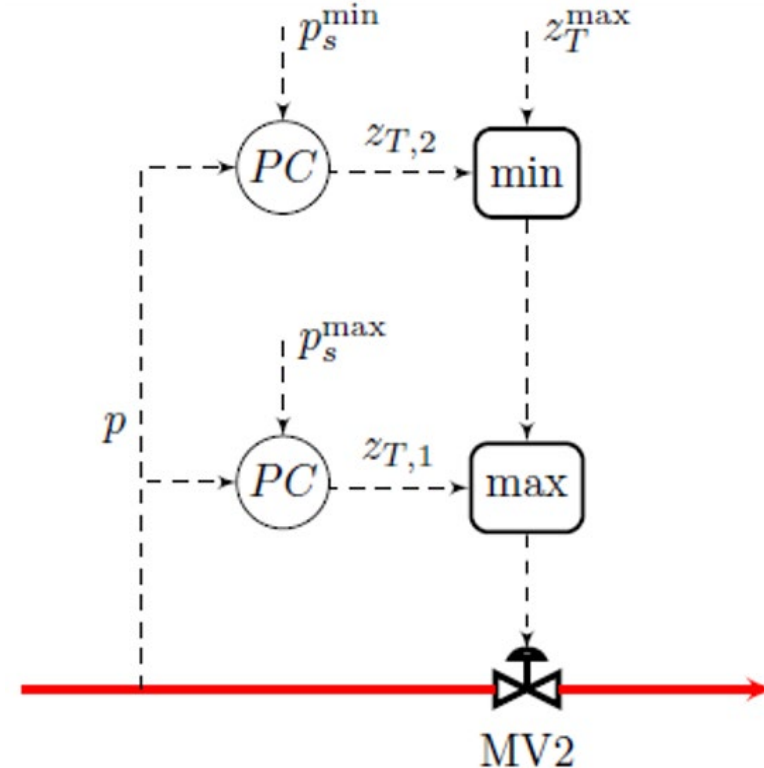
Conclusion: yes, it works.

BUT....

Comment 1: But note that there is also a "hidden" min-selector just before the valve because of the valve which has z_{max} . And also a "hidden" max-selector because of z_{min} (a fully closed valve). These constraints may be inconsistent with the pressure constraints.

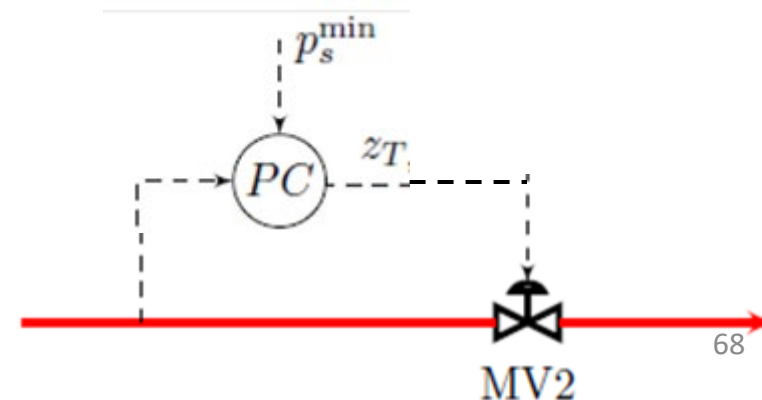
Comment 2: Since the order of the two selectors does not matter in this case, one may instead use the "equivalent" alternative with the max-block first. But we then see clearly that **the constraint on pmax will never be activated**, because $z_{T,max}$ is large. I guess this makes sense since you want to have the valve as open as possible, so then the you will always be at the pmin-constraint or have a fully open valve. **So you can cut the pmax-constraint (and thus the max-selector)** as you anyway want to open the valve as much as possible.

In addition, you can also cut the min-selector because there is already a "hidden" min-selector with z_{max} . (On the other hand, it will not be wrong to keep them.)



Final conclusion: Yes, it works, but it's much too complicated.

- All what is shown can be replaced by a pressure controller (PC) with setpoint p^{min} .



Challenges selectors

- Standard approach requires pairing of each active constraint with a single input
 - May not be possible in complex cases
 - See RTO/feedback-based RTO
- Stability analysis of switched systems is still an open problem
 - Undesired switching may be avoided in many ways:
 - Filtering of measurement
 - Tuning of anti-windup scheme
 - Minimum time between switching
 - Minimum input change

MV-MV switching



- One CV, many MVs (to cover whole steady-state range because primary MV may saturate)*
- Use one MV at a time

Three alternatives:

Alt.1 Split-range control (SRC)

- Plus Generalized SRC (baton strategy)

Alt.2 Several controllers (one for each MV) with different setpoints for the single CV

Alt.3 Valve position control (VPC)

Which is best? It depends on the case!

*Optimal Operation with Changing Active Constraint Regions using Classical Advanced Control, Adriana Reyes-Lua Cristina Zotica, Sigurd Skogestad, Adchem Conference, Shenyang, China. July 2018 ,

Example MV-MV switching

- Break and gas pedal in a car
- Use only one at a time,
- «manual split range control»

E5. Split-range control (SRC)

⁷Note the blue saturation elements for the inputs in Figure 21 and other block diagrams. Saturation can occur for any physical input, but they are explicitly shown for cases where the saturation is either the reason for or part of the control logic. For example, in Figure 21, the reason for using u_2 is that u_1 may saturate.

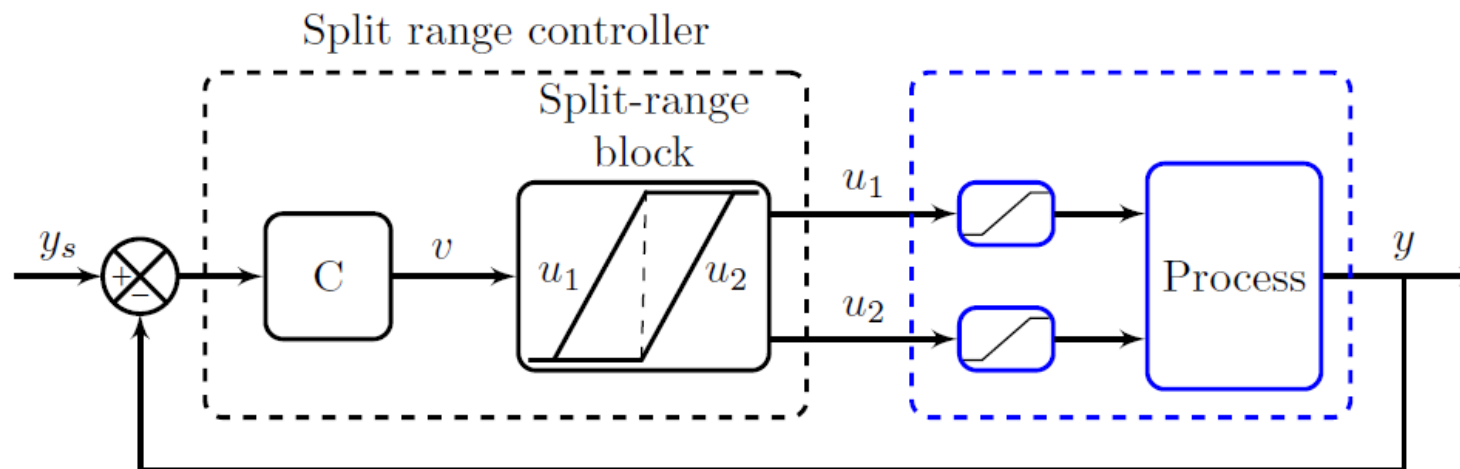


Figure 21: Split range control for MV-MV switching.

For MVs (u) that have same effect (same sign) on the output (y) (Fig. 21), we need to define the order in which the MVs will be used. This is done by the order in the SR-block.

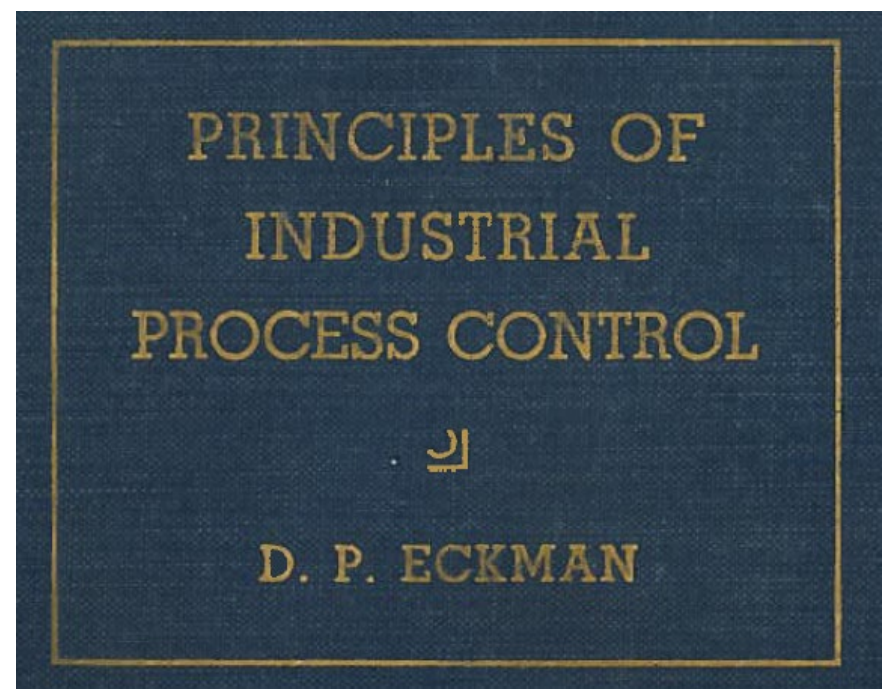
Example: With two heating sources, we need to decide which to use first (see next Example)

Advantage: SRC is easy to understand and implement!

Disadvantages:

1. Only one controller $C \Rightarrow$ Same integral time for all inputs u_i (MVs)
 - Controller gains can be adjusted with slopes in SR-block!
2. Does not work well for cases where constraint values for u_i change

Split range control: Donald Eckman (1945)

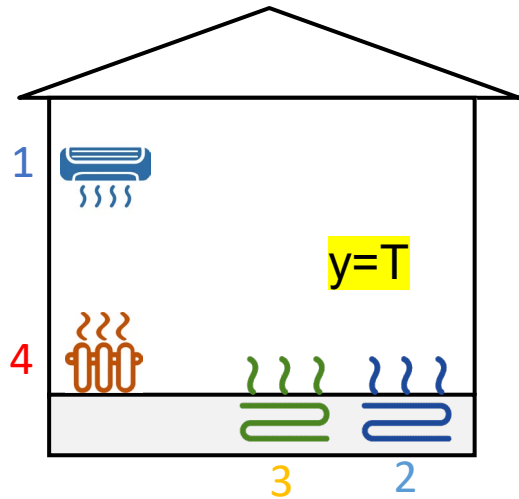


The temperature of plating tanks is controlled by means of dual control agents. The temperature of the circulating water is controlled by admitting steam when the temperature is low, or cold water when it is high. Figure 10-12 illustrates a system where pneumatic proportional control and diaphragm valves with split ranges are used. The steam valve is closed at 8.5 lb per sq in. pressure from the controller, and fully open at 14.5 lb per sq in. pressure. The cold water valve is closed at 8 lb per sq in. air pressure and fully open at 2 lb per sq in. air pressure.

If more accurate valve settings are required, pneumatic valve positioners will accomplish the same function. The zero, action, and range adjustments of valve positioners are set so that both the steam and cold water valves are closed at 8 lb per sq in. controller output pressure. The advantages gained with valve positioners are that at a given

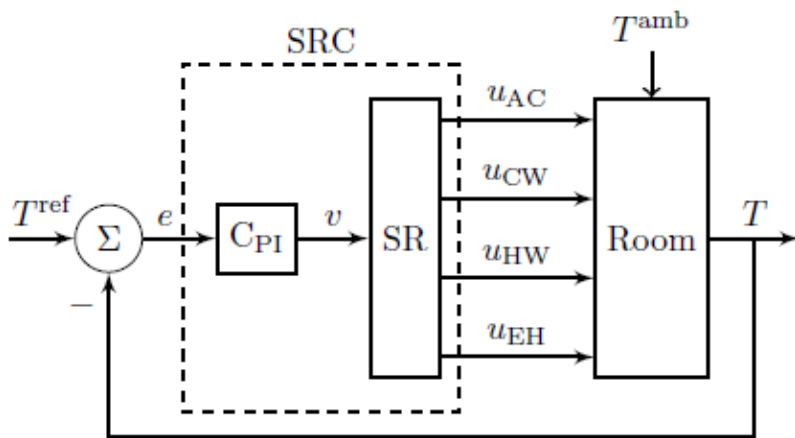
FIG. 10-12. Dual-Agent Control System for Adjusting Heating and Cooling of Bath.

