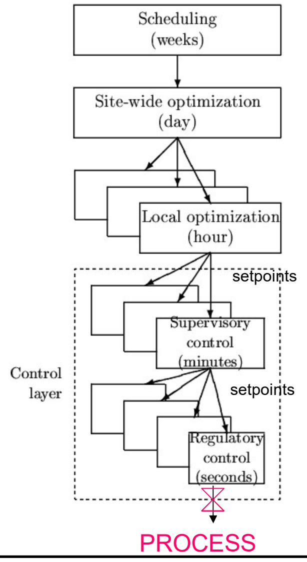
The decision hierarchy is based on “time scale separation”



Scheduling (weeks)

Site-wide optimization (day)

Local optimization (hour)

setpoints

Supervisory control (minutes)

Control layer

setpoints

Regulatory control (seconds)

PROCESS

“Advanced control” (MPC)

Fast “regulatory” control (PID)

“Advanced control” If single-loop feedback control (PID) alone is not good enough

Design based on simple elements

1. Cascade control (measure and control internal variable, y_2)
2. Feedforward control (measure disturbance, d)
    - Including ratio control
3. Selectors (max,min)
4. Input resetting (=midranging = valve position control)
5. Split range control
6. Multivariable control
    - Single-loop control (decentralisexd)
        - RGA
    - Decoupling
        - For interactive process (where RGA is unfavorable)

1. Cascade control

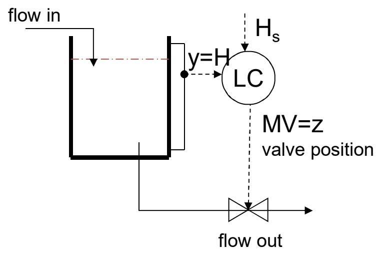
Idea: make use of extra “local” output measurement (y_2)

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

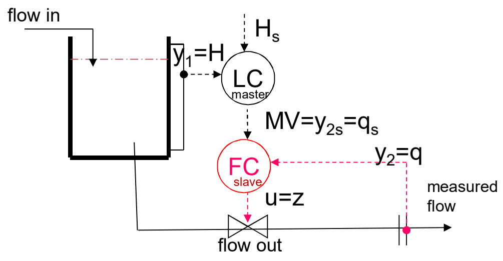
- Without cascade: “Master” controller directly adjusts u (input, MV) to control y (primary output, sometimes called y_1)
- With cascade: Local “slave” controller uses u to control “extra”/fast measurement (y_2). *
  “Master” controller adjusts setpoint y_{2s}.

- **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

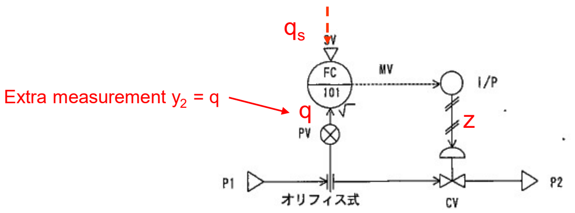


WITH CASCADE

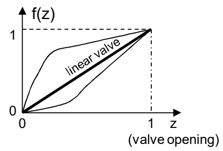


*Comment: Another approach that uses extra measurements to improve control is «**Full state feedback**».

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

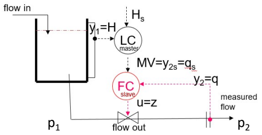


$$\text{Flow rate: } q = C_v f(z)\sqrt{\frac{p_1 - p_2}{\rho}} \quad [\mathrm{m}^3/\mathrm{s}]$$



1. Counteracts nonlinearity in valve, f(z)
   - 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)

Block diagram



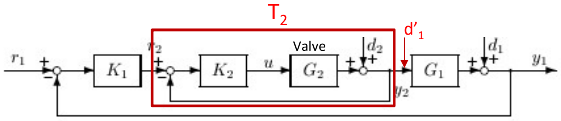


Figure 10.11: Common case of cascade control where the primary output y_1 depends directly on the extra measurement y_2

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
K_2 = Flow controller
$T_2 = G_2K_2/(1+G_2K_2) \sim 1/(\tau_{C2}\, s + 1)$

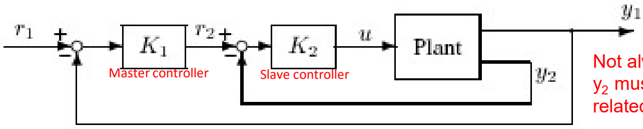
Counteracting nonlinearity using cascade control:

Process gain variation (G_2) -> Closed-loop time constant variation in slave loop (T_2)

Proof:

- Slave controller with u_2 = z (valve position) and y_2=q (flow) .
- Assume nonlinear valve, so slope of valve characteristic f(z) varies with z
- Valve model: First order with varying gain k (k is given by slope of f(z)) :
  $G_2 = k(z) / (\tau s+1)$
- K_2: PI-controller with gain K_c and integral time $\tau_I = \tau$.
  $K_2(s) = K_c(\tau s+1) / (\tau s)$
- Loop transfer function
  $L_2 = G_2K_2 = K_c k / (\tau s) = 1 / (\tau_{c2} s)$
  where $\tau_{C2} = \tau/(k\, K_c)$
- With slave (flow) controller:
  - T_2= Transfer function from y_{2s} to y_2 («effective input dynamics» for master loop):
    $T_2 = L_2/(1+L_2) = 1/(\tau_{C2}\, s + 1)$
- **So variation in k translates into variation in τ_{C2}**
  - In practise this gives a variation in the effective time delay in the master loop
  - Low gain k for valve gives large τ_{C2} (bad)
  - IMPORTANT: Need τ_{C1} (time constant master) > 5 τ_{C2} **(approx.)** for the variation in τ_{C2} **to be unimportant, so inner loop should be fast,**

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")

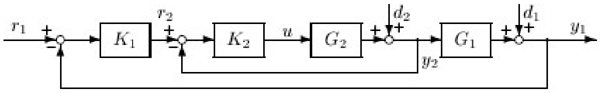


Figure 10.11: Common case of cascade control where the primary output y_1 depends directly on the extra measurement y_2

From book (series cascade control):

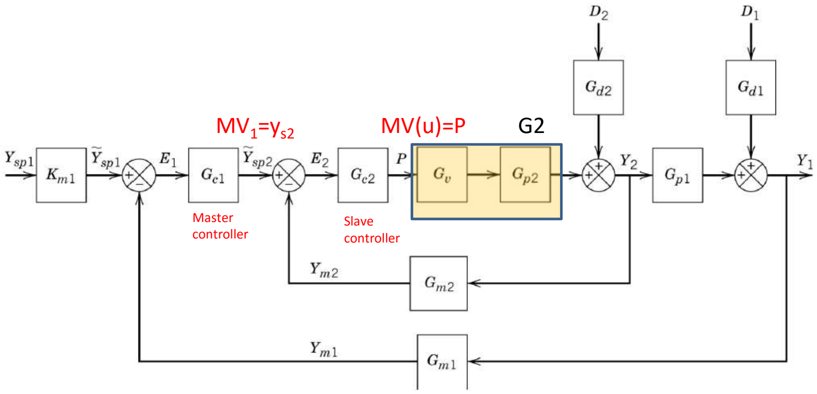


Figure 15.4

When use (series) cascade ?

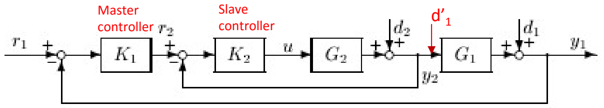


Figure 10.11: Common case of cascade control where the primary output y_1 depends directly on the extra measurement y_2

1. Disturbances d_2 arising within the secondary loop (before y_2) are corrected by the secondary controller before they can influence the primary variable y_1
2. Phase lag existing in the secondary part of the process (G_2) is reduced by the secondary loop. This improves the speed of response of the primary loop.
   - Example: Double integrating process. G_2=1/s, G_1= $e^{-\theta s}$/s
3. Gain variations in G_2 are overcome within its own loop.

Thus, use cascade control (with an extra secondary measurement y_2) when:

1. The disturbance d_2 is significant
2. G_1 has a large effective delay
3. The plant G_2 is nonlinear or varies with time or is uncertain.

Design / tuning (see also in tuning-part):

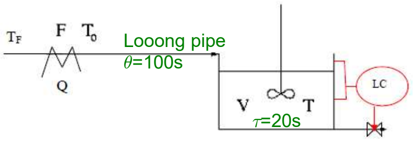
- First design K_2 ("fast loop") to deal with d_2
- Then design K_1 to deal with d_1

Example: Flow cascade for level control
u = z, y_2=q, y_1=H,
K_1= LC, K_2= FC

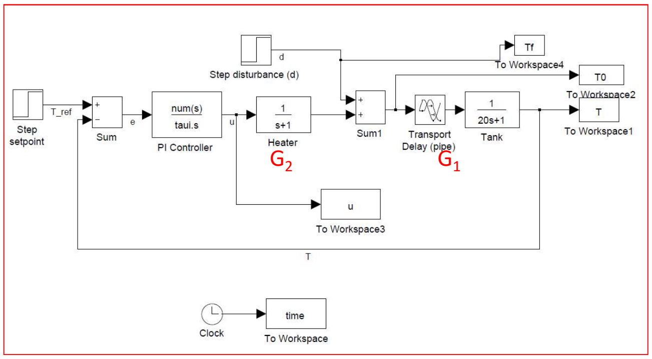
Tuning for common "series cascade"

- First tune fast inner loop ("slave")
  - Design K_2 based on model G_2
- Then with slave closed, tune slower outer loop ("master"):
  - Design K_1 based on model T_2*G_1
    - where T_2 = G_2 K_2/(1+G_2 K_2) is closed-loop response from y_{2s} to y_2
    - With SIMC, T_2 ~ $e^{-\theta_2 s}/(\tau_{c2}$ s+1)
      - Comment: Note that T_2 has gain 1 provided K_2 has integral action (independent of G_2!), which explains why cascade control counteracts nonlinearity in G_2

Example 1: Similar to shower process



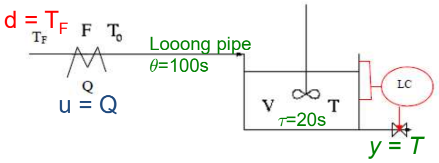
$u = Q$
$y = T$
$d = T_F$

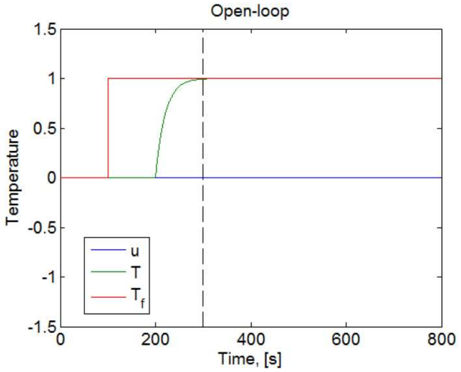


Simulink model: tunepid1_ex1

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

Disturbance response with no control





$u = Q$
$y = T$
$d = T_F$

```
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])
```

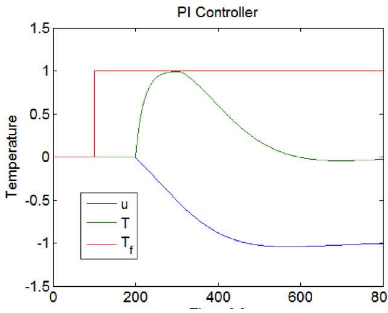
Without cascade: SIMC PI control

G = G_1G_2= exp(-100s)/(20s+1)(s+1)

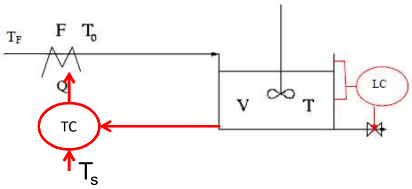
11) SIMC PI tuning rule with $\tau_c = \theta = 100$.

