# Multivariable control

I. Single-loop control (decentralized)
II. Decoupling (similar to feedforward)
III. Model predictive control (MPC)

# I. Multivariable control using single loops

- Interactions
- Choice of pairings (RGA)

# Example (Exercise 12): Shower (mix hot and cold water)

- $u = (q_h \; q_c)$
- $y = (T \; q)$

What pairing?

**Problem 3: Water mixer**

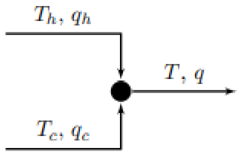



Figure 1: Mixer system

Consider the process of mixing hot and cold water, as shown in Figure 1. The process has inputs $u_1 = \Delta q_h$ [$\ell$/s], $u_2 = \Delta q_c$ [$\ell$/s], and outputs $y_1 = \Delta T$ [°C], $y_2 = \Delta q$ [$\ell$/s].

The control objective is to have a mixing temperature $T = 40$°C and a total flow leaving the mixer of $q = 1$ $\ell$/s. At the nominal operating point we have $T_c = 30$°C and $T_h = 60$°C.

1. Formulate the energy and mass balances. The dynamics of this process are very fast; so, a steady-state model is sufficient to get $T$ and $q$.
2. Linearize the model and show that the linear model can be written $y = Gu$, where:

$$G = \begin{bmatrix} k_1 & k_2 \\ 1 & 1 \end{bmatrix}$$

with: $\quad k_1 = (T_h^* - T^*)/q^* \qquad k_2 = (T_c^* - T^*)/q^* \qquad u = [u_1 \quad u_2]^T$

The symbol * denotes the steady state value.

3. What are the steady state values for $q_c$ and $q_h$?
4. Find the gain matrix $G$ at the nominal operating point.
5. Based on $G$, which stream ($q_h$ or $q_c$) would you use to control the temperature ($T$)? Explain briefly.

*In Exercise 13, you will find out if your intuition was right.*

# Multivariable process

**Distillation column**

**"Increasing reflux L from 1.0 to 1.1 changes $y_D$ from 0.95 to 0.97, and $x_B$ from 0.02 to 0.03"**

**"Increasing boilup V from 1.5 to 1.6 changes $y_D$ from 0.95 to 0.94, and $x_B$ from 0.02 to 0.01"**

**Steady-State Gain Matrix**

$$\begin{pmatrix}\Delta Y_D \\ \Delta x_B\end{pmatrix} = G(0)\begin{pmatrix}\Delta L \\ \Delta V\end{pmatrix}$$

$$G(0) = \begin{bmatrix} g_{11} & g_{12}(0) \\ g_{21} & g_{22}(0) \end{bmatrix} = \begin{bmatrix} \frac{0.97-0.95}{1.1-1.0} & \frac{0.94-0.95}{1.6-1.5} \\ \frac{0.03-0.02}{1.1-1.0} & \frac{0.01-0.02}{1.6-1.5} \end{bmatrix} = \begin{bmatrix} 0.2 & -0.1 \\ 0.1 & -0.1 \end{bmatrix}$$

Effect of input $1(\Delta L)$ on output $2(\Delta x_B)$

Can also include dynamics :

$$G_{(s)} = \begin{bmatrix} \frac{0.2}{1+50s} & -\frac{0.1}{1+50s} \\ \frac{0.1}{1+40s} & -\frac{0.1}{1+40s} \end{bmatrix} \begin{matrix} \to \Delta y_D \\ \to \Delta x_B \end{matrix}$$

(Time constant 50 min for $y_D$)

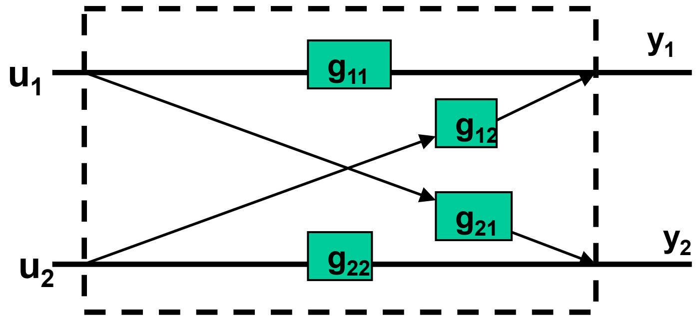
(time constant 40 min for $x_B$)

# Analysis of Multivariable processes

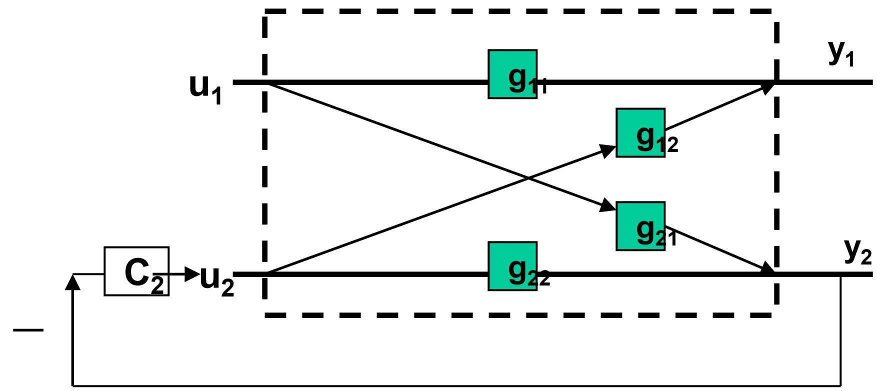
**Process Model 2x2**

$$\text{"Open- loop"}$$
$$y_1(s) = g_{11}(s)u_1(s) + g_{12}(s)u_2(s) \qquad (1)$$
$$y_2(s) = g_{21}(s)u_1(s) + g_{22}(s)u_2(s) \qquad (2)$$

**INTERACTIONS: Caused by nonzero offdiagonal elements ($g_{12}$ and/or $g_{21}$)**



# RGA: Consider effect of $u_1$ on $y_1$




1) "Open-loop" ($C_2 = 0$): $y_1 = g_{11}(s) \cdot u_1$

2) "Closed-loop" (close loop 2, $C_2 \neq 0$): $y_1 = \left( g_{11}(s) - \dfrac{g_{12} g_{21} \cdot C_2}{1 + g_{22} \cdot C_2} \right) u_1$

Change caused by "interactions"

Derivation.

Close loop 2: $u_2 = -c_2(y_2 - y_{2s})$

Here: $y_2 = g_{21}u_1 + g_{22}u_2$ and assume $y_{2s} = 0$:

$\Rightarrow u_2 = -c_2(g_{21}u_1 + g_{22}u_2) \Rightarrow u_2 = \frac{-c_2 g_{12}}{1+g_{22}c_2} u_1$

Effect of $u_1$ on $y_1$ with loop 2 closed is then:

$$y_1 = g_{11}u_1 + g_{12}u_2 = \underbrace{g_{11}\left(1 - \frac{g_{12}g_{21}c_2}{1 + g_{22}c_2}\right)}_{\widehat{g}_{11}} u_1$$

**Limiting Case $C_2 \to \infty$ (perfect control of $y_2$)** *

$$y_1 = \left( g_{11}(s) - \frac{g_{12}\, g_{21}}{g_{22}} \right) u_1 = g_{11}(1/\lambda_{11}) \cdot u_1$$

**How much has "gain" from $u_1$ to $y_1$ changed by closing loop 2 with perfect control?**

$$\text{Relative Gain} = \frac{(y_1/u_1)_{OL}}{(y_1/u_1)_{CL}} = \frac{g_{11}}{g_{11} - \frac{g_{12}\, g_{21}}{g_{22}}} = \frac{1}{1 - \frac{g_{12}\, g_{21}}{g_{11}\, g_{22}}} \stackrel{\text{def}}{=} \lambda_{11}^{RGA}$$

