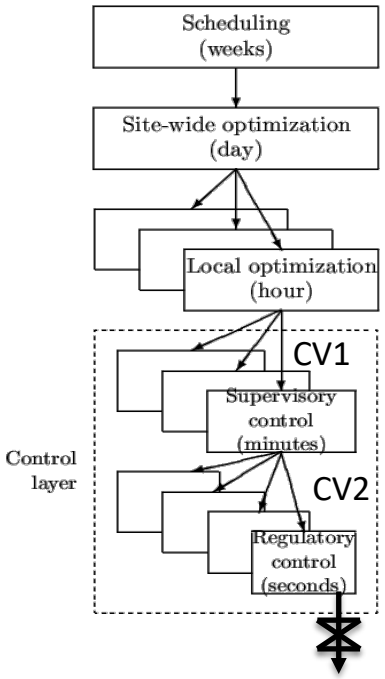
«Advanced» control

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

Two fundamental ways of decomposing the controller

- Vertical (hierarchical; cascade)
- Based on time scale separation
- Decision: Selection of CVs that connect layers



- Horizontal (decentralized)
- Usually based on distance
- Decision: Pairing of MVs and CVs within layers

In addition: Decomposition of controller into smaller elements (blocks):
Feedforward element, nonlinear element, estimators (soft sensors), switching elements

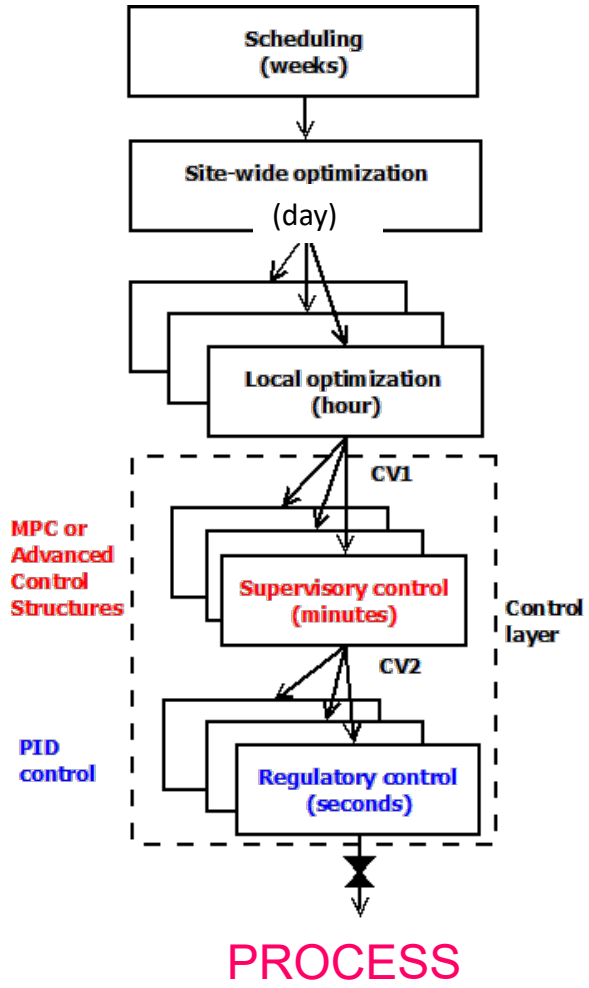
The decision hierarchy is based on "time scale separation"

Control layers:

- **Supervisory control ("Advanced classical control" or MPC):**
  - Follow set points for CV1 from economic optimization layer
  - Switch between active constraints (change CV1)
  - Look after regulatory layer (avoid that MVs saturate, etc.)

- **Regulatory control (PID):**
  - Stable operation (CV2)

CV = controlled variable



Standard Advanced control elements

Sigurd Skogestad, "Advanced control using decomposition and simple elements".
Annual Reviews in Control, vol. 56 (2023), Article 100903 (44 pages).

- 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 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

[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.

If PID feedback control is not working well

- First try retuning (SIMC-PID) or changing the pairing (E0)
  - Of course, you need anti-windup (E8) if the input (MV) may saturate (dynamically)

Next: Advanced control:

- The first thing to consider is cascade control (E1)
  - Measure an output closer to the disturbance
  - **Often flow control slave**
- The next thing is feedforward control (E11)
  - **Ratio control (E2) is a must for mixing!**
  - **Nonlinear feedforward based on steady-state model, $y = f_0(u,d)$: Use transformed input $v=f_0(u,d)$.** (E14)
- Next consider adding extra «fast» input
  - Possibly with valve position control (E3) for original input

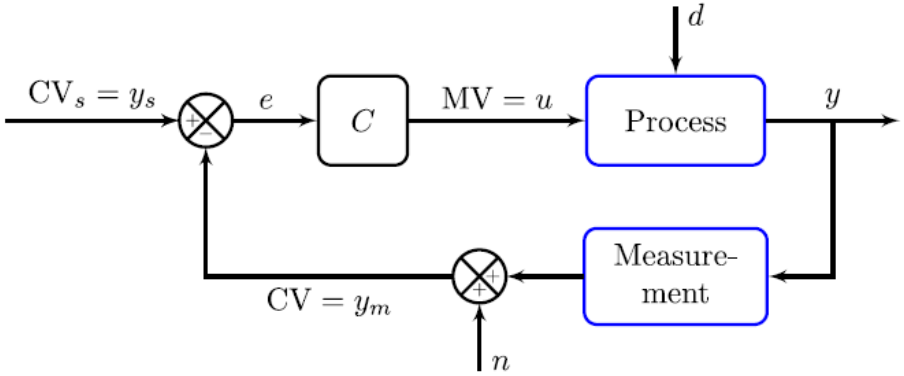
What about Constraints?

- Switching between constraints: Three cases: MV-MV (E5,E6,E7), CV-CV (E4), MV-CV

Multivariable control. If interactions are important

- Start with Single-loop control (Pairing)
- May try: Decoupling (E12, E14)
- May consider MPC if very interactive

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



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

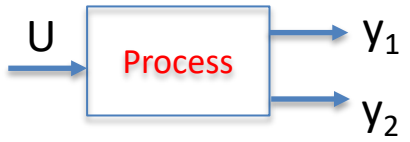
1. ***"Pair-close rule" : T****he MV should have a large, fast, and direct effect on the CV.*
2. ***"Input saturation rule":*** *Pair a MV that may saturate with a CV that can be .given up (when the MV saturates).**
   - *Exception: Have extra MV so we use MV-MV switching (e.g., split range control)*

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

E1. Cascade control



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

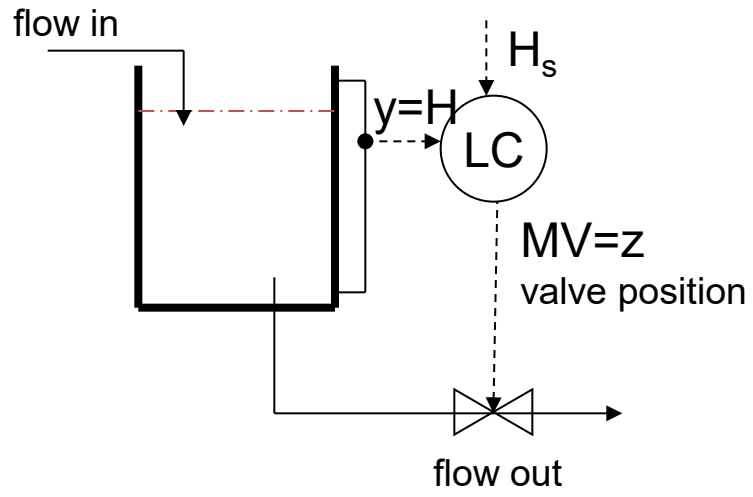
Idea: make use of extra "local" output measurement (y_2)

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

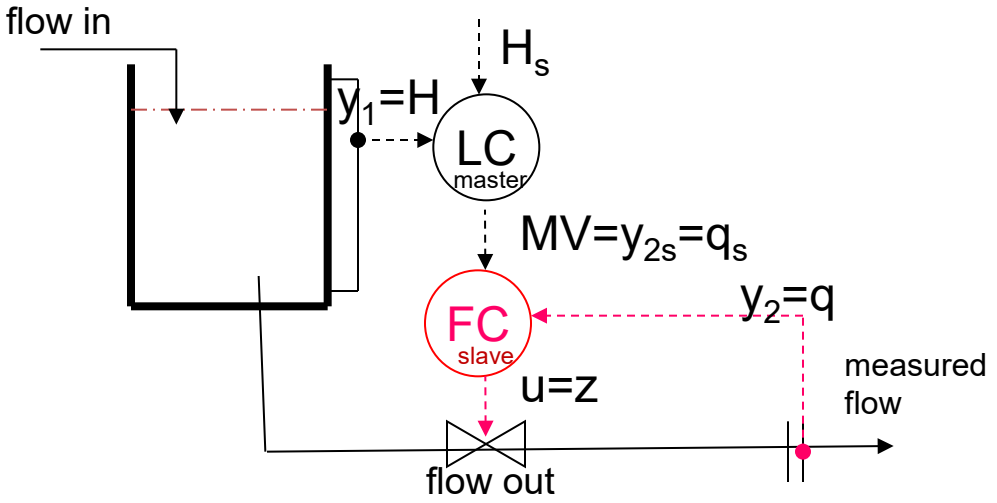
- Without cascade: "Master" controller directly adjusts u (input, MV) to control y (primary output, sometimes called y_1)
- With cascade: Local "slave" controller(fast) uses u to control "extra"/fast measurement (y_2). *
  "Master" controller (at least five times slower than slave) adjusts setpoint y_{2s}.

- **Example: Flow controller on valve (very common!)**
  - y = level H in tank (or could be temperature etc.)
  - u = valve position (z)
  - y_2 = flowrate q through valve

WITHOUT CASCADE

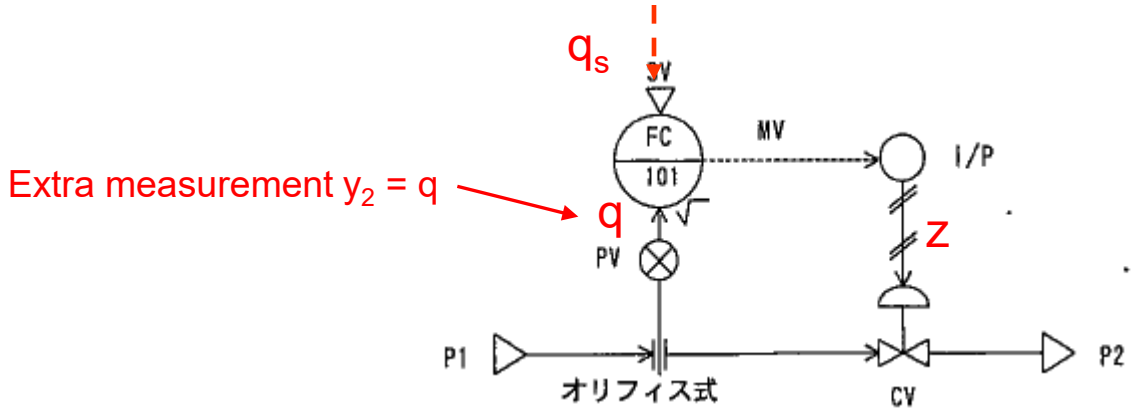


WITH CASCADE



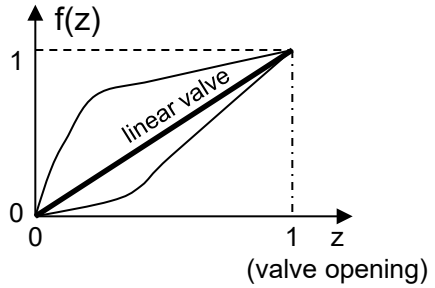
*Comment: Another approach that uses extra measurements to improve control is «**Full state feedback**».

