Part 3: Constraint switching. Standard control elements

Standard Advanced control elements

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

E1*. Cascade control[2]
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

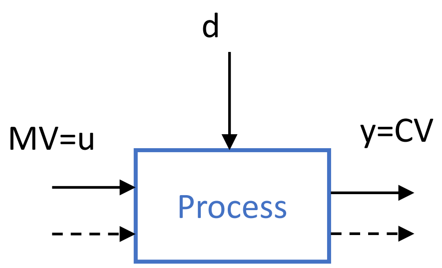
E16. Simple nonlinear static elements (like multiplication, division, square root, dead zone, dead band, limiter (saturation element), on/off)
E17*. Simple linear dynamic elements (like lead–lag filter, time delay, etc.)
E18. Standard logic elements

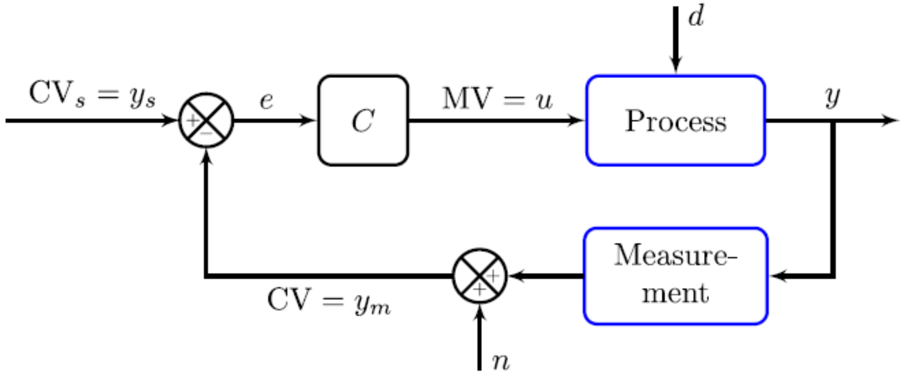
Gives a decomposed control system:

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

Introduction to pairing and switching

Most basic element: Single-loop PID control





MV-CV Pairing. Two main rules:

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)*

Additional rule:

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.*

Need to control active constraints
But active constraints may change during operation

Four cases:

- A. MV-MV switching
- B. CV-CV switching
- MV-CV switching
  - C. Simple (if we follow input saturation rule)
  - D. Complex (combine MV-MV and CV-CV)

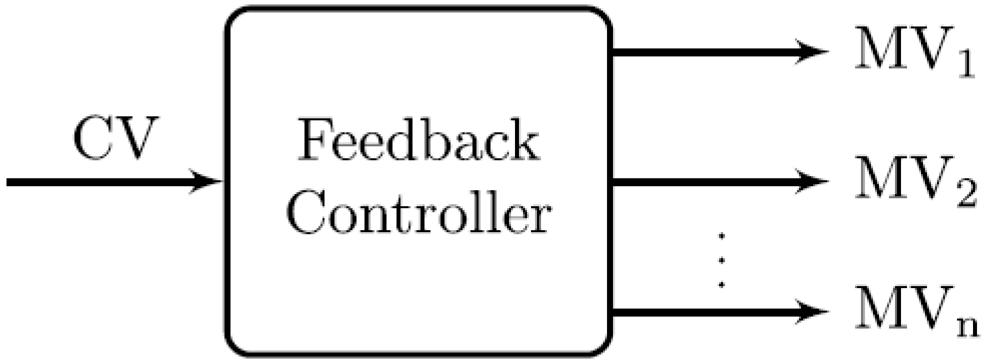


Fig. 5. MV-MV switching is used when we have multiple MVs to control one CV, but only one MV should be used at a time. The block "feedback controller" usually consists of several elements, for example, a controller and a split range block.

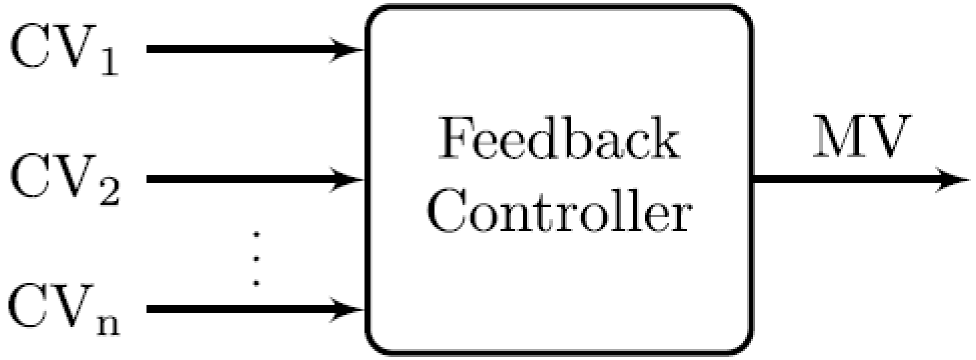


Fig. 6. CV-CV switching is used when we have one MV to control multiple CVs, but the MV should control only one CV at a time. The block "feedback controller" usually consists of several elements, typically several PID-controllers and a selector.

A. MV-MV switching

- Need several MVs to cover whole <u>steady-state</u> range (because primary MV may saturate)*
- Note that we only want to use one MV at the time.

Three main solutions for "selecting the right MV":

Alt.1: (Standard) Split-range control (SRC) (one controller)

Alt 1': Generalized SRC (many controllers)

Alt.2 Many controllers with different setpoints

Alt.3 Valve position control

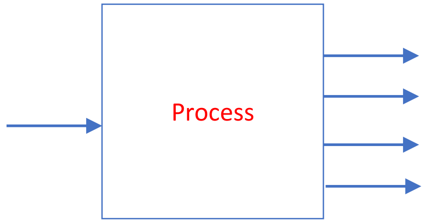
In addition: MPC

Which is best? It depends on the case!

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

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)

B. CV-CV switching



- One MV
- Many CVs, but control only one at a time
- Solution: Selector

The four 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

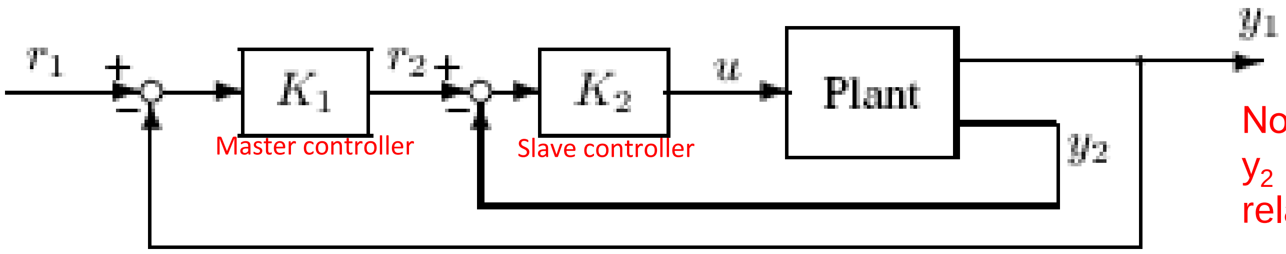
Adriana Reyes-Lua and Sigurd Skogestad, Systematic Design of Active Constraint Switching Using Classical Advanced Control Structures, Ind.Eng.Chem.Res, 2020

Standard advanced control elements

- E1-E18

E1. Cascade control

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")

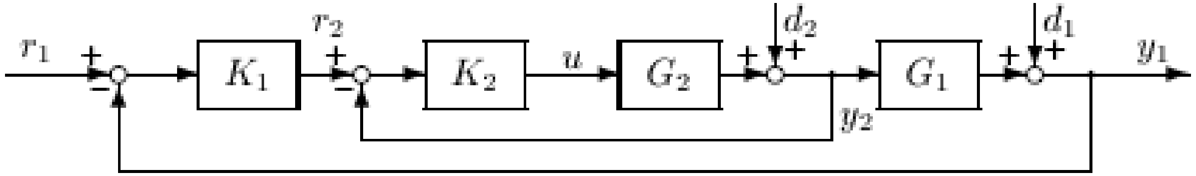
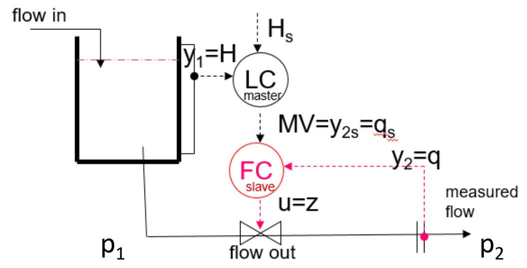


Figure 10.11: Common case of cascade control where the primary output y_1 depends directly on the extra measurement y_2

Block diagram flow controller



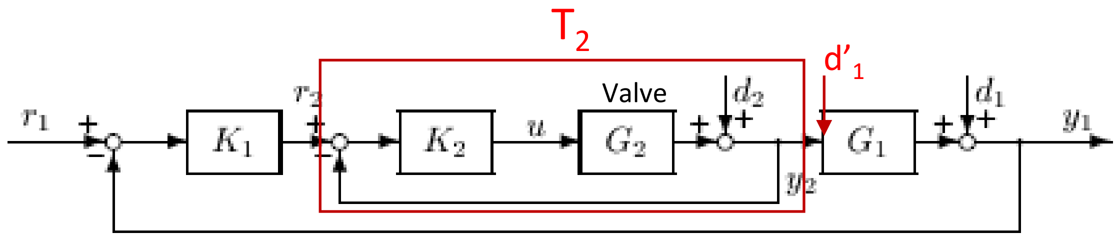
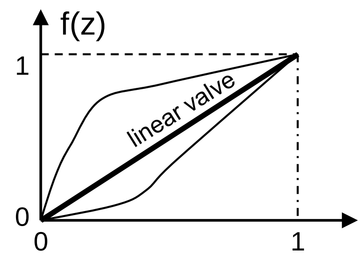


Figure 10.11: Common case of cascade control where the primary output y_1 depends directly on the extra measurement y_2

Example: Level control with slave flow controller:

u = z (valve position, flow out)
y_1 = H
y_2 = q
d'_1 = 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 = Level controller (master)
K_2 = Flow controller (slave)



k(z) = slope df/dz

More about cascade control

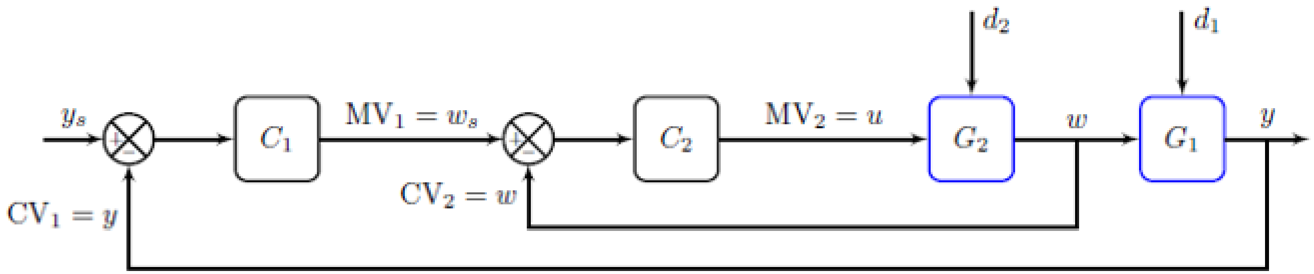


Figure 10: Cascade control for series process where the objective is to control y and w is an intermediate measurement. All blocks are possibly nonlinear.