Example split range control: Room temperature with 4 MVs

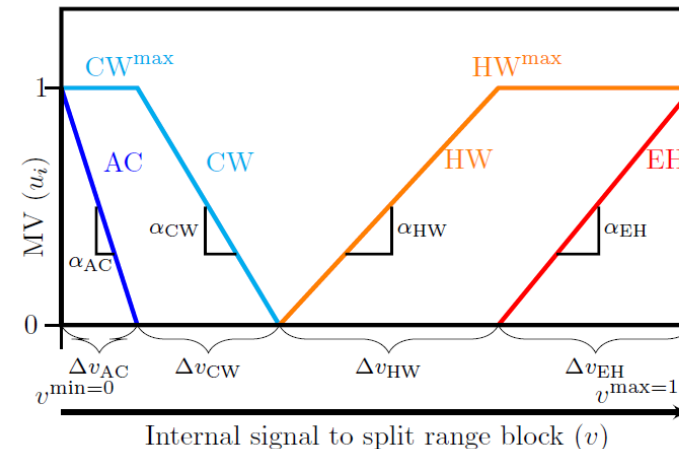


MVs (two for summer and two for winter):

1. AC (expensive cooling)
2. CW (cooling water, cheap)
3. HW (hot water, quite cheap)
4. Electric heat, EH (expensive)

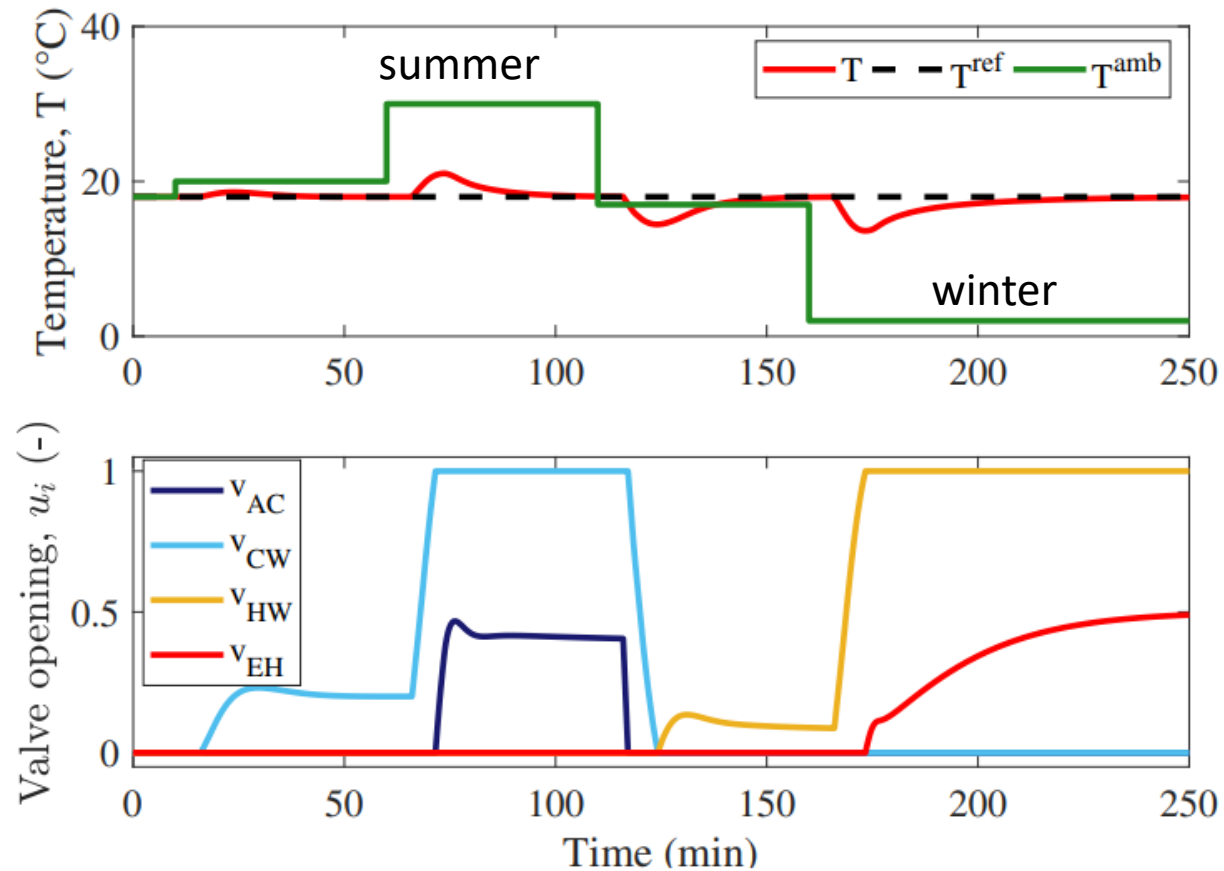
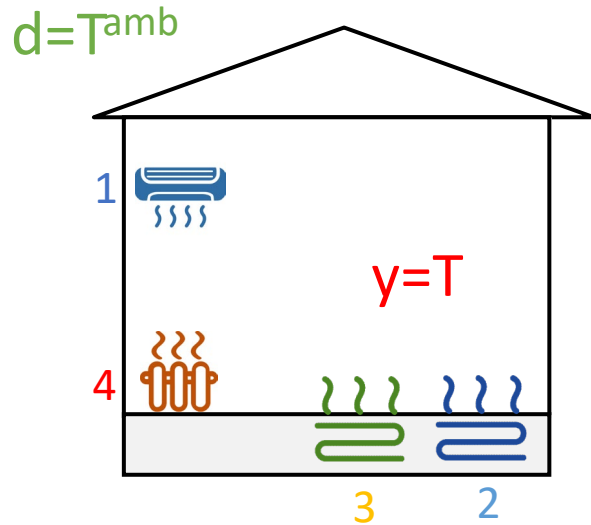


SR-block:



C_{PI} – same controller for all inputs (one integral time)
 But get different gains by adjusting slopes α in SR-block

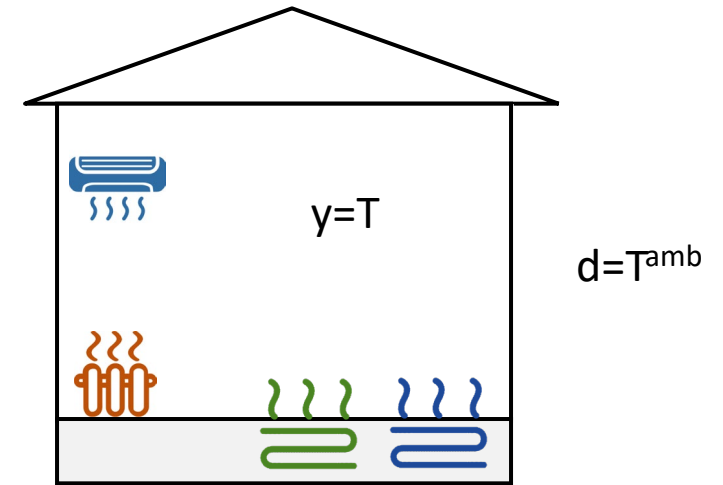
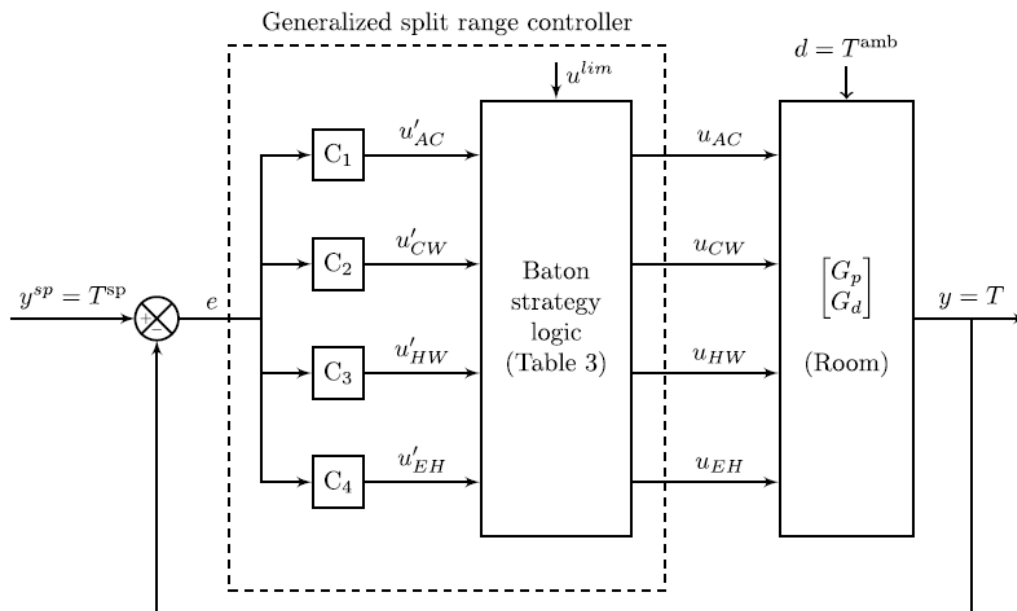
Simulation Split-range control (SRC)



Disadvantages Standard Split-range control (SRC):

1. Must use same integral/derivative time for all MVs
2. Does not work well when constraint values change (SR-block problem)

Alternative: Generalized SRC (Baton strategy: multiple independent controllers)



All four controllers need anti-windup

Table 3
Baton strategy logic for case study.

Value of u'_k	Active input (input with baton, u_k)			
	$u_1 = u_{AC}$	$u_2 = u_{CW}$	$u_3 = u_{HW}$	$u_4 = u_{EH}$
$u'_k < u'_k < u'_k$	Keep u_1 active $u_1 \leftarrow u'_1$ $u_2 \leftarrow u'_2$ $u_3 \leftarrow u'_3$ $u_4 \leftarrow u'_4$	Keep u_2 active $u_1 \leftarrow u_1^{min}$ $u_2 \leftarrow u'_2$ $u_3 \leftarrow u_3^{min}$ $u_4 \leftarrow u_4^{min}$	Keep u_3 active $u_1 \leftarrow u_1^{min}$ $u_2 \leftarrow u_2^{min}$ $u_3 \leftarrow u'_3$ $u_4 \leftarrow u_4^{min}$	Keep u_4 active $u_1 \leftarrow u_1^{min}$ $u_2 \leftarrow u_2^{min}$ $u_3 \leftarrow u_3^{max}$ $u_4 \leftarrow u'_4$
$u'_k \geq u'_k$	Keep u_1 active (max. cooling)	Baton to u_1 $u_1^0 = u_1^{min}$	Baton to u_4 $u_4^0 = u_4^{min}$	Keep u_4 active (max. heating)
$u'_k \leq u'_k$	Baton to u_2 $u_2^0 = u_2^{max}$	Baton to u_3 $u_3^0 = u_3^{min}$	Baton to u_2 $u_2^0 = u_2^{min}$	Baton to u_3 $u_3^0 = u_3^{max}$

A. Reyes-Lúa and S. Skogestad. "Multi-input single-output control for extending the operating range: Generalized split range control using the baton strategy". Journal of Process Control 91 (2020)

Disadvantages Standard Split-range control (SRC):

1. Must use same integral/derivative time for all MVs
2. Does not work well when constraint values change (SR-block problem)

Alternative: Generalized SRC (Baton strategy: multiple independent controllers)

Table 3
Baton strategy logic for case study.

Value of u'_k	Active input (input with <i>baton</i> , u_k)			
	$u_1 = u_{AC}$	$u_2 = u_{CW}$	$u_3 = u_{HW}$	$u_4 = u_{EH}$
$u_k^{min} < u'_k < u_k^{max}$	Keep u_1 active $u_1 \leftarrow u'_1$ $u_2 \leftarrow u_2^{max}$ $u_3 \leftarrow u_3^{min}$ $u_4 \leftarrow u_4^{min}$	Keep u_2 active $u_1 \leftarrow u_1^{min}$ $u_2 \leftarrow u'_2$ $u_3 \leftarrow u_3^{min}$ $u_4 \leftarrow u_4^{min}$	Keep u_3 active $u_1 \leftarrow u_1^{min}$ $u_2 \leftarrow u_2^{min}$ $u_3 \leftarrow u'_3$ $u_4 \leftarrow u_4^{min}$	Keep u_4 active $u_1 \leftarrow u_1^{min}$ $u_2 \leftarrow u_2^{min}$ $u_3 \leftarrow u_3^{max}$ $u_4 \leftarrow u'_4$
$u'_k \geq u_k^{max}$	Keep u_1 active (max. cooling)	Baton to u_1 $u_1^0 = u_1^{min}$	Baton to u_4 $u_4^0 = u_4^{min}$	Keep u_4 active (max. heating)
$u'_k \leq u_k^{min}$	Baton to u_2 $u_2^0 = u_2^{max}$	Baton to u_3 $u_3^0 = u_3^{min}$	Baton to u_2 $u_2^0 = u_2^{min}$	Baton to u_3 $u_3^0 = u_3^{max}$