The relative Gain Array (RGA) is the matrix formed by considering all the relative gains

$$RGA = \Lambda = \begin{bmatrix} \lambda_{11} & \lambda_{12} \\ \lambda_{21} & \lambda_{22} \end{bmatrix} = \begin{bmatrix} \frac{(y_1/u_1)_{OL}}{(y_1/u_1)_{CL}} & \frac{(y_1/u_2)_{OL}}{(y_1/u_2)_{CL}} \\ \frac{(y_2/u_1)_{OL}}{(y_2/u_1)_{CL}} & \frac{(y_2/u_2)_{OL}}{(y_2/u_2)_{CL}} \end{bmatrix}$$

* Alternative : Can derive by setting y2=0 in (2) and put resulting u2 into (1)

NTNU

# Example from before

$$G = \begin{bmatrix} 0.2 & -0.1 \\ 0.1 & -0.1 \end{bmatrix}, \quad \lambda_{11} = \frac{1}{1 - \underbrace{\frac{0.1 \; 0.1}{0.2 \; 0.1}}_{0.5}} = 2$$

$$RGA = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$$

**Only acceptable pairings :**

$u_1 \leftrightarrow y_1$

$u_2 \leftrightarrow y_2$

**Not recommended :**

$u_1 \leftrightarrow y_2$

$u_2 \leftrightarrow y_1$

**With integral action :**

**Negative RGA $\Rightarrow$ individual loop unstable OR overall system unstable when individual loops saturates**

# Use of RGA:

## (1) Interactions

- RGA-element ($\lambda$)> 1: Smaller gain by closing other loops ("fighting loops" gives slower control)
- RGA-element ($\lambda$) <1: Larger gain by closing other loops (can be dangerous)
- RGA-element ($\lambda$) negative: Gain reversal by closing other loops (Oops!)

**Rule 1. Avoid pairing on negative steady-state relative gain – otherwise you get instability if one of the loops become inactive (e.g. because of saturation)**

**Rule 2. Choose pairings corresponding to RGA-elements close to 1**

**Traditional: Consider Steady-state**

**Better (improved Rule 2): Consider frequency corresponding to closed-loop time constant**

**Property of RGA:**

- **Columns and rows always sum to 1**
- **RGA independent of scaling (units) for u and y.**

RGA for general case:

$$[RGA]_{ij} = (g_{ij})_{OL}/(g_{ij})_{CL} = [G]_{ij}[G^{-1}]_{ji}$$

$=$ element-by-element multiplication of $G$ and $G^{-1^T}$.

Matlab: `RGA = G.*pinv(G).'`

Example

G = [5 10 1; 20 -10 0; 18 0 2]

G =

| | | |
|---|---|---|
| 5 | 10 | 1 |
| 20 | -10 | 0 |
| 18 | 0 | 2 |

>> rga=G.*pinv(G).'

rga =

| | | |
|---|---|---|
| 0.3125 | $\boxed{1.2500}$ | -0.5625 |
| $\boxed{1.2500}$ | -0.2500 | 0 |
| -0.5625 | 0 | $\boxed{1.5625}$ |

Conclusion: of the 6 possible pairings only one has positive RGA's

# Example 3x3 process: 6 possible pairing options

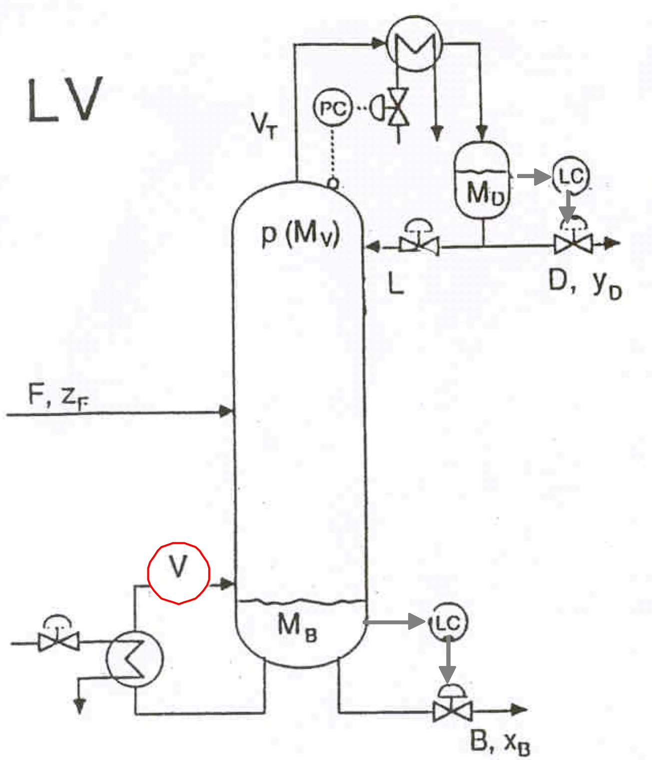
$$G = \begin{pmatrix} 16.8 & 30.5 & 4.30 \\ -16.7 & 31.0 & -1.41 \\ 1.27 & 54.1 & 5.40 \end{pmatrix}, \quad RGA(G) = \begin{pmatrix} 1.50 & 0.99 & -1.48 \\ -0.41 & 0.97 & 0.45 \\ -0.08 & -0.95 & 2.03 \end{pmatrix}$$

Only diagonal pairings give positive steady-state RGA's!

# Distillation

$$y = \begin{pmatrix} y_D \\ x_B \end{pmatrix}, \quad u = \begin{pmatrix} L \\ V \end{pmatrix}$$

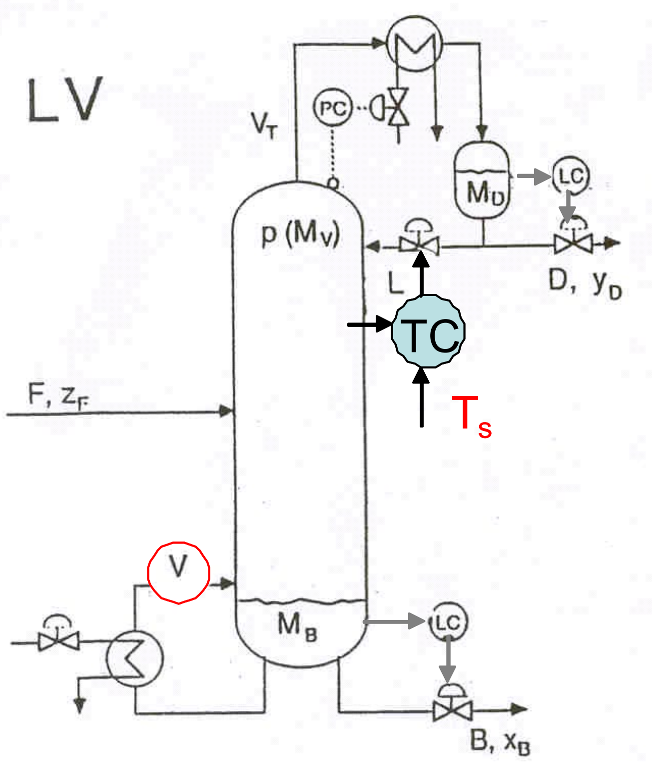
$$G(0) = \begin{pmatrix} 87.8 & -86.4 \\ 108.2 & -109.6 \end{pmatrix}, \quad RGA(0) = \begin{pmatrix} 35 & -34 \\ -34 & 35 \end{pmatrix}$$