- C_1=primary/outer controller (slow), G_1 =primary process
- C_2=secondary/inner controller (fast). G_2 =secondary process

An early and very good description of the benefits of cascade control is given by Shinskey (1967). With reference to Fig. 10, he writes (page 154):

The principal advantages of cascade control are these:

1. Disturbances arising within the secondary loop (d_2) are corrected by the secondary controller (C_2) before they can influence the primary variable (y).
2. Phase lag existing in the secondary part of the process (G_2) 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 (G_2) are overcome within its own loop.
4. The secondary loop permits an exact manipulation of the flow of mass or energy (w) by the primary controller.

As given by the rule of thumb in Section 2.5, the time scale separation τ_{c1}/τ_{c2} between the loops should typically be between 4 and 10. A larger time separation helps to protect against process gain variations in both the inner and outer loops, but it "eats up" more of the available time window.

The third advantage is related to the important linearizing effect of "high-gain" feedback, which is usually not mentioned in control textbooks. Specifically, consider the inner loop in Fig. 10 with a feedback controller C_2 and process model G_2. For the linear case, the inner loop transfer function is $L_2 = G_2C_2$ and the closed-loop response from the setpoint w_s to the output w becomes $w = T_2w_s$ with

$$T_2 = L_2(I + L_2)^{-1}$$

Without the inner loop, the process transfer function (from u to y) is G_1G_2. However, with the inner loop closed, the transformed process (from w_s to y) for tuning the outer controller C_1 becomes G_1T_2. With high-gain feedback in C_2, we get $\|L_2\| \gg 1$ and we have $T_2 \approx I$ (perfect linear response), or equivalently $w \approx w_s$, independent of the model G_2. Thus, we have the (seemingly incredible) fact that the response is independent of the model G_2, so it does not matter if G_2 varies, for example, due to nonlinearity. A typical example is when G_2 is a valve with a nonlinear gain characteristic, u is the valve position and w is the flow measurement. However, it should be noted that gain variations in G_2 translate into changes in the dynamics (response time) in T_2. This illustrated in Appendix B.2 where we find that a process gain increase of 50% translate into a corresponding reduction in the closed-loop time constant τ_{c2} (from 4 s to 4/1.5=2.67 s for the specific example).

When use (series) cascade ?

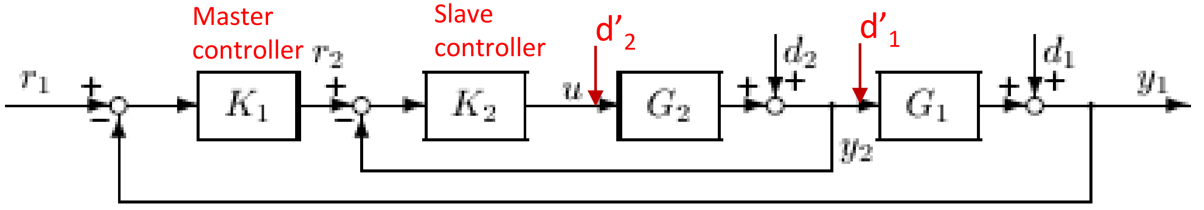


Figure 10.11: Common case of cascade control where the primary output y_1 depends directly on the extra measurement y_2

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_2' 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. Integrating dynamics (including slow dynamics or unstable) in both G_1 and G_2, (because without cascade double integrating plant G_1G_2 is difficult to control)
4. Measurement delay for y_1
   - **Note: In the flowsheet above, y1 is the measured output, so any measurement delay is included in G1**

Design / tuning

- First design K_2 ("fast loop") to deal with d_2 and d_2'
- Then design K_1 to deal with d_1 and d_1'

Transfer functions and tuning

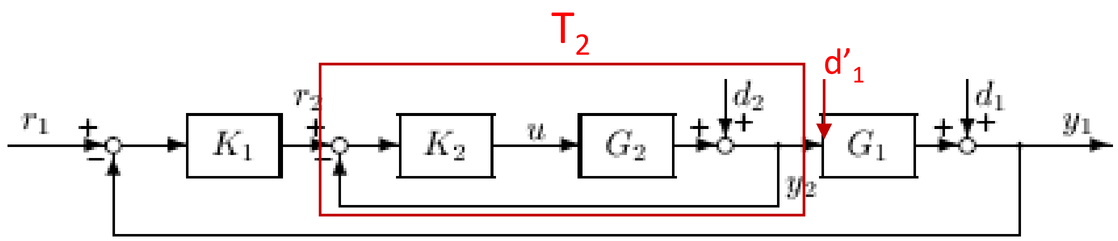


Figure 10.11: Common case of cascade control where the primary output y_1 depends directly on the extra measurement y_2

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: $T_2 \sim e^{-\Theta 2s}/(\tau_{c2}s+1)$

Nonlinearity: Gain variations (in G_2) translate into variations in time constant τ_{C2}

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

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

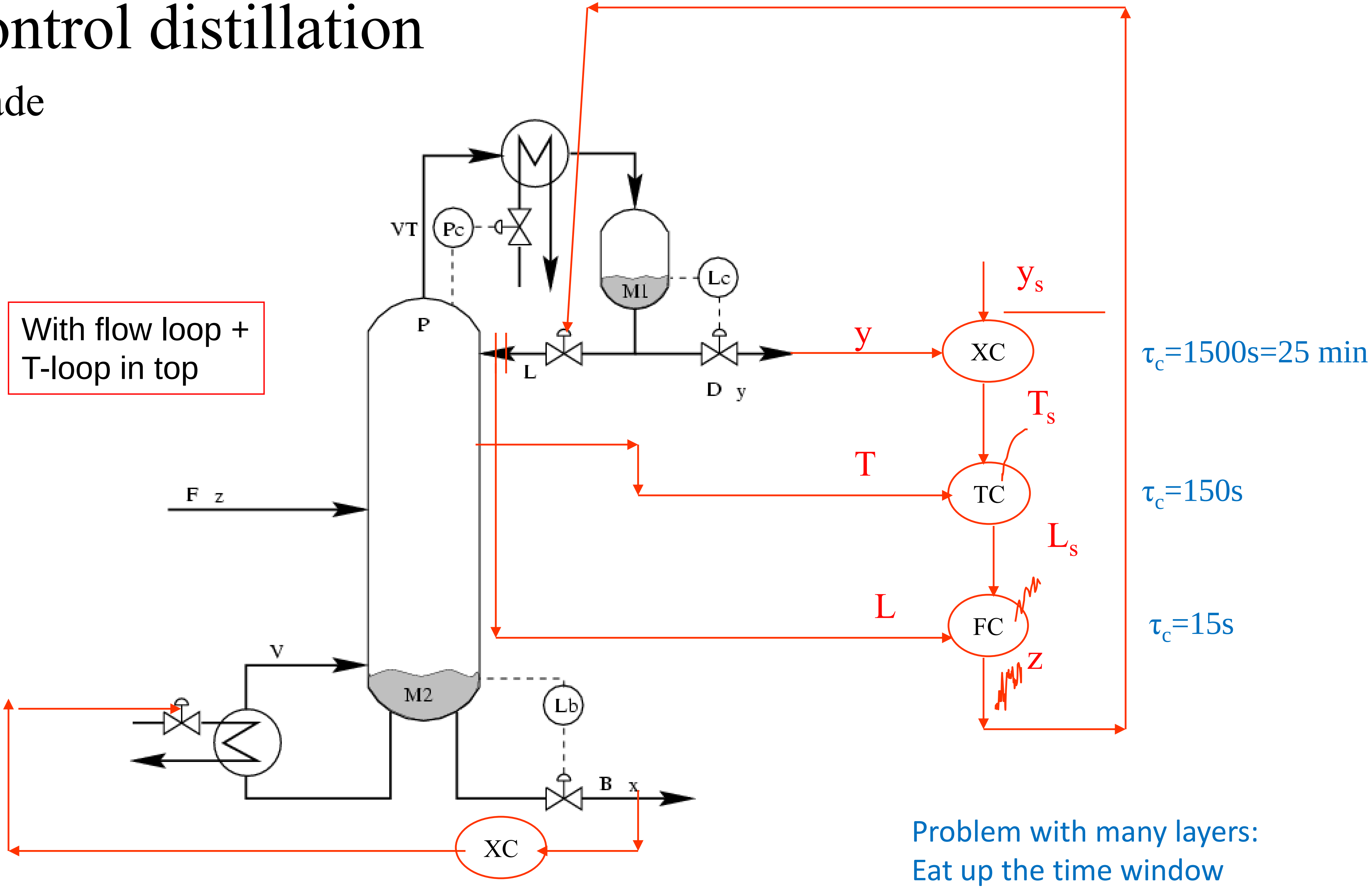
Can often set T2=1 if inner loop is fast!