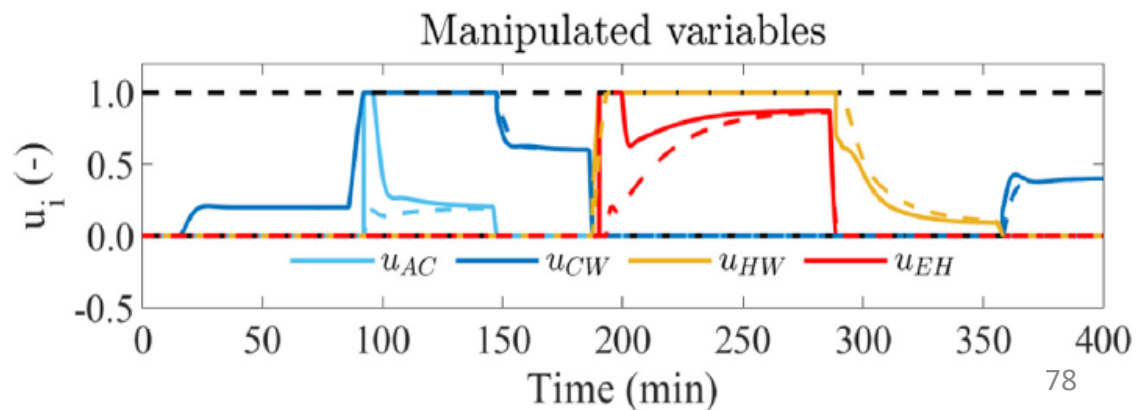
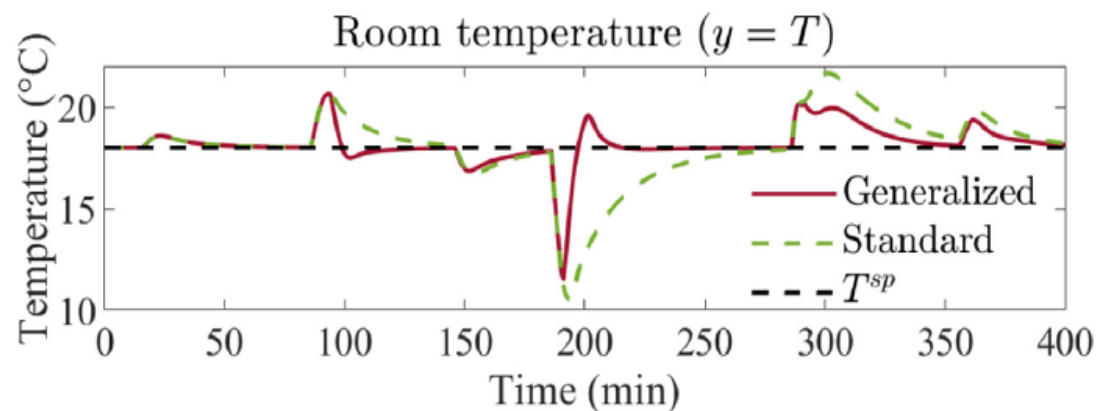
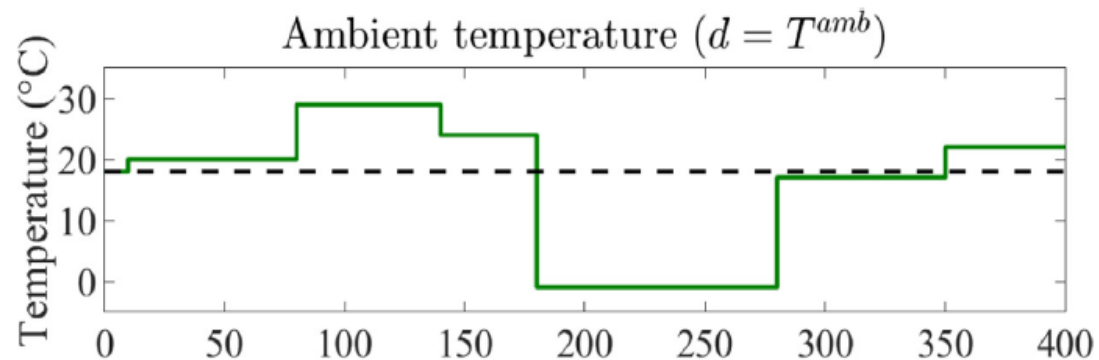
ib

All four controllers need anti-windup

Comparison of standard and generalized SRC

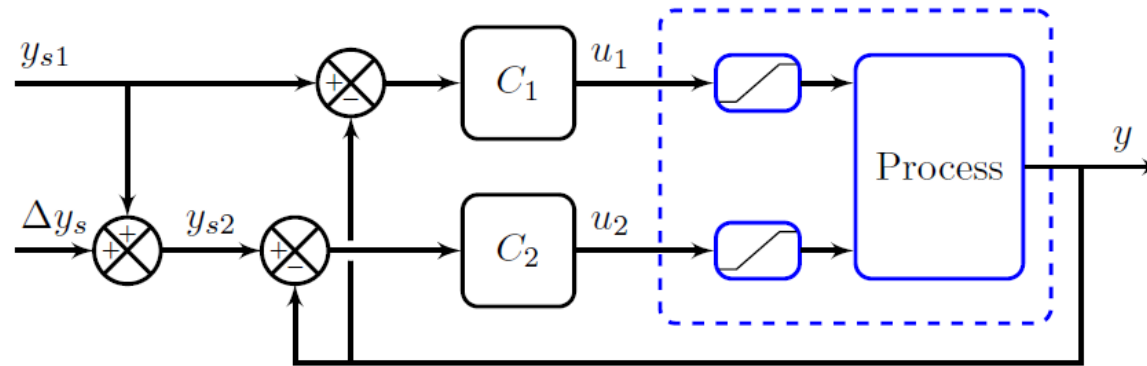
Generalized split range control:

- Different (smaller) integral times for each input
- Gives faster settling for most inputs



What about Model Predictive control (MPC)?

E6. Separate controllers with different setpoints (for MV-MV switching)



Comment on the blue blocks: Saturation can occur for any physical input, but they are explicitly shown for cases where the saturation is either the reason for or part of the control logic.

Figure 22: Separate controllers with different setpoints for MV-MV switching.

The setpoints (y_{s1}, y_{s2}, \dots) should be in the same order as we want to use the MVs. The setpoint differences (e.g., $\Delta y_s = y_{s2} - y_{s1}$ in Fig. 22) should be large enough so that, in spite of disturbances and measurement noise for y , only one controller (and its associated MV) is active at a given time (with the other MVs at their relevant limits).

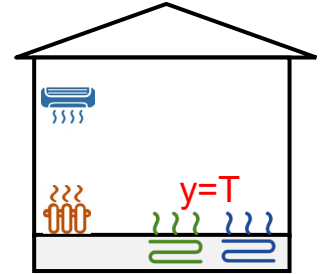
Advantages E6 (compared to split range control, E5):

1. Simple to implement (no logic)
2. Controllers can be tuned independently (different integral times)
3. Switching by feedback: Do not need to know constraint values
 - Big advantage when switching point varies (complex MV-CV switching)

Disadvantages:

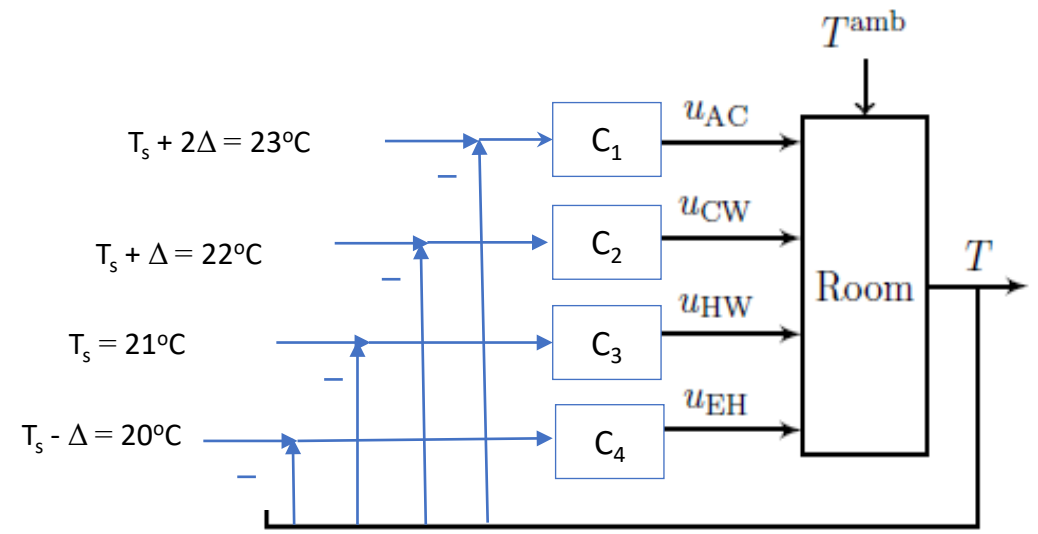
1. Temporary loose control during switching
2. Setpoint not constant
 - Can be an advantage (for example, may give energy savings for room heating)

Example: Room heating with one CV (T) and 4 MVs



- MVs (two for summer and two for winter):
1. AC (expensive cooling)
 2. CW (cooling water, cheap)
 3. HW (hot water, quite cheap)
 4. Electric heat, EH (expensive)

Alt. A2 for MV-MV switching. Multiple controllers with different setpoints

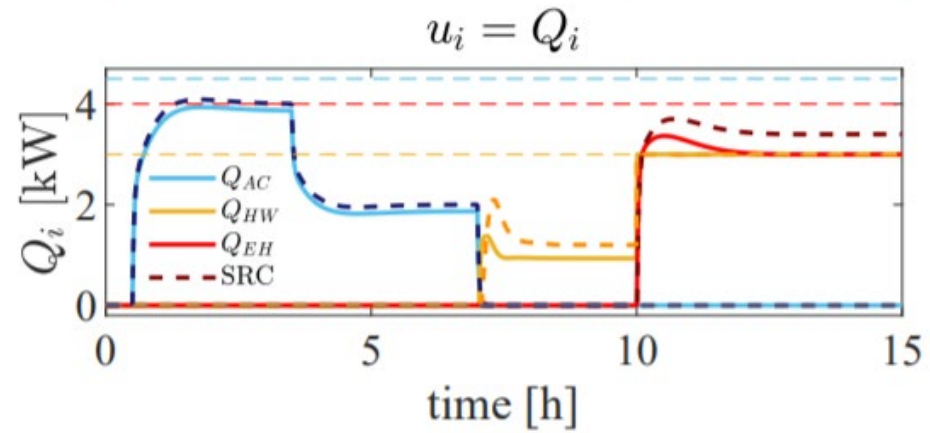
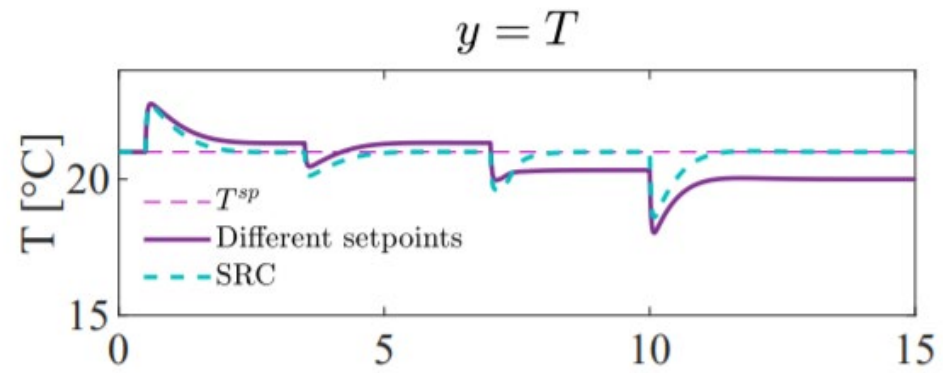
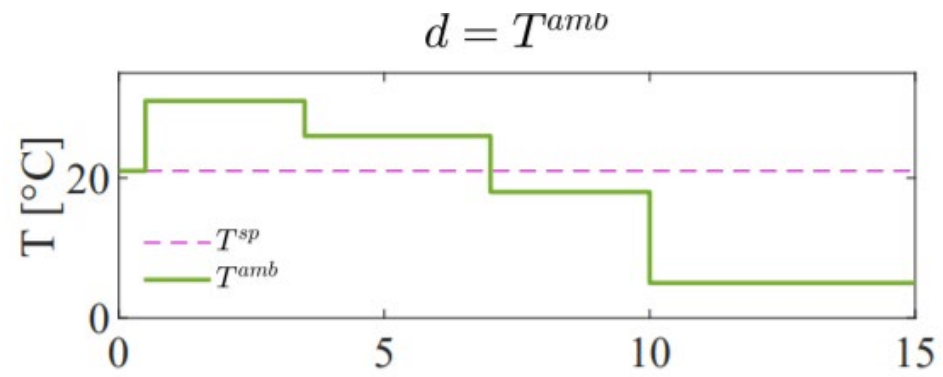
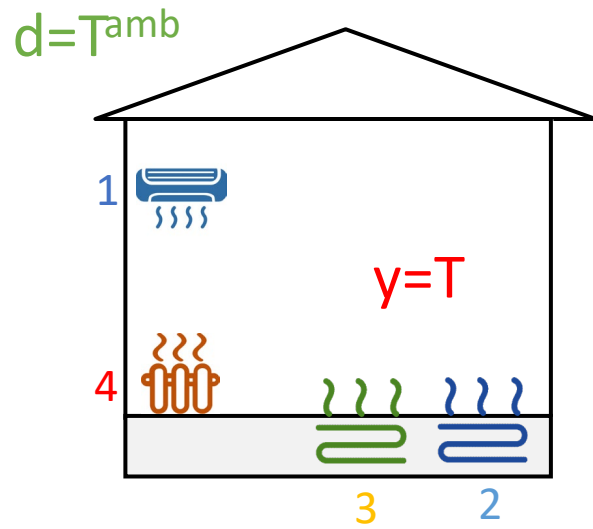


Disadvantage (comfort):

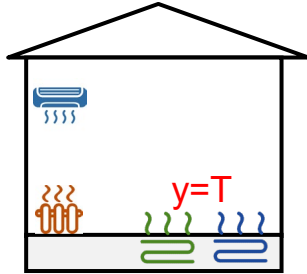
- Different setpoints

Advantage (economics):

- Different setpoints (energy savings)