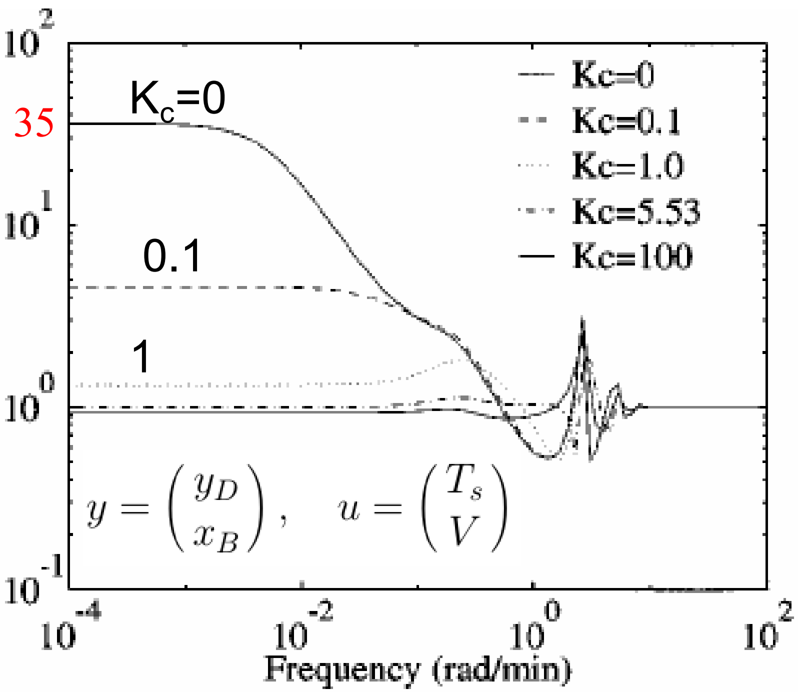
Distillation

$$y = \begin{pmatrix} y_D \\ x_B \end{pmatrix}, \quad u = \begin{pmatrix} L \\ V \end{pmatrix}$$

$$G(0) = \begin{pmatrix} 87.8 & -86.4 \\ 108.2 & -109.6 \end{pmatrix}, \quad RGA(0) = \begin{pmatrix} 35 & -34 \\ -34 & 35 \end{pmatrix}$$




Can break interactions with cascade:
Frequency-dependent RGA with TC



# Sometimes useful: Iterative RGA

- For large processes, lots of pairing alternatives
- RGA evaluated iteratively is helpful for quick screening

$$\mathrm{RGA}(G) = \Lambda(G) = G \times (G^{-1})^T$$

$$\Lambda^2(G) = \Lambda(\Lambda(G))$$

$$\boxed{\Lambda^\infty = \lim_{k\to\infty} \Lambda^k(G)}$$

- Converges to "Permuted Identity" matrix (correct pairings) for generalized diagonally dominant processes.
- Can converge to incorrect pairings, when no alternatives are dominant.
- Usually converges in 5-6 iterations

NTNU

# Example of Iterative RGA

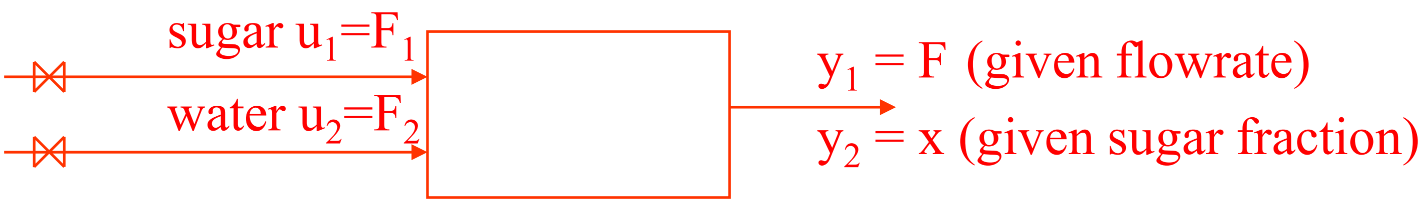
$$G = \begin{bmatrix} 1 & 2 \\ -1 & 1 \end{bmatrix}$$

$$\Lambda = \begin{bmatrix} 0.33 & 0.67 \\ 0.67 & 0.33 \end{bmatrix} \qquad \Lambda^2 = \begin{bmatrix} -0.33 & 1.33 \\ 1.33 & -0.33 \end{bmatrix}$$

$$\Lambda^3 = \begin{bmatrix} -0.07 & 1.07 \\ 1.07 & -0.07 \end{bmatrix} \qquad \Lambda^4 = \begin{bmatrix} 0.00 & \boxed{1.00} \\ \boxed{1.00} & 0.00 \end{bmatrix}$$

Correct pairing

# Exercise. Blending process




- Mass balances (no dynamics)
  - Total: $F_1 + F_2 = F$
  - Sugar: $F_1 = x\,F$

(a) Linearize balances and introduce: $u_1$=$dF_1$, $u_2$=$dF_2$, $y_1$=$F_1$, $y_2$=x,

(b) Obtain gain matrix G (y = G u)

(c) Nominal values are x=0.2 [kg/kg] and F=2 [kg/s]. Find G

(d) Compute RGA and suggest pairings

(e) Does the pairing choice agree with “common sense”?

**Solution.**
(a) The balances "mass in = mass out" for total mass and sugar mass are

$$F_1 + F_2 = F; \; F_1 = xF$$

Note that the mixing process itself has no dynamics. Linearization yields

$$dF_1 + dF_2 = dF : \; dF_1 = x^* dF + F^* dx$$

With $u_1 = dF_1, u_2 = dF_2, y_1 = dF$ and $y_2 = dx$ we then get the model

$$y_1 = u_1 + u_2$$
$$y_2 = \frac{1-x^*}{F^*} u_1 - \frac{x^*}{F^*} u_2$$

where $x^* = 0.2$ is the nominal steady-state sugar fraction and $F^* = 2$ kg/s is the nominal amount.
(b,c) The transfer matrix then becomes

$$G(s) = \begin{pmatrix} 1 & 1 \\ \frac{1-x^*}{F^*} & -\frac{x^*}{F^*} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0.4 & -0.1 \end{pmatrix}$$

(d) The corresponding RGA matrix is (at all frequencies)

$$\Lambda = \begin{pmatrix} x^* & 1-x^* \\ 1-x^* & x^* \end{pmatrix} = \begin{pmatrix} 0.2 & 0.8 \\ 0.8 & 0.2 \end{pmatrix}$$

For decentralized control, it then follows from pairing rule 1 ("prefer pairing on RGA elements close to 1") that we should pair on the off-diagonal elements; that is, use $u_1$ to control $y_2$ and use $u_2$ to control $y_1$.
(e) This corresponds to using the largest stream (water, $u_2$) to control the amount ($y_1 = F$), which is reasonable from a physical point of view. Also note that the RGA-elements are always between 0 and 1 for this process, and the RGA-elements are all 0.5, corrresponding to "switching" the pairings, when $x^* = 0.5$, which is when the two feed streams are equal.