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



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

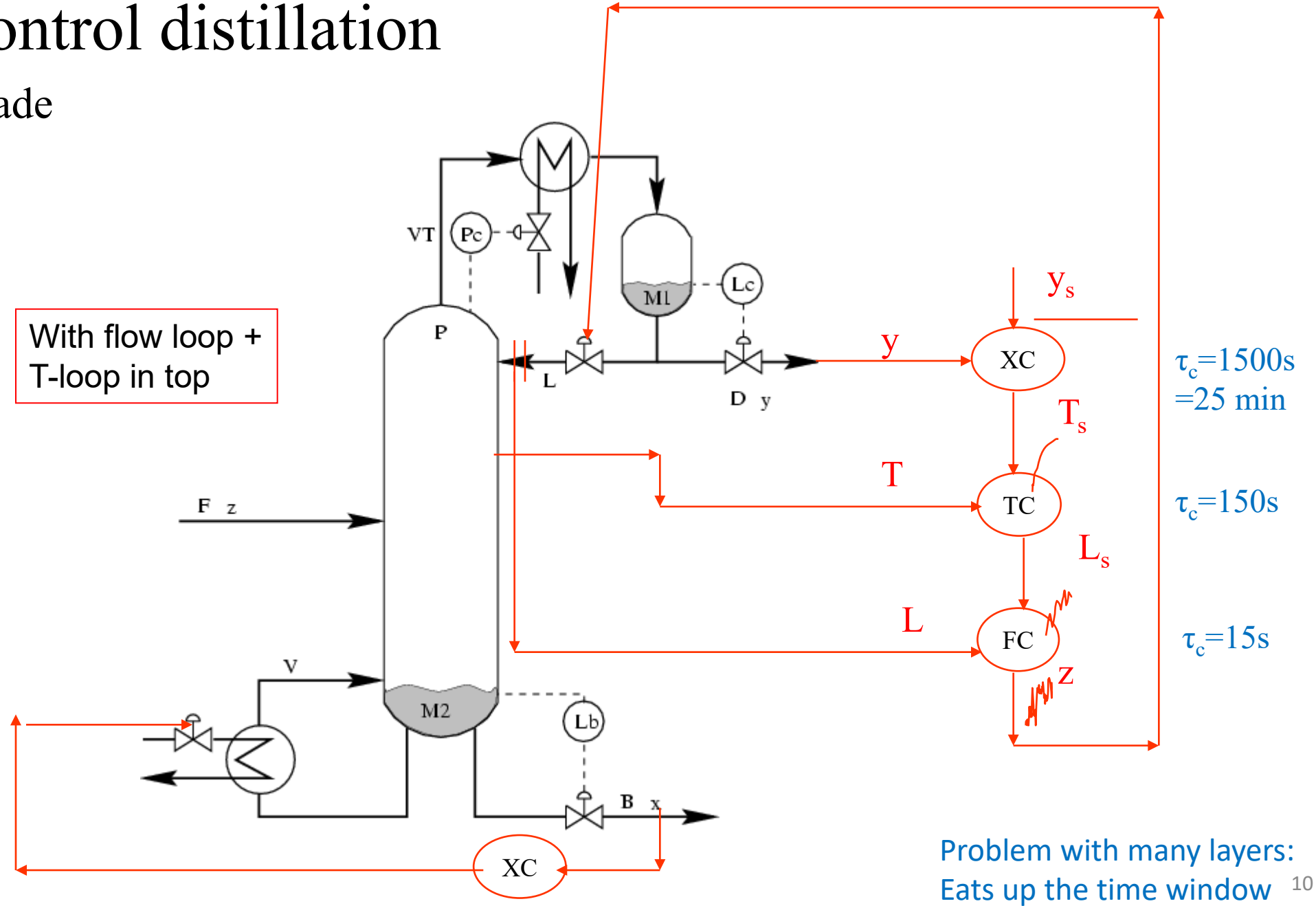
1. Counteracts nonlinearity in valve, f(z)
   - High gain in inner loop eliminates nonlinearity inside inner loop
   - With fast flow control we can assume q = q_s



2. Eliminates effect of disturbances in p1 and p2
   (FC reacts faster than outer level loop)

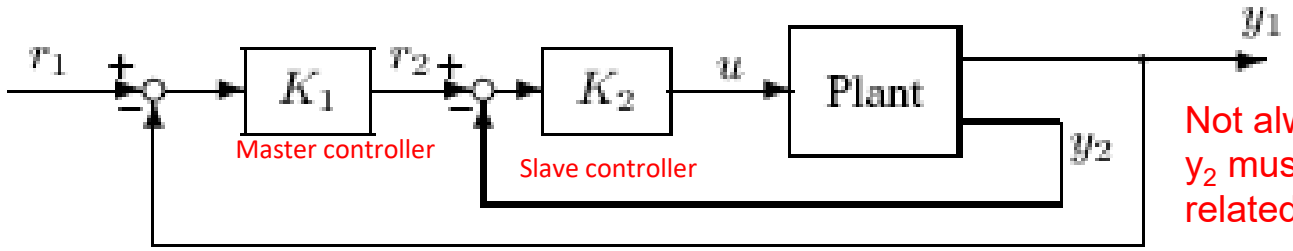
Cascade control distillation

3 layers of cascade



Problem with many layers:
Eats up the time window

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")

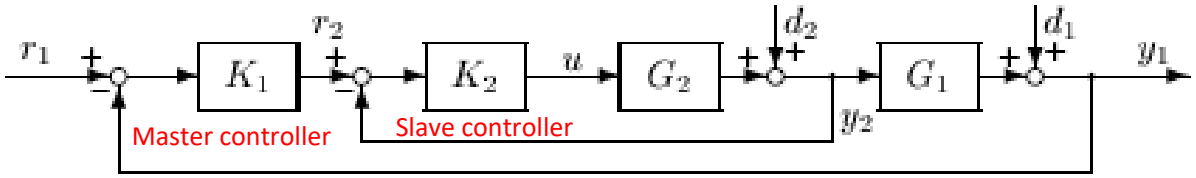
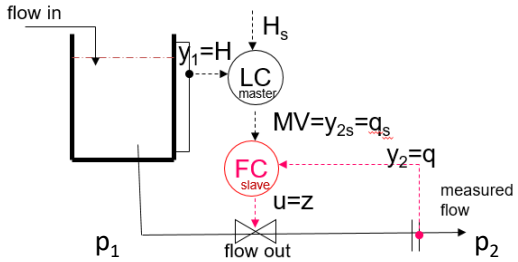


Figure 10.11: Common case of cascade control where the primary output y_1 depends directly on the extra measurement y_2

Block diagram flow controller



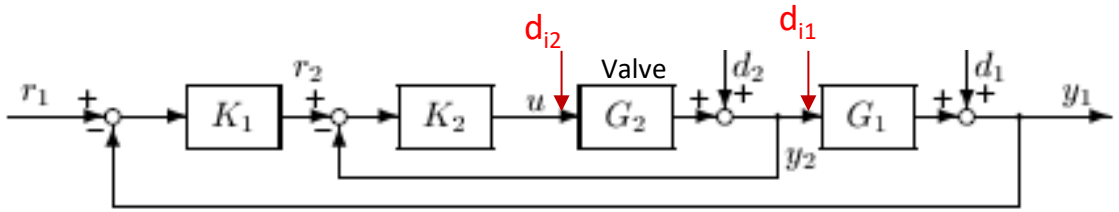
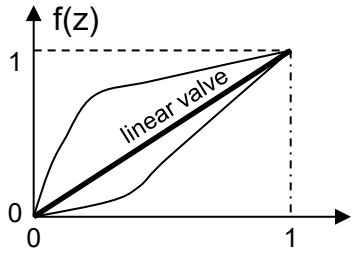


Figure 10.11: Common case of cascade control where the primary output y_1 depends directly on the extra measurement y_2

Example: Level control with slave flow controller:

u = z (valve position, flow out)
y_1 = H
y_2 = q
d_{11} = flow in
d_{i2} = p_1-p_2

Transfer functions:
G_2 = k(z)/(τs+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

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 ?

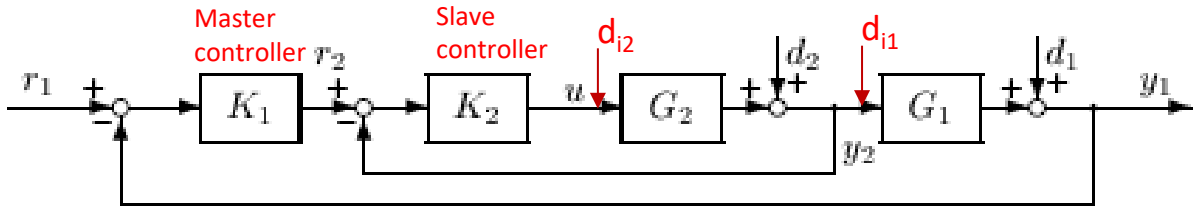


Figure 10.11: Common case of cascade control where the primary output y_1 depends directly on the extra measurement y_2

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

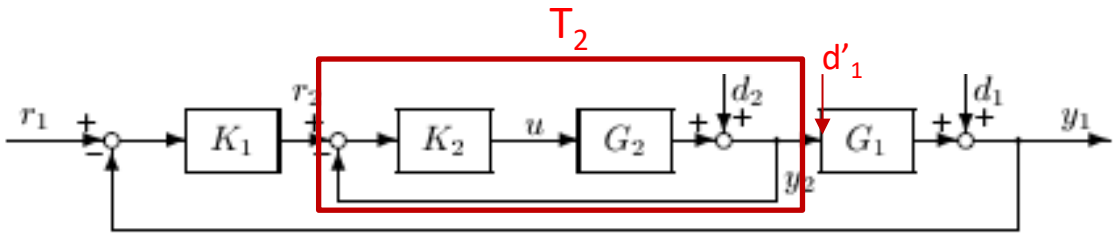


Figure 10.11: Common case of cascade control where the primary output y_1 depends directly on the extra measurement y_2

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

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, $T_2 \approx e^{-\Theta 2s}/(\tau_{c2}s+1) \approx e^{-(\Theta 2+ \tau c2)\ s}$
- 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 (σ =10) is to avoid that control gets too slow, especially if we have many layers

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

Counteracting nonlinearity using cascade control

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

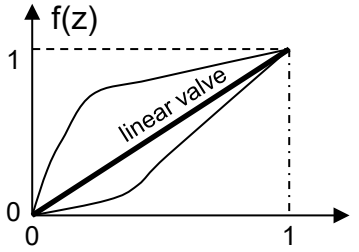
- Nonlinear valve with varying gain k_2: $G_2(s) = k_2(z) / (\tau_2 s+1)$
- Slave (flow) controller K_2: PI-controller with gain K_{c2} and integral time $\tau_I = \tau_2$ (SIMC-rule). Get

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

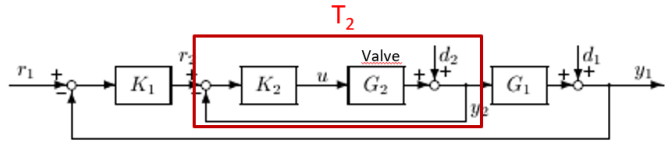
- With slave controller: Transfer function from y_{2s} to y_2 (as seen from outer loop):

$$T_2 = L_2/(1+L_2) = 1/(\tau_{C2}\, s + 1), \text{ where } \tau_{C2} = \tau_2/(k_2\, K_{c2})$$

- 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_1T_2.
  - A smaller process gain k_2 results in a larger τ_{C2} and thus a large effective delay, whuch 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



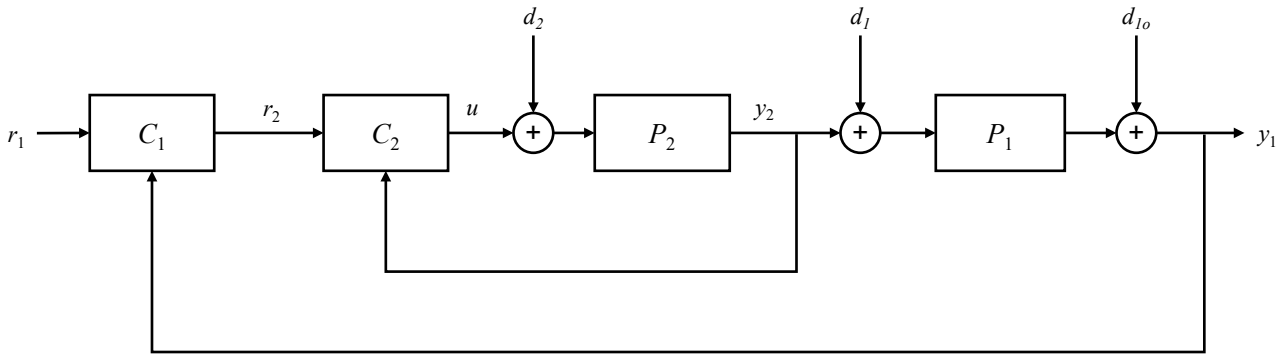
$k_2(z)$ = slope = df/dz



G_1T_2 = «Process» for tuning master controller K_1

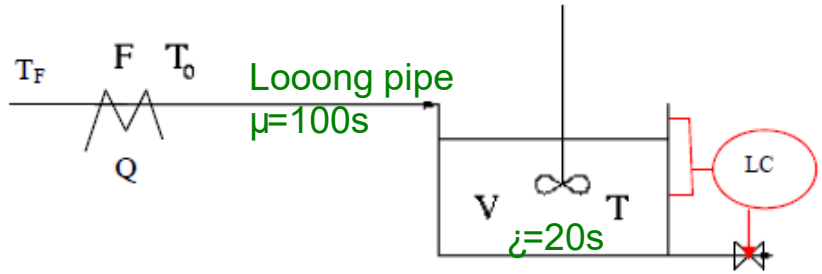
Cascade control block diagram

- Which disturbances motivate the use of cascade control?