Typical choice: $\tau_{c1} = 10\ \tau_{c2}$

Cascade control distillation

3 layers of cascade

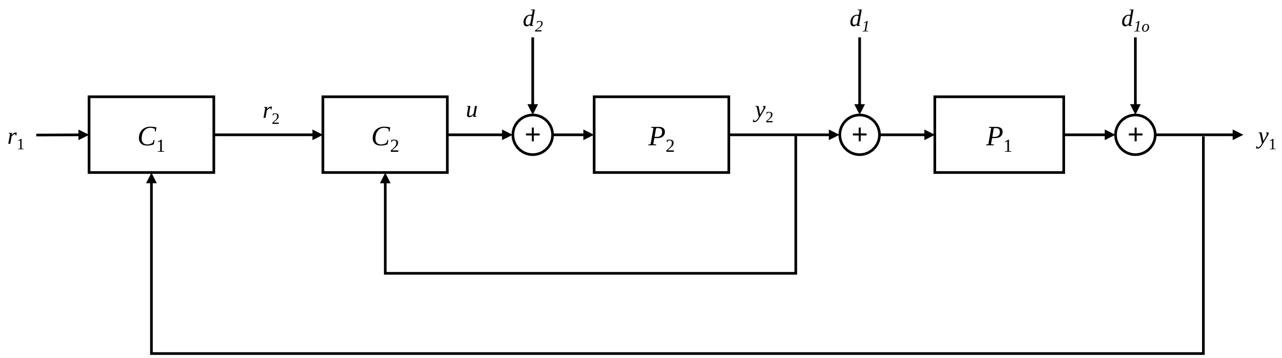
With flow loop +
T-loop in top



Problem with many layers:
Eat up the time window

Cascade control block diagram

- Which disturbances motivate the use of cascade control?

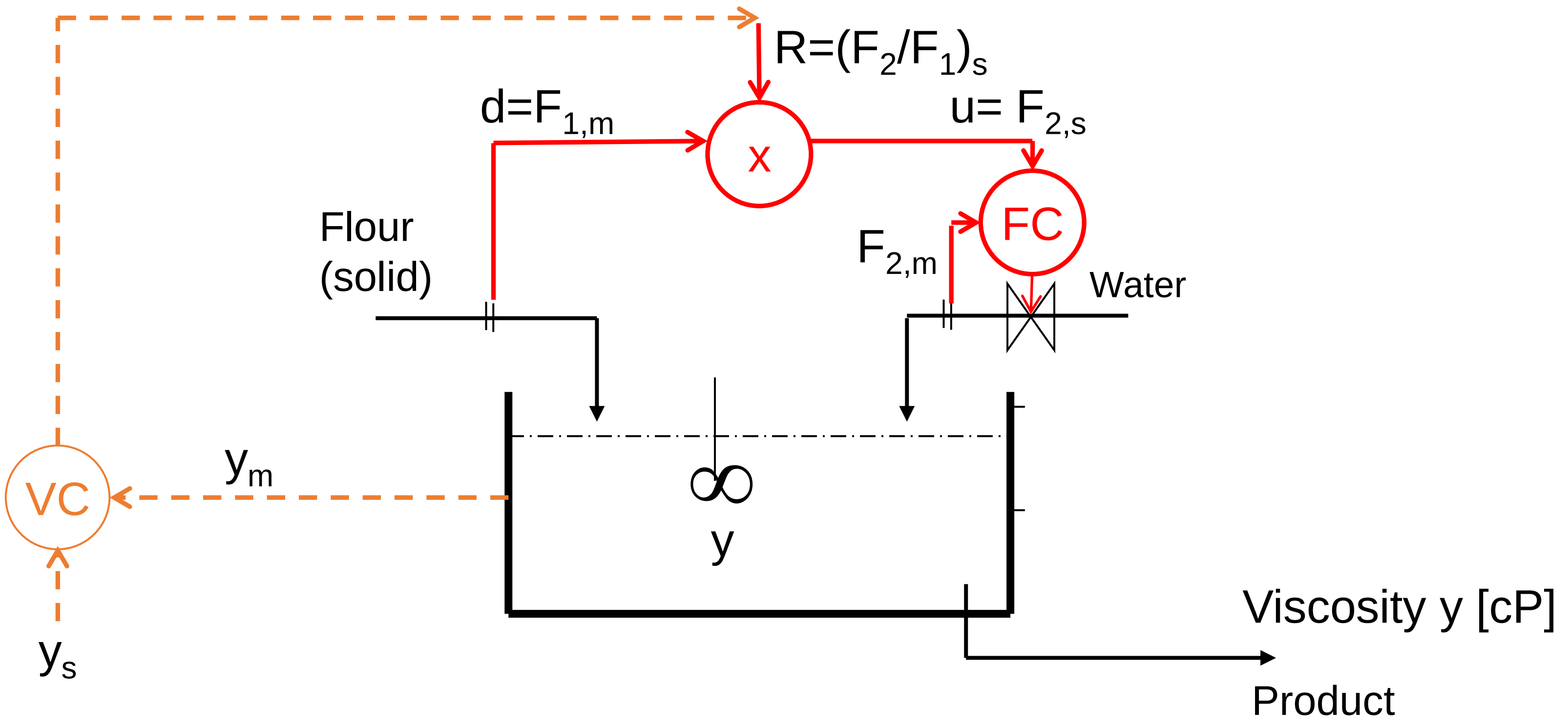


Answer: d_2

E2. Ratio control

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:

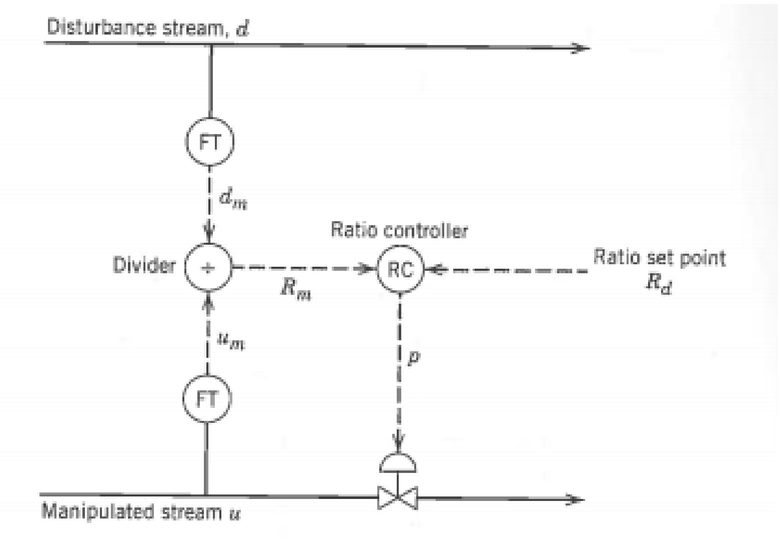


Figure 14.5 Ratio control, Method I.

Bad solution

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

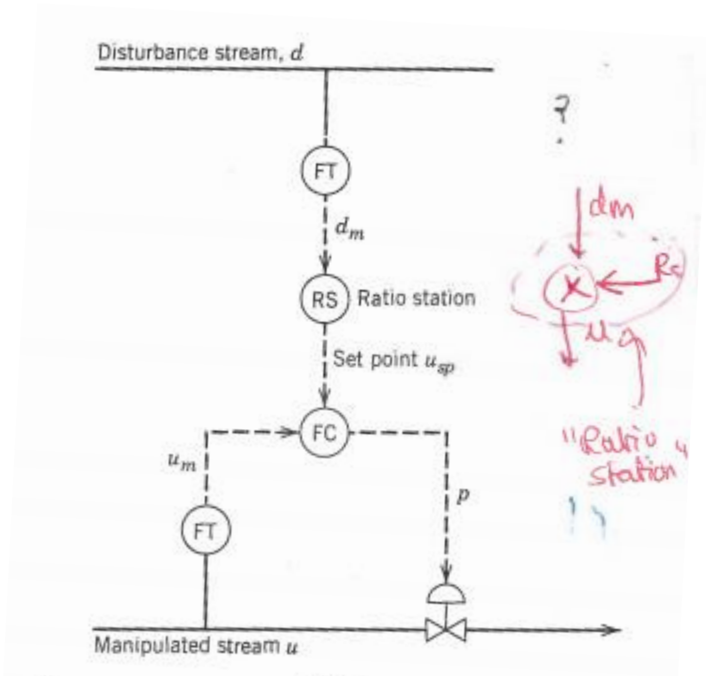


Figure 14.6 Ratio control, Method II.

This is complicated. What is RS?

Ok if implemented as shown in red at right

Ratio control

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

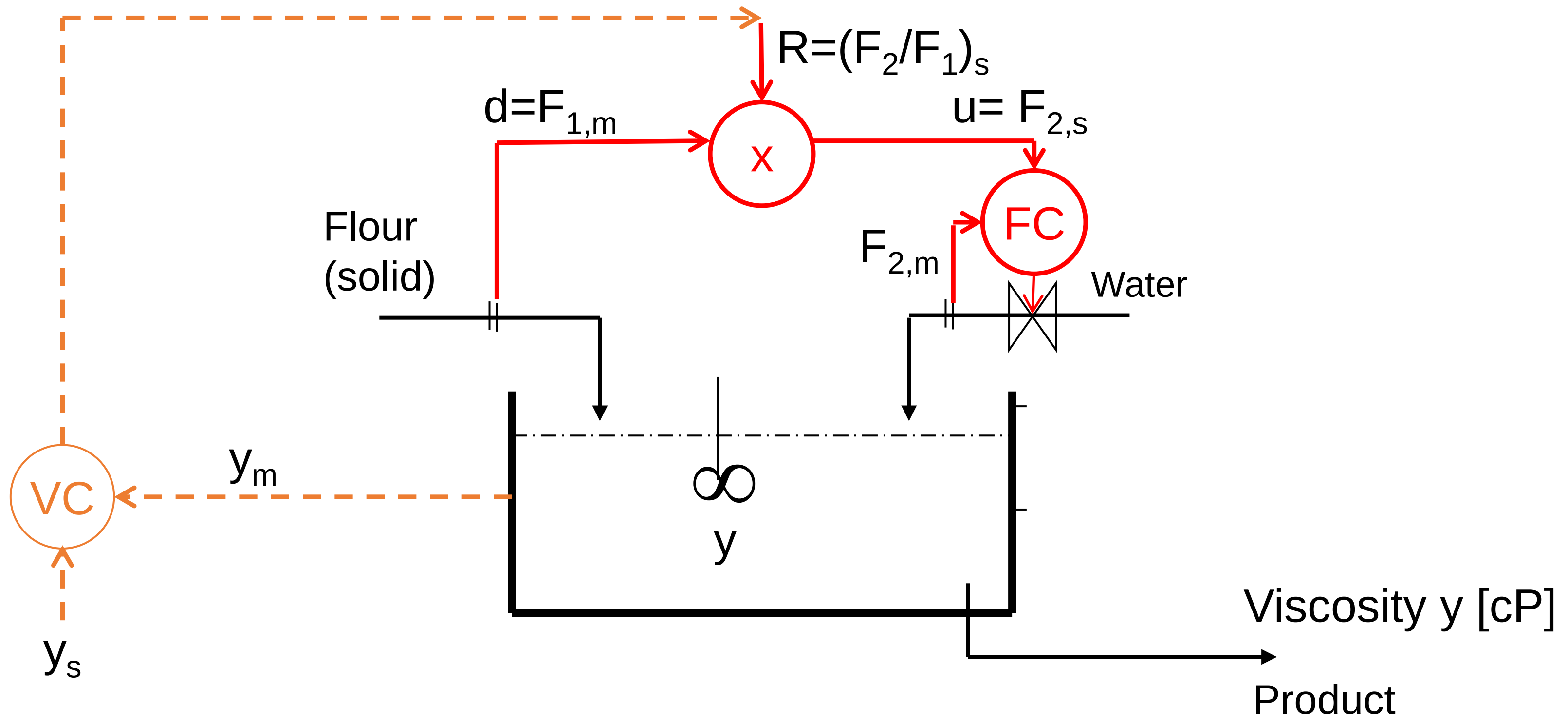
Theoretical basis of ratio control

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 require an explicit model for y. For a mixing process, the "scaling property" or "scaling assumption" says 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.