# 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

# Summary
# Single-loop control = Decentralized control

*Use for*: Noninteracting process

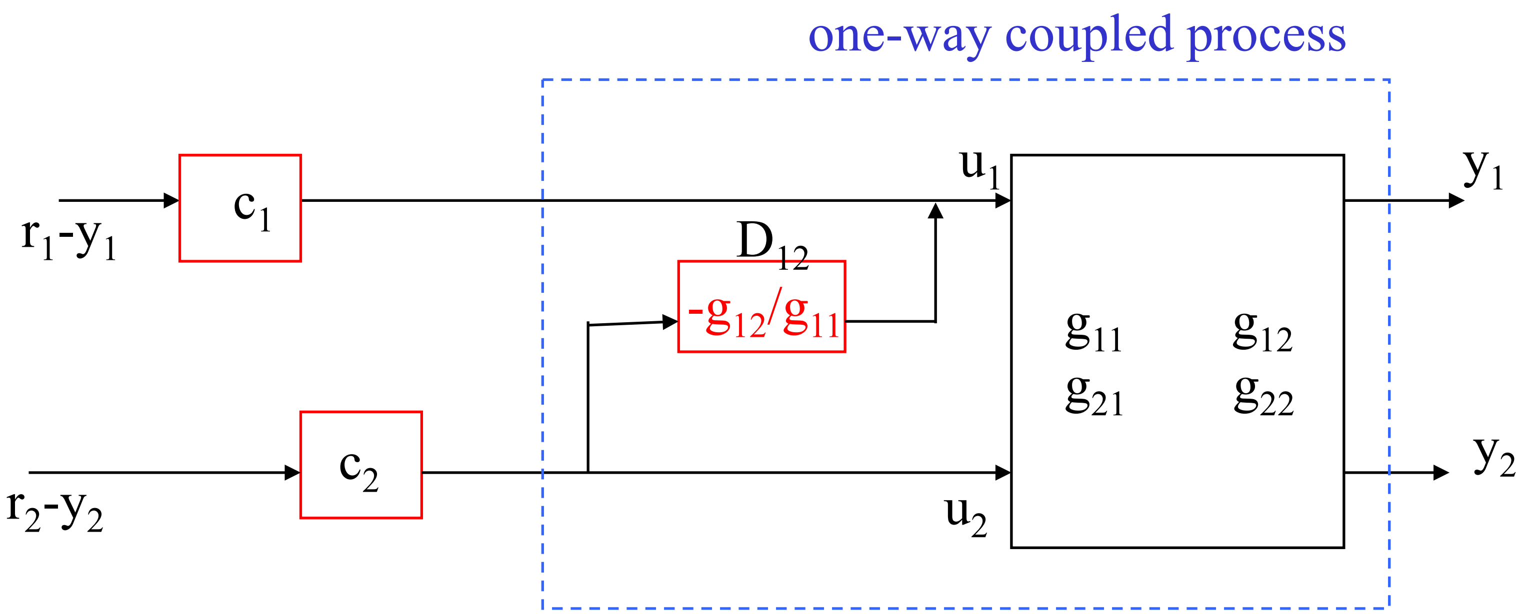
+ Tuning may be done on-line
+ No or minimal model requirements
+ Easy to fix and change

- Need to determine pairing
- Performance loss compared to multivariable control

# Multivariable control

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

## II. Decoupling
## a) One-way Decoupling (improved control of $y_1$)



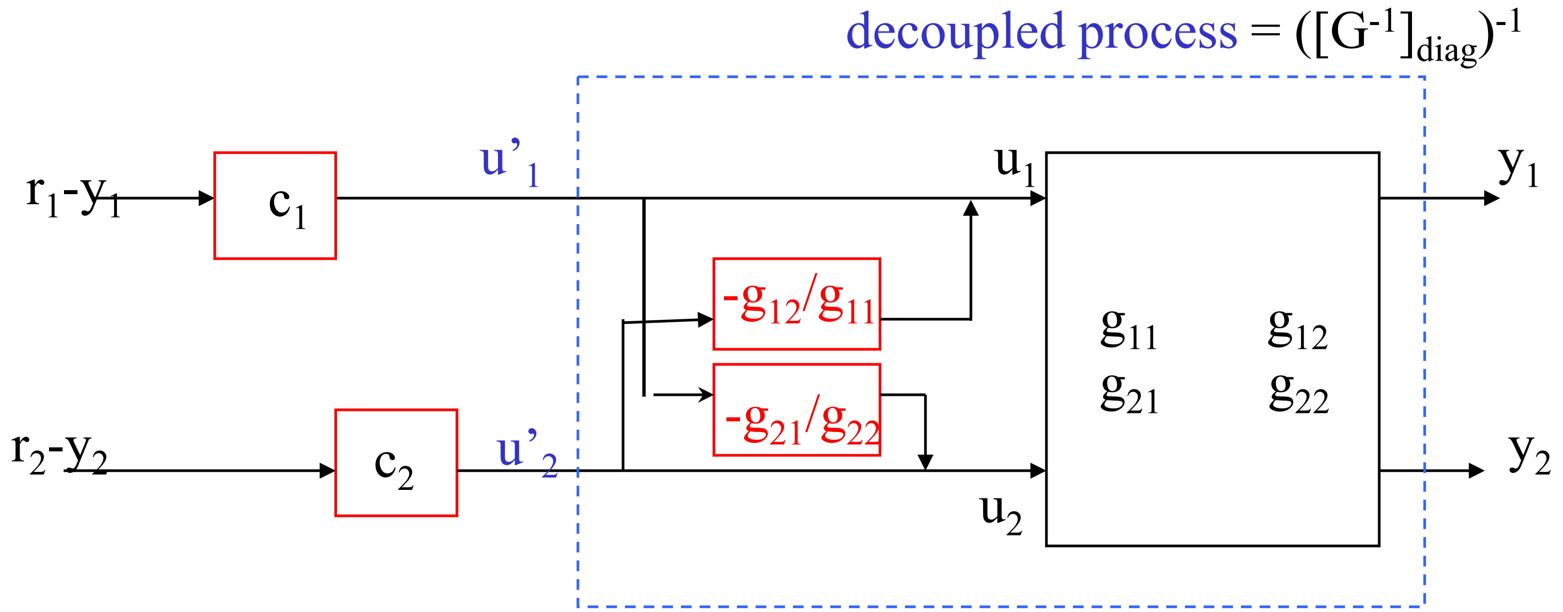

DERIVATION

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 $u_1 = -(g_{12}/g_{11})\, u_2$