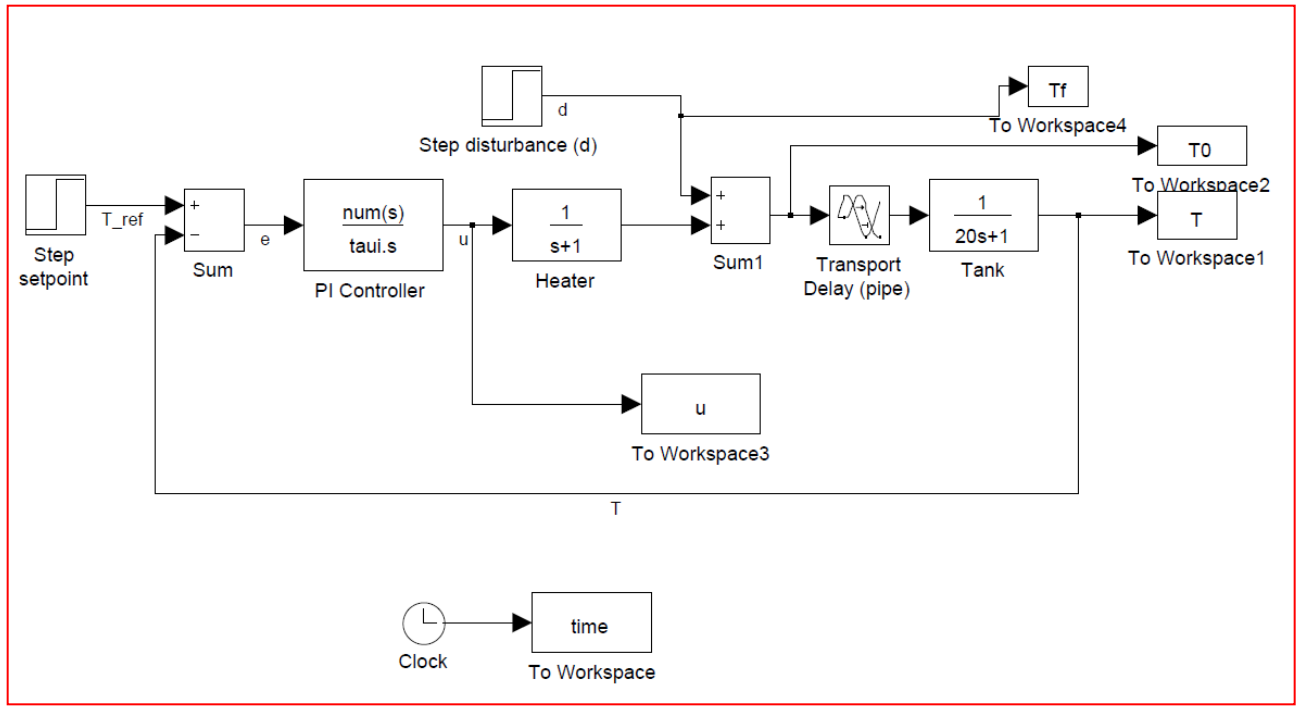


Answer: d_2

Example: Similar to shower process



u = Q
y1 = T
d = T_F

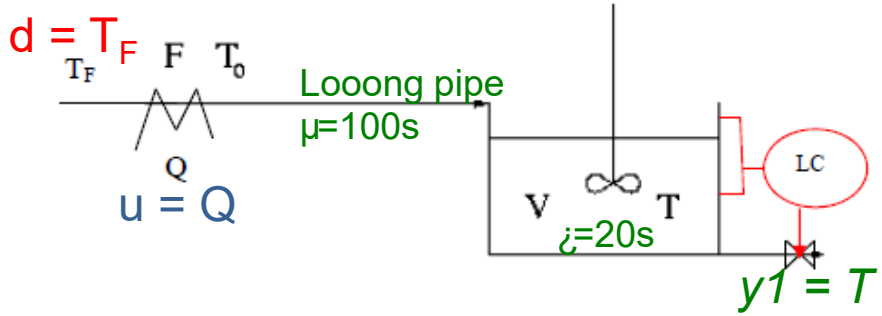


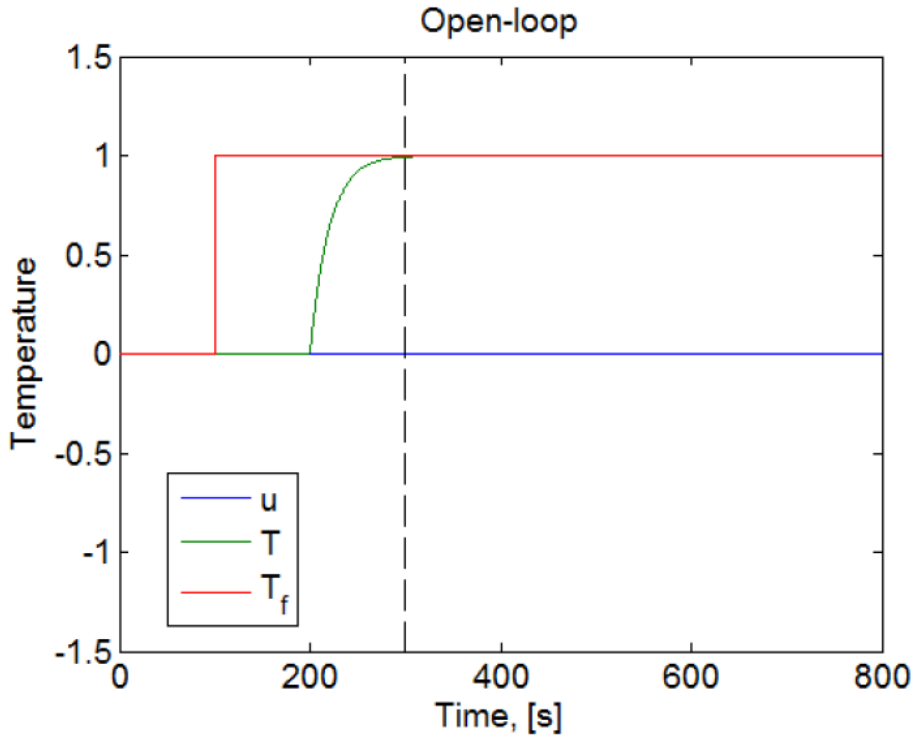
Simulink model: tunepid1_ex1

Simulink model: tunepid1_ex1

Note: level control not explicitly included in simulation (assume constant level)

Disturbance response with no control





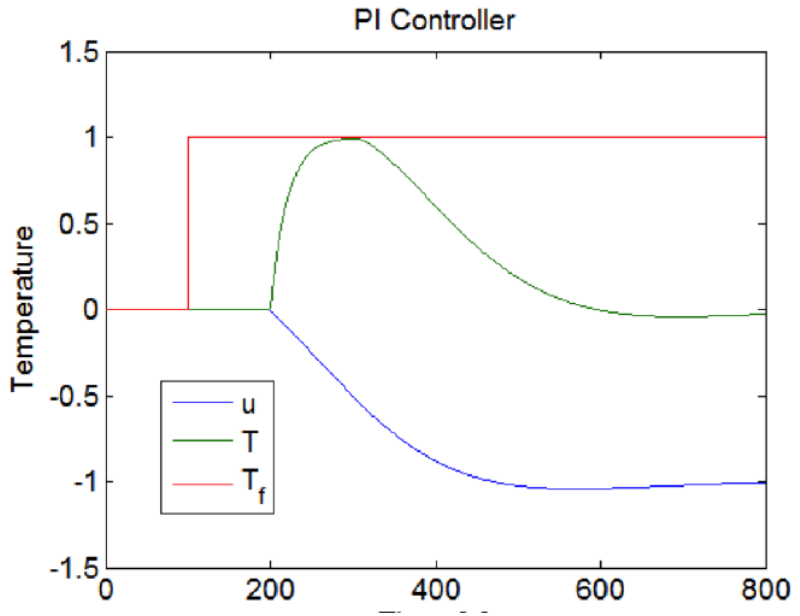
u = Q
y1 = T
d = T_F

```
Kc=0; taui=9999; % no control
%start simulation (press green button)
plot(time,u,time,T,time,Tf), axis([0 800 -1.5 1.5])
```

Conventional PI control (measure y1)

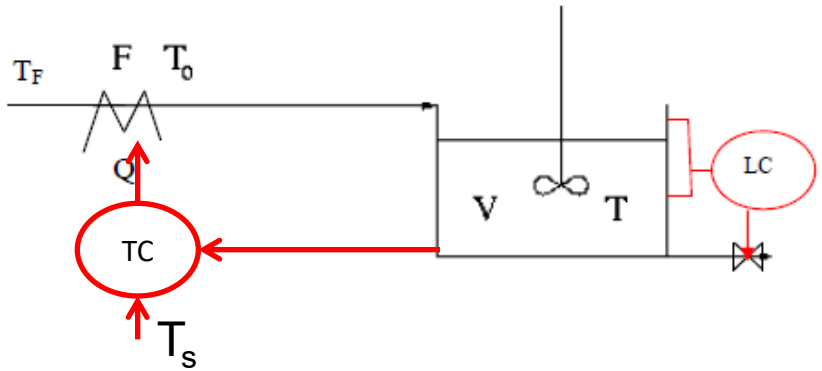
11) SIMC PI tuning rule with $\tau_c = \theta = 100$.

$$K_c = (1/k)\tau_1/(\tau_c + \theta) = 20/200 = 0.1; \tau_I = \min(\tau_1, 4(\tau_c + \theta) = 20$$



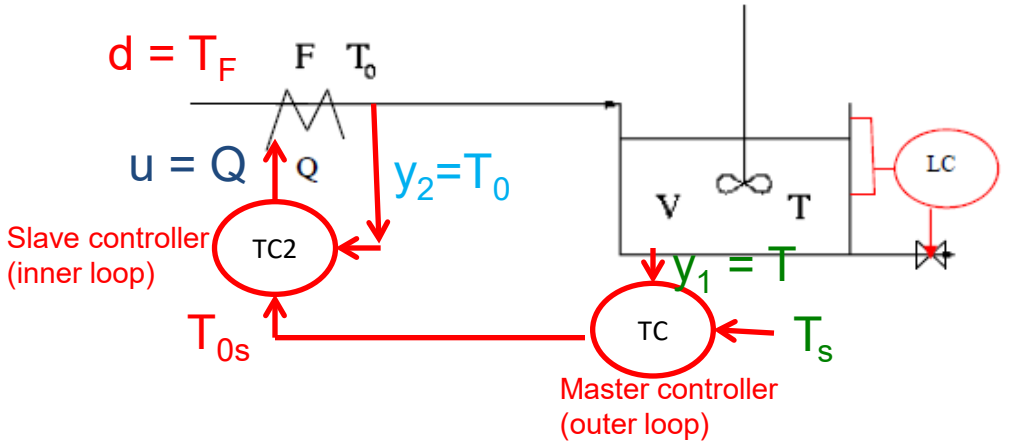
u = Q
y1 = T
d = T_F

No offset, but slow




```
Kc=0.1; taui=20; % SIMC PI-control
%start simulation (press green button)
plot(time,u,time,T,time,Tf), axis([0 800 -1.5 1.5])
```