LINEARITY OF RATIO CONTROL when done correctly



Note : This way of implementing ratio control makes it easy to tune the outer feedback loop (CC: composition controller) because the gain from MV = R_s to CV=y does not depend on disturbance d=F_1.

Proof of last statement

- “Note : <u>This way </u>of implementing ratio control makes it easy to tune the outer feedback loop (CC: composition controller) because the gain from MV = $(q2/q1)_s$ to CV=c does not depend on disturbances in q1.”
- One may think that the last statement is fairly obvious, because we are talking about just scaling all flowrates by the same factor and then the composition c should remain constant. But actually, I wrote the following in 2021 (and earlier).

  WRONG: “Potential problem for outer feedback loop (CC: composition controller): Gain from MV = $(q2/q1)_s$ to CV=c will vary because of multiplication with $q_{1,m}$. So outer loop must have robust tunings to get high gain margin (large tauc)”.
- In fact, it’s opposite, there are less gain variations when the outer controller manipulates $(q_2/q_1)_s$ than when it manipulates q_2 directly
- ***Proof.*** The component balance gives: $CV=c=(c_1q_1 + c_2q_2)/(q_1+q_2)$
- We are here considering disturbances in q1, so assume that c1 and c2 are constant.
- We also assume that there is an outer loop so that c remains constant. From the component balance we see that c=constant implies that at as we change q1 (disturbance) we will have that q1/q2=constant and also that R_1=q1/(q1+q2) = constant.
- With no ratio control: The gain from MV=q_2 to CV=c is:
  - K = (c2-c1)q1/(q1+q2)^2 = (c2-c1)R1/(q1+q2)
  - From the above argument K = constant/(q1+q2) so the gain K will change with operation, which will be a problem for the outer feedback controller (CC). Actually, we find that K=infinity when q=q1+q2 goes to zero, so we may get instability in the outer feedback loop at low flowrates.
- With ratio control: The gain from MV=$(q_2/q_1)_s$ to CV=c is:
  - K_r = (c2-c1)q1^2/(q1+q2)^2 = (c2-c1) R_1^2
  - From the above argument we have that R1=constant so we get Kr= constant independent of the value of the disturbance (q1)! So the outer loop always has the same gain and there no reason to be careful about the tunings.
- 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=q1.) In this case we get the problem with process gain variation for the feedback controller CC). So ratio control is the best!
- But note that we should not always use ratio control for flow disturbances; it only holds if you are controlling temperature or composition (which are intensive variables). If you are controlling an extensive variable like total flow or level then you should add or subtract the disturbance. To the right an example:
- Challenge to myself (Sigurd): prove this more generally using theory of 1) ratio control and 2) input transformations.

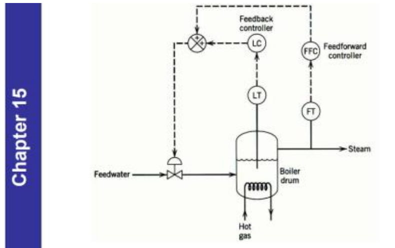


Figure 15.4 The feedfoward-feedback control of the boiler drum level.

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

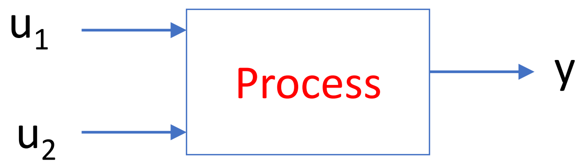
E3. VPC on extra dynamic input

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)

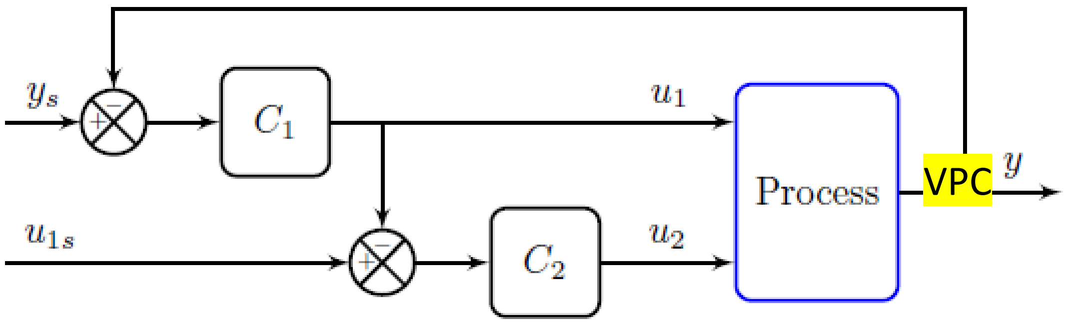


Figure 12: Valve (input) position control (VPC) for the case when an "extra" MV (u_1) is used to improve the dynamic response. A typical example is when u_1 is a small fast valve and u_2 is a large slower valve.
C_1 = fast controller for y using u_1.
C_2 = slow valve position controller for u_1 using 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 parallell) 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
- The small valve (u_1) has less stiction and gives good flow control, but it's too small to use alone

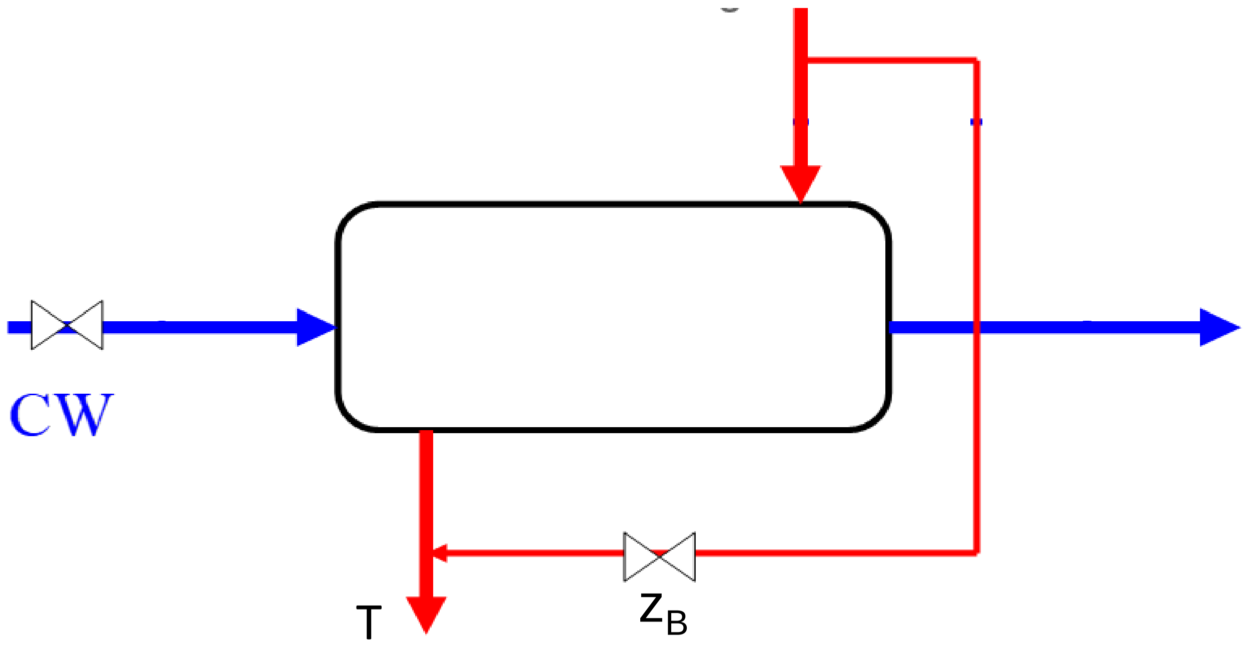
Example 2: Strong base (u_2) and weak base (u_1) for neutralizing acid (disturbance) to control y=pH

- 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. u1 controls the pH in the last tank (y)

Alternative term for dynamic VPC:

- Mid-ranging control (Sweden)

Example: Heat exchanger with bypass

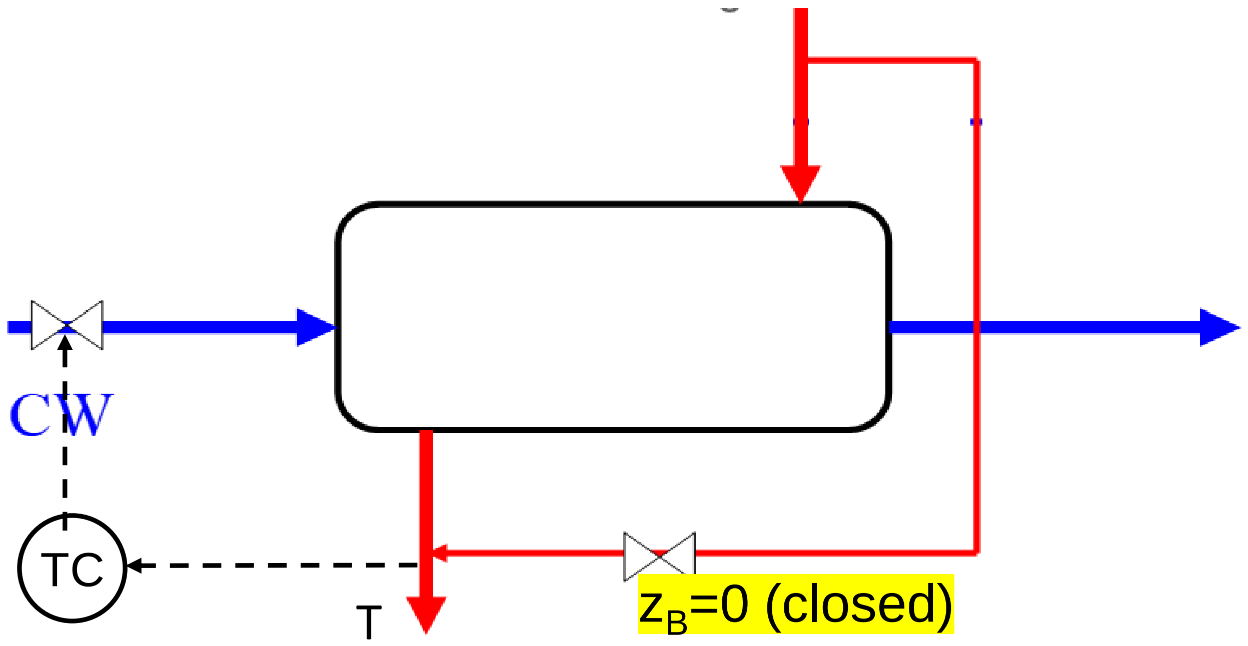


Want tight control of $y=T$.