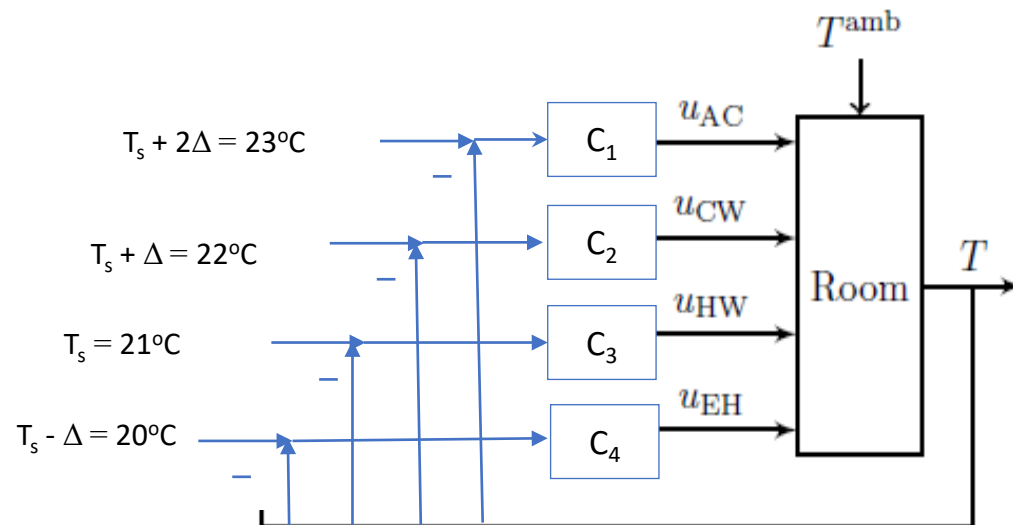
Example: Room heating with one CV (T) and 4 MVs



MVs (two for summer and two for winter):

1. AC (expensive cooling)
2. CW (cooling water, cheap)
3. HW (hot water, quite cheap)
4. Electric heat, EH (expensive)

Alt. A2 for MV-MV switching. Multiple controllers with different setpoints

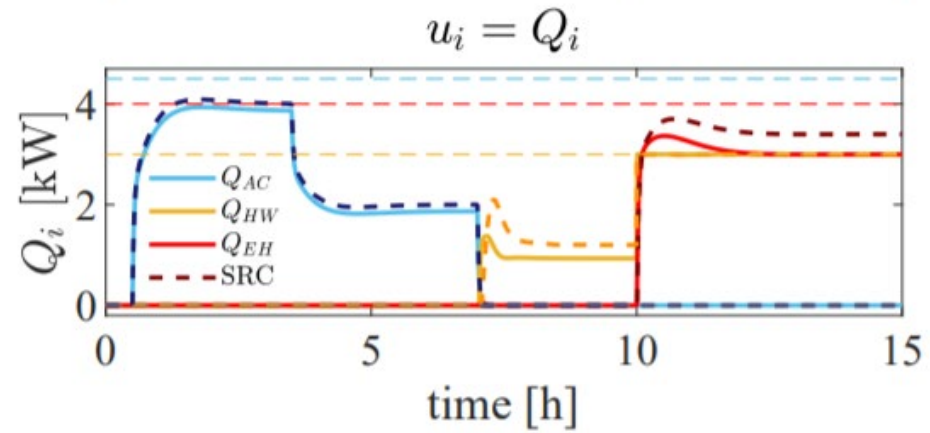
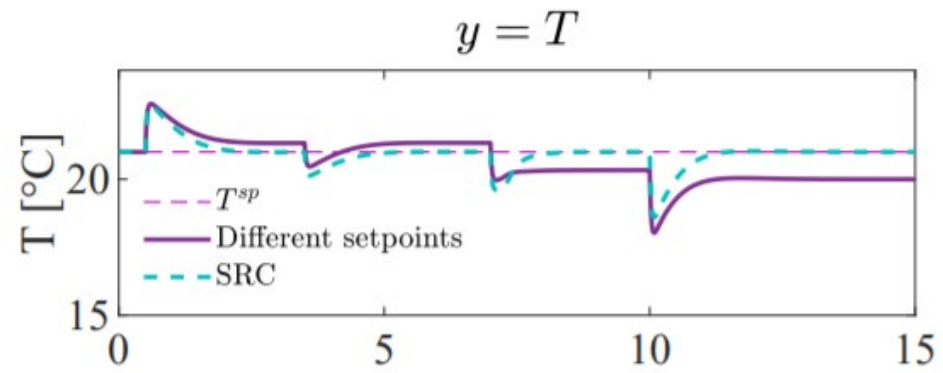
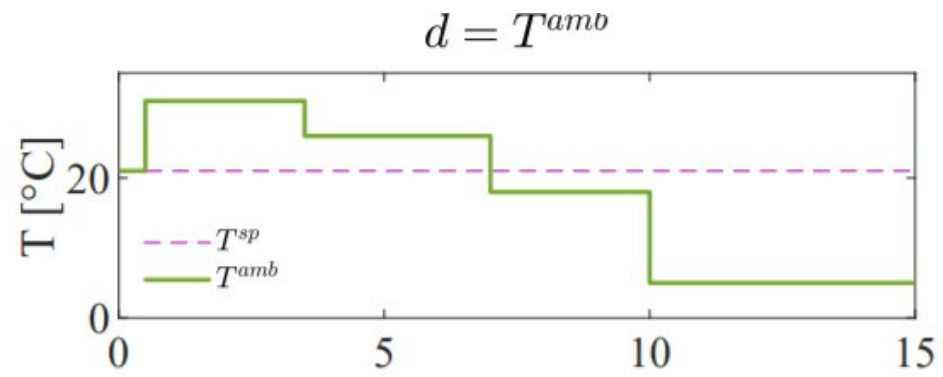
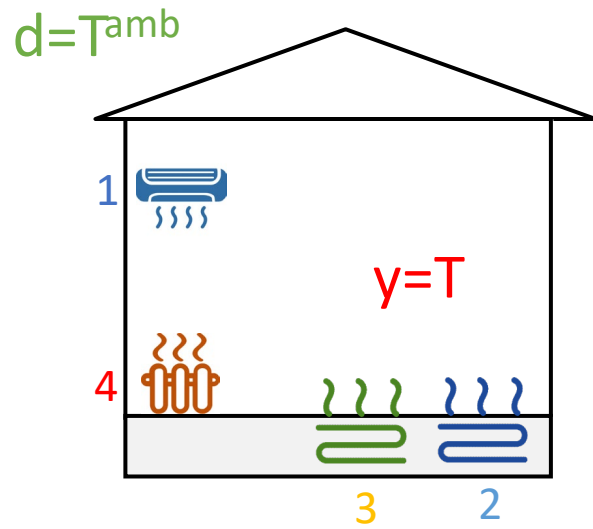


Disadvantage (comfort):

- Different setpoints

Advantage (economics) :

- Different setpoints (energy savings)



Want separate controllers.

Fixes that avoid using different setpoints

Alt.1: «Baton strategy» (a bit complicated).

Alt.2 (simpler, but gives temporary setpoint change at MV-MV switch):
Introduce a (slow) outer cascade (master controller) that resets the setpoint of the active controller to y_s , while maintaining the setpoint distances

Fix: Outer cascade to avoid different setpoints

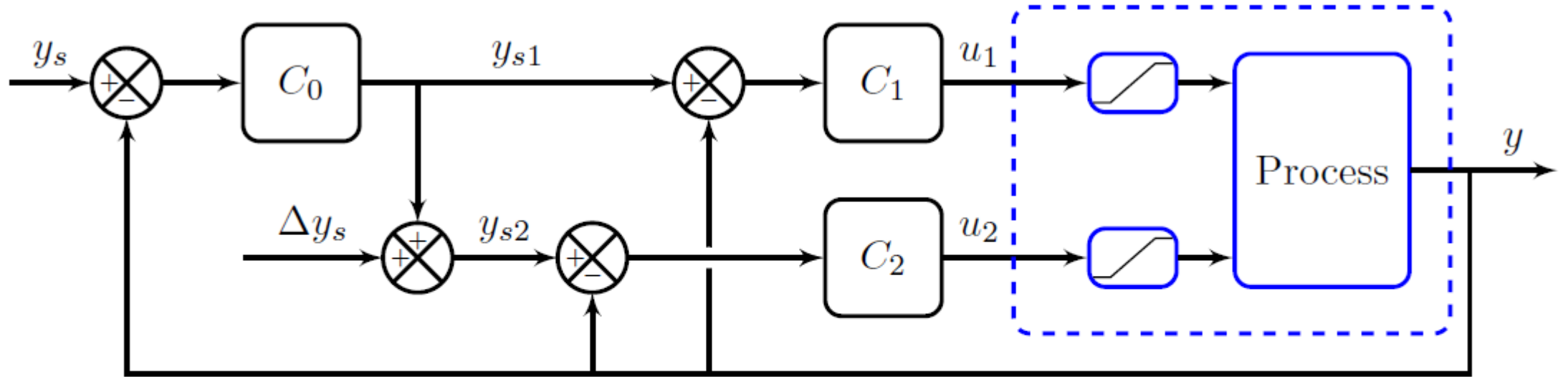


Figure 23: Separate controllers for MV-MV switching with outer resetting of setpoint. This is an extension of the scheme in Figure 22, with a slower outer controller C_0 that resets y_{1s} to keep a fixed setpoint $y = y_s$ at steady state.

E7. VPC on main steady-state input (for MV-MV switching)

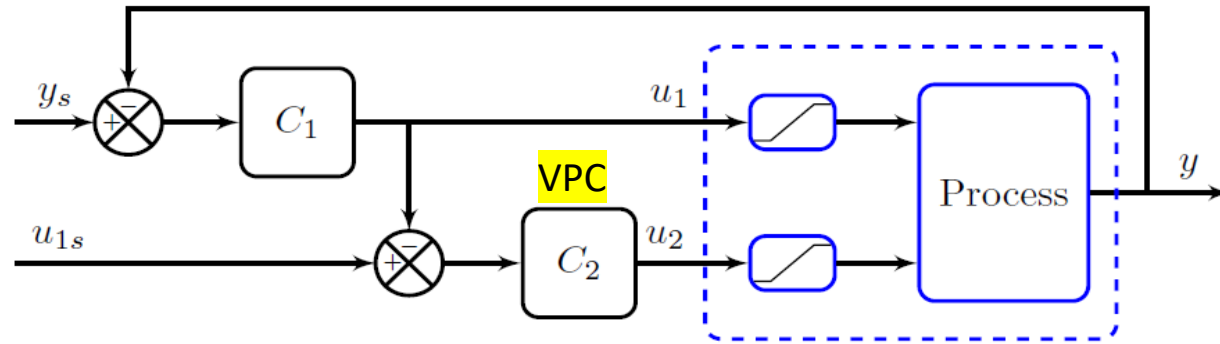


Figure 24: Valve (input) position control for MV-MV switching. A typical example is when u_2 is needed only in fairly rare cases to avoid that u_1 saturates.

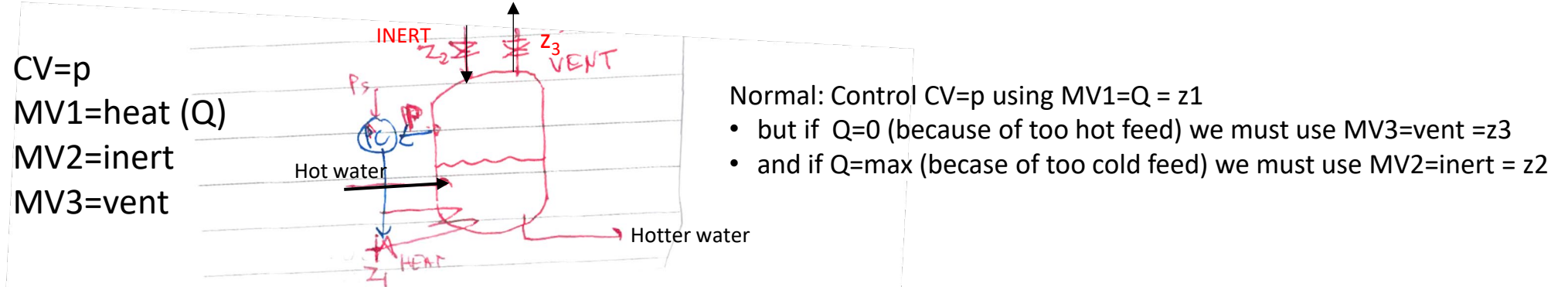
Advantages E7 (for MV-MV switching): Always use u_1 to control y

- For example, u_2 may only allow discrete changes (e.g., $u_2=0,1,2,3$)
- or dynamics for u_2 may be very slow

Disadvantages E7:

1. We cannot let u_1 become fully saturated because then control of y is lost
 - This means that we cannot use the full range for u_1 (potential economic loss)
2. Related: When u_2 is used, we need to keep using a “little” of u_1 .
 - Example: May need to use both heating and cooling at the same time (when u_1 normally should be off).