Measure also $y_2=T_0$: Cascade control is much better



G_1= exp(-100s)/(20s+1)
G_2=1/(s+1)

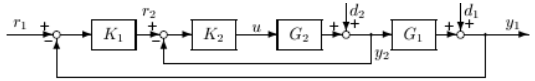
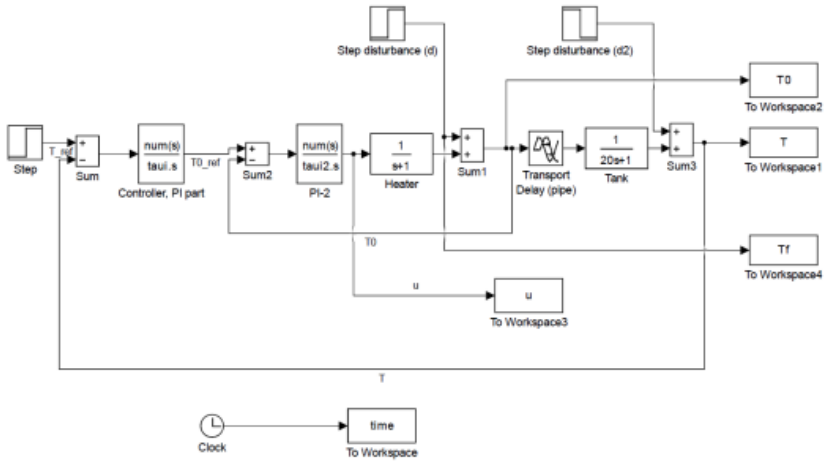
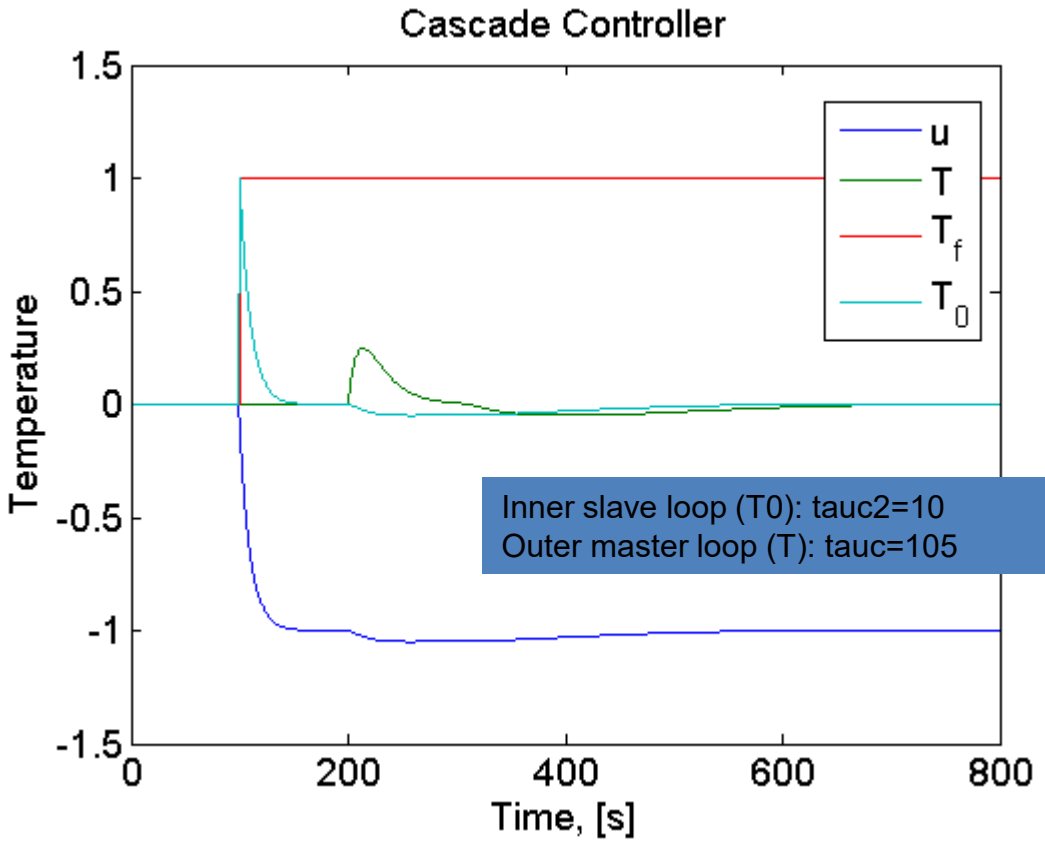


Figure 10.11: Common case of cascade control where the primary output y_1 depends directly on the extra measurement y_2





Inner slave loop (T0): tauc2=10
Outer master loop (T): tauc=105

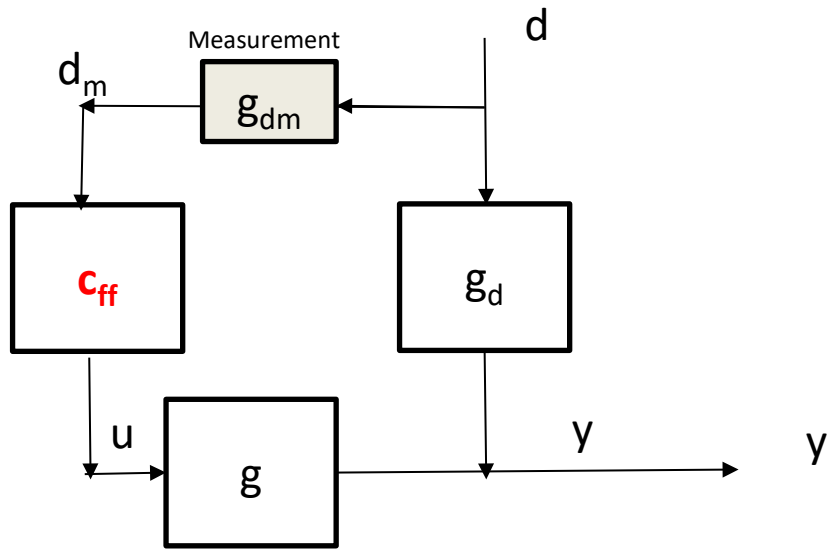
```
Kc2=0.1;taui2=1; % inner loop with tauc2=10
Kc=0.119; taui=25; % outer loop with tauc=105
sim('tunepid1_ex1_cascade') %start simulation
plot(time,u,time,T,time,Tf,time,T0), axis([0 800 -1.5 1.5])
```

Question: Will setpoint tracking for y_1 =T be improved with cascade (in this case)?

- No, since there was essentially no dynamics in G2=1/(s+1), it is actually slightly worse (tauc increased from 100 to 105).
- But if G2 was an integrating process, cascade could improve seytpoint response

E11. Feedforward (FF) control

Mainly: For disturbances where feedback control is not good enough.



- Model: $y = g\,u + g_d\,d$
- Measured disturbance: $d_m = g_{dm}\,d$
- Feedforward controller: $u = \mathbf{c_{FF}}\,d_m$
- Get $y = (g\,\mathbf{c_{FF}}\,g_{dm} + g_d)\,d$
- Ideal feedforward controller:

  $y = 0\,d \;\rightarrow\; c_{FF,ideal} = -(g_d / (g_{dm}\,g))$
  - But often not realizable

  For $c_{FF}(s)$ to be realizable:
  1. Order pole polynomial c_{FF} ≥ order zero polynomial c_{FF}
  2. No prediction allowed (μ in c_{FF} cannot be negative)
  3. And must avoid that C_{FF} has too high gain (to avoid aggressive input changes)

  – Common simplification is to use static FF: $c_{FF} = k$

  – General. Approximate $c_{FF,ideal}$ by

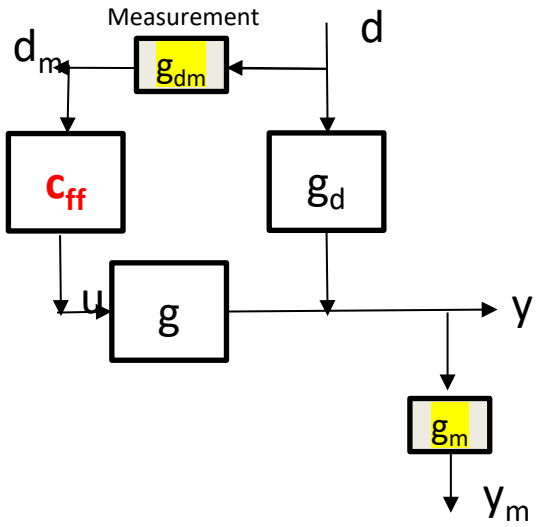
$$c_{FF}(s) = k\frac{(T_1 s+1)\cdots}{(\tau_1 s+1)(\tau_2 s+1)\cdots}e^{-\theta s}$$

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

Note: Feedforward control is model-based.
Since true process is nonlinear, we often include nonlinearity, for example:

- Make k depend on operating point.
- Ratio control
- Nonlinear feedforward control (input transformation)

When use feedforward?



Recall: $c_{FF,ideal} = - g_d / (g_{dm}\, g)$

Use FF when

1. There is an effective delay in the "total process" from u to y_m (in $g_m\, g$) (so that feedback cannot be fast)
   - and in particular if there is delay in the measurement of y (g_m)
2. Preferably some "delay" in g_d
   - Feedforward then has "more time" to take the right action
   - Ideally the delay in g_d is larger or equal to the delay in $g_{dm}\, g$ (so $c_{FF,ideal}$ can be realized).
   - Note: No delay is needed in g_d if the delay in g_m is large (because then feedforward always has a potentially large benefit)

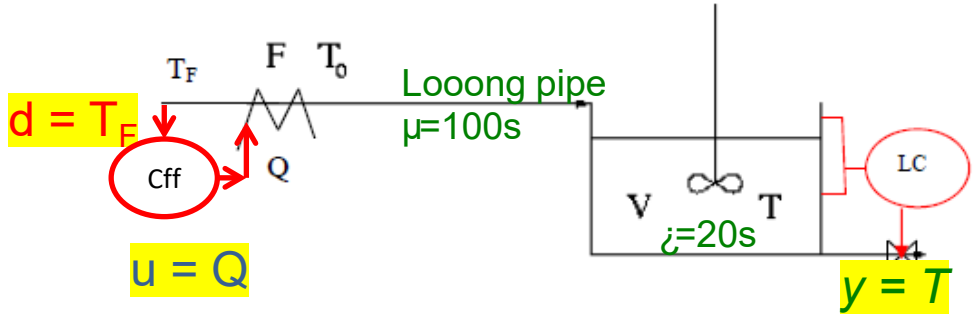
3. And: Have good models

Note: The two yellow measurement transfer functions are not the same.