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

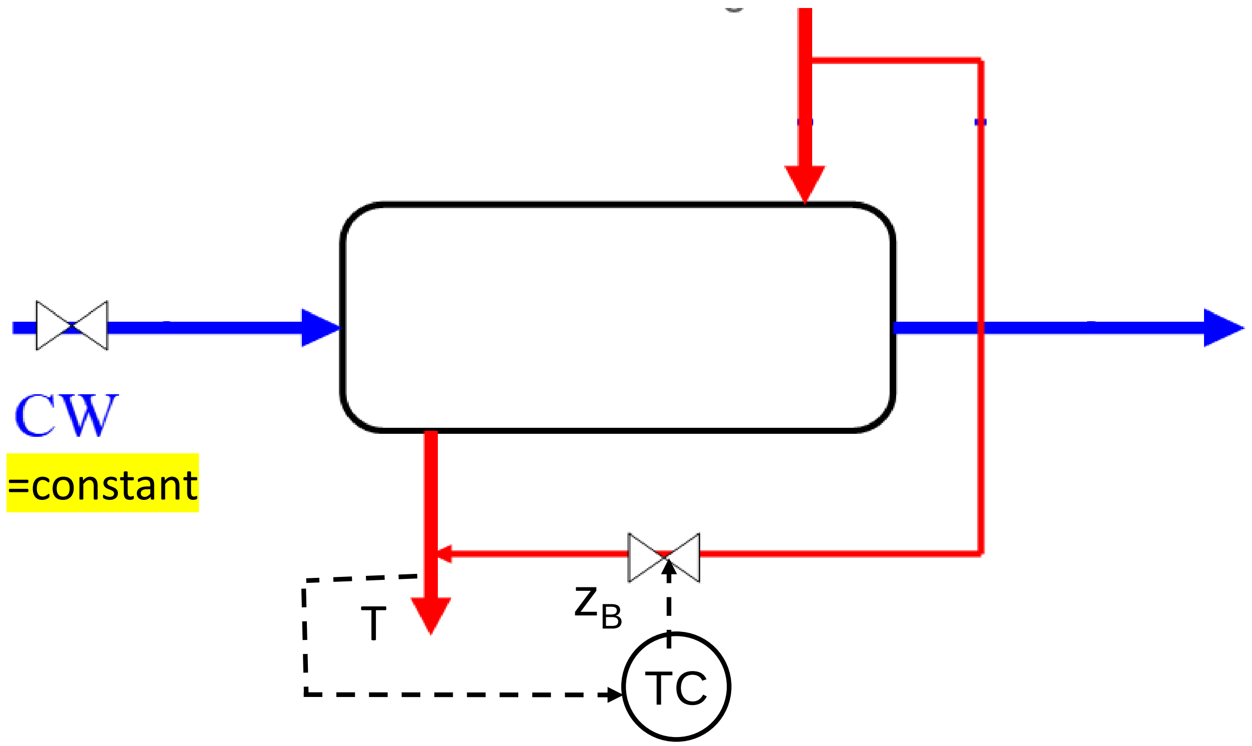
Proposed control structure?

Attempt 1. Use u_2=cooling water: TOO SLOW



Attempt 2. Use $u_1 = z_B$=bypass. SATURATES

(at z_B=0=closed if CW too small)

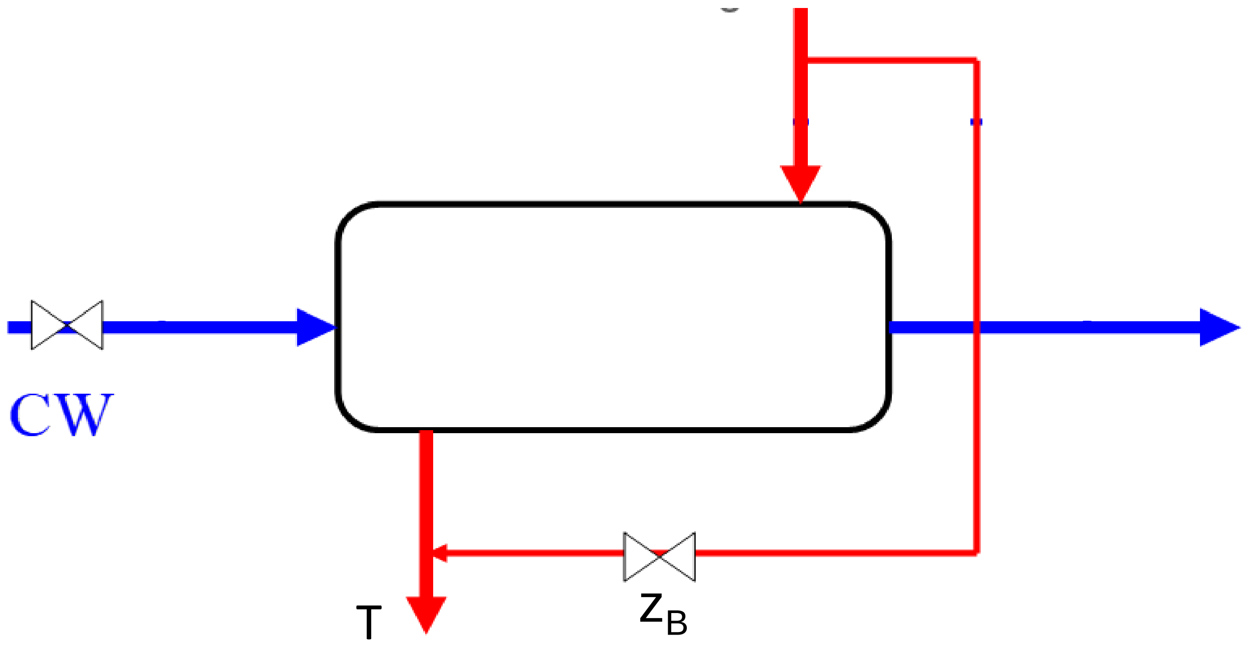


Advantage: Very fast response (no delay)

Problem: z_B is too small to cover whole range

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

What about VPC?

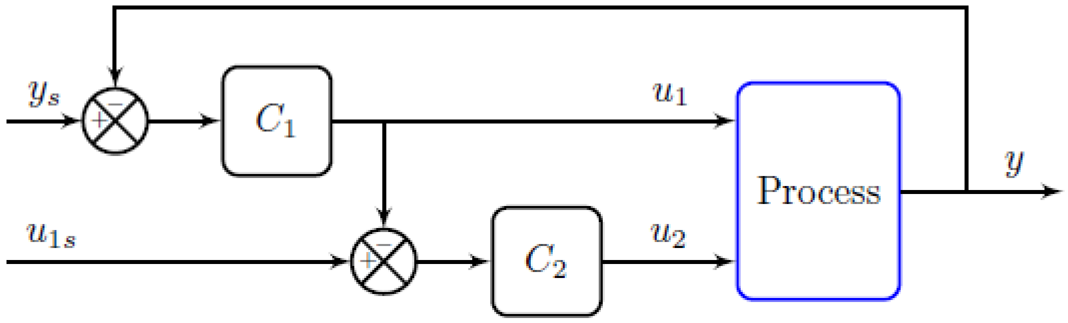


Want tight control of y=T.

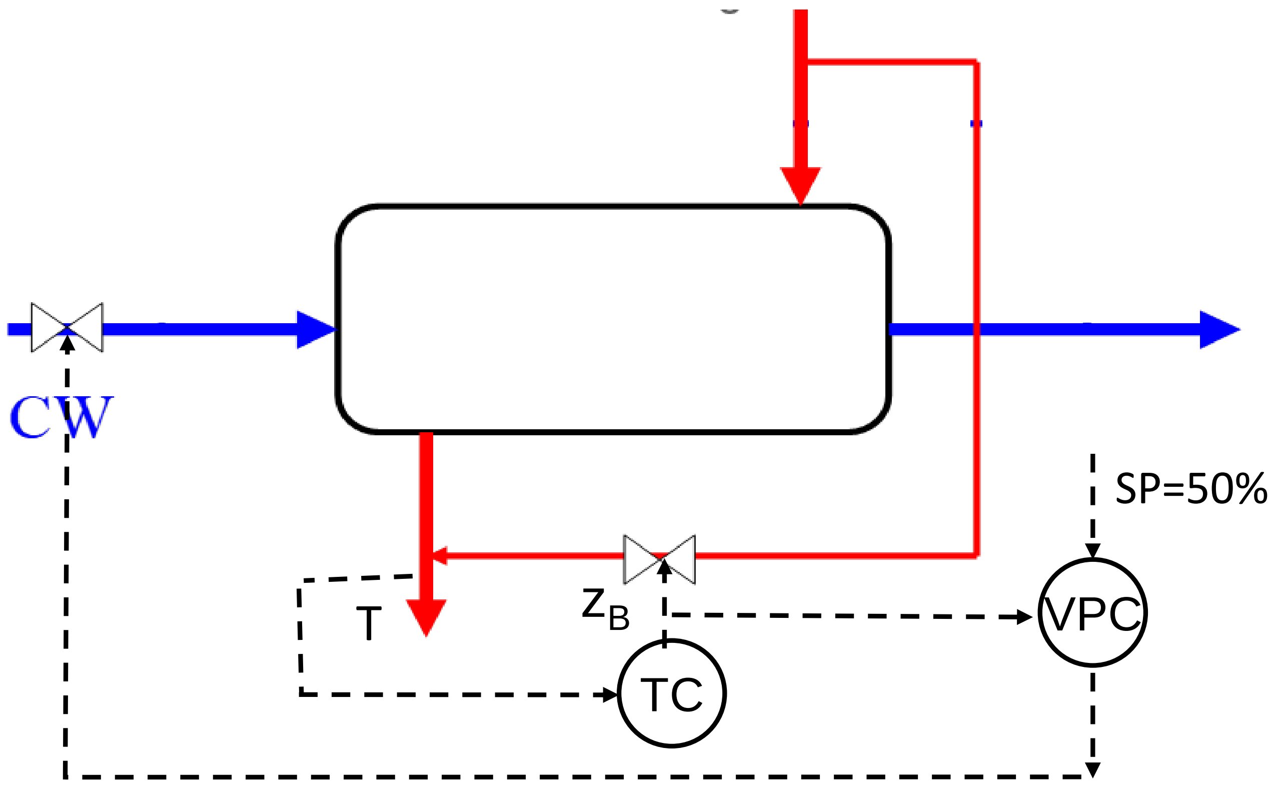
- $u_1 = z_B$
- u_2=CW

Proposed control structure?