Example MV-MV switching: Pressure control (Alt. 3 may be the best in this case)



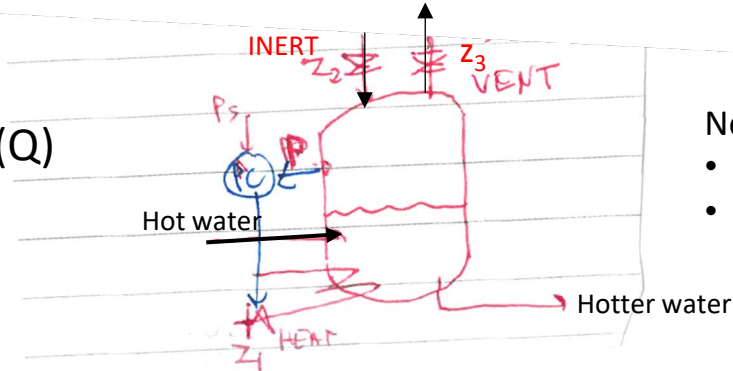
Example: Heating water to 213C = Control steam pressure at 20 bar*.

- «Inert» (z2) could be HP steam.

* Rule of thumb: $p[\text{bar}] = [T[\text{C}]/100]^4$

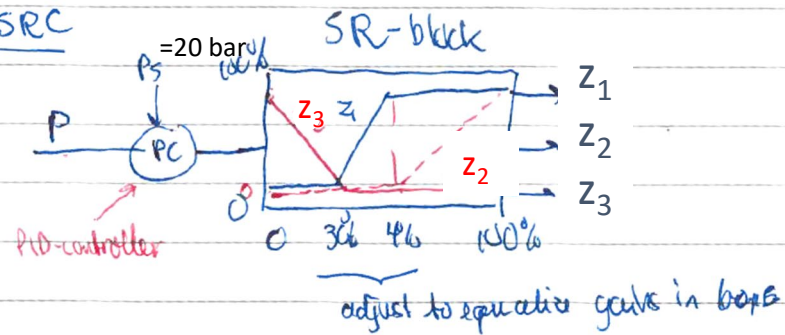
Example MV-MV switching: Pressure control (Alt. 3 may be the best in this case)

CV=p
MV1=heat (Q)
MV2=inert
MV3=vent



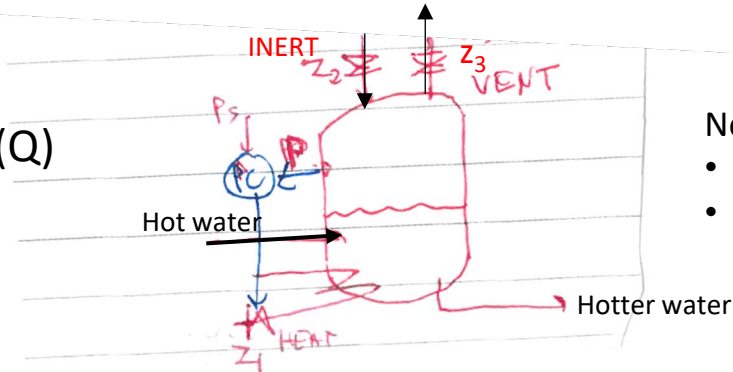
- Normal: Control CV=p using MV1=Q
- but if Q=0 we must use MV3=vent
 - and if Q=max we must use MV2=inert

Alt. 1. SRC



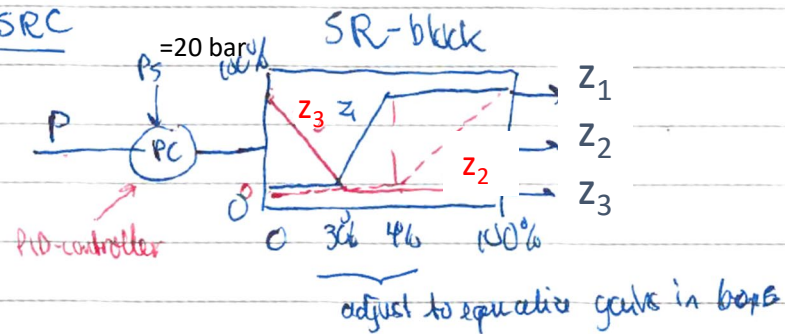
Example MV-MV switching: Pressure control (Alt. 3 may be the best in this case)

CV=p
 MV1=heat (Q)
 MV2=inert
 MV3=vent

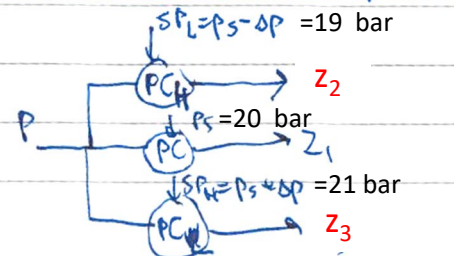


- Normal: Control CV=p using MV1=Q
- but if $Q=0$ we must use MV3=vent
 - and if $Q=\max$ we must use MV2=inert

Alt. 1. SRC

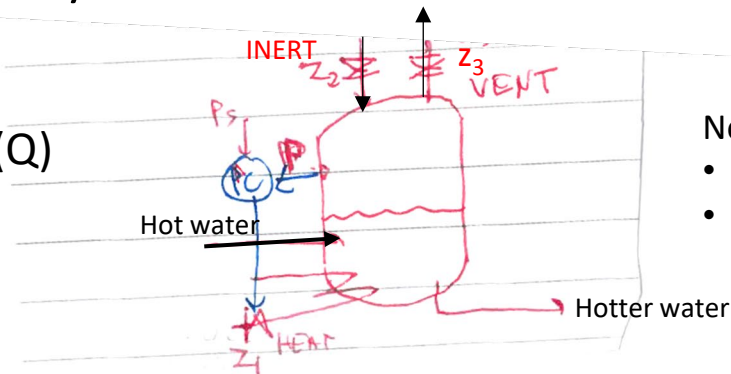


Alt. 2. Three controllers with different setpoints.



Example MV-MV switching: Pressure control (Alt. 3 may be the best in this case)

CV=p
MV1=heat (Q)
MV2=inert
MV3=vent



- Normal: Control CV=p using MV1=Q
- but if $Q=0$ we must use MV3=vent
 - and if $Q=\max$ we must use MV2=inert

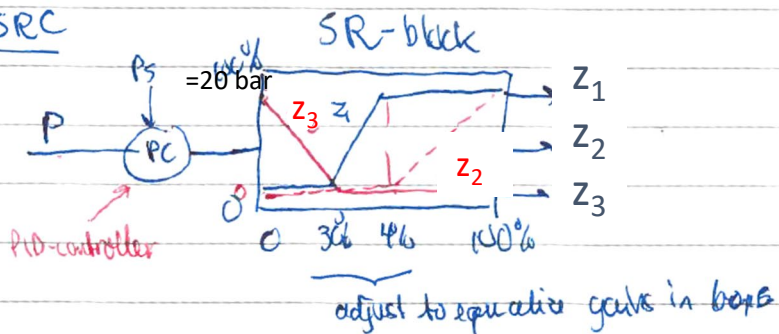
Alt.3: VPC (z2 and z3 could here even be on/off valves)

Always use Q (z1) to control p.

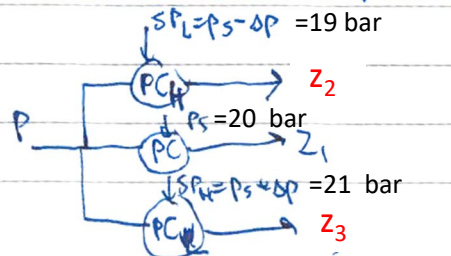
Need two VPC's:

- Use vent (z3) to avoid Q small ($z1=0.1$)
- Use inert (z2) to avoid Q large ($z1=0.9$)
- $z2=0$ and $z3=0$ when $0.1 < z1 < 0.9$

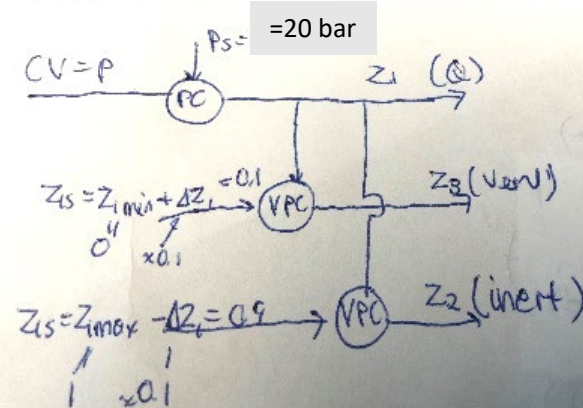
Alt.1. SRC



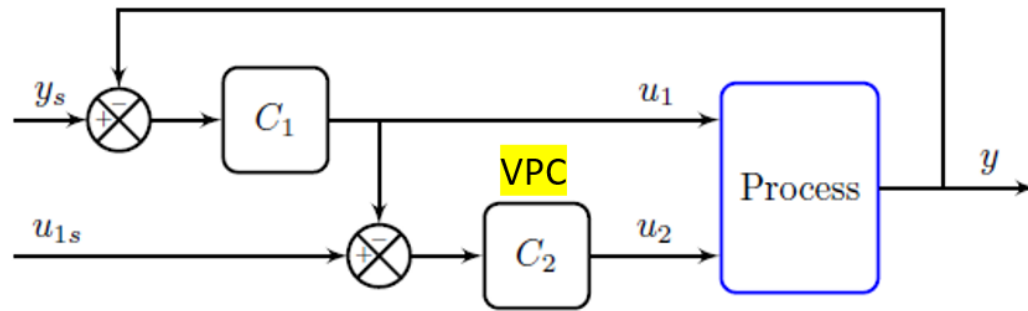
Alt.2. Three controllers with different setpoints.



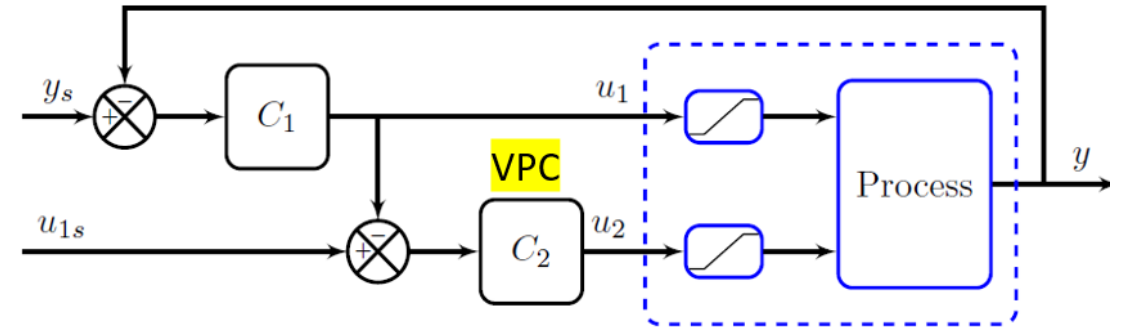
Alt.3 VPC



Beware: Two different applications of VPC (E3 and E7)



E3 (Fig. 12)



E7 (Fig. 24)

The VPC schemes in Figure 12 (E3 - VPC on dynamic input) and Figure 24 (E7) seem to be the same

- In fact, they are the same - except for the blue saturation elements - which tells that in Figure 24 (E7) the saturation has to be there for the structure to work as expected

But their behavior is very different!

- In Figure 12 (E3) both inputs are used all the time
 - u_1 is used to improve the dynamic response
 - u_2 is the main steady-state input (and used all the time)
 - u_{1s} is typically 50% (mid-range)
- In Figure 24 (E7)
 - u_1 is the main input (and used all the time)
 - u_2 is only used when u_1 approaches saturation (for MV-MV switching)
 - The setpoint u_{1s} is typically close to the expected saturation constraint (10% or 90%)

I frequently see people confuse these two elements - which is very understandable!

The four switching cases in more detail

A. MV-MV switching (because MV may saturate)

- Need many MVs to cover whole steady-state range
- Use only one MV at a time
- **Three options:**
 - A1. Split-range control,
 - A2. Different setpoints,
 - A3. Valve position control (VPC)

B. CV-CV switching (because we may reach new CV constraint)

- Must select between CVs
- **One option:** Many controllers with Max-or min-selector

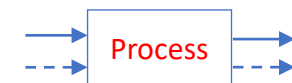
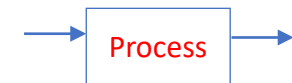
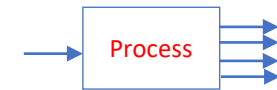
Plus the combination: MV-CV switching

C. **Simple** MV-CV switching: CV can be given up

- We followed «input saturation rule»
- Don't need to do anything (except anti-windup in controller)

D. **Complex** MV-CV switching: CV cannot be given up (need to «re-pair loops»)

- Must combine MV-MV switching (three options) with CV-CV switching (selector)



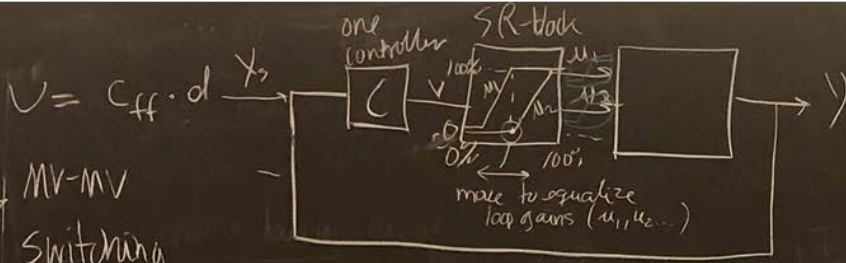
Note: we are here assuming that the constraints are not conflicting so that switching is possible

Summary MV-MV switching

E5. Split range control

Used when we need several MVs to cover whole range (of disturbances) at steady state
 - want to use one MV at a time.