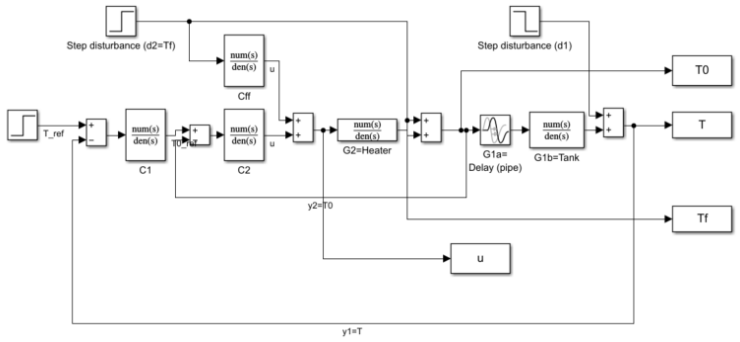
Example: Similar to shower process

$$g_d = \frac{e^{-100s}}{20s+1}, g = \frac{k}{s+1} g_d$$

gd has delay,
Promising for FF!
Will use C_{ff} = -1 (see details below)



u = Q
y = T
d = T_F (measured)
y_2=T_0 (yellow)




```
Kc2=0;  % without feedback
Kff=-1   % Feedforward controller, Cff=Kff=-1
sim('tunepid1_ex1_cascade_ff') %start simulation
plot(time,u,"blue",time,T,"green",time,Tf,"red",time,T0,"orange"), axis([0 800 -1.5 1.5])
```

Details: Model heater: $T_0 = T_F + \frac{k}{s+1} Q$

Model pipe and tank: $T = gd(s)T_0$; $g_d = \frac{e^{-100s}}{20s+1}$

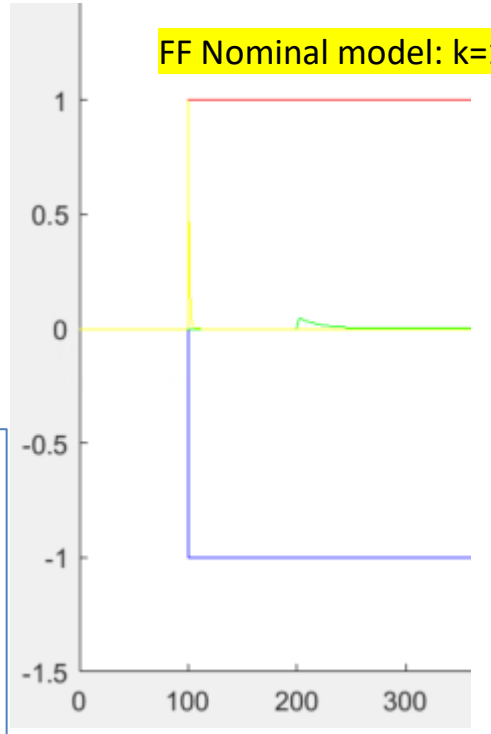
Nominal: k=1,

Assume perfect measurements: $g_{dm} = g_m = 1$

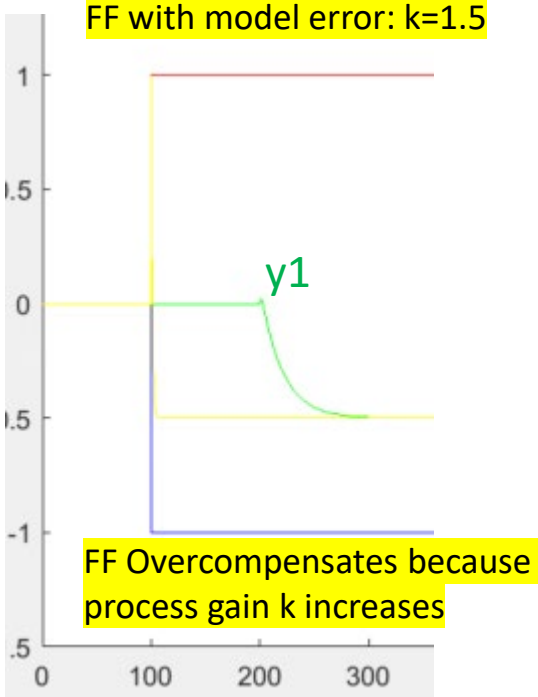
Ideal FF: $C_{ff,ideal} = -g_d / (g_{dm}\, g) = -\frac{G_1}{1 \cdot \frac{k}{s+1} \cdot G_1} =$ -(s+1)

(more zeros than poles, not realizable)

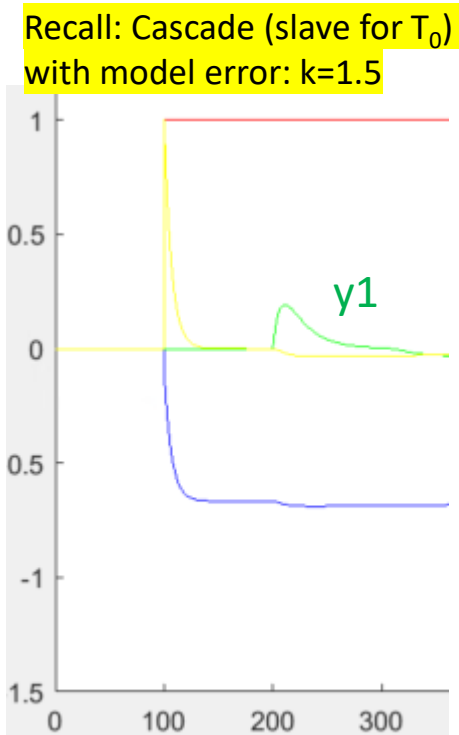
Simplified: C_{ff} = -1



Feedforward almost perfect (nominal)
(Kff=-1, Kc2=0)



Feedforward with model error
(Change gain in heater from 1 to 1.5)



Cascade with same model error
(Kc2=0.1, Kff=0)

Conclusion: Feedforward sensitive to model error. Feedback/cascade robust.

Example 2 FF (extra). Try different simple c_{FF}

- g = 5*exp(-2*s)/((3*s+1)*(1.5*s+1)) % Long delay: Feedback alone is not OK
- gd = 2*exp(-4*s)/(3*s+1) % Good for FF: even longer delay in gd
- gdm = 1/(0.1*s+1)

- Disturbance: d=1 occurs at t=0
- Desired: y small, u small

cff0 = -0.4*(1.5*s+1)*(0.1*s+1)*exp(-2*s)/(0.001*s+1)^2 % Cff,ideal (black)

% Great performance (y=0), but input u much too large!

cff1 = -0.4*exp(-2*s) % Simple. Take away zeros. RED

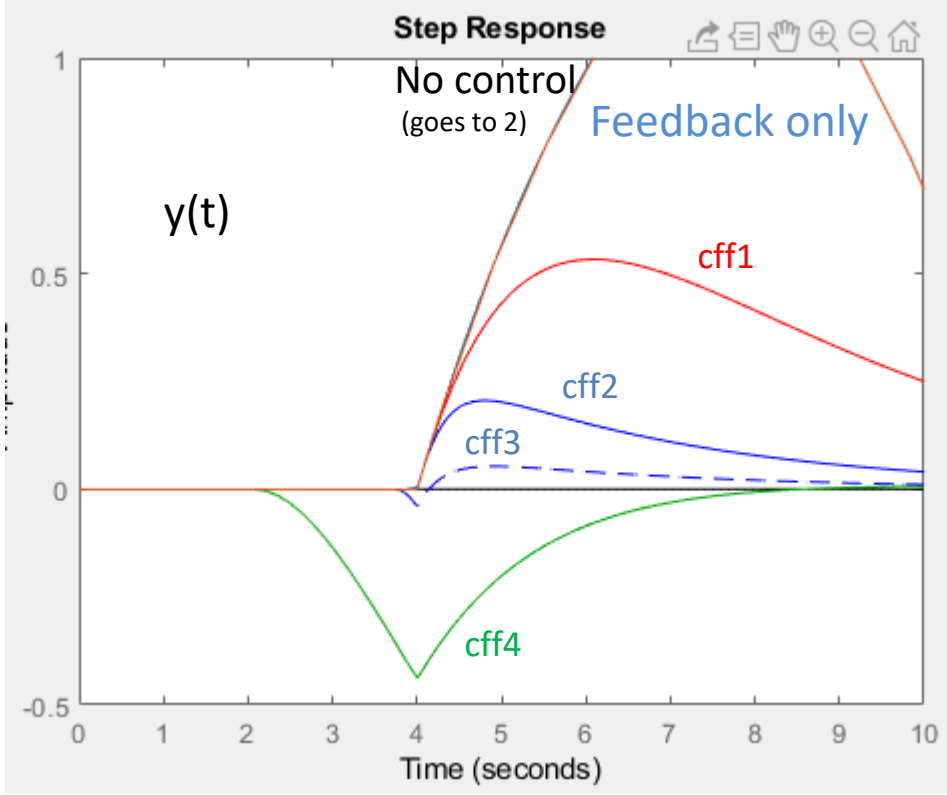
cff4 = -0.4 % % Even simpler: Static gain. Simplest (green)

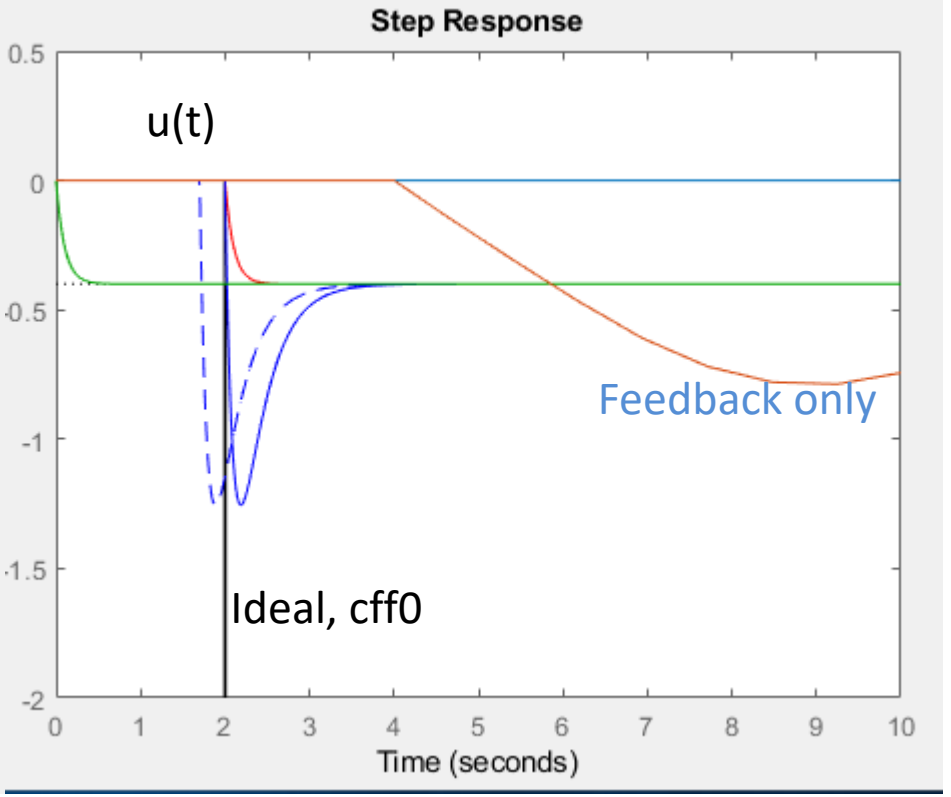
% Improvements

cff2 = -0.4*(1.5*s+1)*exp(-2*s)/(0.3*s+1) % Add (1.5s+1) +add filter to cff1, Solid blue

cff3 = -0.4*(1.5*s+1)*exp(-1.7*s)/(0.3*s+1) % Same… but reduce delay when adding filter, dashed blue

```
s=tf('s')
gd = 2*exp(-4*s)/(3*s+1)
g  = 5*exp(-2*s)/((3*s+1)*(1.5*s+1))
gdm = 1/(0.1*s+1)

cff0 = -0.4*(1.5*s+1)*(0.1*s+1)*exp(-2*s)/(0.001*s+1)^2  % Cff,ideal (black)
cff1 = -0.4*exp(-2*s)                                    % Simple with delay. RED
cff2 = -0.4*(1.5*s+1)*exp(-2*s)/(0.3*s+1)        % Add (1.5s+1) + filter, Solid blue
cff3 = -0.4*(1.5*s+1)*exp(-1.7*s)/(0.3*s+1)      % Adjust delay, dashed blue
cff4 = -0.4   %                                    % Simplest (green)
cffno = 0   %                                     % no control

%SIMC PI-controller
theta=2+1.5/2; tau=3+1.5/2; k=5;
tauc=theta;
Kc=(1/k)*(tau/(tauc+theta)), taui=min(tau,3*(tauc+theta))
c = Kc*(1+1/(taui*s));
S=1/(1+g*c);
Td = S*gd;                  % with PI feedback only
Tdu = -c*S*gd;

T0 = g*cff0*gdm + gd
T1 = g*cff1*gdm + gd
T2 = g*cff2*gdm + gd
T3 = g*cff3*gdm + gd
T4 = g*cff4*gdm + gd
T5 = S*T1                   % combine Cff=-0.4 with feedback
Tno = g*cffno*gdm + gd

T0u = gdm*cff0
T1u = gdm*cff1
T2u = gdm*cff2
T3u = gdm*cff3
T4u = gdm*cff4
T5u = S*T1u + Tdu   % combine Cff=-0.4 with feedback
Tnou = gdm*cffno

figure(1),step(T0,'black',T1,'red',T2,'blue',T3,'--',T4,'green',T5, 'magenta', Tno,Td), axis([0 20 -0.5 1])
figure(2),step(T0u,'black',T1u,'red',T2u,'blue',T3u,'--',T4u, 'green ', T5u, 'magenta', Tnou,Tdu),axis([0 20 -2 0.5])
```