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



Attempt 3 (proposed): VPC



- Fast control of y: $u_1 = z_B$
- Main control (VPC): u_2=CW (slow loop)
- Need time scale separation between the two loops

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 (u1) and CW flow (u2) have an effect on T (CV)

- There are two «unbalanced» cases:
  - 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: Parallell control

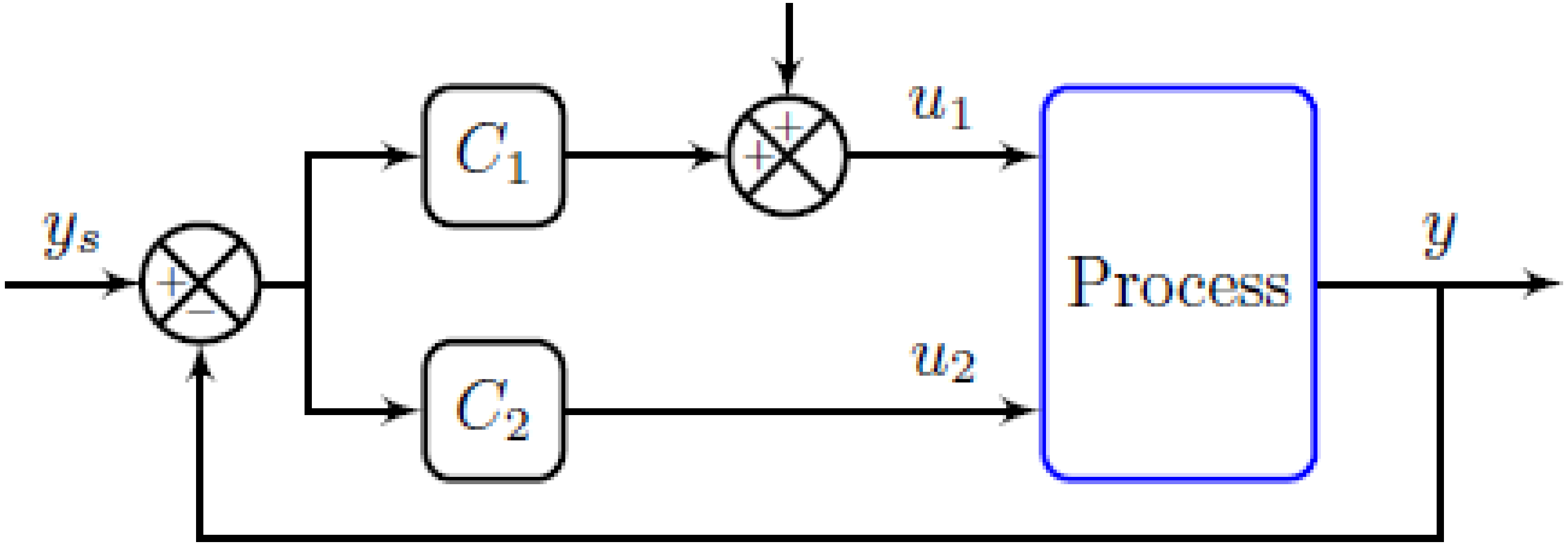


Figure 13: Parallel control to improve dynamic response - as an alternative to the VPC solution in Figure 12.
The "extra" MV (u_1) is used to improve the dynamic response, but at steady-state it is reset to u_{1s}. The loop with C_2 has more integral action and wins a steady state.

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

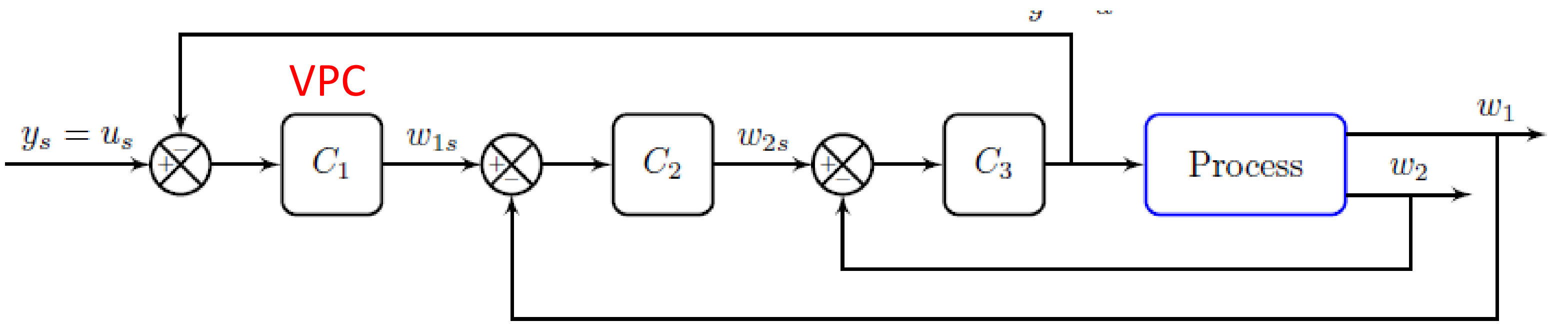
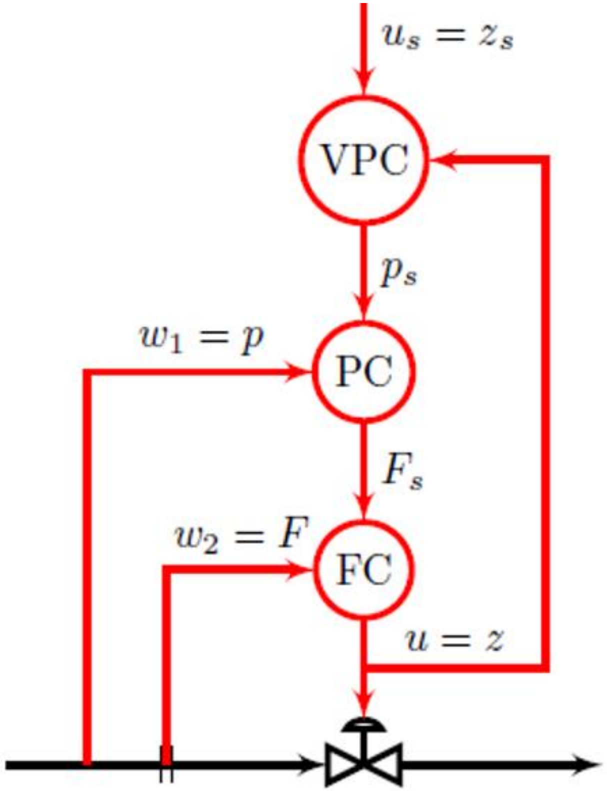


Figure 14: Stabilizing control of variable w_1 combined with valve position control (VPC) for u (=valve position) and inner flow controller ($w_2 = F$).
It corresponds to the flowsheet in Figure 15 with $w_1 = p$ (pressure), C_1 = outer VPC (slow), C_2 = stabilizing controller (fast), C_3 = inner flow 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

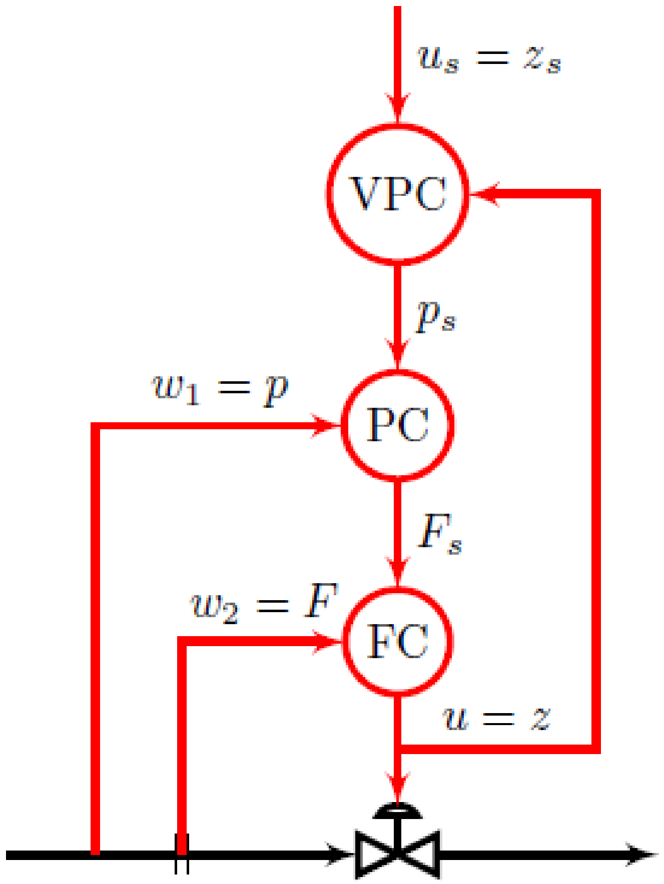


Figure 15: Anti-slug control where the pressure controller (PC) is used to stabilize a desired non-slugging flow regime. The inner flow controller (FC) (fast) provides linearization and disturbance rejection. The outer valve position controller (VPC) (slow) resets the valve position to its desired steady-state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

Note that this is a cascade control system, where we need at least a factor 4 (and preferably 10) between each layer. This implies that the outer VPC (C_1) must be at least 16 (and preferably 100) times slower than the inner flow controller (C_3). This may not be a problem for this application, because flow controllers can be tuned to be fast, with τ_c less than 10 seconds (Smuts, 2011).