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

u = Q
y = T
d = T_F

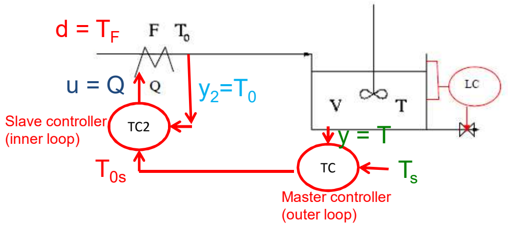


No offset



```
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 T_0: Cascade control is much better



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

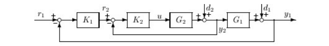
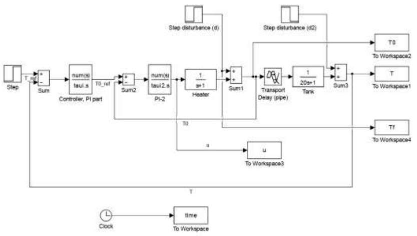
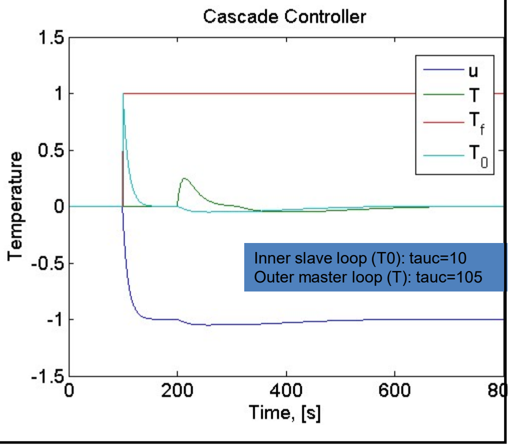


Figure 10.11: Common case of cascade control where the primary output y_1 depends directly on the extra measurement y_2



```
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])
```



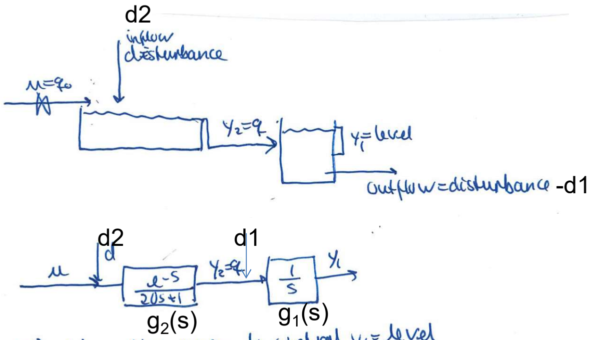
Inner slave loop (T0): tauc=10
Outer master loop (T): tauc=105

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

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).

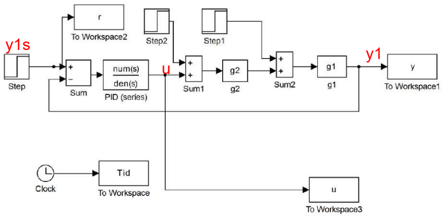
Example 2
(almost double integrating process)



Objective: Use u=q0 to control y1=level
cascade = Add flow controller for y2=q (inner loop)

PI-control: Without cascade

- Integrating process with large effective delay -> control poor