cff0 = -0.4*(1.5*s+1)*(0.1*s+1)*exp(-2*s)/(0.001*s+1) % cff,ideal (black) PERFECT y! But Not realistic. TOO LARGE u

cff4 = -0.4 % Simplest: static gain; overcompensates (y in wrong direction) but otwerwise OK (green)

cff1 = -0.4*exp(-2*s) % Add delay. Surprisinlgly, it does not really help compared to cff4 (red)

cff2 = -0.4*(1.5*s+1)*exp(-2*s)/(0.3*s+1) % Add (1.5s+1) + add filter to avoid too large u, Good! (Solid blue)

cff3 = -0.4*(1.5*s+1)*exp(-1.7*s)/(0.3*s+1) % Adjust delay so 1.7+0.3=2. Very good! (BUT with perfect model) (dashed blue)

- Feedback only is too slow!
- All simulations are without model error.
- Note simplest case (green line): OK response (compared to feedback only), but y(t) goes in opposite direction initially because it acts too early. This may confuse operators.
- MPC may be a better approach in this case (it is generall good for FF)

Main problem Feedforward (FF): need good models

"If process gain increases by more than a factor 2, then ideal feedforward control is worse than no control"

(makes sense from Shower example! If FF overcompensates too much then it is worse than with no control)

- Proof: $y = g\,u + g_d\,d$ where $u = c_{FF}\,d$
- Let $c_{FF} = -g_d/g$
- Ideal: Get $y = g\,c_{FF}\,d + g_d\,d = 0$

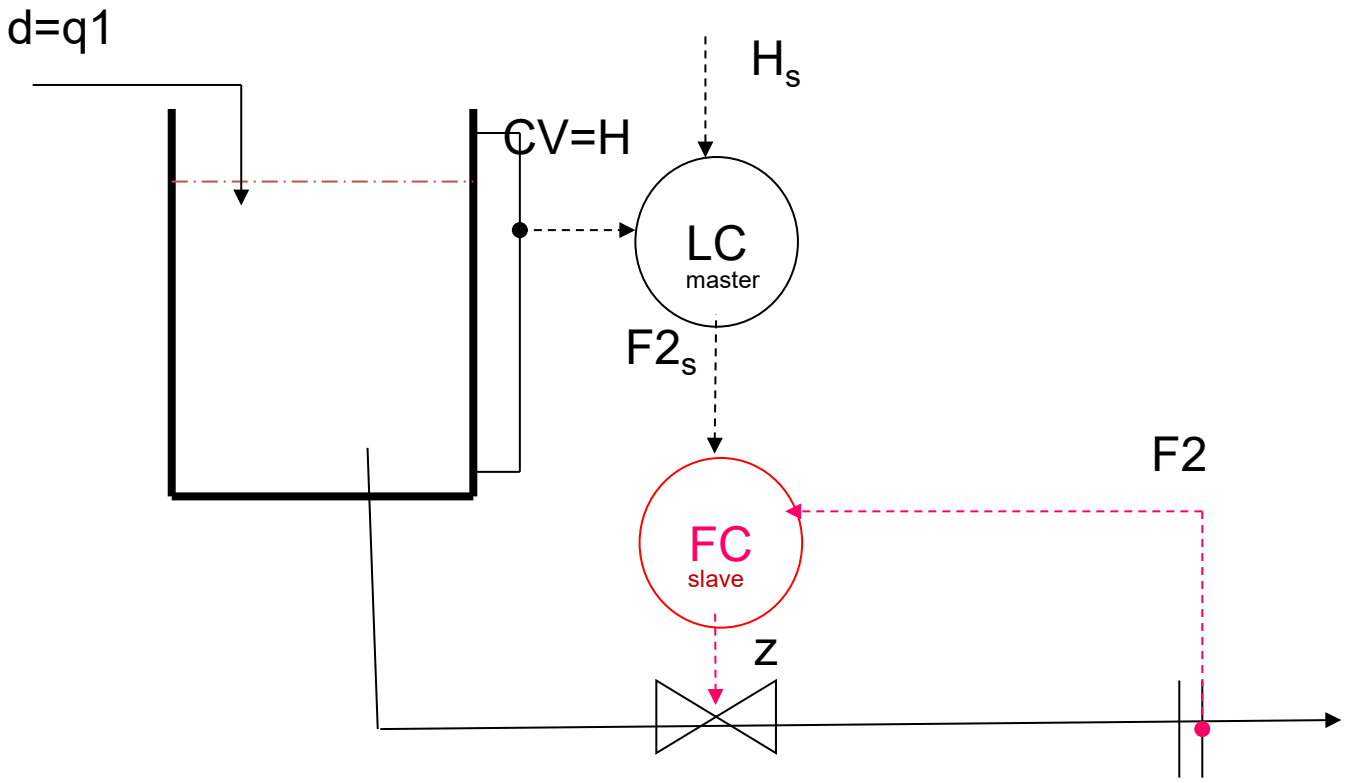
  so term "$g\,c_{FF}\,d$" is equal to "$-g_d\,d$"
- Real: If g has increased by a factor x then

  $y = x(-g_d\,d) + g_d\,d = (-x+1)\,g_d\,d$

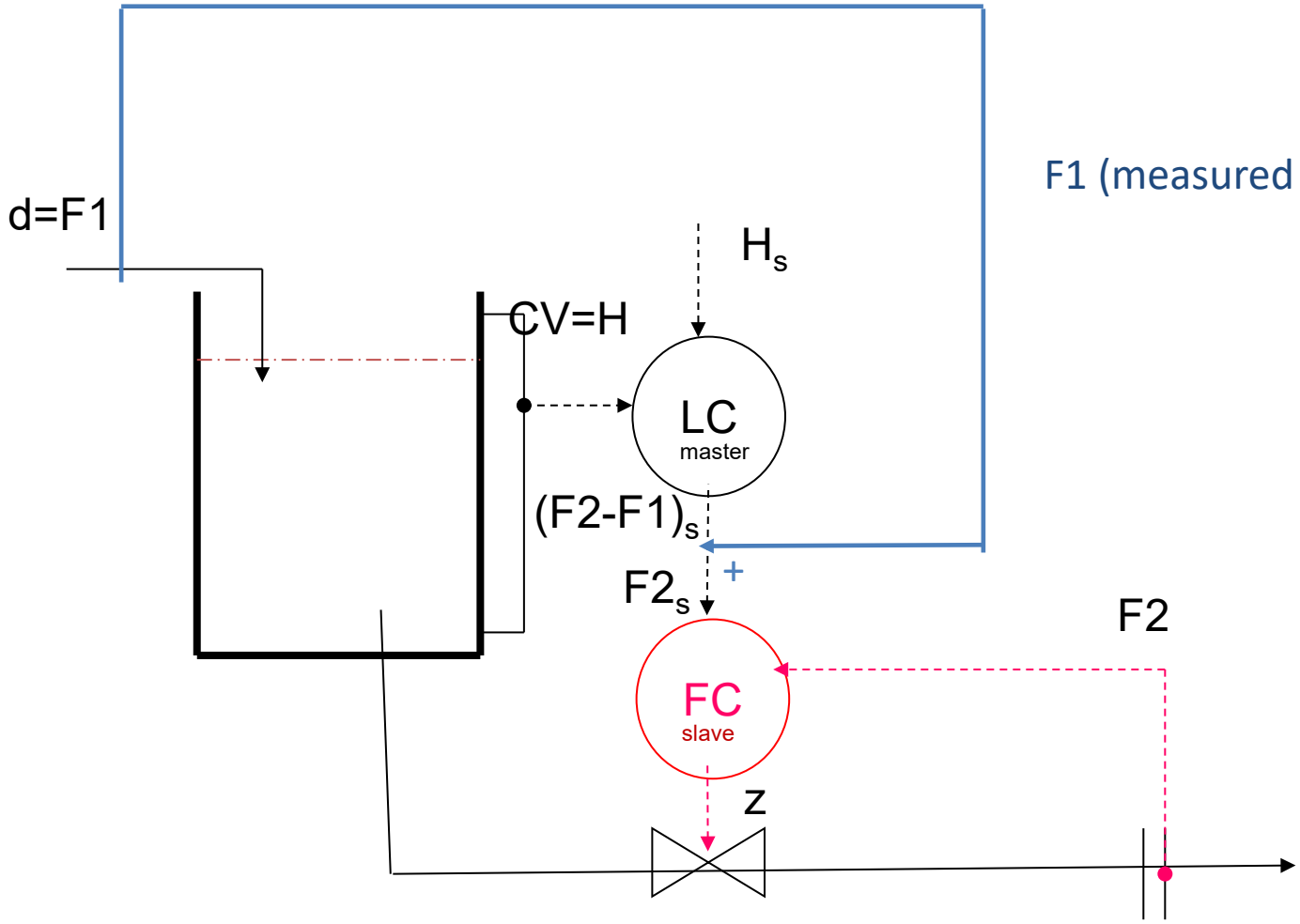
And we have that $|-x+1| > 1$ (worse than no control) for $x > 2$.

- Example: $x = 2.1$, $g_d\,d = 1$
  - Ideal $y = g_d\,d = 1$
  - Real: $y = (-2.1+1)*1 = -1.1$ (which is greater than 1 in magnitude, so y overshoots y by more than 1 on the other side)

Quiz: How can we add feedforward?



Solution: How can we add feedforward?

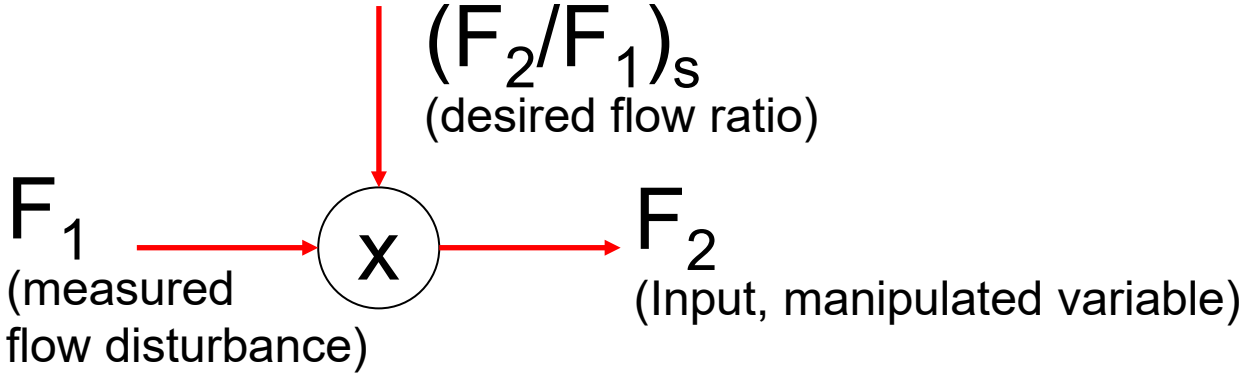


F1 (measured flow disturbance)

E2. Ratio control (most common case of feedforward)

- Note: Disturbance needs to be a flow (or more generally an extensive variable)
- Very common when a unit has several feed streams.
- Example: Process with two feeds F_1(d) and F_2 (u), where ratio should be constant.