Another more fundamental problem is that any unstable mode (RHP pole) in the process will appear as an unstable (RHP) zero as seen from the VPC (C_1) (Storkaas & Skogestad, 2004), which will limit the achievable speed (bandwidth) for resetting the valve to its desired position $u_s = z_s$.

Title: Cascade Control of unstable systems with application to stabilization of slug flow Author: Espen Storkaas and Sigurd Skogestad Presented at IFAC-symposium Adchem'2003, Hong Kong, Jan. 2004 (original date

Example: Stabilize bycycle

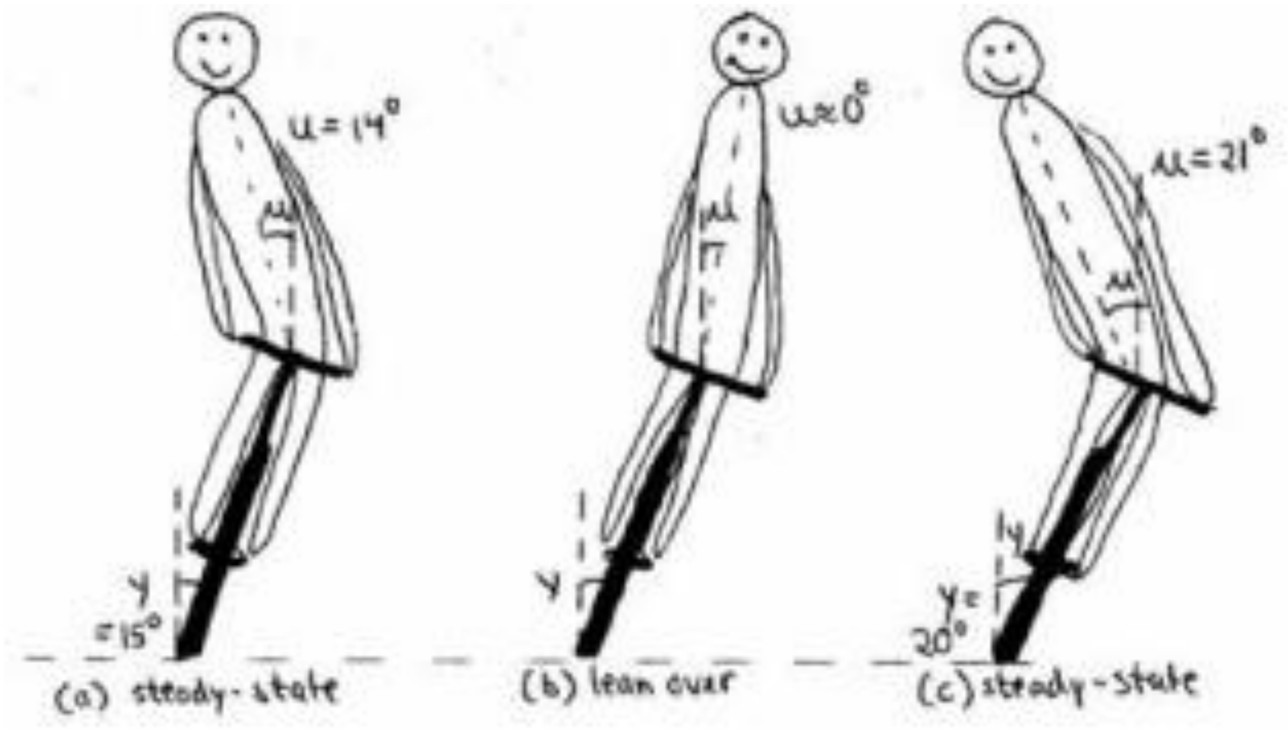


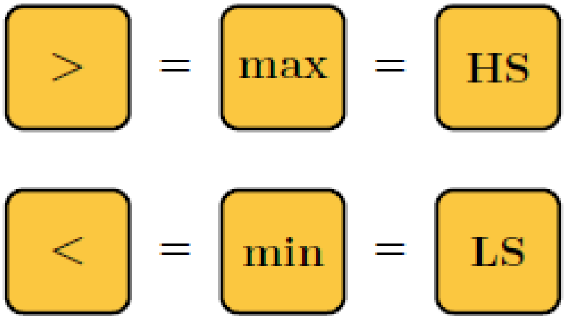
Fig. 2. Inverse response for a bicycle caused by an underlying instability

Consider Figure 2 where the aim is to tilt the bike from an initial angle $y = 15°$ (Fig. 2a) using your body (u) to an angle $y = 20°$ (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.

Espen Storkaas and Sigurd Skogestad, "Cascade Control of unstable systems with application to stabilization of slug flow", IFAC-symposium Adchem'2003, Hong Kong,

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

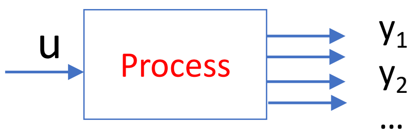


Note: Selectors are logic blocks

- Sometimes called “override”
  - But this term may be misleading
- 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.

Implementation selector



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

- Selector on MVs
  - Must have anti windup in c1 and c2 !

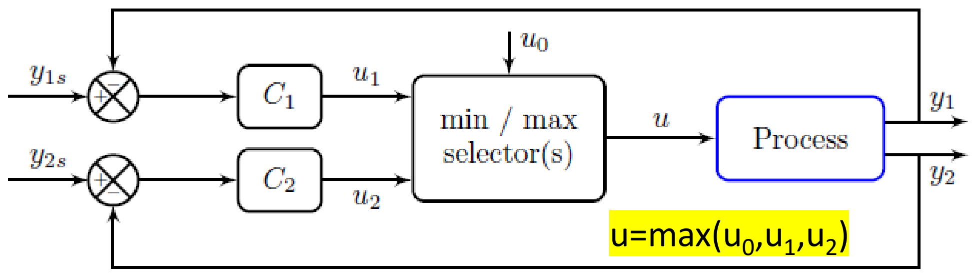
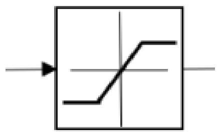


Figure 17: CV-CV switching with selector on MV (input u).

Alt. II (Less general) Controllers in cascade

- Selector on CV setpoint
- In this case: Selector may be replaced by saturation element (with y2s as the max) or min)



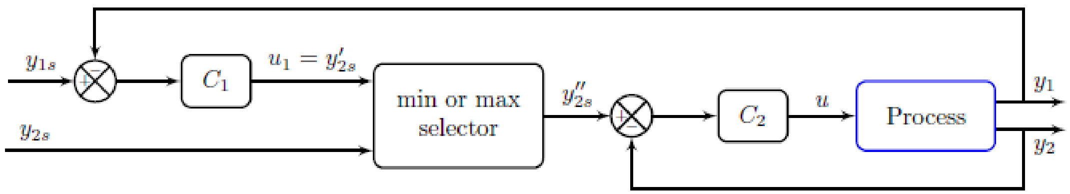
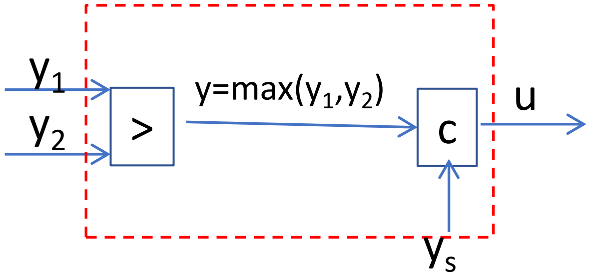


Figure 19: Alternative cascade CV-CV switching implementation with selector on the setpoint. In many cases, y_{1s} and y_{2s} are constraint limits.

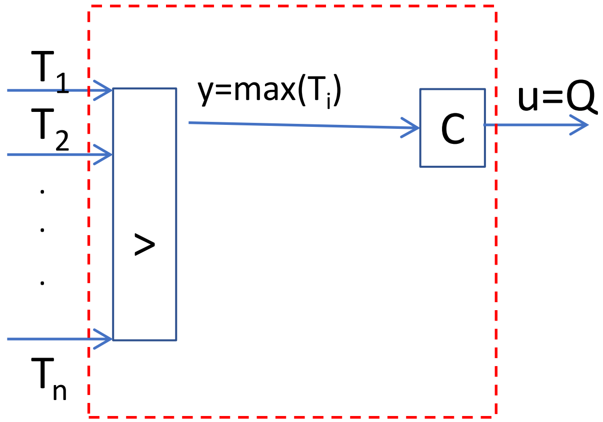
Alt. III (For special case where all CVs have same bound). One controller

- 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.



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,…) computing the heat input ($u_1=Q_1$, $u_2=Q_2$, ….) and then select u = **min**(u_1, u_2, …), but it is more complicated.