```
s=tf('s')
g1 = 1/s
g2 = exp(-s)/(20*s+1)
```

Without cascade:

$y_1 = gu$ where $g = g_1 g_2 = \frac{e^{-s}}{s(20s+1)}$

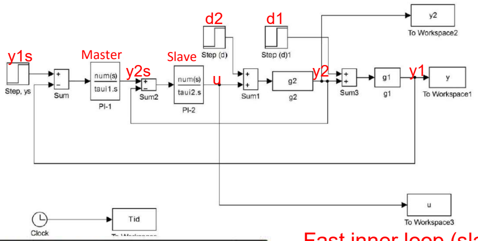
Half rule for integrating process (time constant $\tau \to \infty$): $g \approx \frac{e^{-11s}}{s}$

SIMC PI-tunings with $\tau_c = \theta = 11$ (Note: large effective delay!):

$K_c = \frac{1}{k'}\frac{1}{\tau_c+\theta} = \frac{1}{1}\frac{1}{11+11} = 0.045$

$\tau_I = \min\{\tau, 4(\tau_c+\theta)\} = \min\{\infty, 4(11+11)\} = 88$

PI-control: With cascade



Fast inner loop with $y_2 = q$ (control of flow between tanks):

$y_2 = g_2 u$ where $g_2 = \frac{e^{-s}}{(20s+1)}$

SIMC PI-tunings. We are a bit conservative and use $\tau_c = 2\theta = 2$:
$K_{c2} = \frac{1}{k}\frac{\tau}{\tau_c+\theta} = \frac{1}{1}\frac{20}{2+11} = 6.7$
$\tau_{I2} = \min\{\tau, 4(\tau_c+\theta)\} = \min\{20, 4(2+1)\} = 12$

Fast inner loop (slave loop): Takes care of disturbances inside slave loop (d2).

Also eliminates nonlinearity in g2: provides linear response from y2s (MV for master) to y2

In this case:
G_2 has dynamics so also have benefit of faster outer loop (master loop has tauc reduced from 11 to 3):
Get better rejection also of disturbances outside slave loop (d1) + better setpoint response (y1s)

Slower outer loop with y_1 = level and MV $= y_{2s} = q_s$:

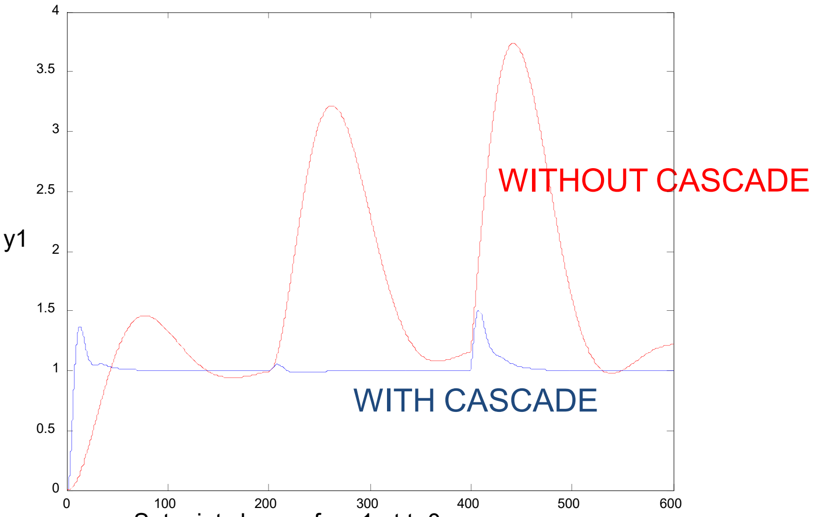
$y_1 = \hat{g}_1 y_{2s}$

Model $\hat{g}_1$ can be found experimentally with inner loop closed.

Alternative: use model. For series cascade process:
$\hat{g}_1 = g_1 \frac{c_2 g_2}{1+c_2 g_2}$ where $g_1 = \frac{1}{s}$
Approximation of inner loop (SIMC): $\frac{c_2 g_2}{1+c_2 g_2} \approx \frac{e^{-\theta s}}{\tau_c s+1} = \frac{e^{-s}}{2s+1}$
Resulting model using half rule: $\hat{g}_1 = \frac{e^{-s}}{s(2s+1)} \approx \frac{e^{-2s}}{s}$

SIMC PI-tunings. We are again bit conservative and use $\tau_c = 1.5\theta = 3$:
$K_{c1} = \frac{1}{k'}\frac{1}{\tau_c+\theta} = \frac{1}{1}\frac{1}{3+2} = 0.2$
$\tau_{I1} = \min\{\tau, 4(\tau_c+\theta)\} = \min\{\infty, 4(3+2)\} = 20$

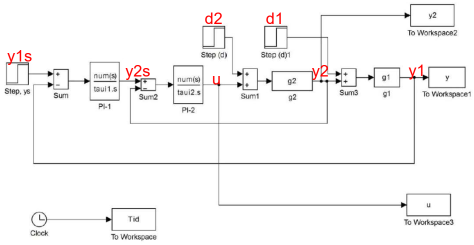
Simulation cascade control



Setpoint change for y1 at t=0
Disturbance d2=0.1 (at input to g2, inside slave loop) at t=200
Disturbance d1=0.1 (at input to g1, outside slave loop) at t=400

Details

PI-control: With cascade



1. Fast inner loop (slave) with $y_2 = q$ (control of flow between tanks):

$y_2 = g_2 u$ where $g_2 = \frac{e^{-s}}{(20s+1)}$

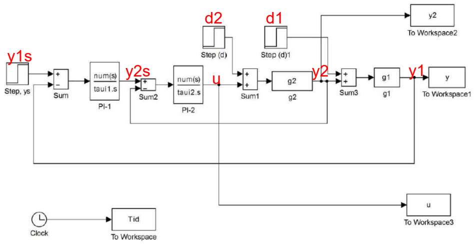
SIMC PI-tunings. We $\tau_c = 2\theta = 2$ (which seems a bit conservative, but it actually gives a "speedup" of a factor 20/2=10 compared to the open-loop):

$K_{c2} = \frac{1}{k}\frac{\tau}{\tau_c+\theta} = \frac{1}{1}\frac{20}{2+11} = 6.7$

$\tau_{I2} = \min\{\tau, 4(\tau_c+\theta)\} = \min\{20, 4(2+1)\} = 12$

Details

PI-control: With cascade



2. **Slower outer loop (master)** with $y_1 =$ level and MV $= y_{2s} = q_s$:

$y_1 = \hat{g}_1 y_{2s}$

Model $\hat{g}_1(c_2)$ can be found experimentally with inner loop closed.

Alternative: use model. For series cascade process:

$\hat{g}_1 = g_1 \frac{c_2 g_2}{1+c_2 g_2}$ where $g_1 = \frac{1}{s}$

Approximation of inner loop (SIMC): $\frac{c_2 g_2}{1+c_2 g_2} \approx \frac{e^{-\theta_2 s}}{\tau_{c2}s+1} = \frac{e^{-s}}{2s+1}$

Resulting model (for tuning of master loop) using half rule: $\hat{g}_1 = \frac{e^{-s}}{s(2s+1)} \approx \frac{e^{-2s}}{s}$

SIMC PI-tunings. We are again bit conservative and use $\tau_c = 1.5\theta = 3$:

$K_{c1} = \frac{1}{k'}\frac{1}{\tau_c+\theta} = \frac{1}{1}\frac{1}{3+2} = 0.2$

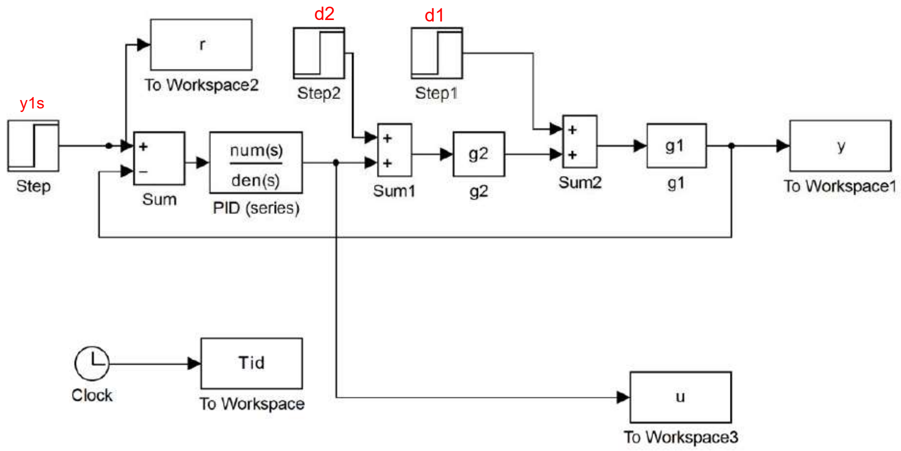
$\tau_{I1} = \min\{\tau, 4(\tau_c+\theta)\} = \min\{\infty, 4(3+2)\} = 20$

Details

WITHOUT CASCADE

s=tf('s')
g1 = 1/s
g2 = exp(-s)/(20*s+1)

Kc = 0.0455
taui = 88
taud = 0



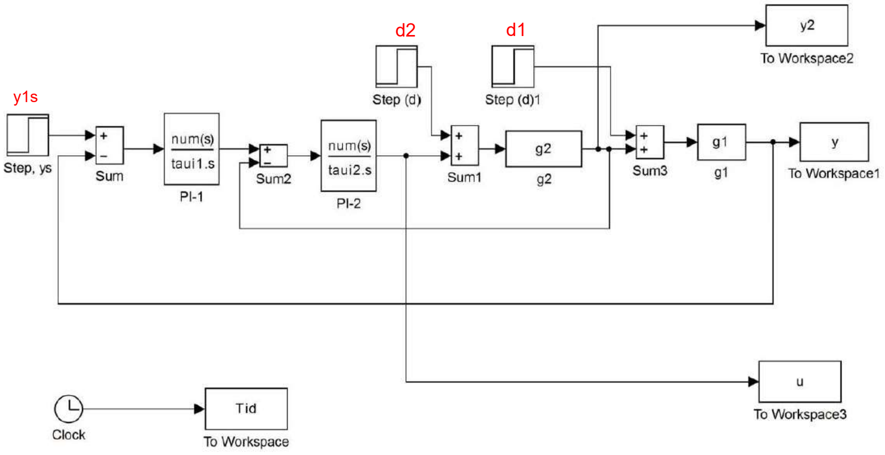
File: tunepid4_cascade0

Details

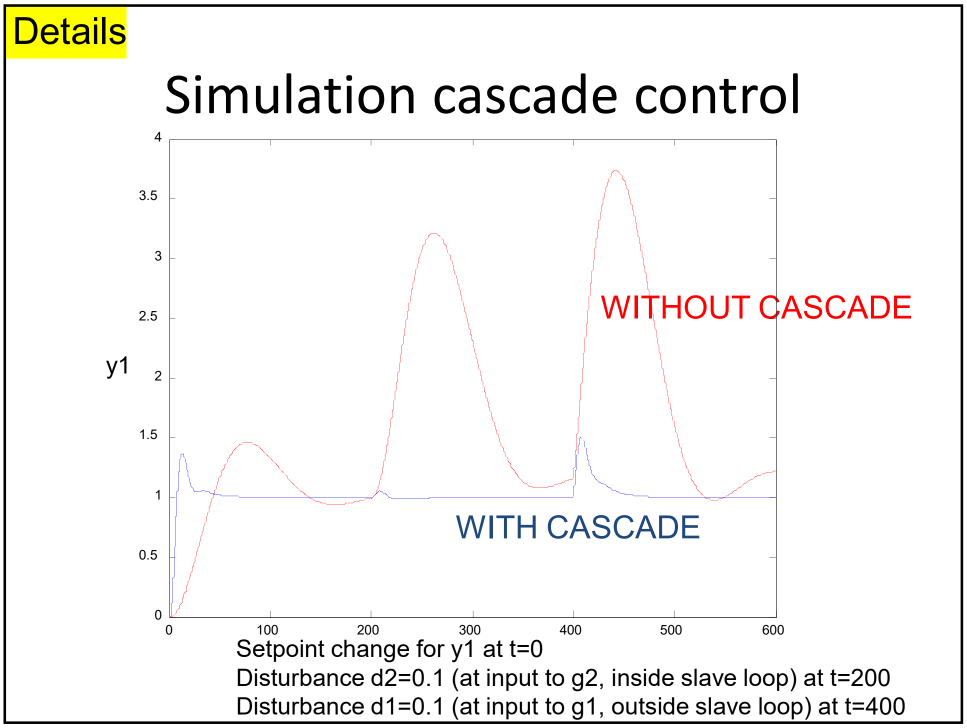
WITH CASCADE

s=tf('s')
g1 = 1/s
g2 = exp(-s)/(20*s+1)

Kc1 = 0.2000
taui1 = 20
Kc2 = 6.7000
taui2 = 12



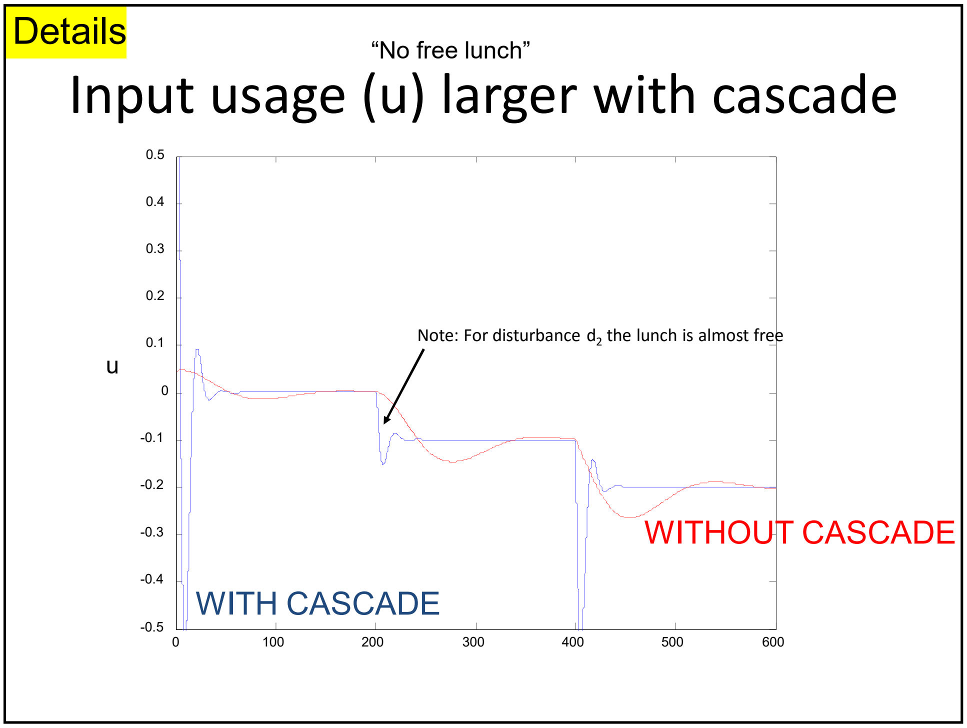
File: tunepid4_cascade



Details

Simulation cascade control

Setpoint change for y1 at t=0
Disturbance d2=0.1 (at input to g2, inside slave loop) at t=200
Disturbance d1=0.1 (at input to g1, outside slave loop) at t=400



Details

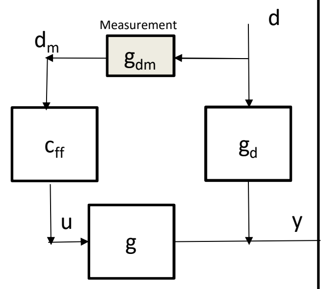
"No free lunch"

Input usage (u) larger with cascade

Note: For disturbance d_2 the lunch is almost free

2. Feedforward 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: $u = c_{FF}\,d_m$
- Get $y = (g\,c_{FF}\,g_{dm} + g_d)\,d$
- Ideal feedforward:

  $y = 0$ -> $c_{FF,ideal} = -\,(g_d / (g_{dm}\,g)$