# b) Two-way Decoupling: Standard implementation (Seborg)



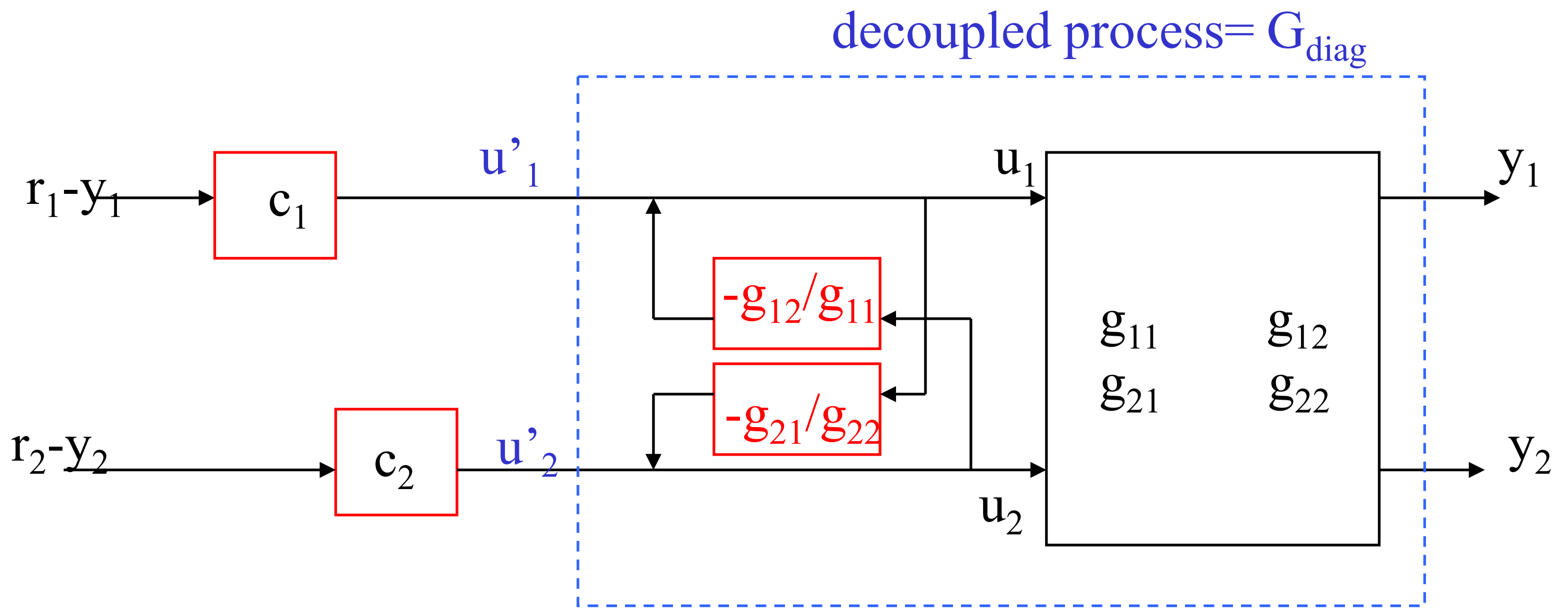

… but note that diagonal elements of decoupled process are different from G
Problem for tuning!

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

Sigurd recommends this alternative!

NTNU

# 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: $y_2 = 0^*u_1' + g_{22} u_2'$

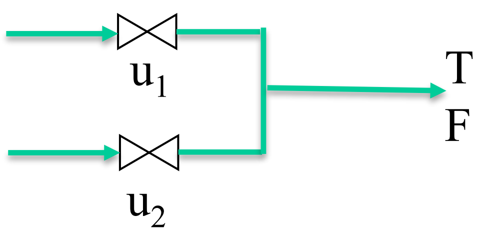
# Pairing and decoupling

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

# Nonlinear decoupling

- It's often easier to make nonlinear decoupler based on static model or insight

# Example: Mixing of hot ($u_1$) and cold ($u_2$) water

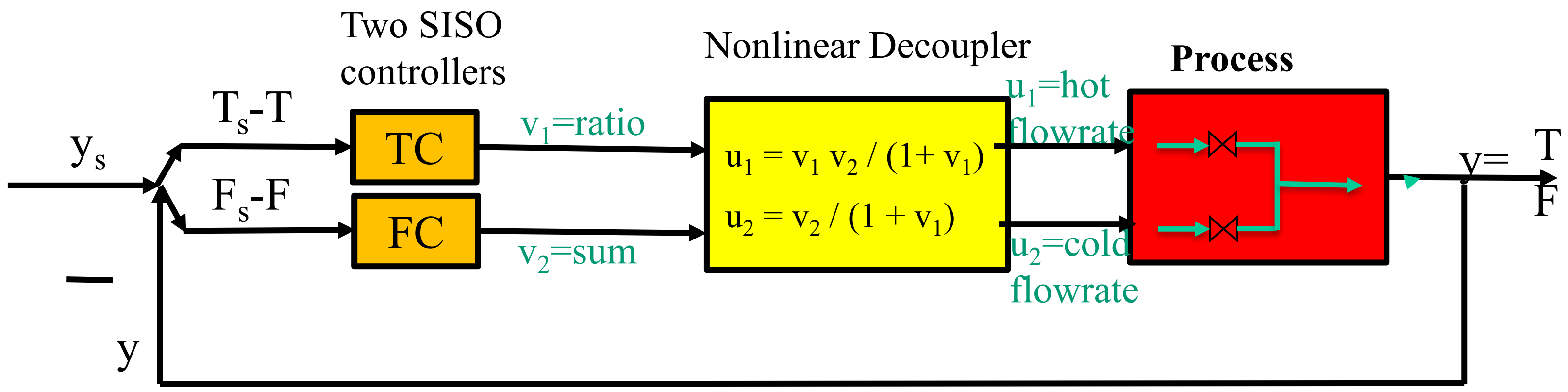




- Want to control
  - $y_1$ = Temperature T
  - $y_2$ = total flow F
- Inputs, u=flowrates
- May use two SISO PI-controllers
  - TC
  - FC
- Insight: Get decoupled response with transformed inputs
  - TC sets flow ratio, $v_1 = u_1/u_2$
  - FC sets flow sum, $v_2 = u_1 + u_2$
- Decoupler: Need «static calculation block» to solve for inputs

$$u_1 = v_1 v_2 / (1+ v_1)$$

$$u_2 = v_2 / (1 + v_1)$$




Pairings:

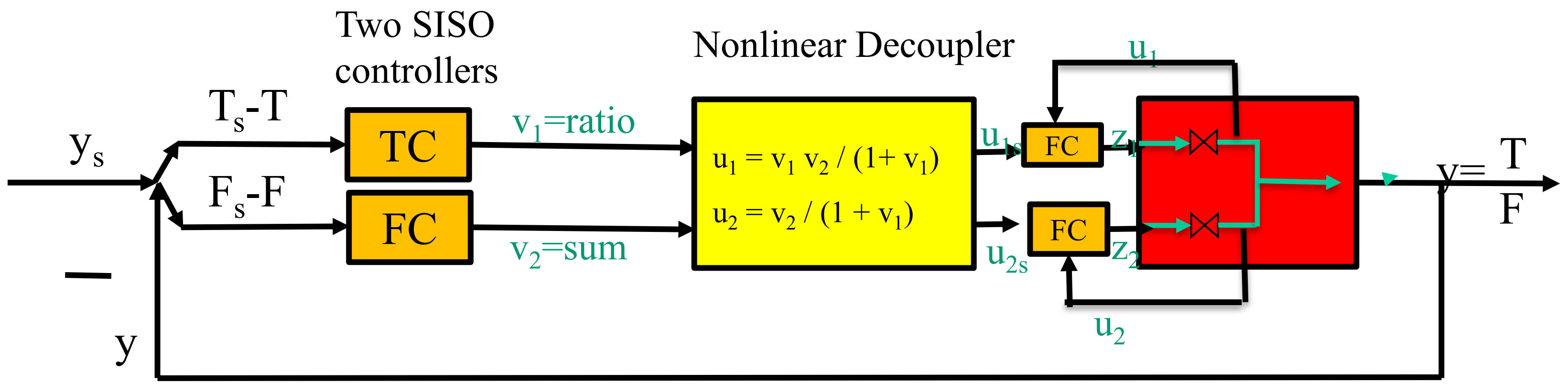
- $T - v_1$
- $F - v_2$

No interactions for setpoint change

Note:

- In practice u=valve position (z)
- So must add two flow controllers
  - These generate inverse by feedback