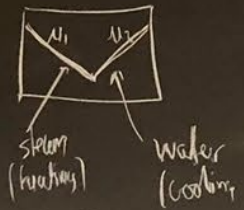
Three alternatives: simplest to understand is E5.



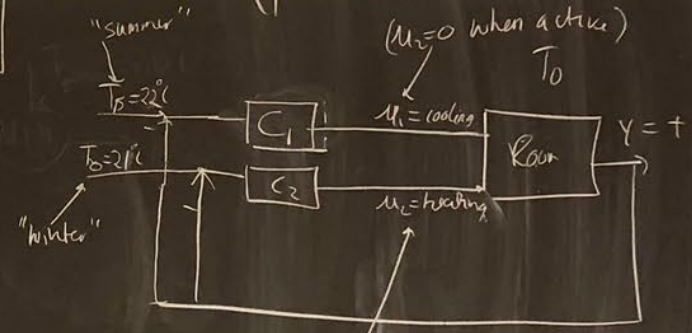
$U = C_{ff} \cdot d$
 MV-MV Switching.

$u_1 = \text{hot water (cheap)}$
 $u_2 = \text{electric heat}$

$y = T$

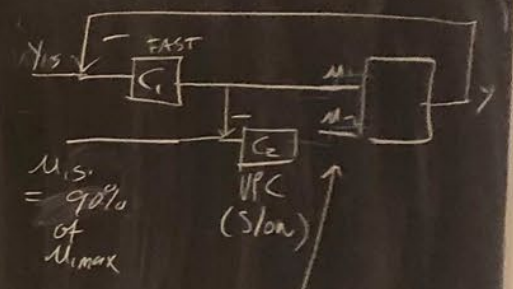


E6. Separate controllers with different setpoints (for MV-MV switching)



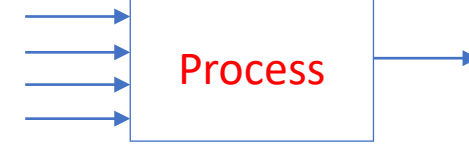
"Smart idea!"
 Need anti-windup!

E7. Use VP (on main steady state) input (for MV-MV switching)



u_2 : used when u_1 approaches saturation
 Normally $u_2 = 0$
 Advantage: Always use u_1 to control y !

Summary MV-MV switching



Split-range control (E5)

Advantage: SRC is easy to understand and implement!

Disadvantages:

1. Only one controller $C \Rightarrow$ Same integral time for all inputs u_i (MVs)
 - Controller gains can be adjusted with slopes in SR-block!
2. Does not work well for cases where constraint values for u_i change

Multiple controllers with different setpoints (E6)

Advantages several controllers (compared to split range control, E5):

1. Simple to implement (no logic)
2. Controllers can be tuned independently (different integral times)
3. Switching by feedback: Do not need to know constraint values
 - Big advantage when switching point varies (complex MV-CV switching)

Disadvantages:

1. Temporary loose control during switching
2. Setpoint not constant
 - Can be an advantage (for example, may give energy savings for room heating)

VPC for MV-MV switching) (E7)

Advantages VPC : Always use u_1 to control y

Disadvantages:

1. We cannot let u_1 become fully saturated because then control of y is lost
 - potential economic loss
2. Related: When use u_2 , need keep using a “little” of u_1 .
 - Example: May use both heating and cooling at the same time

E8. Anti-windup for the integral mode

Without anti-windup:

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

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 “exter-

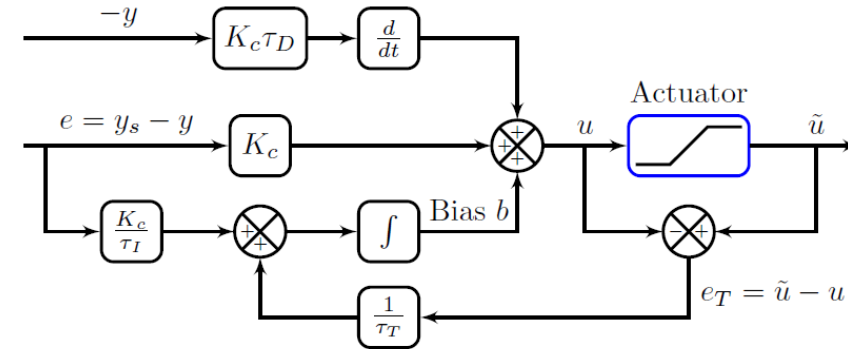


Figure 7: Recommended PID-controller implementation with anti-windup using tracking of the actual controller output (\tilde{u}), and without D-action on the setpoint. (Åström & Hägglund, 1988).

With anti-windup using tracking:

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

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

Anti-windup with cascade control

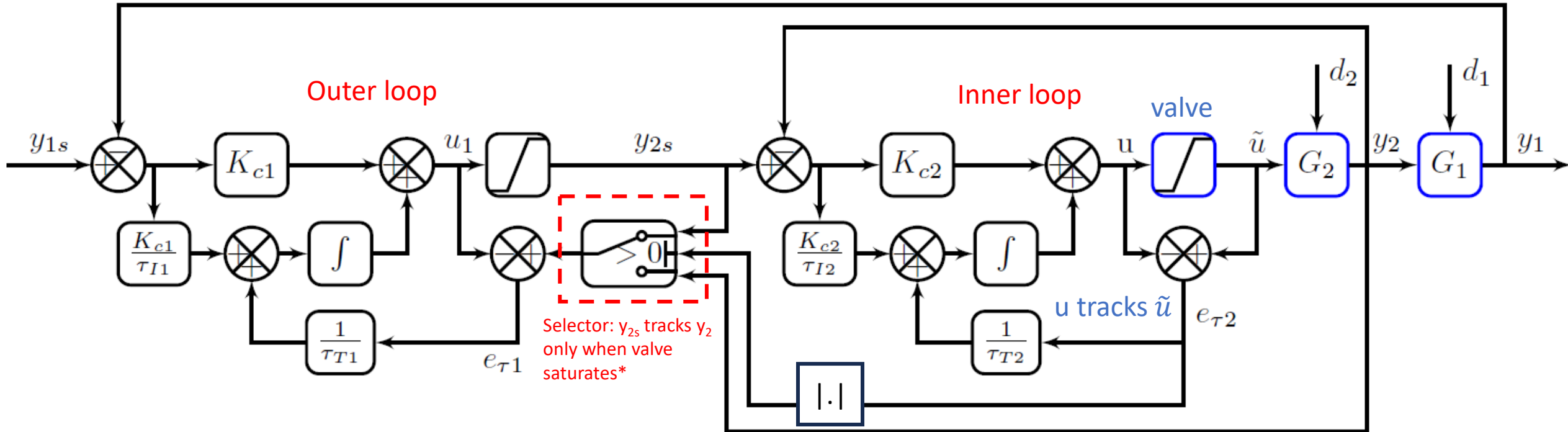


Figure 25: Cascade control with anti windup using the industrial switching approach (Leal et al., 2021).

* 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.

E9. Two degrees-of-freedom control

- One degree-of freedom control: Controller uses $e=y_s-y$
- Two degrees-of freedom control: y and y_s used differently.
 - For example, no derivative action on setpoint,
 - More generally, setpoint filter F_s :

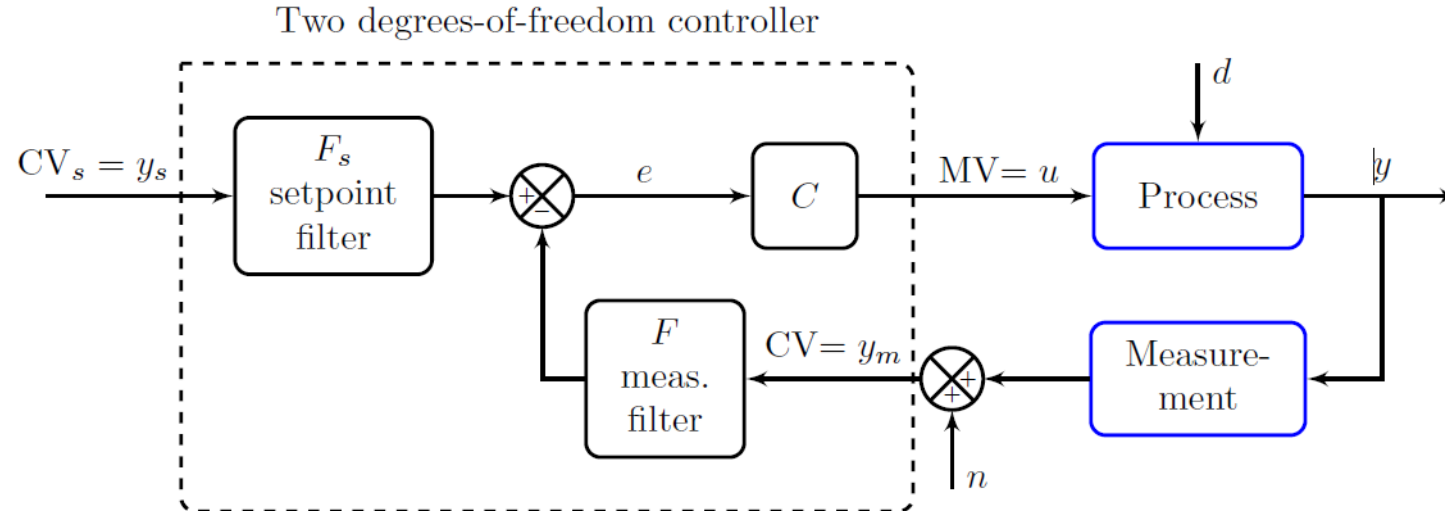
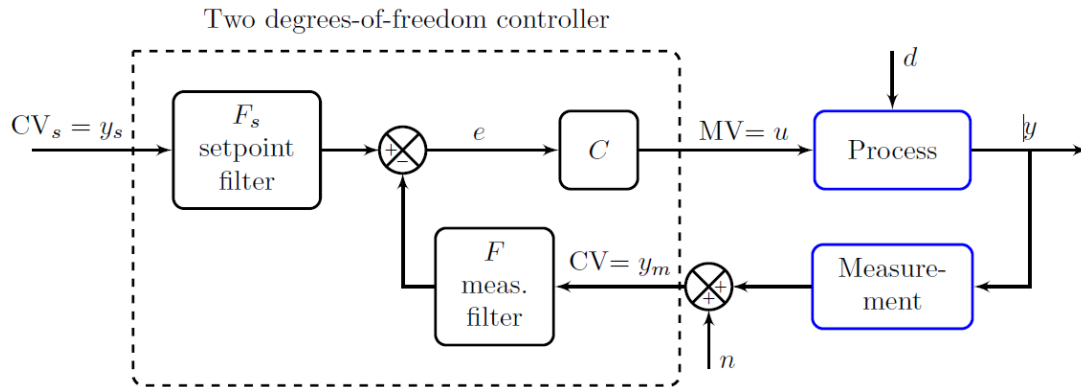


Figure A.41: Two degrees-of-freedom control system with setpoint filter F_s and measurement filter F . All blocks are possibly nonlinear.

Measurement filter $F(s)$



- Very common, especially with noisy measurements
 - Used also alone (without F_s)
 - Most common: First-order filter

$$F(s) = \frac{1}{\tau_F s + 1}$$

Recommended: $\tau_F \leq \frac{\tau_c}{2}$ (preferably smaller, typically $\tau_F = 0.1 \tau_c$)

- τ_c : Closed-loop time constant (SIMC)

Here τ_F is the measurement filter time constant, and the inverse ($\omega_F = 1/\tau_F$) is known as the cutoff frequency. However, one should be careful about selecting a too large filter time constant τ_F as it acts as a effective delay as seen from the controller C .

King (2011) (page xii) writes in this respect: “Many engineers are guilty of installing excessive filtering to deal with noisy measurements. Often implemented only to make trends look better they introduce additional lag and can have a detrimental impact on controller performance.” To reduce the effective delay (lag) introduced by filtering, Sigifredo Nino (personal email communication, 30 March 2023), who has extensive industrial experience, suggests using a second-order Butterworth filter,

$$F(s) = \frac{1}{\tau_F^2 s^2 + 1.414 \tau_F s + 1} \quad (\text{A.4})$$

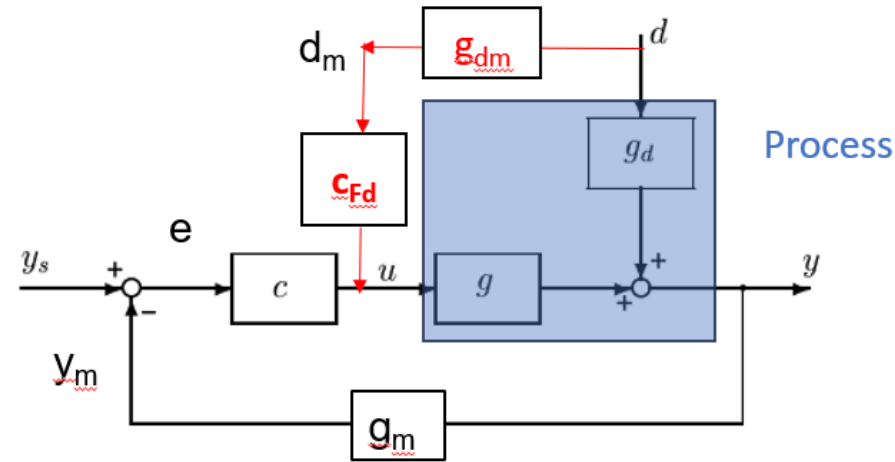
E10. Gain scheduling

- Very popular for PID within EE and ME, e.g., airplanes, automotive.
- Controller (PID) tunings change as a given function of the scheduling variable, e.g.,
 - disturbance d
 - process input u
 - process output y
 - setpoint y_s
 - control error $e=y_s-y$

E11. Feedforward control

Linear feedforward, $u = C_{Fd}d_m + c e$

If perfect measurements: $g_m=1, g_{dm}=1$



Ideal Feedforward (with no feedback, $c=0$): Want $y=0$.