- Actual feedforward:

  $y = (g\,c_{FF}\,g_{dm} + g_d)\,d = (1 - c_{FF}/\,c_{FF,ideal})\,g_d\,d$

  where $c_{FF}(s)$ must be realizable
    - Order pole polynomial > order zero polynomial
    - No prediction allowed (θ can not be negative)
    - And must avoid that C_{FF} has too high gain (to avoid aggressive input changes)
  - Common simplification: $c_{FF} = k$
  - 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

When use feedforward?

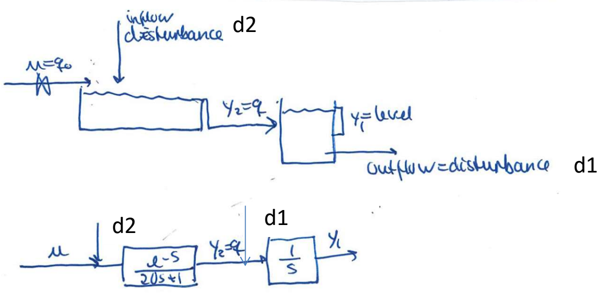
Feedforward is helpful if

1. The reason for poor feedback control is that the measurement of y has a long delay (but g itself has a short delay)
2. The disturbance response (g_d) is "slow" compared to input response (g)
   - $c_{FF,ideal} = -\,g_d / (g_{dm}\,g)$ is then realizable, which means that the feedforward has "enough time" to take the right action
   - For example, if g_d has a larger delay than g (so that $c_{FF,ideal}$ has a delay) or if g_d has a larger time constant than g

- Note: If the reason for poor feedback control is a large delay in g, then adding feedforward will not help very much

Note: g does not include the measurement dynamics for y

Example



Objective: Use u=q0 to control y1= level

Example

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

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 $c_{FF1} = -1$.

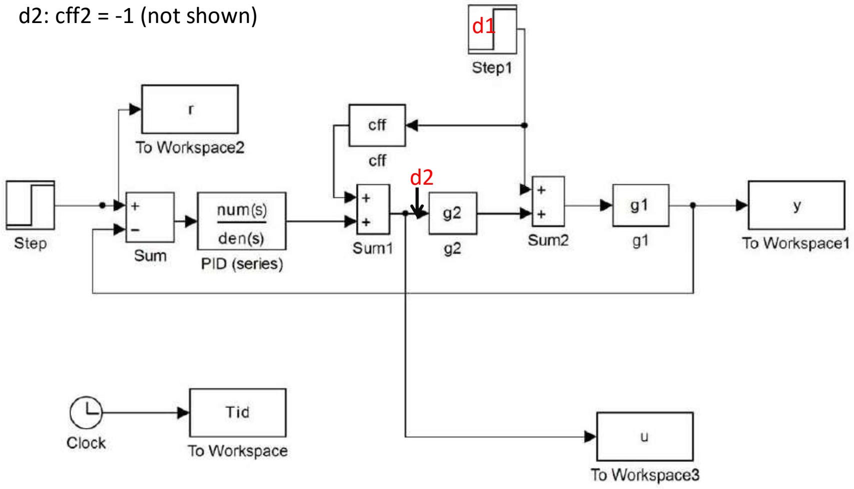
Disturbance 2:
Ideal: $c_{FF2} = -1$
Actual: $c_{FF2} = -1$

«Chicken factor»

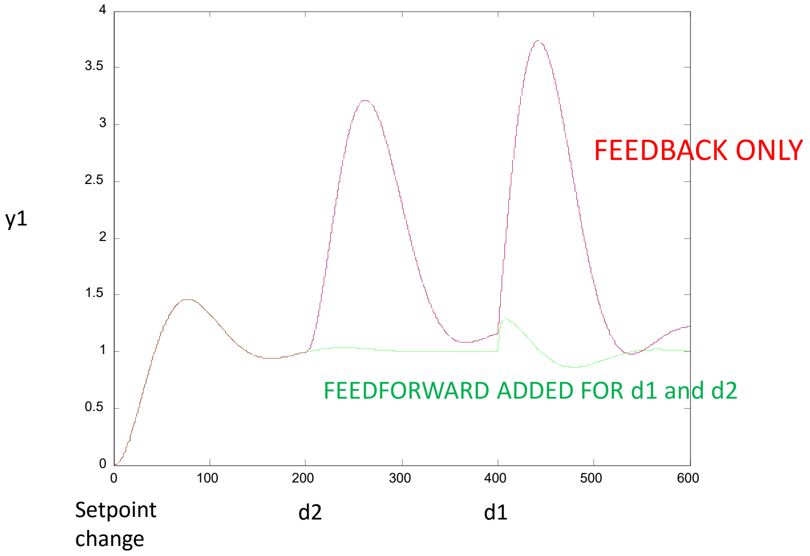
Comment: In practice, one often sets the feedforward gain about 80% of the theoretical, that is, $c_{FF2} = -0.8$. This is to avoid that the feedforward controller overreacts, which may confuse the operators. It also makes the feedforward action more robust.

Feedforward control (measure d1 & d2)

d1: cff = -(20*s+1)/(2*s+1)
d2: cff2 = -1 (not shown)



Simulation feedforward



Main problem feedforward: Need good models

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

- Proof: $y = gu + g_d\,d$ where $u=c_{FF}\,d$
- Ideal: $y = g\,c_{FF}\,d + g_d\,d = 0$
  - So want term "$g\,c_{FF}\,d$" equal to "$-g_d\,d$" (gives $c_{FF,ideal} = -g_d/g$)
- Real: If g has increased by a factor x then

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

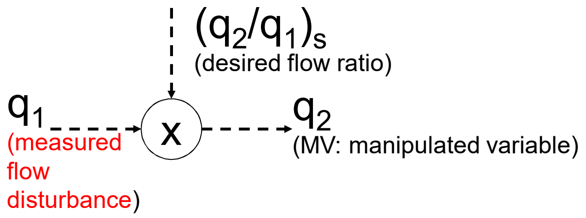
For x>2: $|-x+1|>1$ (worse than no control)….

- Example, x=2.1, $g_d\,d=1$,
  - No control: $y = g_d\,d = 1$
  - Ideal: $y = 0$
  - 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…)

Ratio control (most common case of feedforward)

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

Use multiplication block (x):



$(q_2/q_1)_s$

(desired flow ratio)

q_1

(measured flow disturbance)

x

q_2

(MV: manipulated variable)

"Measure disturbance ($d=q_1$) and adjust input ($u=q_2$) such that ratio is at given value $(q_2/q_1)_s$"

Usually: Combine ratio (feedforward) with feedback

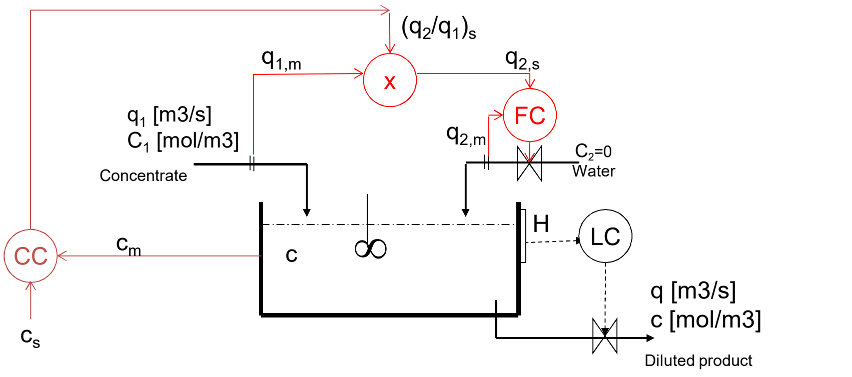
- Adjust $(q1/q2)_s$ based on feedback from process, for example, composition controller.
  - This is a special case of cascade control

  – ***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: MIXING PROCESS

RATIO CONTROL with outer cascade (to adjust ratio setpoint)



Potential problem outer feedback loop (CC: composition controller):
Gain from MV = $(q2/q1)_s$ to CV=c will vary because of multiplication with q1,m.
So outer loop must have robust tunings to get high gain margin (large tauc)

Ratio control

- It is simple
- Book has some strange suggestions, for example, Figure 14.5

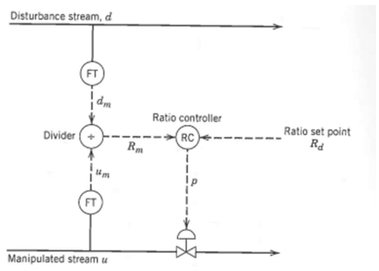


Figure 14.5 Ratio control, Method I.

Bad solution

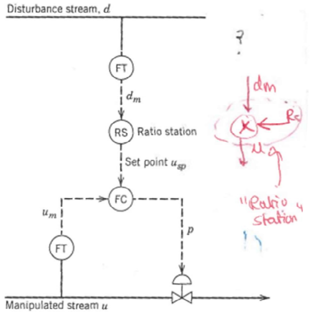


Figure 14.6 Ratio control, Method II.

Ok if implemented as shown in red

3. Other control configuration elements

- **Control configuration.** *The restrictions imposed on the overall controller by decomposing it into a set of local controllers (subcontrollers, units, elements, blocks) with predetermined links and with a possibly predetermined design sequence where subcontrollers are designed locally.*

Some control configuration elements:

- Cascade controllers (One MV, two CVs but only one has setpoint)
- Decentralized controllers (One MV, One CV)
- Feedforward elements (Measure DV, Adjust MV)
- Decoupling elements (Measure another MV2, Adjust MV1)
- **Split-range control: Two MVs needed to cover whole range at steady state one CV**
- **Input resetting/Valve position control/Midranging control: Two MVs (one to improve dynamics) , one CV**
- **Constraint control : One MV, one CV: But MV is used only on one side of constraint, Cvlimit=CVs.**