Furnace control with safety constraint (Alt. I)

Input (MV)

u = Fuel gas flowrate

Output (CV)

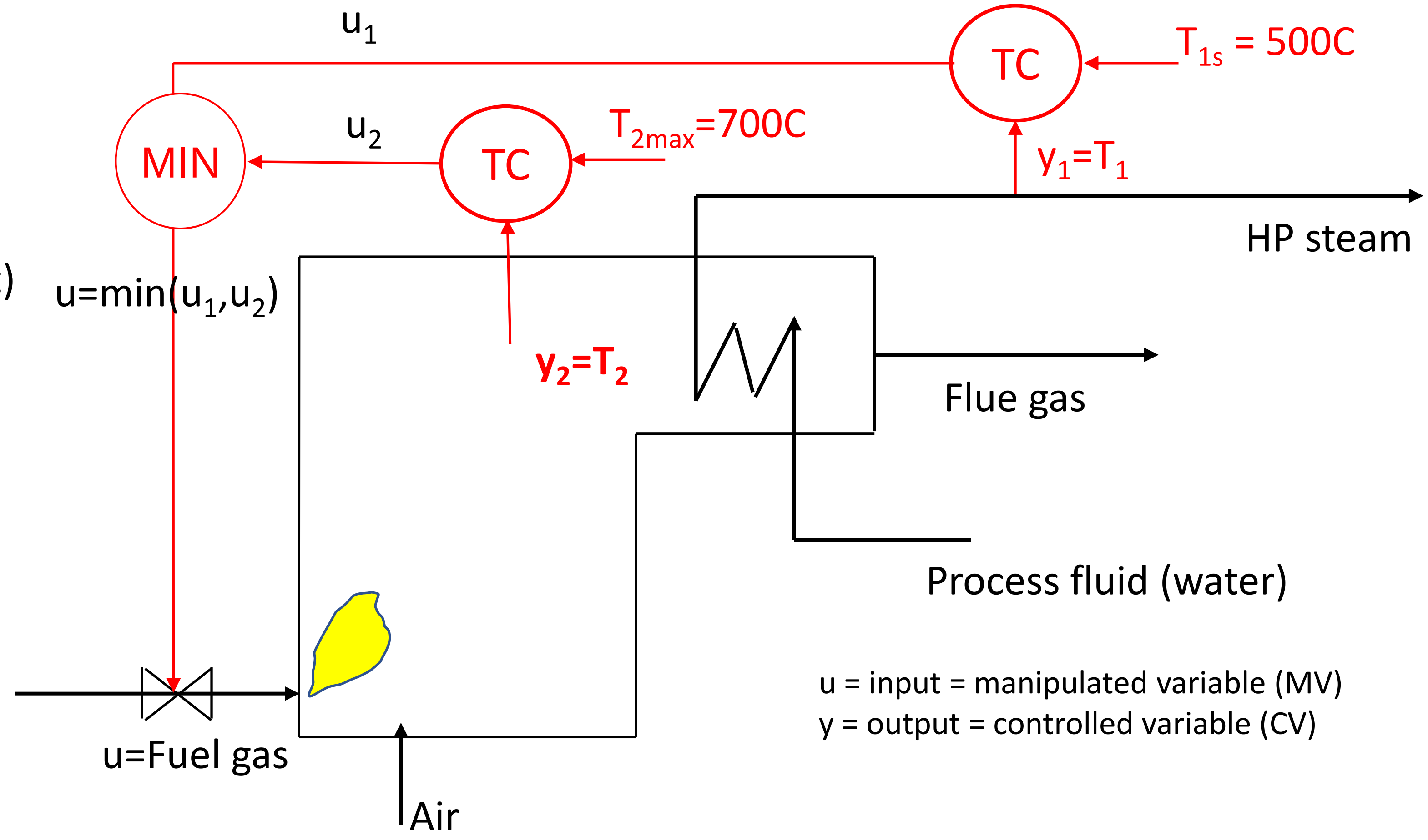
y_1 = process temperature T_1

(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

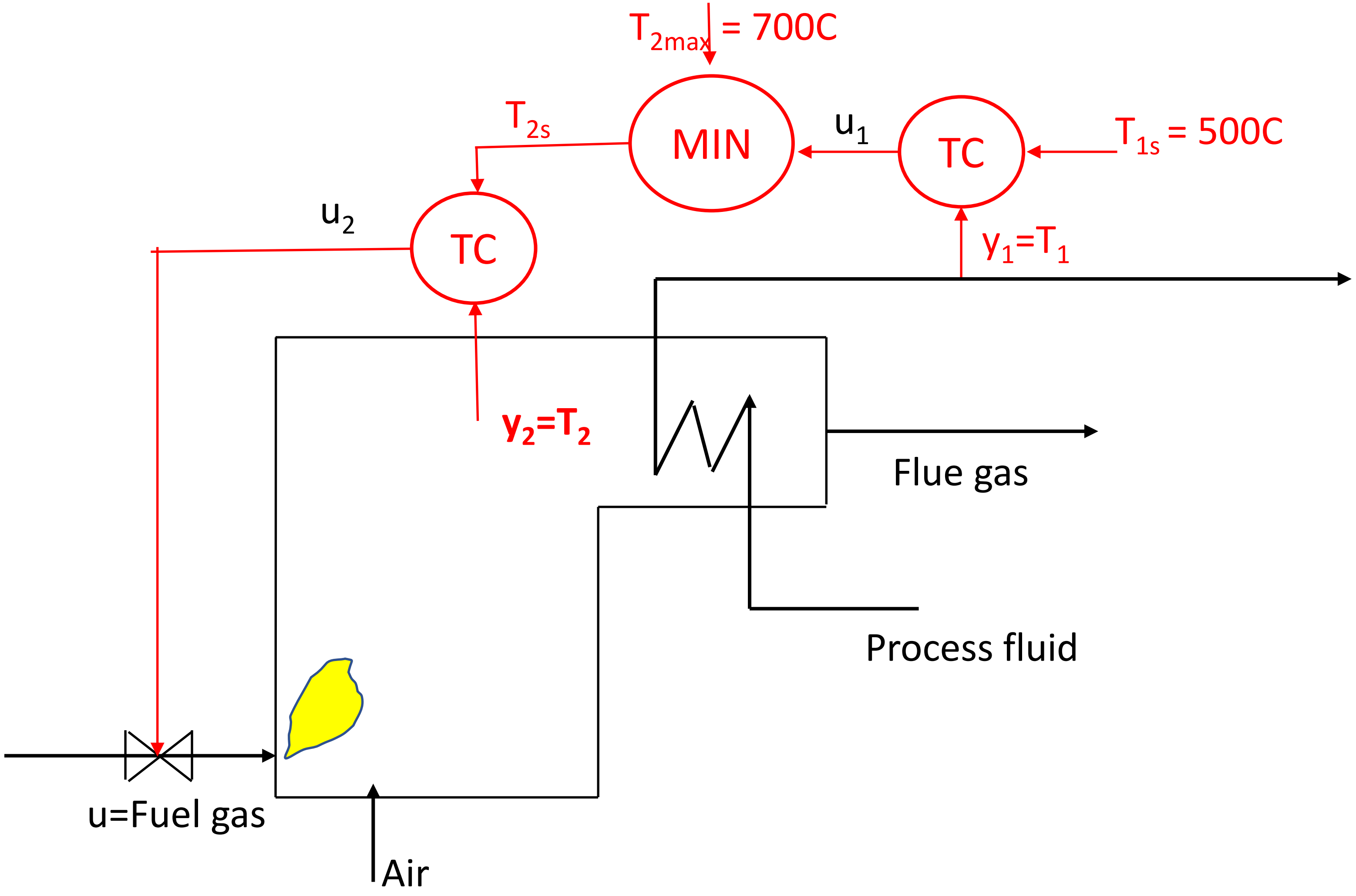


u = input = manipulated variable (MV)

y = output = controlled variable (CV)

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.



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
- 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)

Rule 2 (order of selectors)

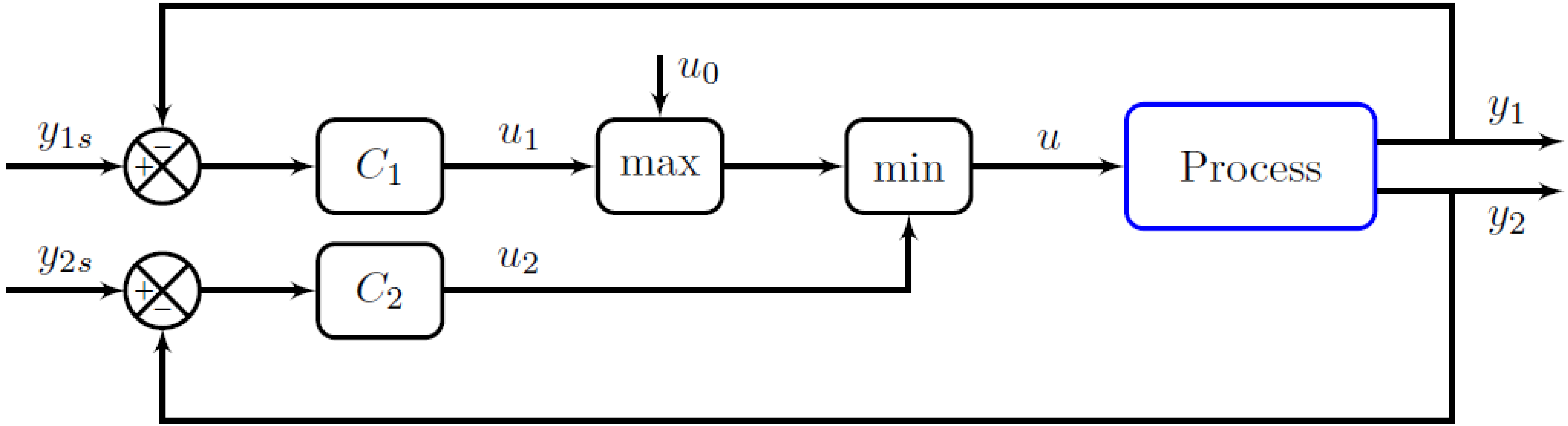


Figure 18: CV-CV switching for case with possibly conflicting constraints. In this case, constraint y_{1s} requires a max-selector and y_{2s}) requires a min-selector. The selector block corresponding to the most important constraint (here y_{2s}) should be at the end (Rule 2).

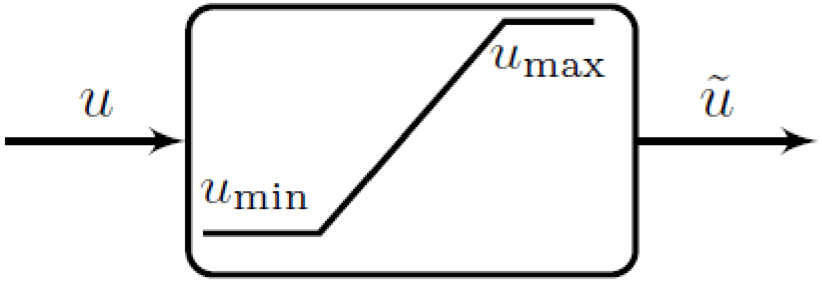
To understand the logic with selectors in series, start reading from the first selector. In this case, this is the max-selector: The constraint on y_1 is satisfied by a large value for u which requires a max-selector (Rule 1). u_0 is the desired input for cases when no constraints are encountered, but 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.

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).
- Another way to see this is to note that a valve works as a saturation element



The order of the “built-in” max- and min -selector in (8) does not matter because there is no possibility for conflict, as the two constraints (limits), u_{min} and u_{max}, cannot be active at the same time. However, in general, the order of the selectors does matter, and in cases of conflict, Rule 2 says that we should put the most important constraint at the end. Note that the “built-in” max- and min-selector

Question: Why doesn’t order matter here?

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

Challenges selectors

- Standard approach requires pairing each active constraint with a single input
  - May not be possible in complex cases
- 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