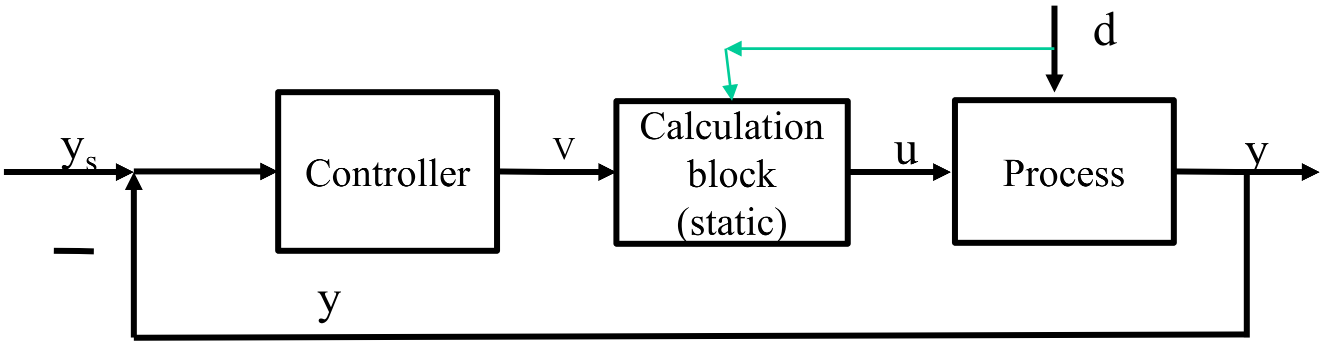
In practice must add two slave flow controllers




v = transformed inputs
u = flowrates
z = valve positions

# General approach: Combined Nonlinear decoupling, feedforward and linearization using Transformed Inputs *

- Linear decoupling and feedforward often work poorly because of nonlinearity
- Example of nonlinear feedforward: Ratio control
- Generalization: Nonlinear calculation block




Genaral Method: Select «transformed inputs» v as right hand side of steady state model equations, y = RHS(d,u,..)

*Zotica, Alsop and Skogestad. 2020 IFAC World Congress

# Example: Combined nonlinear decoupling and feedforward. Mixing of hot and cold water

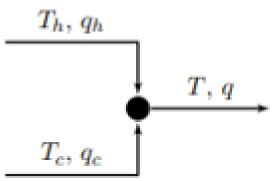
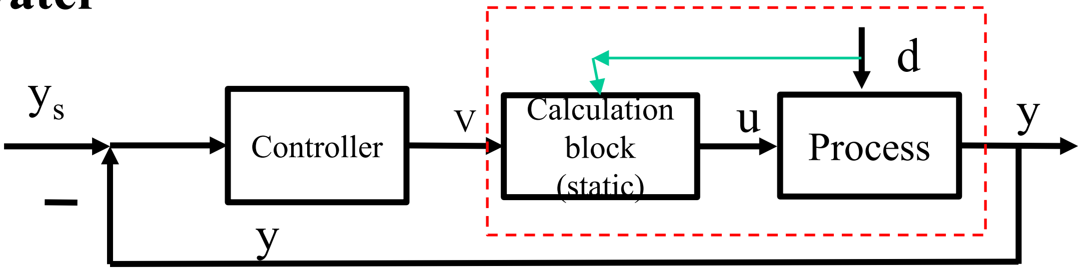



Figure 1: Mixer system




$$u = \begin{pmatrix} q_h \\ q_c \end{pmatrix}$$

$$d = \begin{pmatrix} T_h \\ T_c \end{pmatrix}$$

$$y = \begin{pmatrix} T \\ q \end{pmatrix}$$

Steady-state model written as y=f(u,d):

$$T = \frac{q_h T_h + q_c T_c}{q_h + q_c}$$

$$q = q_c + q_h$$

Select transformed inputs as right hand side, v = f

$$v_1 = \frac{q_h T_h + q_c T_c}{q_h + q_c} \quad (1)$$ Generalized ratio

$$v_2 = q_c + q_h \quad (2)$$

Model from v to y (red box) is then decoupled and with perfect disturbance rejection:

$$T = v_1$$

$$q = v_2$$

- Can then use two single-loop PI controllers for T and q!
  - These controllers are needed to correct for model errors and unmeasured disturbances
- Note that $v_1$ used to control T is a generalized ratio, but it includes also feedforward from Tc and Th.

**Implementation (calculation block)** : Solve (1) and (2) with respect to u=(qc  qh):

Decoupler with feedforward:

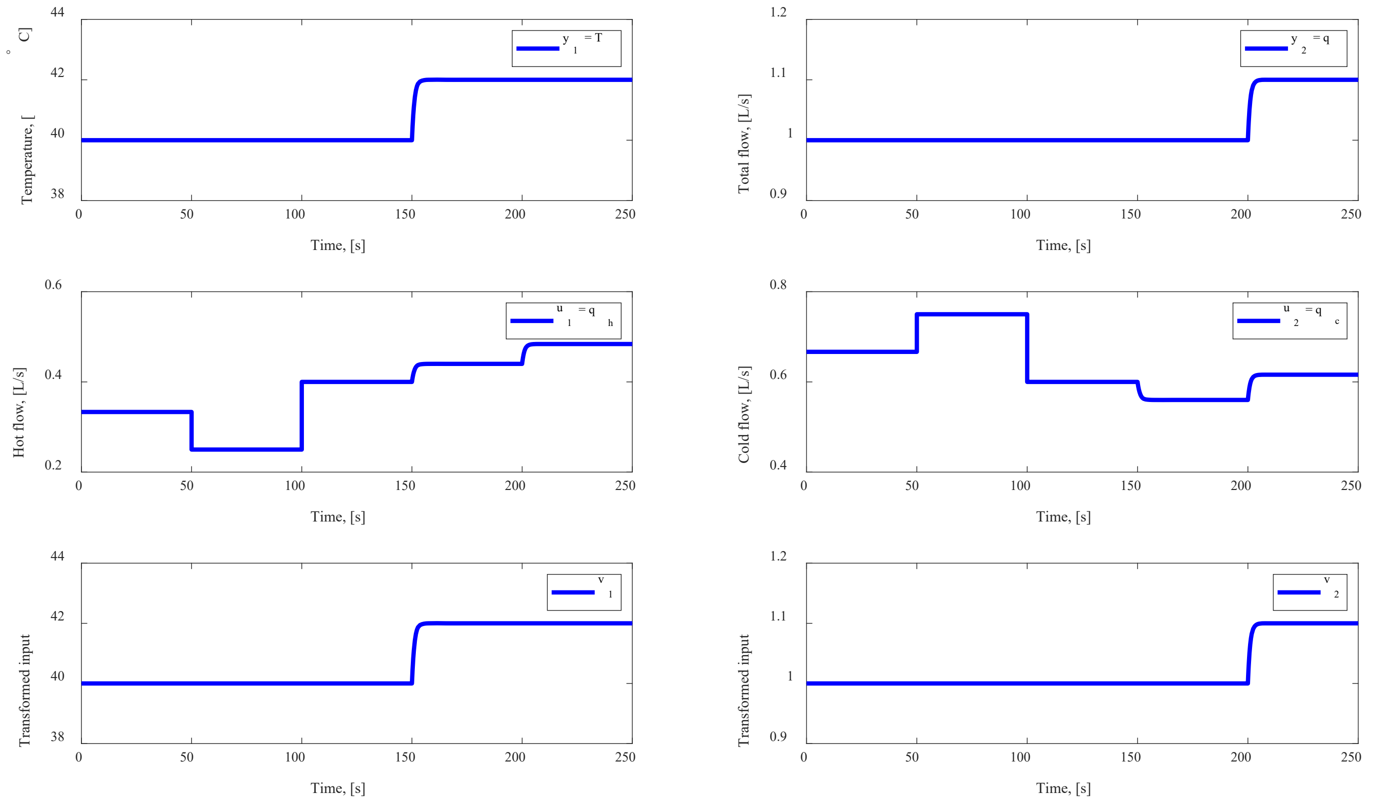
$$q_h = \frac{v_2(v_1 - T_c)}{T_h - T_c}$$

$$q_c = v_2 - q_h$$

Transformed MVs for decupling, linearization and disturbance rejection
Mixing of hot and cold water (static process)
New system: $T=v_1$ and $q=v_2$

Outer loop: Two I-controllers with $\tau_C = 1$ s

1. $T_h$: 60->70 °C at t = 50 s
2. $T_c$: 30->20 °C at t = 100 s
3. $T_h^s$: 40->42 °C at t = 150 s
4. $q^s$: 1->1.1 L/s at t = 200 s



# 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

- no need for logic
- But does not always work

\- 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 can also handle constraints
  - Don't need to add anti-windup

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.