  Example Heater: MC=Q, CV=T. Constraint T>20C. We have Q=0 when T>20C
- **Override selector : One MV, two CVs. CV1 has desired setpoint which may be given up when CV2 exceeds limit**

Use of extra inputs

Two different cases

1. Have extra dynamic inputs (degrees of freedom)

   Cascade implementation: "Input resetting to ideal resting value"

   Example: Heat exchanger with extra bypass

   Also known as: Midranging control, valve position control

2. Need several inputs to cover whole range (because primary input may saturate) (steady-state)

   Split-range control

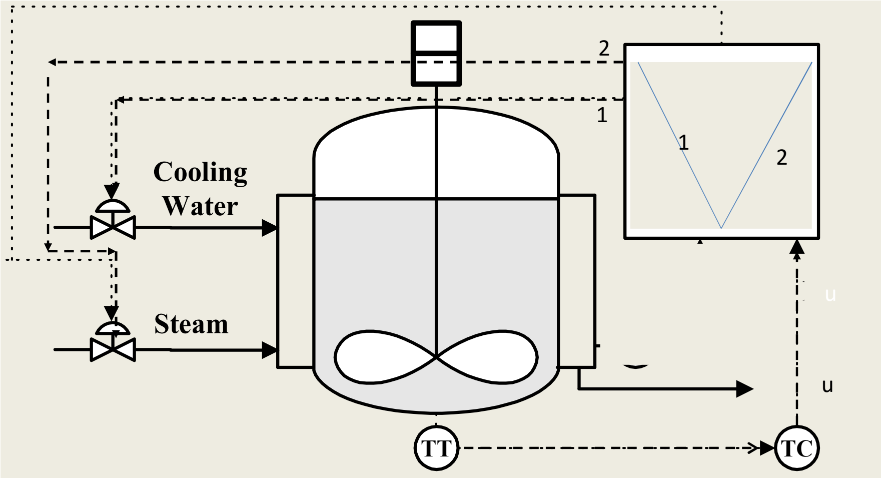
   Example 1: Control of room temperature using AC (summer), heater (winter), fireplace (winter cold)

   Example 2: Pressure control using purge and inert feed (distillation)

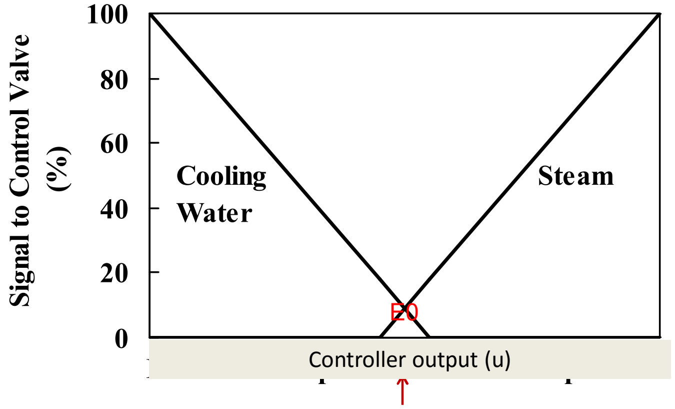
Split Range Temperature Control

(Two MVs needed to cover whole range, one CV)

Split range control is used when we need to inputs to cover the whole output range (at steady state), for example, we need both heating and cooling in a house to control temperature.
The range is split so that only one input is active for control at a time

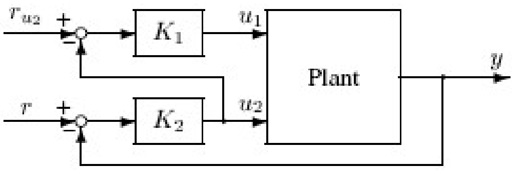


Split Range Temperature Control



Note: may adjust the location er E0 to make process gains equal

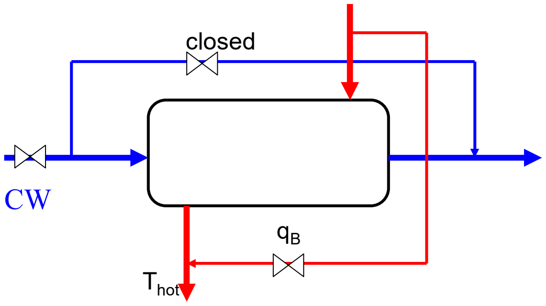
Extra inputs, dynamically



(b) Extra inputs u_2 (input resetting)

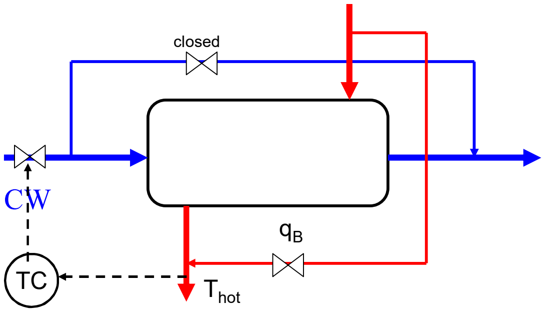
- Exercise: Explain how “valve position control” fits into this framework. As en example consider a heat exchanger with bypass

QUIZ: Heat exchanger with bypass

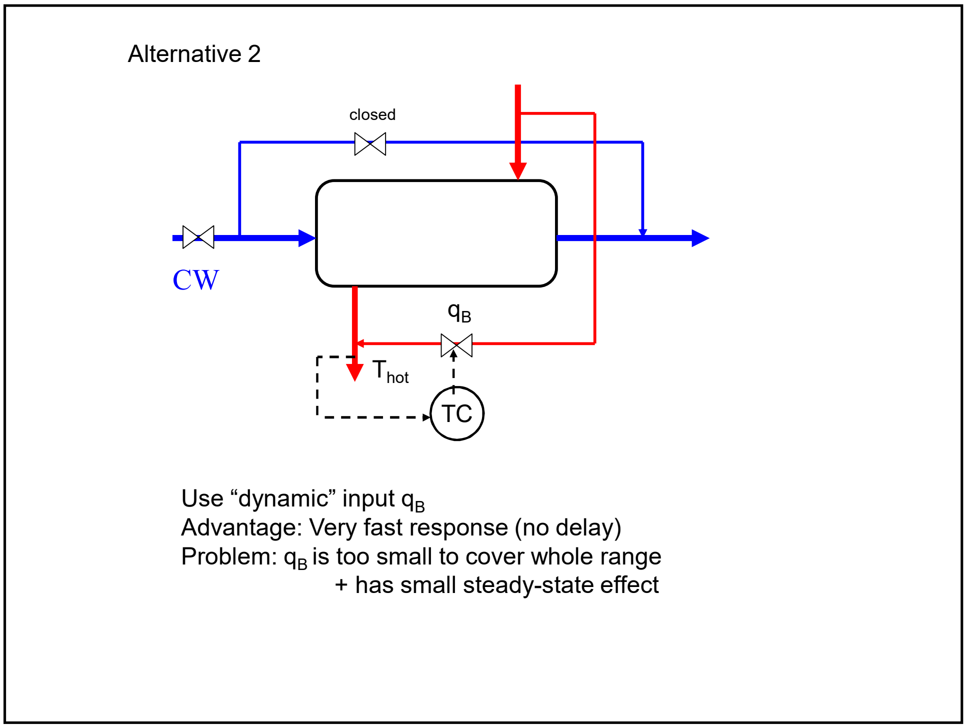


- Want tight control of T_{hot}
- Primary input: CW
- Secondary input: q_B
- Proposed control structure?

Alternative 1

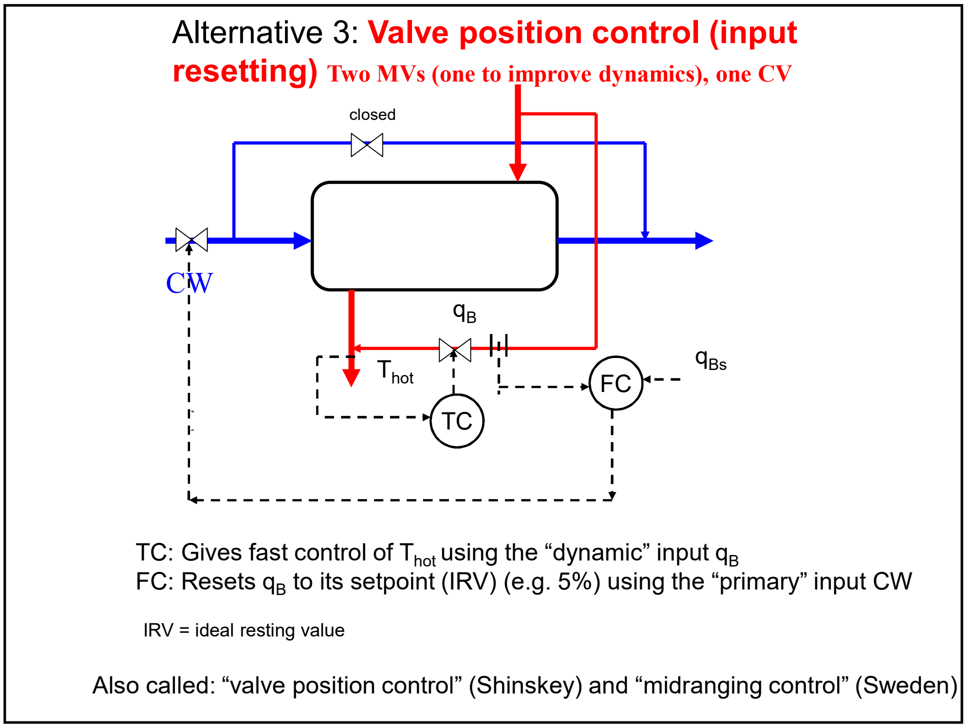


Use primary input CW: TOO SLOW



Alternative 2

Use “dynamic” input q_B
Advantage: Very fast response (no delay)
Problem: q_B is too small to cover whole range
+ has small steady-state effect



Alternative 3: **Valve position control (input resetting)** **Two MVs (one to improve dynamics), one CV**

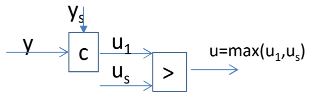
TC: Gives fast control of T_{hot} using the “dynamic” input q_B
FC: Resets q_B to its setpoint (IRV) (e.g. 5%) using the “primary” input CW

IRV = ideal resting value

Also called: “valve position control” (Shinskey) and “midranging control” (Sweden)

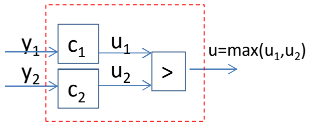
Constraint control

- One MV (u), One CV which only needs to be controlled when it reaches constraint.
- Assume that u=0 (or more generally $u=u_s$) gives acceptable control for some disturbances.
- Controller activates (u>0) when CV (y) is on the undesired side of ys=CVlimit.
- "Works by itself" but make sure you have anti windup in controller.
- Here is a long explanation which probably only confuses
  - Keep CV above or below limit using an available ("extra") input which we would normally do not want to use (and which we therefore may say "works only in one direction").
  - So the controller is only activates (u>0) when CV is on the undesired side of CVs=Cvlimit. Controller will then keep CV=CVs=CVlimit until we again return to u=0 (when the disturbance goes away). At this point we are on the desired side of the constraint and the controller is inactive (u=0 and CV on the right side of CVlimit).
  - Example 1: Keep TV>Tmin = 5C in cabin by using heating (it will normally be hotter so heating is only used when constraint is reached)
  - Example 2: Minimum flow for pump or compressor using recycle valve.
  - Comment: if we need to control CV at CVs=CVlimit all the time, then we need one more input (and may use split range control) to handle some disturbances.

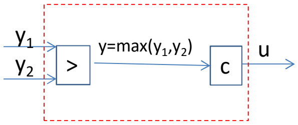


Selector: One MV and two CVs

- Must control output ($y_1=CV_1$, $y_2=CV_2$) with highest priority
  - Selectors
- Implementation:
  - Alt. 1. Several controllers (with the same u=MV)
    - Selects max (or min) MV value, u = max(u1,u2)
    - Often used to handle changes in active constraints
    - Example: one heater for two rooms. T1 = 20C (desired), T2>10C
      - Max-selector
      - Must give up controlling T1 if T2 drops below 10C
      - Could also say that requirement is T1 > 20C, T2> 10C.
    - Example: Petlyuk distillation column
      - Heat input (V) is used to control three compositions using max-selector
      - Two products will be better than setpoint ("overpurified") at any given time



  - Alt. 2. One controller (with several CVs)
    - Selects max or min CV value, y = max(y1,y2)
    - More general, e = max(e1,e2), e = y-ys