In the linear case

$$y = g_d d + g u = (g_d + g C_{Fd} g_m) d$$

To get $y=0$, the ideal dynamic feedforward controller (if realizable) inverts the process:

$$\text{Linear: } C_{Fd,ideal}(s) = g^{-1} g_d g_m^{-1}$$

Annual Reviews in Control 56 (2023) 100903

Nonlinear static feedforward based on input transformations (with setpoint v_0 from feedback):

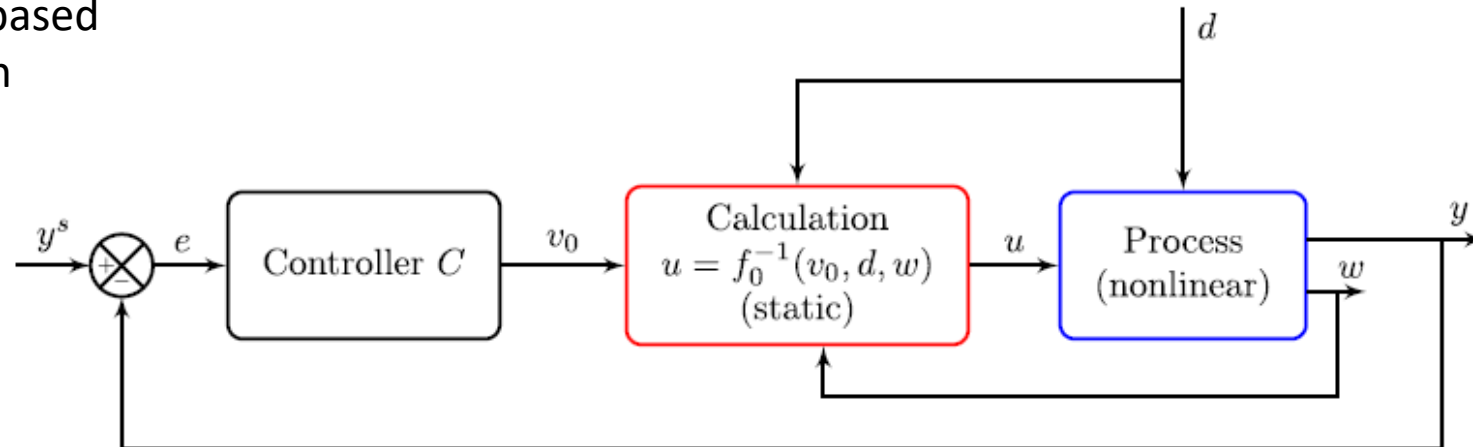


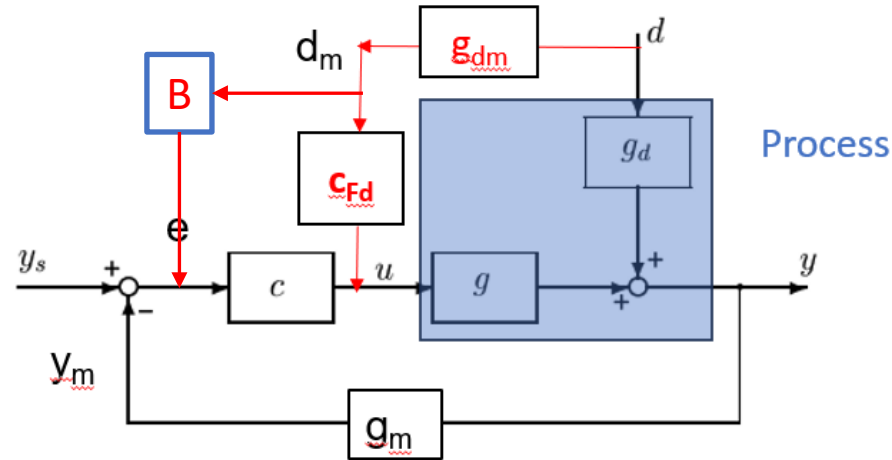
Fig. 29. Feedforward, decoupling and linearization (red calculation block) using transformed inputs $v_0 = f_0(u, d, w)$ based on static model $y = f_0(u, d, w)$. In the ideal case with no model error, the transformed system from v_0 to y (as seen from the controller C) becomes $y = I v_0$ at steady state.

d = measured disturbance

w = measured process state variable.

Main problem feedforward: Sensitive to model error and changes (nonlinearity)

Disturbances: Avoid fighting of feedforward and feedback (with B)



In cases where feedforward is not perfect (typically because of a delay in g), feedback may try to correct for temporary deviations in y (which feedforward will handle, but it needs a little time). To avoid this fighting between feedforward (cF_d) and feedback (c), we want the transfer function (with feedforward control included) from (the measured) d to the feedback controller input (e) to be zero. So we want*

$$B * g_{dm} * d - g_m * (g_d + g * c_{Fd} * g_{dm}) * d = 0$$

Here g_m includes a possible measurement filter F . We get

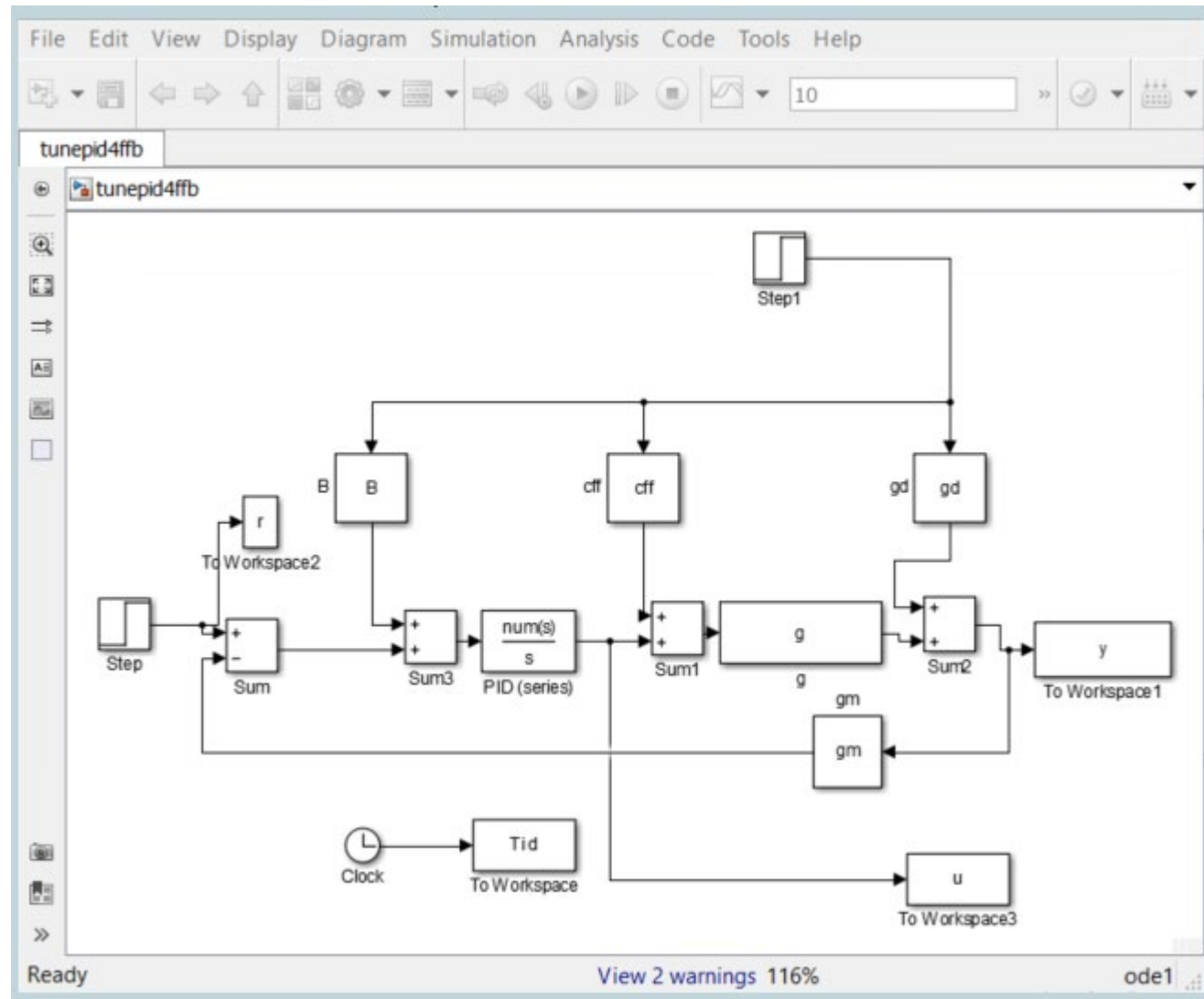
$$B = (g_d + g_{dm} * c_{Fd} * g) * g_m / g_{dm}.$$

This is usually realizable unless g_m has a large delay. Note that $(g_d + g_{dm} * c_{Fd} * g)$ is the expected response from d to y with feedforward.

With perfect feedforward it will be 0 and we get $B=0$.

* This idea was originally proposed by Lang and Ham (1955) for the case with combined feedback and feedforward from setpoints. The paper is referred to from D'azzo and Houpis in their 2nd edition from 1966, but not in the third from 1988. Åström and Hägglund also discuss this structure in great detail in their PID-book from 2006. Also see Guzman and Hägglund (2021) they use $H=B$ who refer to Brosilow and Joseph (2002).
Lang, G., and J. M. Ham. "Conditional feedback systems-A new approach to feedback control." *Transactions of the American Institute of Electrical Engineers, Part II: Applications and Industry* 74.3 (1955): 152-161.

- $g = \exp(-s)$, $gd = 1$, $gm = 1$
- $Cff = -1$
- Pure I-controller with $K_i = 0.5$ (SIMC)
- $B = (c_{ff} * g * g_{dm} + g_d) * g_m = 1 - \exp(-s)$



Setpoints: Avoid fighting of feedforward and feedback (with B)

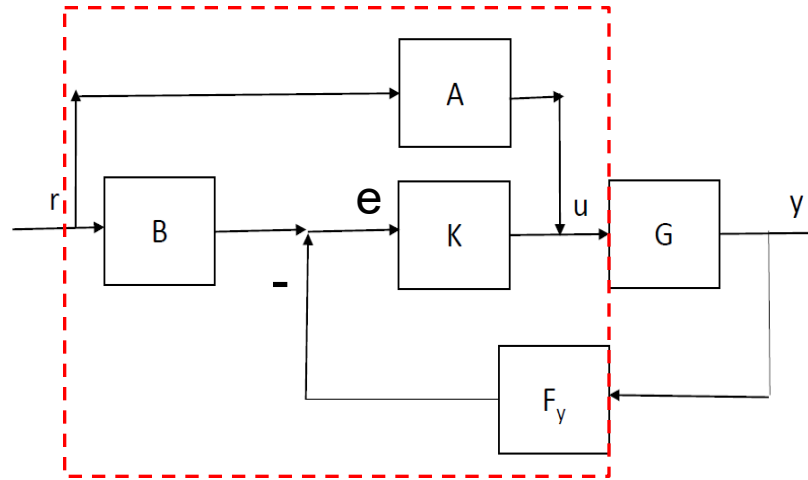


Fig. 3. Two degrees-of-freedom controller with feedforward controller A and prefilter B

- Typically, for a setpoint change r , the feedforward block is $A = G_-^{-1}F_r$ where G_- is the invertible part of G .
 - A typical choice for the prefilter is $F_r = 1/(\tau_r s + 1)$
 - Example. $G = (-3s+1)\exp(-4s)/(7s+1)^2$, $F_v=1$ (perfect measurement) and $\tau_r=2$. Use "IMC-like" design and write $G = G_+ G_-$ where $G_+ = \exp(-4s)(-3s+1)/(3s+1)$ and $G_- = (3s+1)/(7s+1)^2$. Gives $A = (7s+1)^2/(3s+1)(2s+1)$
- We want to choose B such that **A and K can be designed independently!!**
 - **Solution*** (Lang and Ham, 1955): Choose $B = F_y G_+$ so that transfer function from r to the controller input (e) is zero (with perfect model) !!
 - The feedback will then only take action if the feedforward is not working as expected (due to model error).
 - We must have $B(0)=I$ so that we will have no offset ($y=r$ at steady state) even with model error for G
 - Example. $B(s) = F_y G_+ = F_y G_+ F_r = \exp(-4s)(-3s+1)/(3s+1)(2s+1)$. Note that $B(0)=1$.
- The feedback controller K can be designed for disturbance rejection and robustness, e.g., using SIMC rules.
 - Example. Approximate as first-order with delay process with $\theta=4+3+7/2=10.5$ and $\tau_1=7+7/2=10.5$, and use SIMC! With $\tau_{cfc}=\theta$ get $K_c=0.5$ and $\tau_{i1}=10.5$.
- The same approach applies to disturbances ($d=r$) with a feedforward controller C_{ff} and measurement G_{dm} , but then $G A$ is replaced by $(G C_{ff} G_{dm} + G_d)$. So we get $B = F_y (G C_{ff} G_{dm} + G_d)$. Note that $F_y = G_m$ in many cases. See next slide

*See also my note on Two degrees in moka/sis Krister and...