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

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

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

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

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

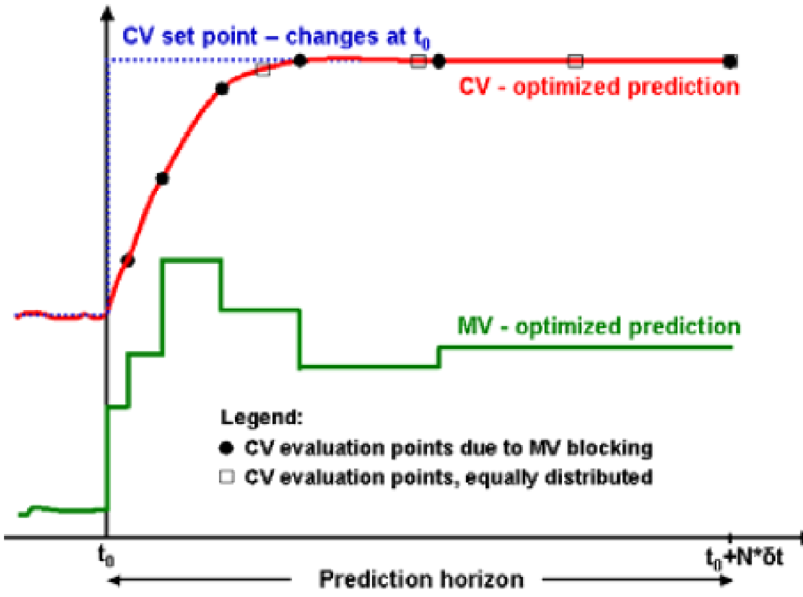



Fig. 1. MV blocking and CV evaluation

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

Discretize in time:

$$y = [y_1\, y_2 \cdots y_n]$$

$$u = [u_1\, u_2 \ldots u_k]$$

$$\Delta u = [\Delta u_1\, \Delta u_2 \ldots \Delta u_k]$$

$$\Delta u_i = u_i - u_{i-1}$$

Note: Implement only current input $\Delta u_1$

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, Equinor)

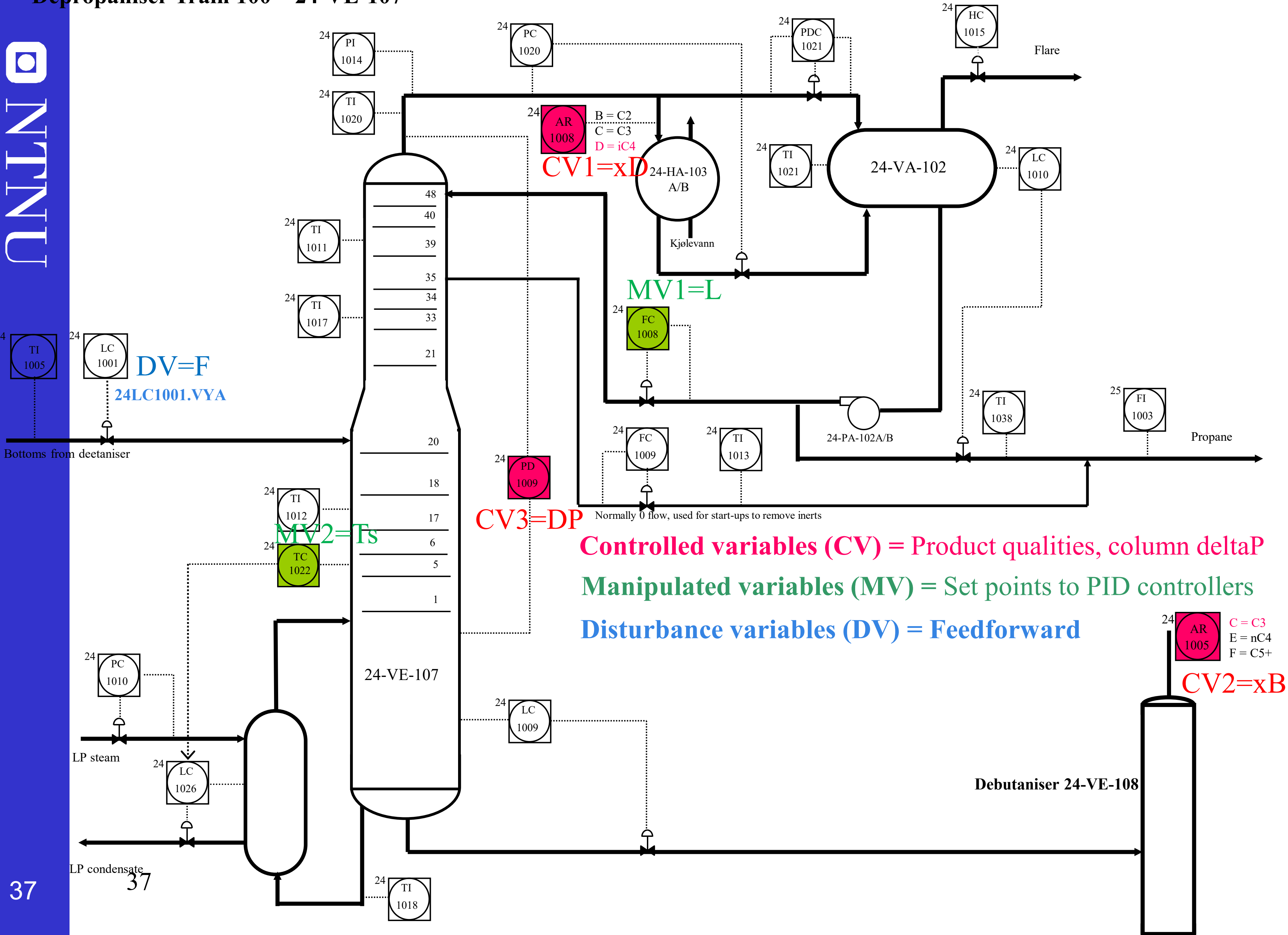
- Initial MV/CV/DV selection
- DCS* preparation (controller tuning, instrumentation, MV handles, communication logics etc)
- Control room operator pre-training and motivation
- Product quality control → Data collection (process/lab) → Inferential model ("soft sensor")