    - Simpler than Alt.1, but dynamics from u to y1 and y2 must be similar
    - Example: Control hot-spot in reactor or furnace.



Override selector: Alt. 1

Override selectors are used when you normally want to keep y1 at a setpoint, but you must make sure that y2 (higher priority) does not exceed a limit. When y2 is controlled one must give up control of y1. (The reason for y2 exceeding its limit may be a disturbance or because the input used to control y2 has saturated)

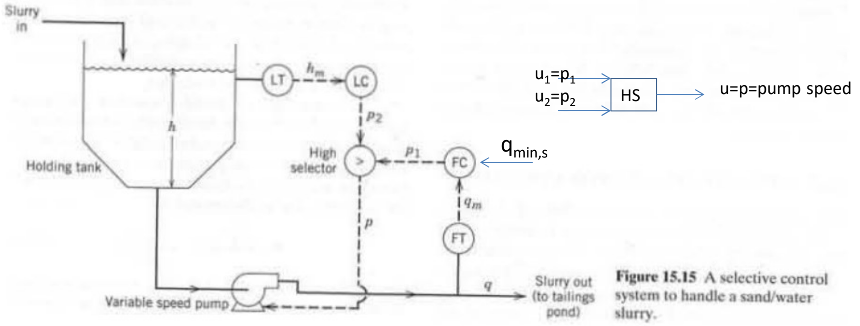
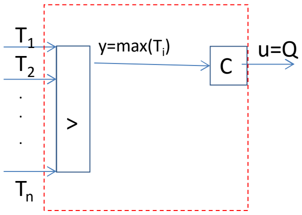


Figure 15.15 A selective control system to handle a sand/water slurry.

Comment: Could instead have a “lower” flow controller which is active all the time, and let LC set q_s. This is probably a better solution

Override selector. Alt. 2

- Hot-spot control in reactor or furnace

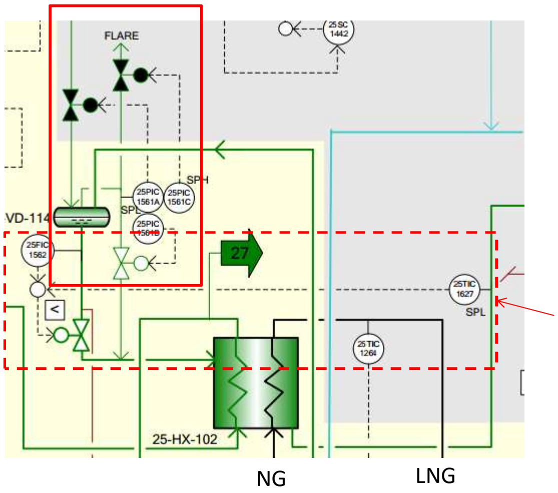


- Comment: Could use Alt. 2 (many controllers) for hot-spot control, with each temperature controller (c_1, c_2,…) computing the heat input ($u_1=Q_1$, $u_2=Q_2$, ….) and then select u = min(u_1, u_2, …), but it is more complicated.
  - Question: Why u = min(u_1, u_2, …) and not u = max(u_1, u_2, …) ?
  - Answer: Because Q is heating (would get max if Q was cooling)

LNG-plant

Black valve = normally closed

Keep pressure within bounds with extra MVs (Different setpoints, alternative to split range).



Override Low Selector (avoids that operator sets too high flow)

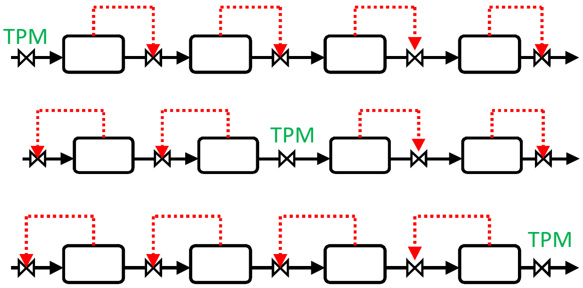
Temperature must be above low limit (superheated) to avoid liquid in compressor

Not finished….

CV with setpoint	CV with limit	MVs	Extra meas. output	Meas. disturbances	Structure	Comment
1		1			SISO	
1*		1		1	Feedforward (including ratio)	*CV not measured
1		1	1		Cascade	
1	1 or more	1			Selector (override)	Low priority: CV1=setpoint High priority: CV2 bound Higher priority: CV3 bound
1		2			Split range	Extra MVs needed for steady state
1		2			Input resetting (mid-ranging)/ Parallell	Extra MVs to improve dynamics (IRV-setpoint for MV2)
	1	1			Constraint control = SISO	Increasing MV moves away from constraint

Process control*: Throughput manipulator (TPM)

- TPM (“gas pedal”): Sets the production rate.
- Where is the TPM located for the process?
  - Usually at the feed, but not always!
  - Important decision because it determines the control structure
- Inventory control (Level and pressure) must be radiating around TPM:

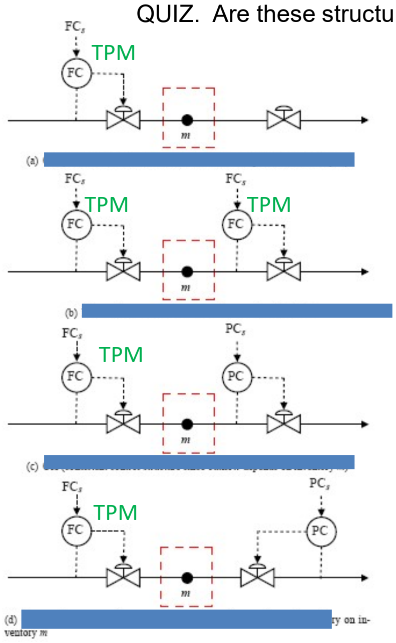


*”If it has a TPM it is process control”

- Usually only one TPM
  - To get consistent mass balance: Can only fix same flow once
- All inventories (level, pressure) must be regulated by
  - Controller, or
  - “self-regulated” (e.g., overflow for level, open valve for pressure)
  - Exception closed system: Must leave one inventory (level) uncontrolled

- Rule for maximizing production: Locate TPM at bottleneck.

QUIZ. Are these structures workable? Yes or No?

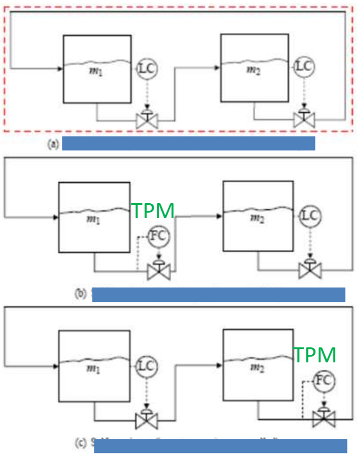


(a) [redacted]

(b) [redacted]

(c) [redacted]

(d) [redacted]ry on inventory m



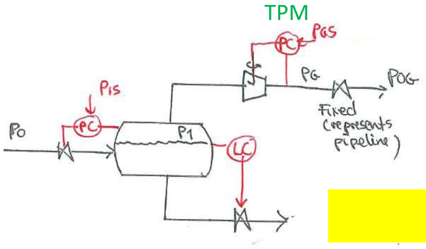
(a) [redacted]

(b) [redacted]

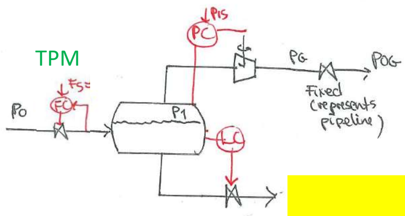
(c) [redacted]

Figure 2.8: Inventory control for closed system.

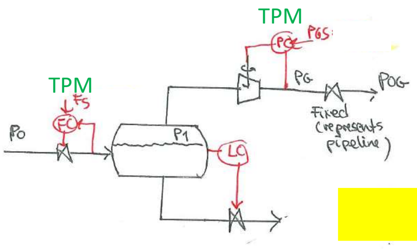
Quiz 2. Workable? Yes or No



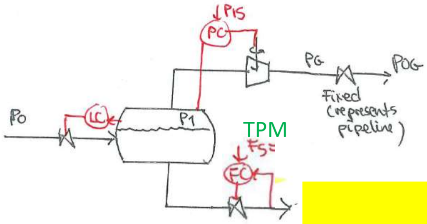
[redacted]



[redacted]



[redacted]



[redacted]

4. Multivariable control

1. Single-loop control (decentralized)
2. Decoupling (similar to feedforward)
3. Model predictive control (MPC)

RGA in here

- For choosing pairings for decentralized control
- See separate slides

Single-loop control = Decentralized control

Use for: Noninteracting process and no change in active constraints

+ Tuning may be done on-line
+ No or minimal model requirements
+ Easy to fix and change
- Need to determine pairing
- Performance loss compared to multivariable control
- Complicated logic required for reconfiguration when active constraints move

Decentralized control tuning

- Independent design
  - Use when small interactions (RGA close to I)
- Sequential design (similar to cascade)
  - Start with fast loop
  - NOTE: If close on negative RGA, system will go unstable of fast (inner) loop saturates
  - Sequential vs. independent design
    - + Generally better performance, but
    - - outer loop gets slow, and
    - - loops depend on each other

If interactions cause poor performance for single-loop control

Possible solutions:

I. Adding fast loop to break interactions (cascade control)
II. Decoupling
III. MPC

Breaking interactions with cascade control (fast slave loop)