E12. Linear decoupling

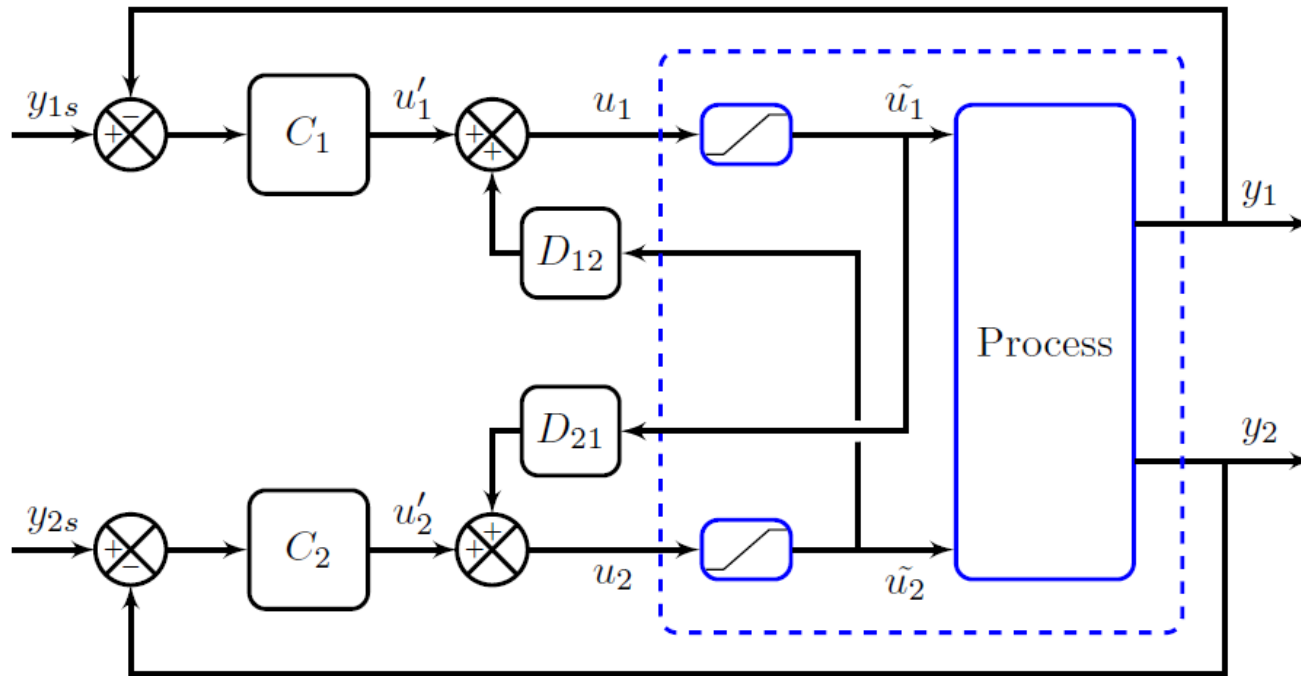


Figure 26: Linear decoupling with feedback (reverse) implementation of Shinskey (1979)

$$G = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \quad D_{12} = -\frac{G_{12}}{G_{11}}, \quad D_{21} = -\frac{G_{21}}{G_{22}}$$

To make the decoupling elements realizable, we need a larger (effective) delay in the off-diagonal elements than in the diagonal elements of G . This means that the “pair close” rule should be followed also when using decoupling. An alternative is to use static decoupling or partial (one-way) decoupling.

Note that Figure 26 uses the feedback decoupling scheme of Shinskey (1979) which is called inverted decoupling (Wade, 1997). Compared with the more common “feedforward” scheme (where the input to the decoupling elements is u' rather than \tilde{u}), the feedback decoupling scheme in Figure 26 has the following nice features (Shinskey, 1979):

1. With inverted decoupling, the model from the controller outputs (u') to the process outputs (y) becomes (assuming no model error) $y_1 = G_{11}u'_1$ and $y_2 = G_{22}u'_2$. Thus, the system, as seen from the controllers C_1 and C_2 , is in addition to being decoupled (as expected), also identical to the original process (without decoupling). This simplifies both controller design and switching between manual and auto mode. In other words, the tuning of C_1 and C_2 can be based on the open loop models (G_{11} and G_{22}).
2. The inverted decoupling works also for cases with input saturation, because the actual inputs (\tilde{u}) are used as inputs to the decoupling elements.

Note that there is potential problem with internal instability with the inverted implementation because of the positive feedback loop $D_{12}D_{21}$ around the two decoupling elements. However, this will not be a problem if we can follow the “pair close” pairing rule. In terms of the relative gain array (RGA), we should avoid pairing on negative RGA-elements. To avoid the stability problem (and also for other reasons, for example, to avoid sensitivity to model uncertainty for strongly coupled processes) one may use one-way decoupling where one of the decoupling elements is zero. For example, if tight control of y_2 is not important, one may select $D_{21} = 0$.

E13. Linearization elements

- Typically, logarithm or nonlinear feedforward blocks
- General approach: See Input transformations

E14. Calculation block based on transformed input


- SEE LATER

E11. Simple static estimators

- Inferential element
- Soft sensor

Additional standard elements

- **E16.** Simple nonlinear static elements

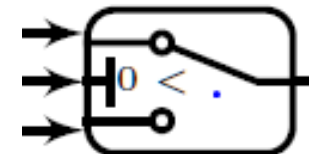
- Multiplication
- Division (avoid or at least be careful)
- Square root
- Dead zone
- Dead band
- Limiter (saturation element) 
- On/off

- **E17.** Simple linear dynamic elements

- Lead-lag filter
- Time delay
- ... more...

- **E18.** Standard logic elements

- If, then, else
- Example: Select depending on sign of another signal:



What about the Smith Predictor? Forget it!

Note that the Smith Predictor (Smith, 1957) is not included in the list of 18 control elements given in the Introduction, although it is a standard element in most industrial control systems to improve the control performance for processes with time delay. The reason why it is not included, is that PID control is usually a better solution, even for processes with a large time delay (Grimholt & Skogestad, 2018b; Ingimundarson & Hägglund, 2002). The exception is cases where the true time delay is known very accurately. There has been a myth that PID control works poorly for processes with delay, but this is not true (Grimholt & Skogestad, 2018b). The origin for the myth is probably that the Ziegler–Nichols PID tuning rules happen to work poorly for static processes with delay.

The Smith Predictor is based on using the process model in a predictive fashion, similar to how the model is used in internal model control (IMC) and model predictive control (MPC). With no model uncertainty this works well. However, if tuned a bit aggressively to get good nominal performance, the Smith Predictor (and thus also IMC and MPC) can be extremely sensitive to changes in the time delay, and even a *smaller* time delay can cause instability. When this sensitivity is taken into account, a PID controller is a better choice for first-order plus delay processes (Grimholt & Skogestad, 2018b).

A smart invention: Cross-limiting control

Industry also makes use of other smart solutions, which do not follow from the standard structures presented in this paper.

One example is cross-limiting control for combustion, where the objective is to mix air (A) and fuel (F) in a given ratio, but during dynamic transients, when there will be deviations from the given ratio, one should make sure that there is always excess of air. The scheme in Fig. 39 with a crossing min- and max-selector achieves this. It is widely used in industry and is mentioned in many industrial books (e.g., Liptak (1973), Nagy (1992) and Wade (2004)). The setpoint for the ratio, $(F_F/F_A)_s$, could be set by a feedback controller (not shown) which controls, for example, the remaining oxygen after the combustion.

The selectors in Fig. 39 are used to handle the dynamic (transient) case, so this is a somewhat rare case where the selectors are not performing a steady state CV-CV switch.

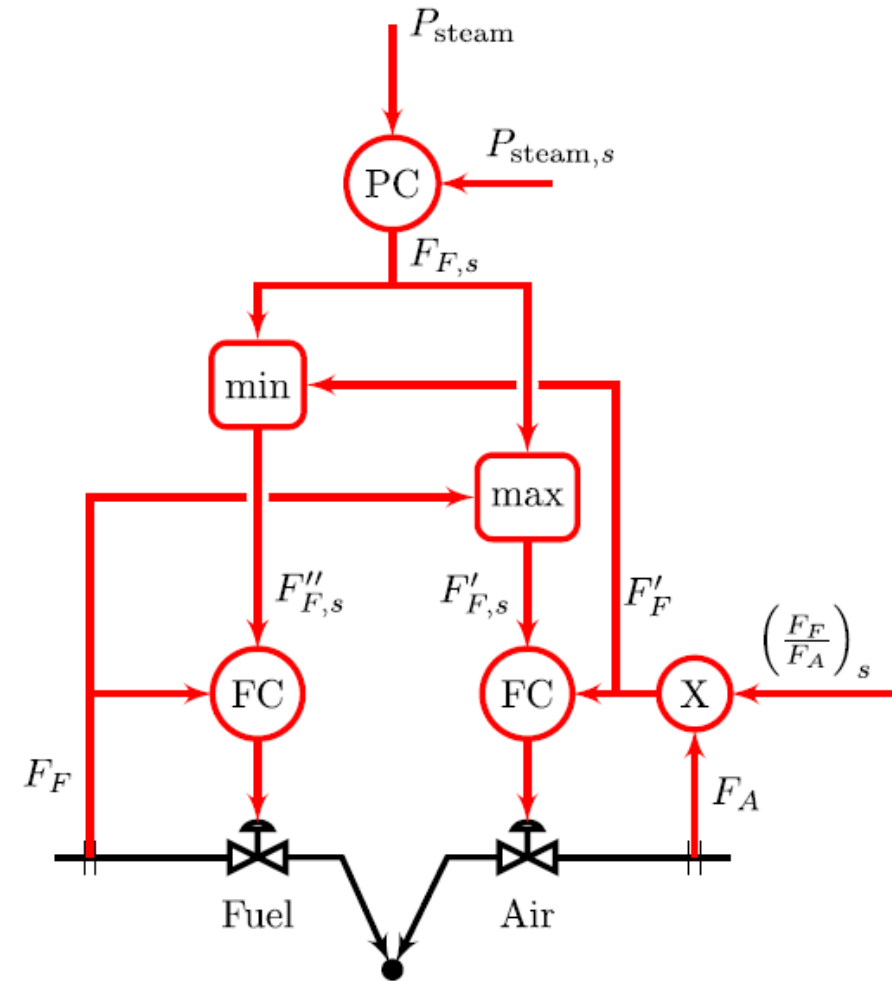


Fig. 39. Cross-limiting control for combustion where air (A) should always be in excess to fuel (F).

Standard advanced control elements studied in this paper.

Control element	Main use	Inputs	Outputs
E1. Cascade control Figs. 9 and 10	Linearization and local disturbance rejection	Outer master controller: • CV_{1s} -CV Inner controller: • CV_{2s} - CV_2	Outer master controller: • CV_{2s} Inner controller: • MV
E2. Ratio control Fig. 11	Feedforward or decoupling without model (assumes that scaling property holds)	• R (desired ratio) • DV or MV_1	• $MV = R \cdot DV$, or • $MV_2 = R \cdot MV_1$
E3. VPC on extra dynamic input Fig. 12	Use extra dynamic input MV_1 to improve dynamic response (because MV_2 alone is not acceptable). MV_1 setpoint is unconstrained (mid-range) and controlled all the time	• $MV_{1s} - MV_1$	• MV_2
E4. Selector Figs. 17, 18 and 19	CV-CV switching: Many CVs (CV_1, CV_2, \dots) controlled by one MV	• MV_1 • MV_2, \dots (generated by separate controllers for CV_1, CV_2, \dots)	• $MV = \max/\min (MV_1, MV_2, \dots)$
E5. Split-range control Figs. 21 and 23	MV-MV switching: One CV controlled by sequence of MVs (using only one controller)	• CV_s -CV	• MV_1 • MV_2, \dots
E6. Separate controllers with different setpoints Fig. 22	MV-MV switching: One CV controlled by sequence of MVs (using individual controllers with different setpoints)	• $CV_{s1} - CV$ • $CV_{s2} - CV$...	• MV_1 • MV_2 ...
E7. VPC on main steady-state input Fig. 24	MV-MV switching: One CV controlled by main MV_1 with use of extra MV_2 to avoid saturation of MV_1 . MV_1 setpoint is close to constraint and only controlled when needed	• $MV_{1s} - MV_1$	• MV_2
E9. Two degrees-of-freedom feedback controller Fig. A.41	Treat setpoint (CV_s) and measurement (CV) differently in controller C	• CV_s • CV	• MV
E11. Feedforward control Fig. A.42	Reduce effect of disturbance (using model from DV and MV to CV)	• DV	• MV
E12. Decoupling element Fig. 26	Reduce interactions (using model from MV_1 and MV_2 to CV)	• MV_1 • MV_2	• MV_2 • MV_1
E14. Calculation block based on transformed input Fig. 27	Static nonlinear feedforward, decoupling and linearization based on nonlinear model from MV, DV and w to CVv	• Transformed input = feedback trim (v) • DV (d) • Extra meas. (w)	• MV (u)

Comment on need for rules

- The human brain (at least mine) has problems in analyzing even quite simple cases
- Two «simple» cases are:
 - choice of max- and min- selectors
 - how to get consistent inventory control
- I frequently need to go back to the «selector rules» or the «radiation rule» to get this right.