Use multiplication block (x):



"Measure disturbance (d=F_1) and adjust input (u=F_2) such that ratio is at given value $(F_2/F_1)_s$"

Don't need a model; just need process insight (about when it can be used)

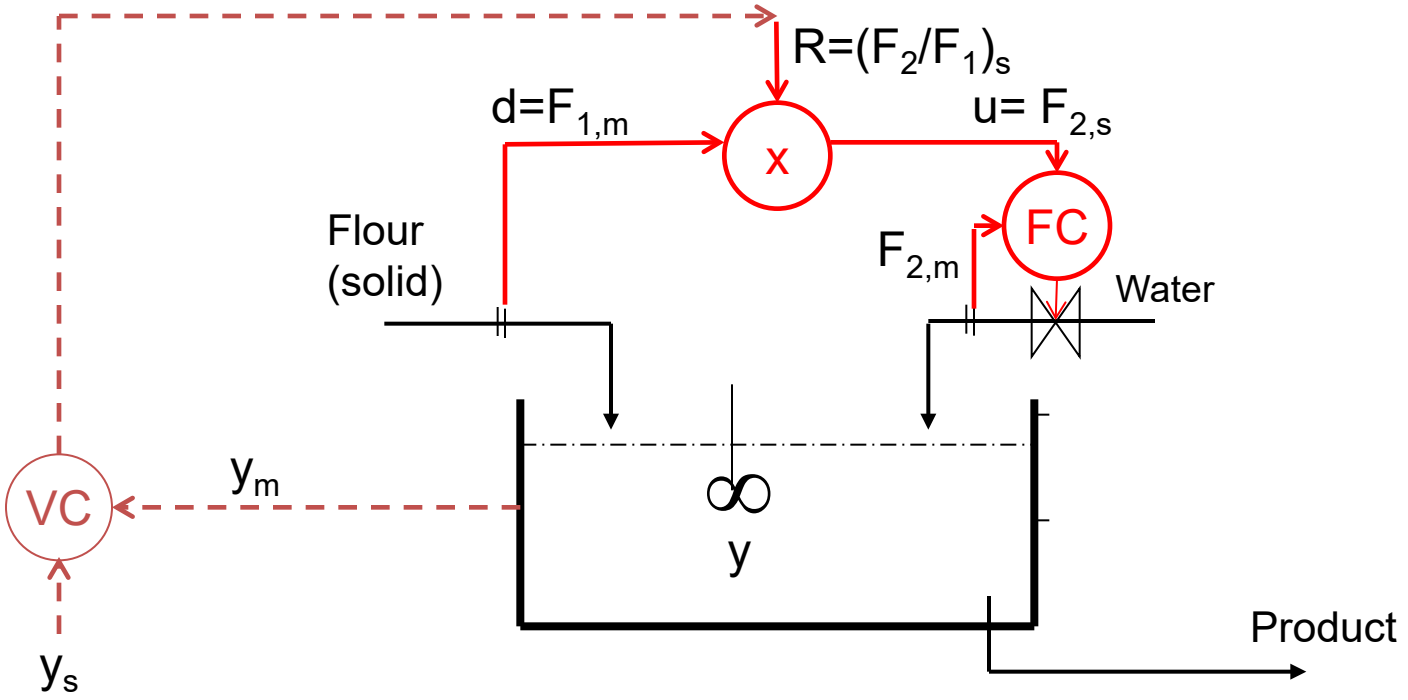
Usually: Combine ratio (feedforward) with feedback

- Adjust $(F1/F2)_s$ based on feedback from process, for example, composition controller.
  - This is a special case of cascade control

  - ***Example cake baking****: Use recipe (ratio control = feedforward), but adjust ratio if result is not as desired (feedback)*
  - ***Example evaporator:*** *Fix ratio q_H/q_F (and use feedback from T to fine tune ratio)*

EXAMPLE: CAKE BAKING MIXING PROCESS

RATIO CONTROL with outer feedback (to adjust ratio setpoint)



Feedback correction («trim»)

Want to control: Viscosity y [cP]
(or any intensive quality variable, like c, ρ or T)

Note : This way of implementing ratio control makes it easy to tune the outer feedback loop (here: VC) because the gain from MV = R_s= $(F2/F1)_s$ to CV=y (here: viscosity) does not depend on the flowrate (that is, it is independent of disturbances in F1).

Ratio control

- Avoid divisions in implementation! (can divide by 0)
- Book has some strange suggestions, for example, Figure 14.5

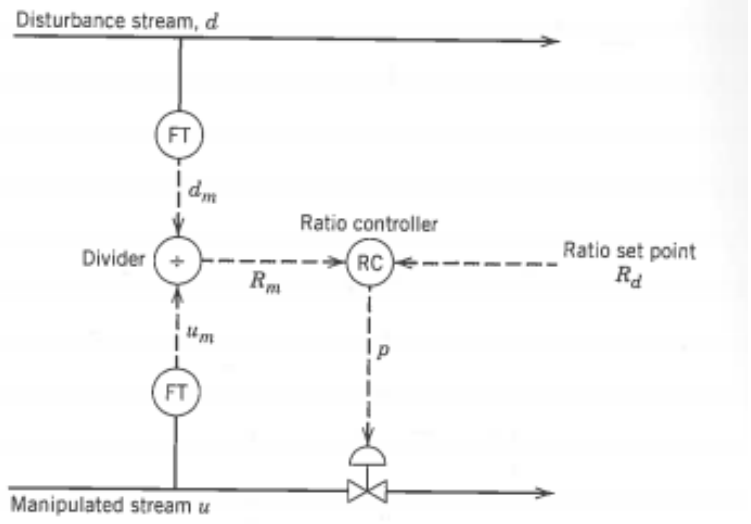


Figure 14.5 Ratio control, Method I.

Bad solution

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

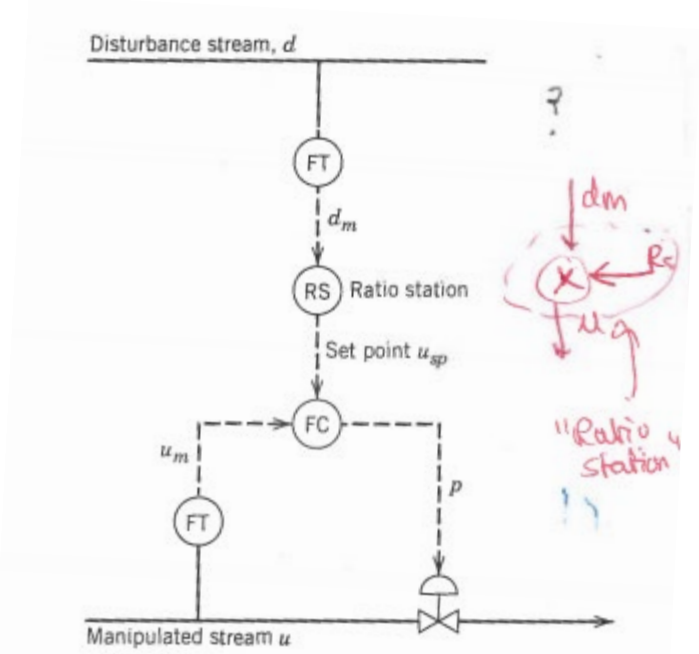


Figure 14.6 Ratio control, Method II.

This is complicated. What is RS?

Ok if implemented as shown in red at right

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
  - May use VPC for MV-MV switching: see later

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)

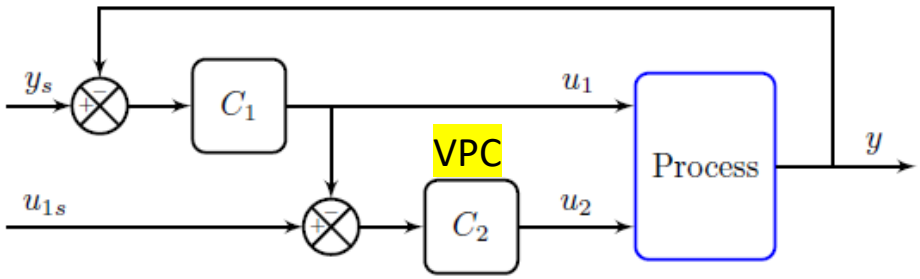


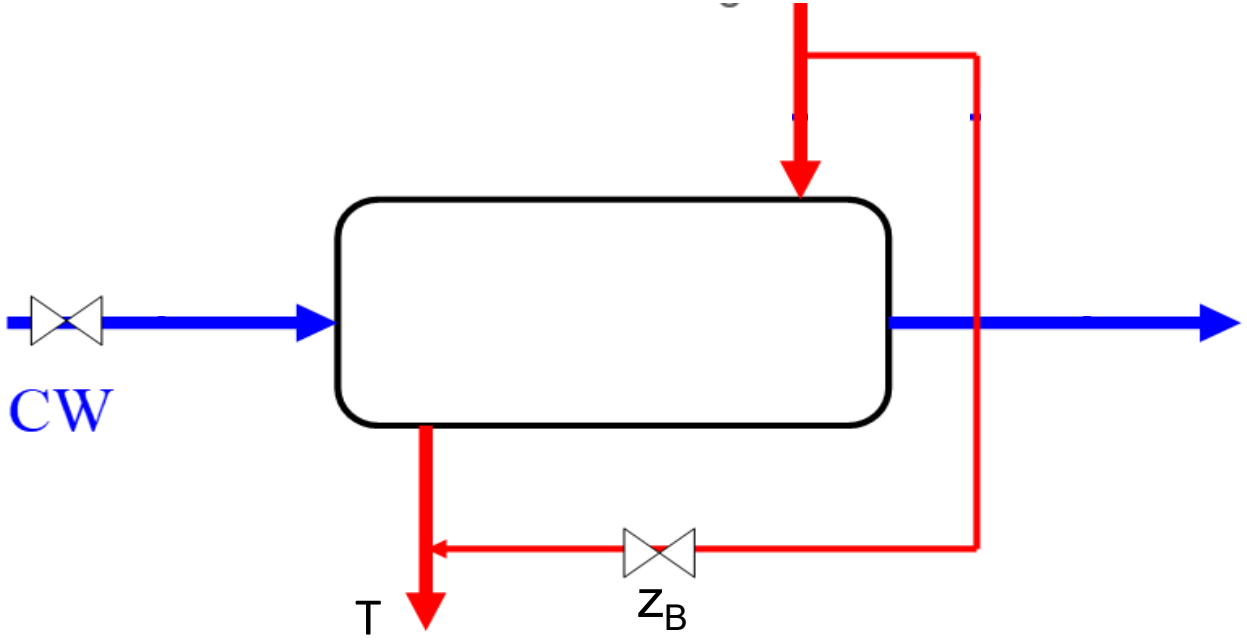
Figure 12: Valve (input) position control (VPC) for the case when an "extra" MV (u_1) is used to improve the dynamic response. A typical example is when u_1 is a small fast valve and u_2 is a large slower valve.
C_1 = fast controller for y using u_1.
C_2 = slow valve position controller for u_1 using u_2 (always operating).
u_{1s} = steady-state resting value for u_1 (typically in mid range. e.g. 50%).