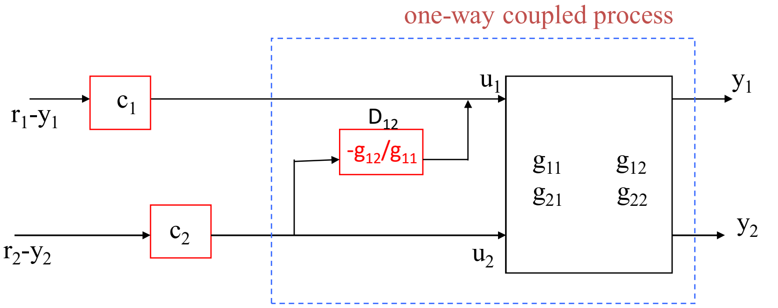
Example 1. Control of level and pressure in separator

- MVs: Valve positions for liquid and gas out
- Highly interactive
- Interactions an be avoided with cascade! How?

Example 2. Control of compositions in distillation column

- MVs: Reflux and heat input (boilup)
- Time delay on composition measurements
- Highly interactive
- Can to some extent be avoided with cascade (inner temperature loop)

One-way Decoupling (improved control of y_1)



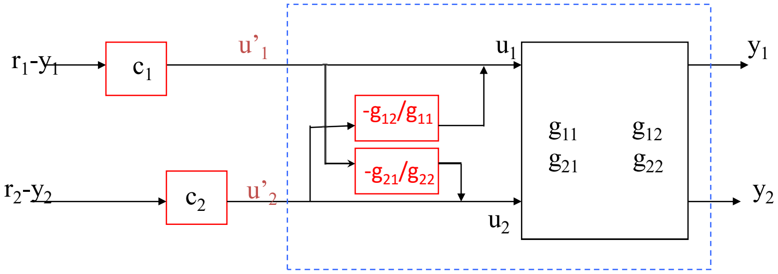
DERIVATION

Process: $y_1 = g_{11}\ u_1 + g_{12}\ u_2$ (1)
$y_2 = g_{21}\ u_1 + g_{22}\ u_2$ (2)

Consider u_2 as disturbance for control of y_1.
Think «feedforward»: Adjust u_1 to make y_1=0. (1) gives $u_1 = -\ (g_{12}/g_{11})\ u_2$

Two-way Decoupling: Standard implementation (Seborg)

decoupled process = $([G^{-1}]_{diag})^{-1}$

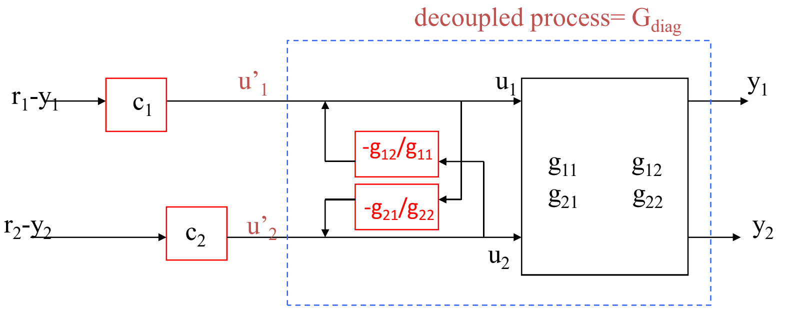


… but note that diagonal elements of decoupled process are different from G
Problem for tuning!

Process: $y_1 = g_{11}\ u_1 + g_{12}\ u_2$
Decoupled process: $y_1 = (g_{11}\text{-}g_{12}{*}g_{21}/g_{22})\ u_1' + 0{*}u_2'$
Similar for y_2.

Sigurd recommends this alternative!

Two-way Decoupling: «Inverted» implementation (Shinskey)

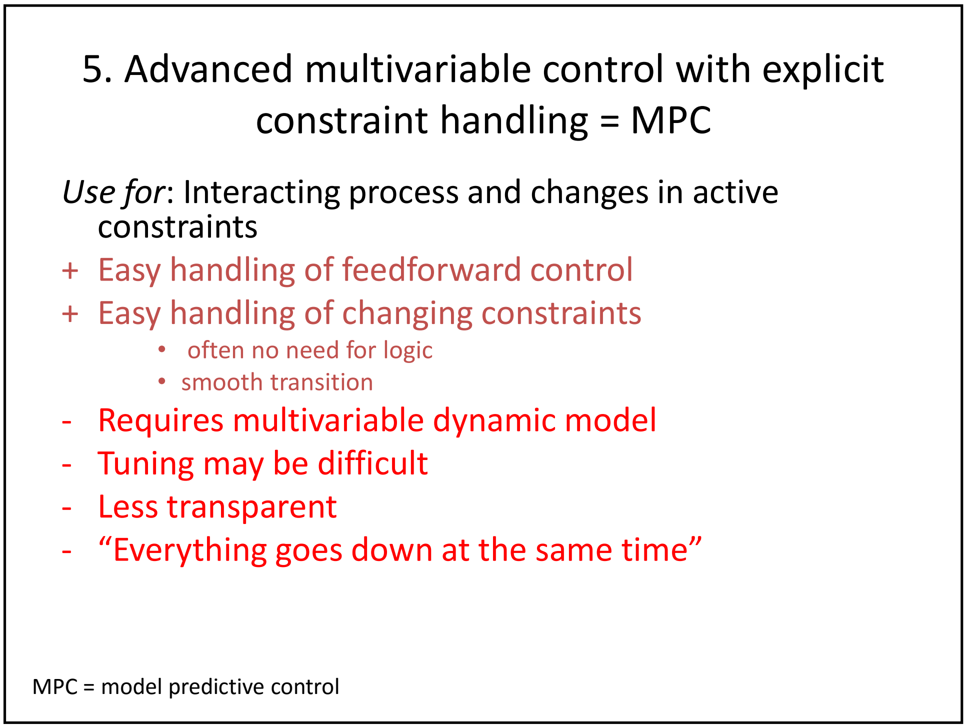


Advantages: (1) Decoupled process has same diagonal elements as G. Easy tuning!
(2) Handles input saturation! (if u1 and u2 are actual inputs)

Proof (1): $y_1 = g_{11} u_1 + g_{12} u_2$, where $u_1 = u_1' - (g_{12}/g_{11})u_2$.
Gives : $y_1 = g_{11} u'_1 + 0^* u_2'$
Similar: $y_2 = 0^*u_1' + g_{22} u_2'$

Pairing and decoupling

- To get ideal decoupling, offdiagonal elements should have smaller effective delay than the diagonal elements
- Thus, we should pair on elements with small effective delay ("pair close rule")
- Pairing on negative steady state RGA elements is not a problem if we use decoupling
  - Because negative RGA-elements are caused by interactions, which is what we are cancelling with decoupling



5. Advanced multivariable control with explicit constraint handling = MPC

Use for: Interacting process and changes in active constraints

+ Easy handling of feedforward control

+ Easy handling of changing constraints

- often no need for logic
- smooth transition

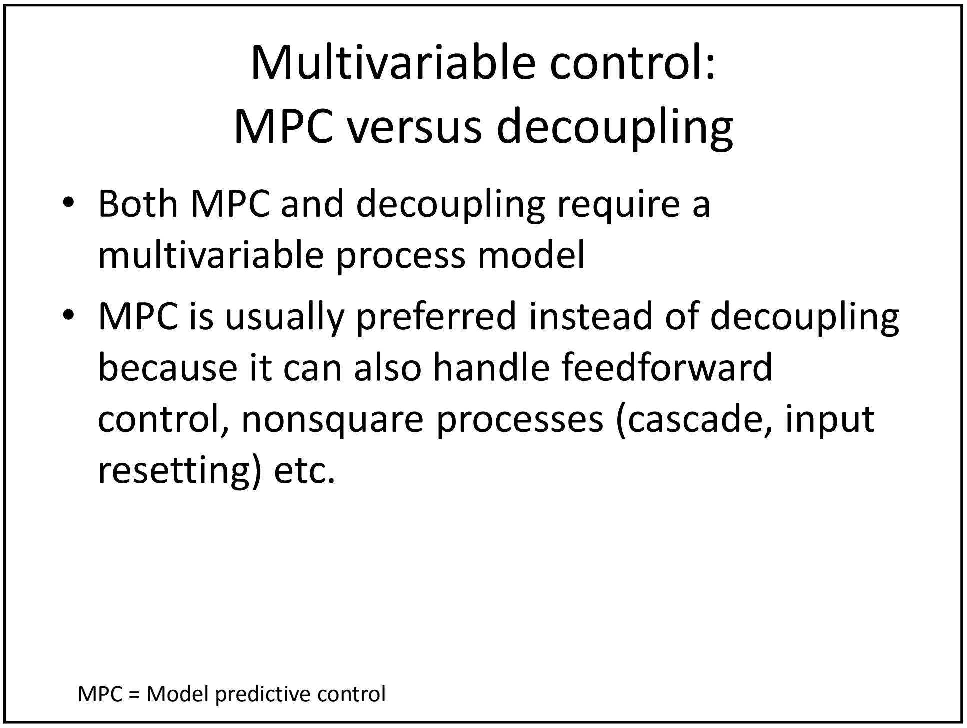
- Requires multivariable dynamic model

- Tuning may be difficult

- Less transparent

- "Everything goes down at the same time"

MPC = model predictive control



Multivariable control: MPC versus decoupling

- Both MPC and decoupling require a multivariable process model
- MPC is usually preferred instead of decoupling because it can also handle feedforward control, nonsquare processes (cascade, input resetting) etc.

MPC = Model predictive control

Model predictive control (MPC) = “online optimal control”

The quadratic program of equations (1)-(5) is solved each control sample to find the optimal control actions.

$$\min_{\Delta u} y_{dev}^T Q_y y_{dev} + u_{dev}^T Q_u u_{dev} + \Delta u^T P \Delta u \qquad (1)$$

$$u_{\min} < u < u_{\max} \qquad (2)$$

$$\Delta u_{\min} < \Delta u < \Delta u_{\max} \qquad (3)$$

$$y_{\min} < y < y_{\max} \qquad (4)$$

$$y = M(y, u, d, v) \qquad (5)$$

y_{dev}=y-y_s
u_{dev}=u-u_s

Discretize in time:

$$y = [y_1\, y_2 \ldots y_n]$$
$$u = [u_1\, u_2 \ldots u_k]$$
$$\Delta u = [\Delta u_1\, \Delta u_2 \ldots \Delta u_k]$$
$$\Delta u_i = u_i - u_{i-1}$$

Note: Implement only current input Δu_1

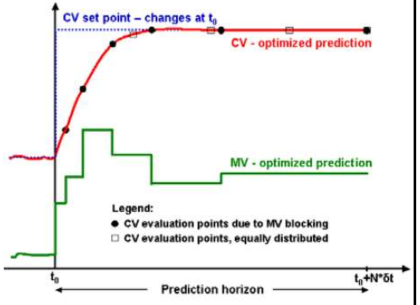


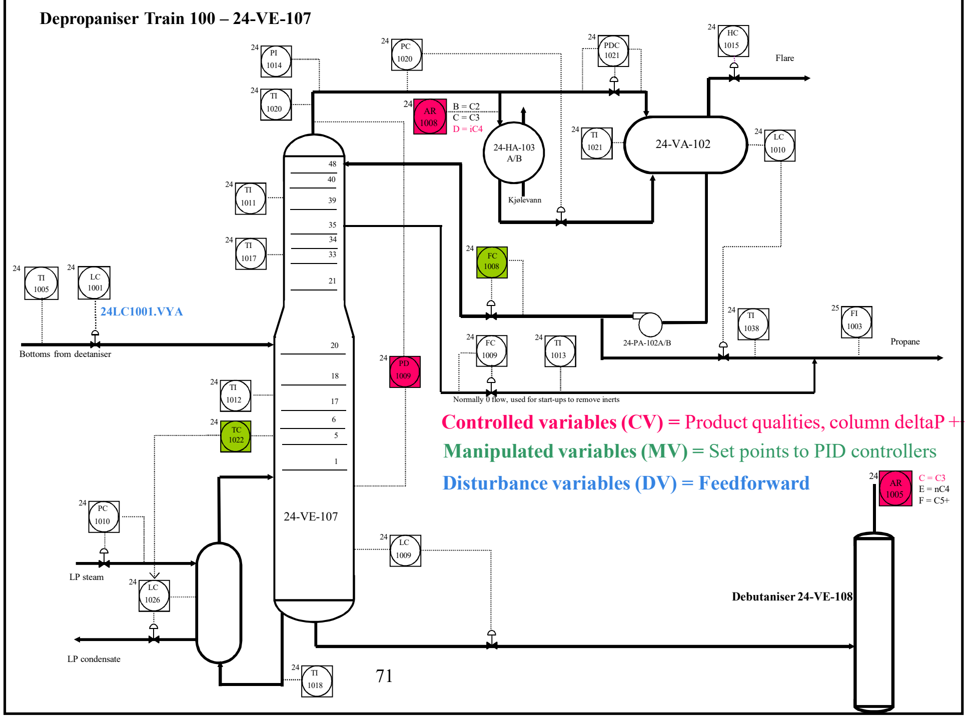
Fig. 1. MV blocking and CV evaluation

The quadratic objective function (1) penalizes CV (y) deviations from set point, MV (u) deviations from ideal values, and MV moves. The constraints are: MV high and low limits (2); MV rate of change limits (3); and CV high and low limits (4). The dynamic model (5) predicts the CV response from past and future CV and MV values as well as past DV (d) values and estimated and optionally predicted unmeasured disturbances v.

Implementation MPC project (Stig Strand, Statoil)

- Initial MV/CV/DV selection
- DCS preparation (controller tuning, instrumentation, MV handles, communication logics etc)
- Control room operator pre-training and motivation
- Product quality control → Data collection (process/lab) → Inferential model
- MV/DV step testing → dynamic models
- Model judgement/singularity analysis → remove models? change models?
- MPC pre-tuning by simulation → MPC activation – step by step and with care – challenging different constraint combinations – adjust models?
- Control room operator training
- MPC in normal operation, with at least 99% service factor

DCS = “distributed control system” = Basic PID control layer



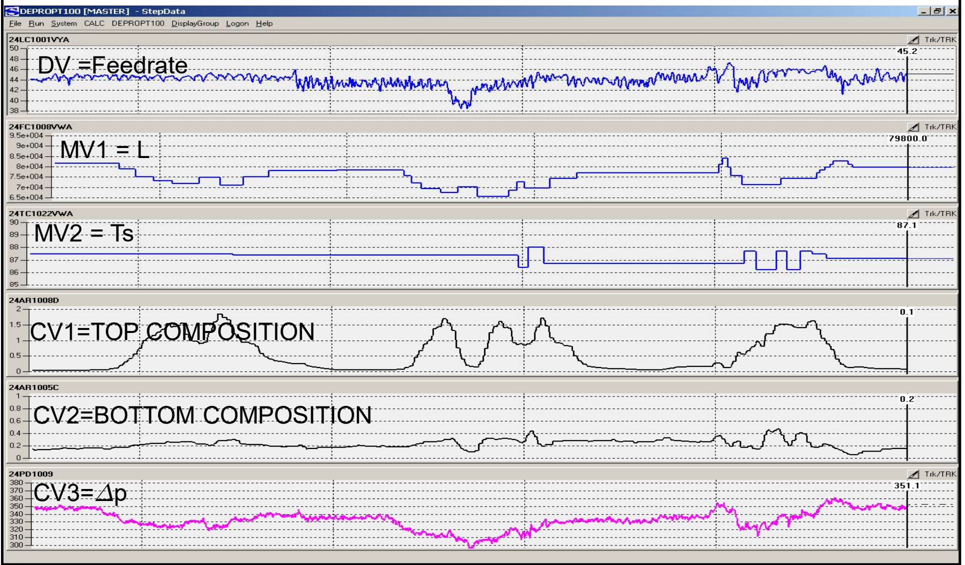
Controlled variables (CV) = Product qualities, column deltaP +

Manipulated variables (MV) = Set points to PID controllers

Disturbance variables (DV) = Feedforward

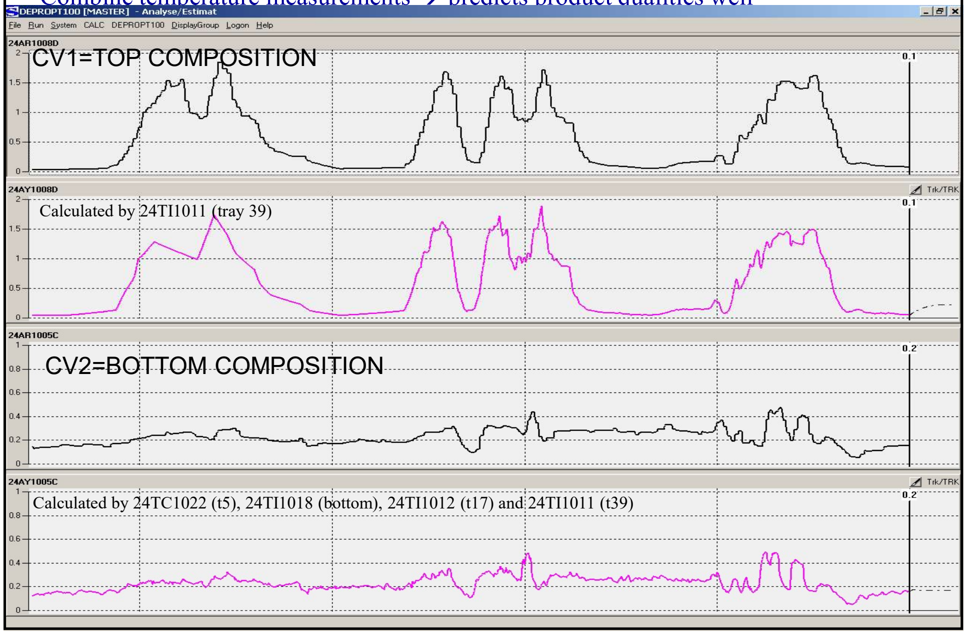
Depropaniser Train100 step testing

- 3 days – normal operation during night
-



Estimator: inferential models

- Analyser responses are delayed – temperature measurements respond 20 min earlier
- Combine temperature measurements → predicts product qualities well


DEPROPT100 [MASTER] - Analyse/Estimat
File Run System CALC DEPROPT100 DisplayGroup Logon Help
24AR1008D
24AY1008D
24AR1005C
24AY1005C
Trk/TRK
0.1
0.2

CV1=TOP COMPOSITION

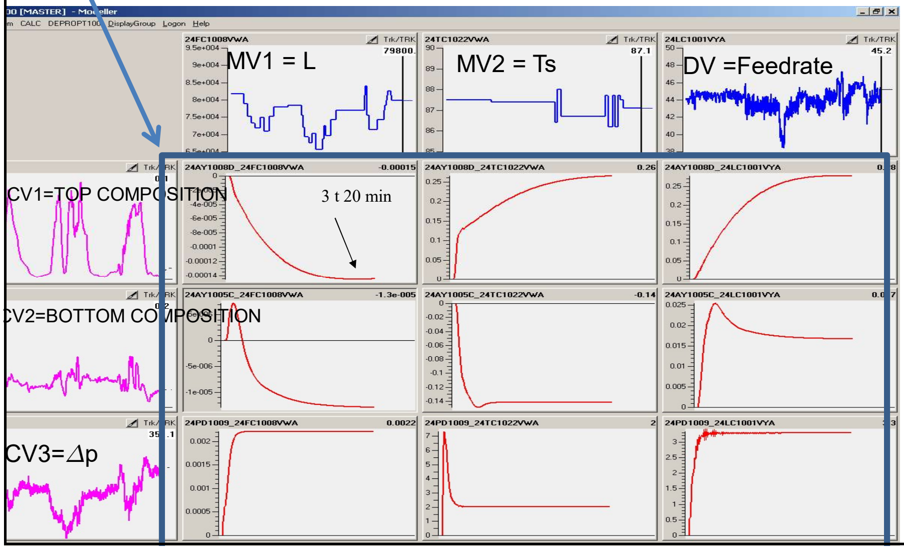
Calculated by 24TI1011 (tray 39)

CV2=BOTTOM COMPOSITION

Calculated by 24TC1022 (t5), 24TI1018 (bottom), 24TI1012 (t17) and 24TI1011 (t39)

Depropaniser Train100 step testing – Final model

- Step response models:
  - MV1=reflux set point increase of 1 kg/h
  - MV2=temperature set point increase of 1 degree C
  - DV=output increase of 1%.



MV1 = L

MV2 = Ts

DV =Feedrate

CV1=TOP COMPOSITION

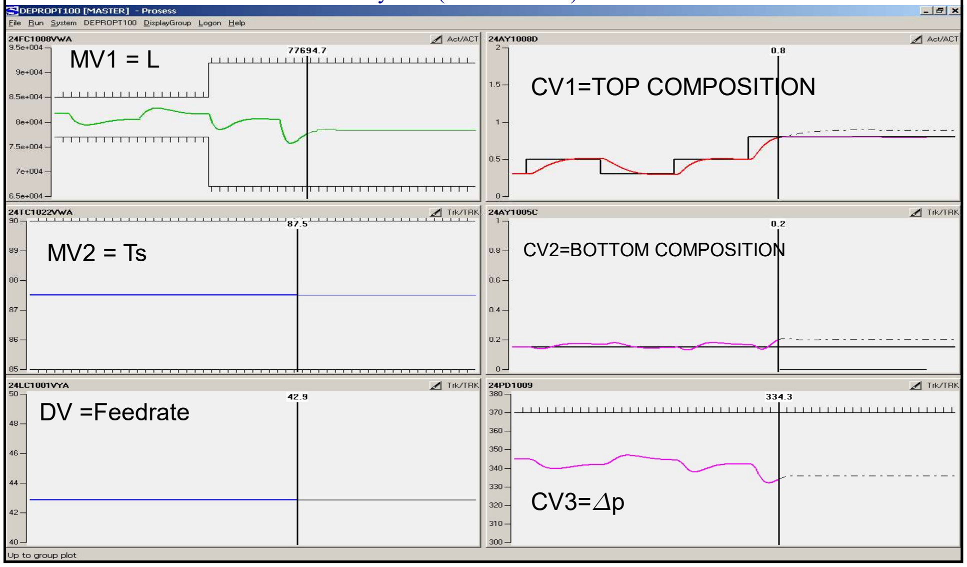
3 t 20 min

CV2=BOTTOM COMPOSITION

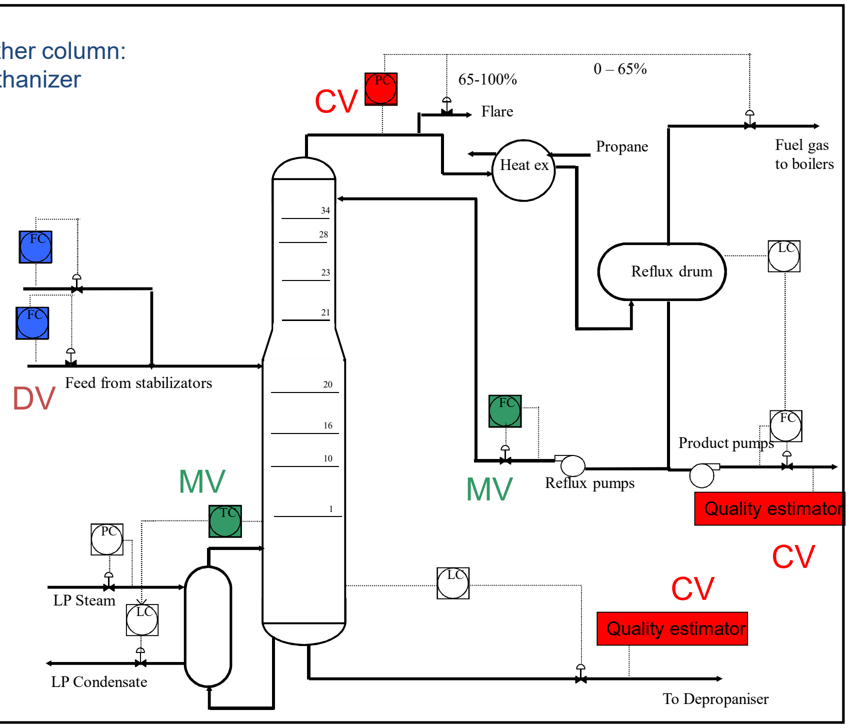
CV3=Δp

Depropaniser Train100 MPC – controller activation

- Starts with 1 MV and 1 CV – CV set point changes, controller tuning, model verification and corrections
- Shifts to another MV/CV pair, same procedure
- Interactions verified – controls 2x2 system (2 MV + 2 CV)

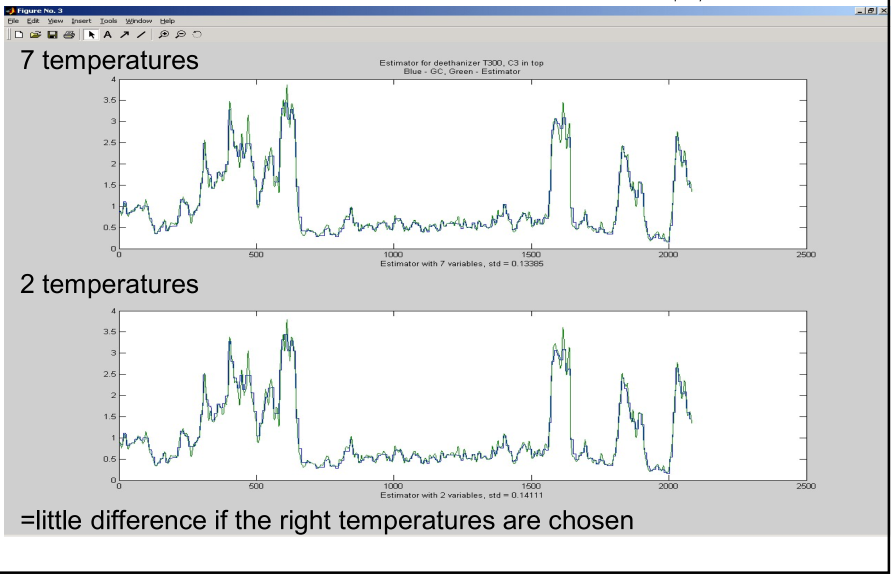


Another column:
Deethanizer



Top: Binary separation in this case
Quality estimator vs. gas chromatograph

(use logarithmic composition to reduce nonlinearity, CV = - ln $x_{impurity}$)

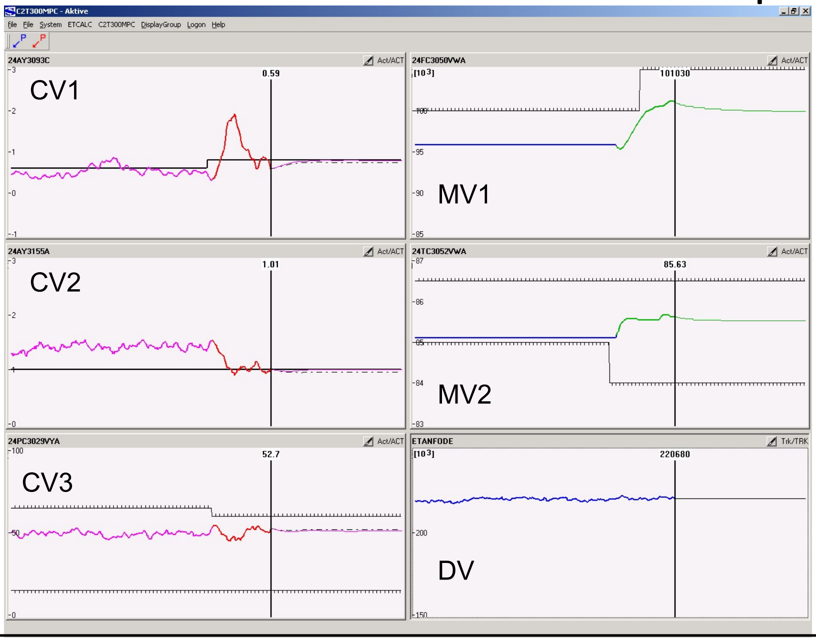


7 temperatures

2 temperatures

=little difference if the right temperatures are chosen

The final test: MPC in closed-loop



Conclusion MPC

- Generally simpler than previous advanced control
- Well accepted by operators
- Statoil: Use of in-house technology and expertise successful

Pole placement (state feedback)

- Place closed-loop poles. Old design method
- Useful for insight, but difficult to use. Not used much in practice, at least not for linear controllers
- Basis:
  - Linear system on state space form

    $$\underline{x} = Ax + Bu$$

  - And State feedback (assuming we know all states)

    $$u = Kx$$

SIMC is “pole placement” (p=-1/tauc), but with output feedback, and we also place zeros

8.5.1 Stability and state feedback

The poles of the transfer function, which are the zeros of its denominator polynomial, determine the dynamic characteristics of the system, in particular its stability and its damping characteristics. Transferring this statement to equation (8.60), it follows that the roots of the equation

$$\det(s \cdot \boldsymbol{I} - \boldsymbol{A}) = 0 \tag{8.67}$$

are essential for the behaviour of the system. The determinant in equation (8.67) is a n-th order polynomial in s and corresponds to the characteristic polynomial. The roots of the determinant in equation (8.67) are also designated as the eigenvalues of the matrix $\boldsymbol{A}$. All of them must exhibit negative real parts, if the system described by the matrix $\boldsymbol{A}$ is supposed to be stable.

which will be combined to yield

$$\dot{\boldsymbol{x}} = (\boldsymbol{A} - \boldsymbol{B} \cdot \boldsymbol{K}) \cdot \boldsymbol{x} \tag{8.71}$$

Equation (8.71) describes a system without any input variables with the system matrix

$$\boldsymbol{A}_K = \boldsymbol{A} - \boldsymbol{B} \cdot \boldsymbol{K} \quad . \tag{8.72}$$

8.5.2 Pole placement

One possibility for the controller design is to select desirable eigenvalues of the matrix $\boldsymbol{A}_K$ and to determine from this and the known matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ the controller or feedback matrix $\boldsymbol{K}$.

As an example a state feedback is to be determined according to the mentioned procedure of pole placement for a transfer system with a single input and a single ouput variable. Figure 8-6 shows the functional diagram of the system with feedback.

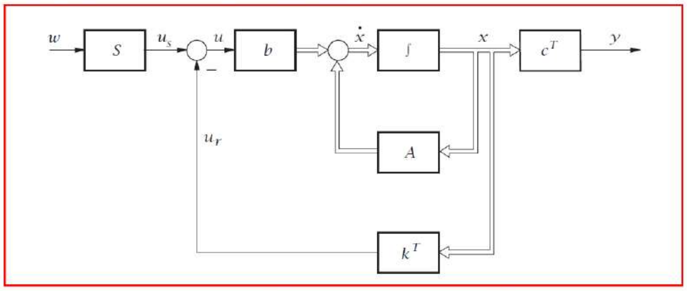


Figure 8-6: SISO-system with state feedback

The transfer system may be be stated in controller canonical form according to equation (8.23). The state variables of the controller canonical form can be obtained for this purpose by transformation of the original state variables in the way described in chapter 8.2. According