Alternative term for dynamic VPC:
- Mid-ranging control (Sweden)

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)

Example: Heat exchanger with bypass

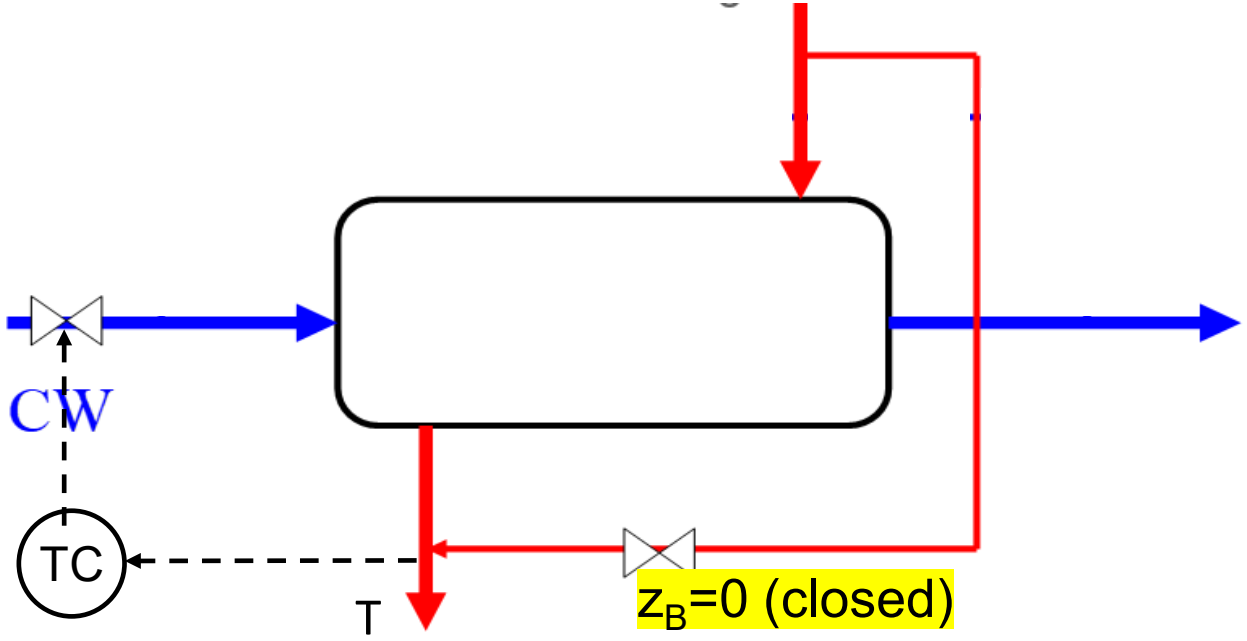


Want tight control of y=T.

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

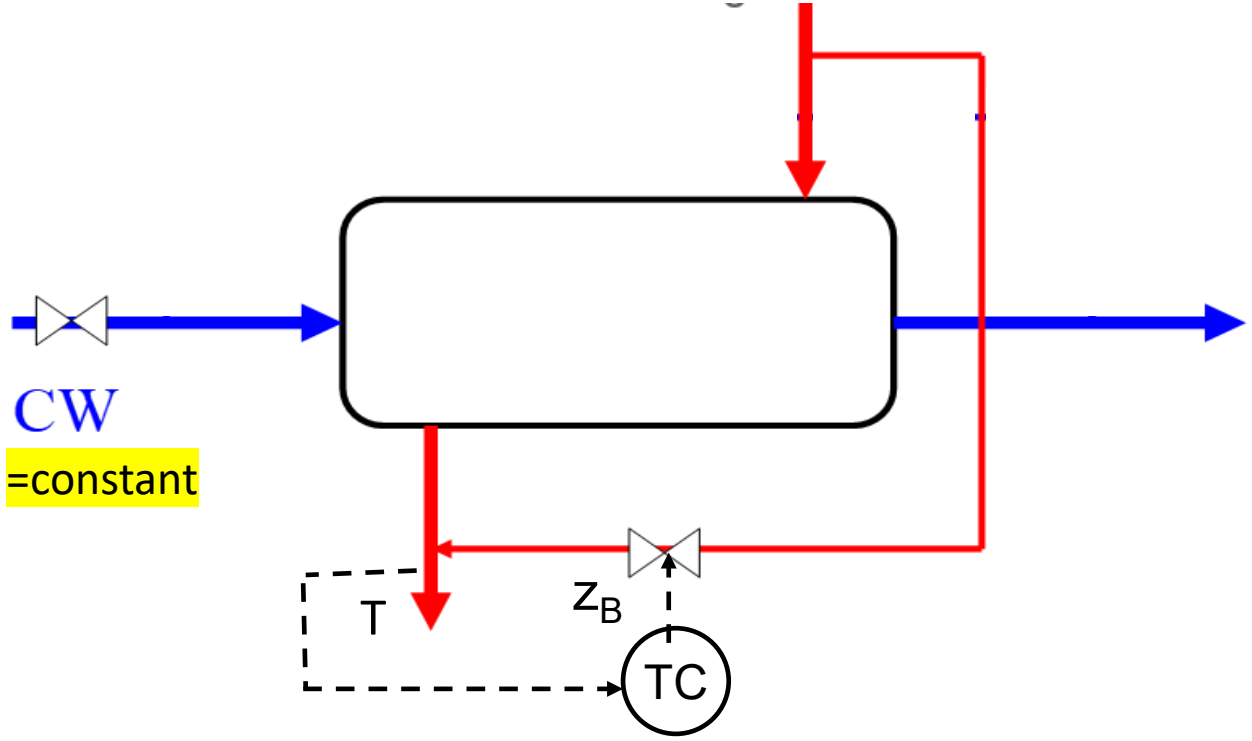
Proposed control structure?

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)

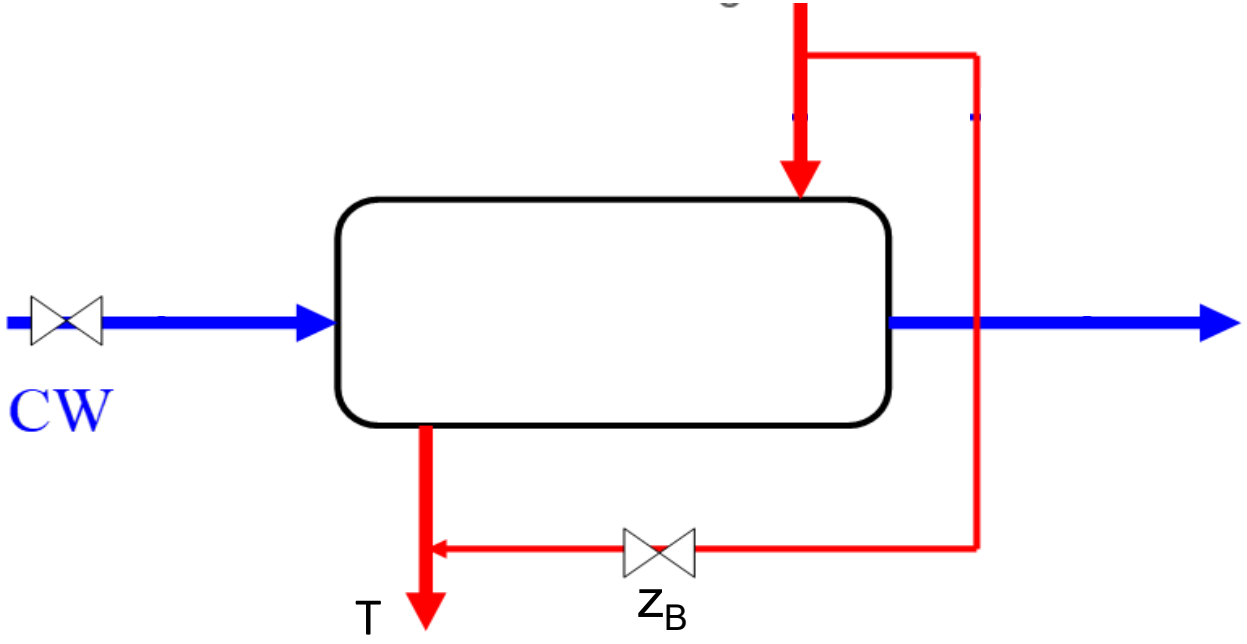


Advantage: Very fast response (no delay)

Problem: z_B is too small to cover whole range

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

What about VPC?

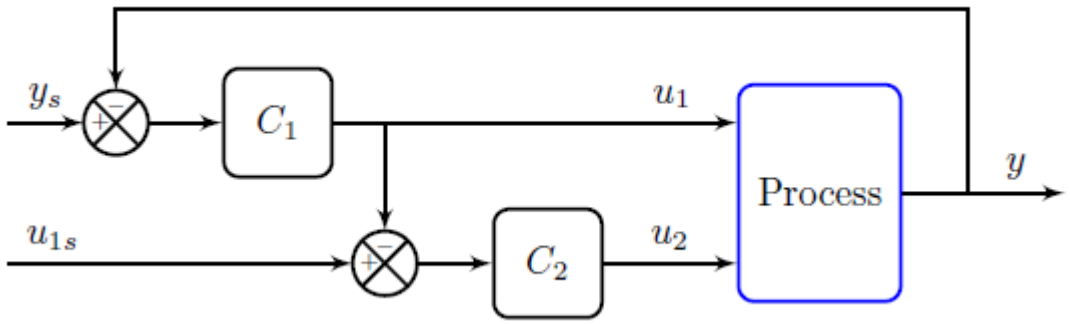


Want tight control of y=T.

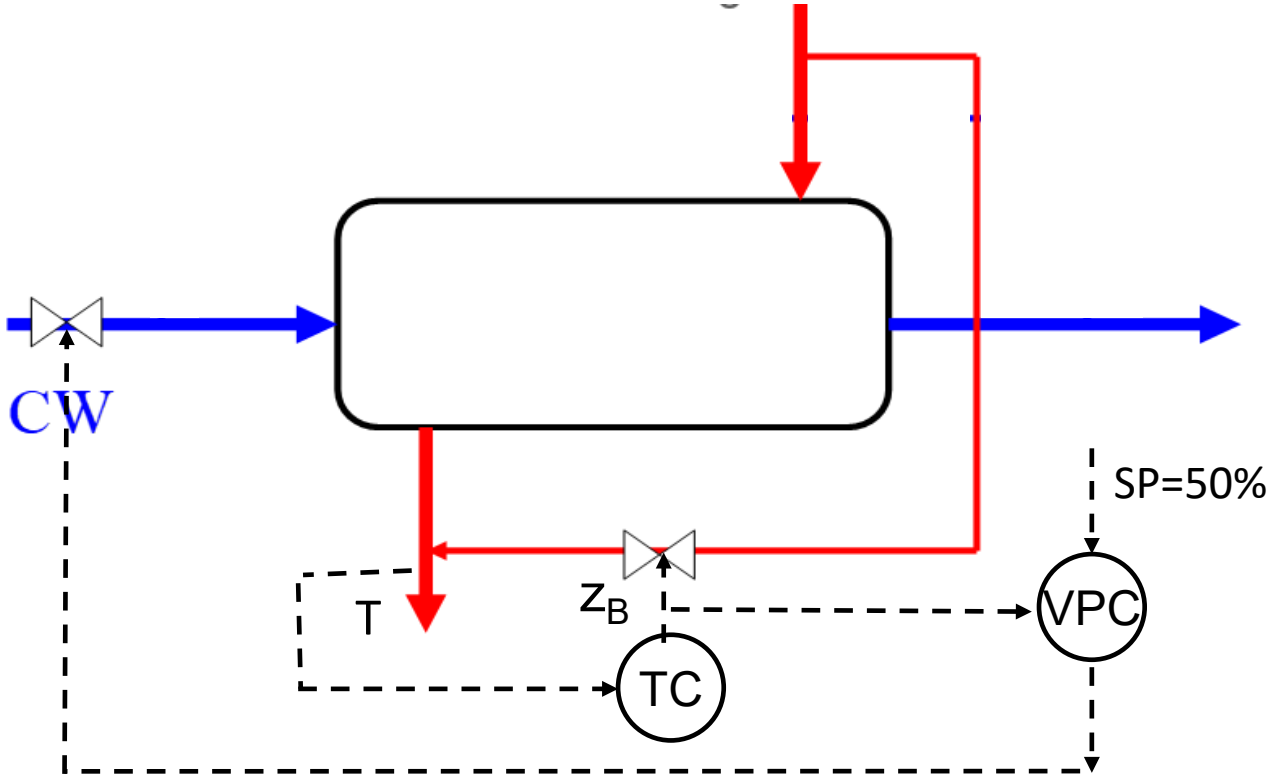
- $u_1 = z_B$
- u_2=CW

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

Standard Advanced control elements

Sigurd Skogestad, "Advanced control using decomposition and simple elements".
Annual Reviews in Control, vol. 56 (2023), Article 100903 (44 pages).

- 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 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

[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.