- 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

## Depropaniser Train 100 – 24-VE-107




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

**Manipulated variables (MV) = Set points to PID controllers**

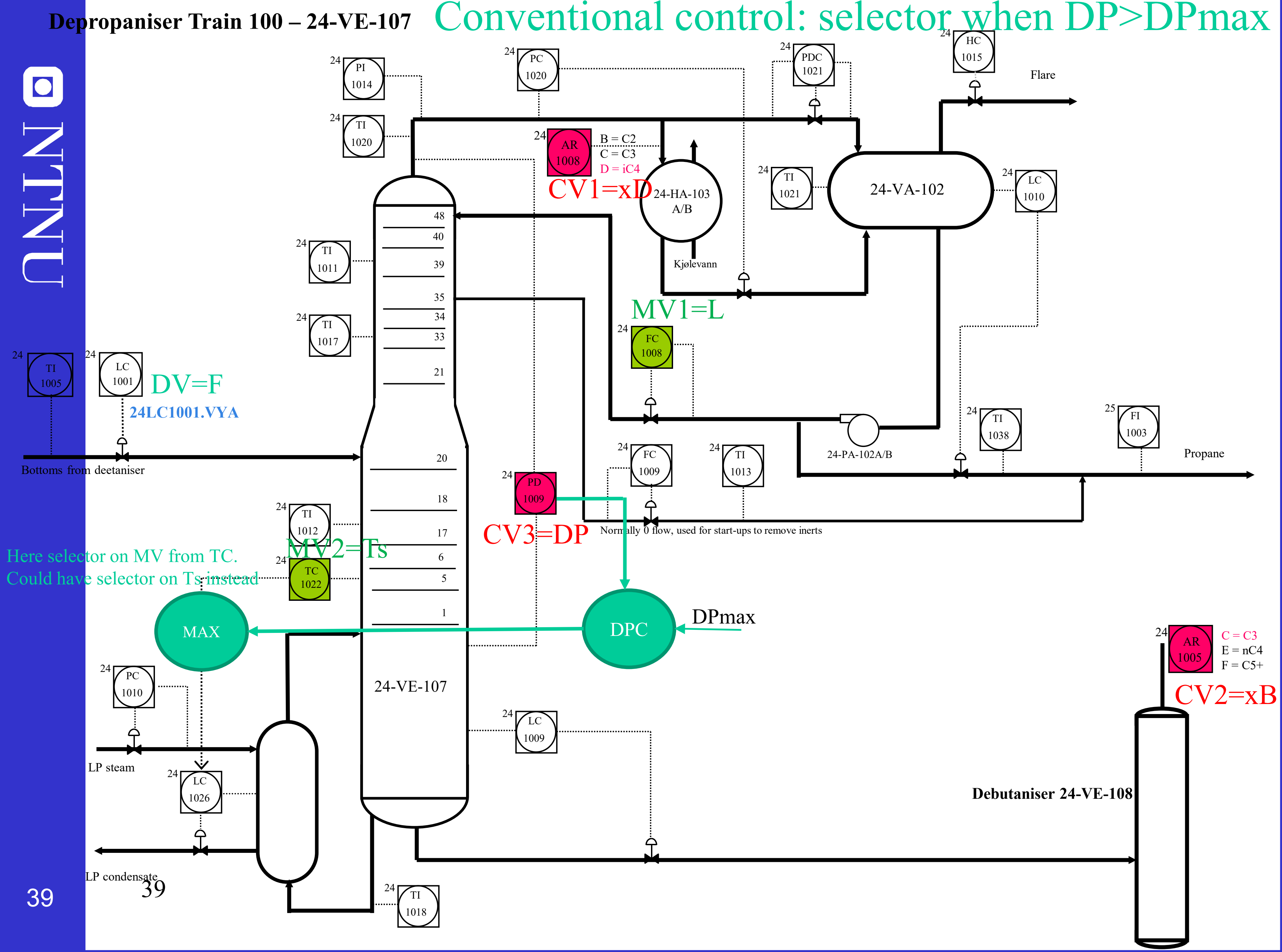
**Disturbance variables (DV) = Feedforward**

**Conventional control (PID)**

Select pairings MV-CV (obvious here: L-xD, Ts-xB)
Selector: Give up xB when we meet DP constraint

Tune 3 PI controllers

# Conventional control: selector when DP>DPmax



**Conventional control (PID)**

Select pairings MV-CV (obvious here: L-xD, Ts-xB)
Selector: Give up xB when we meet DP constraint

Tune 3 PI controllers

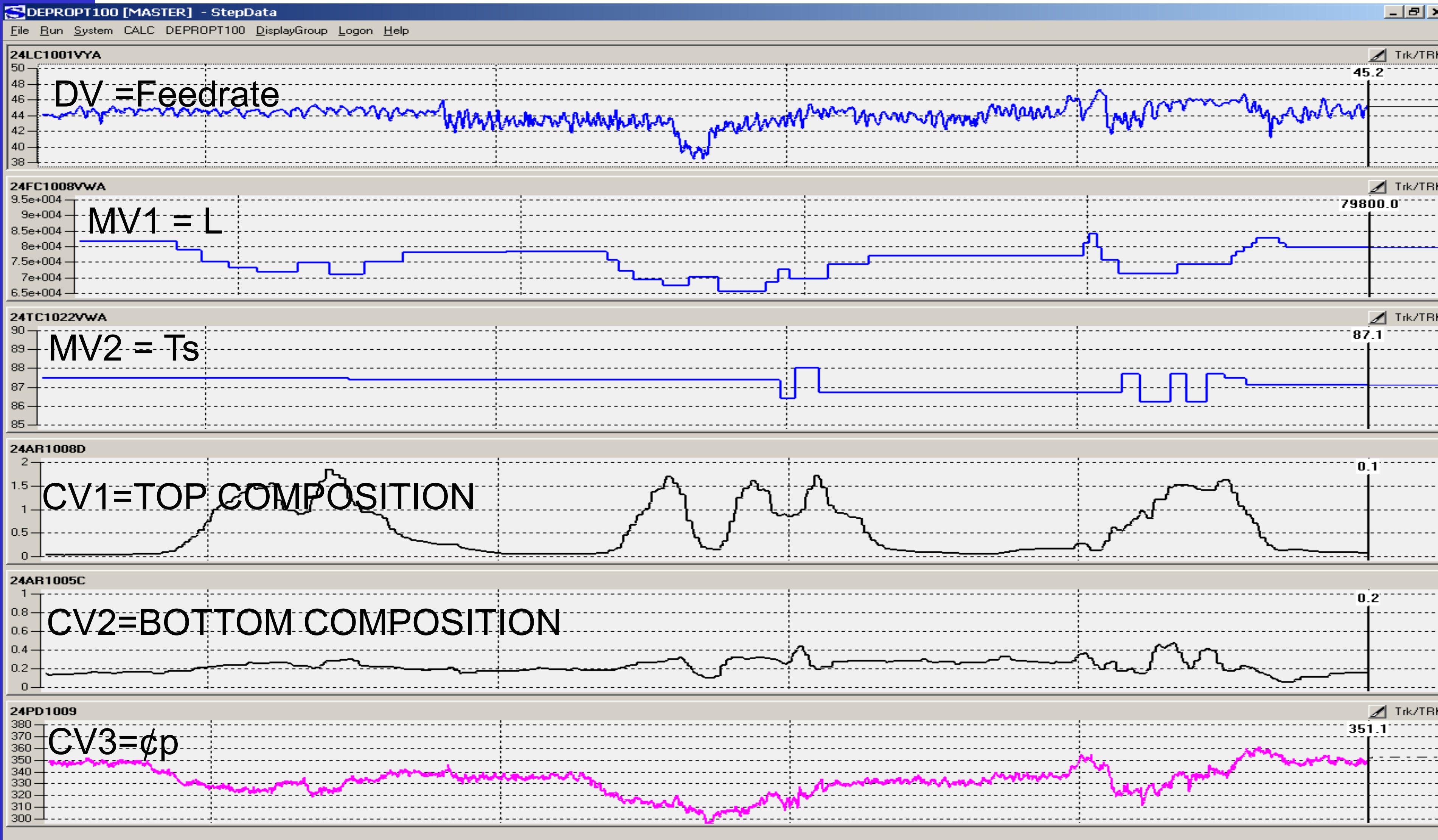
**MPC**

Don't need to make pairing choices

But need model
And need to tune MPC controller

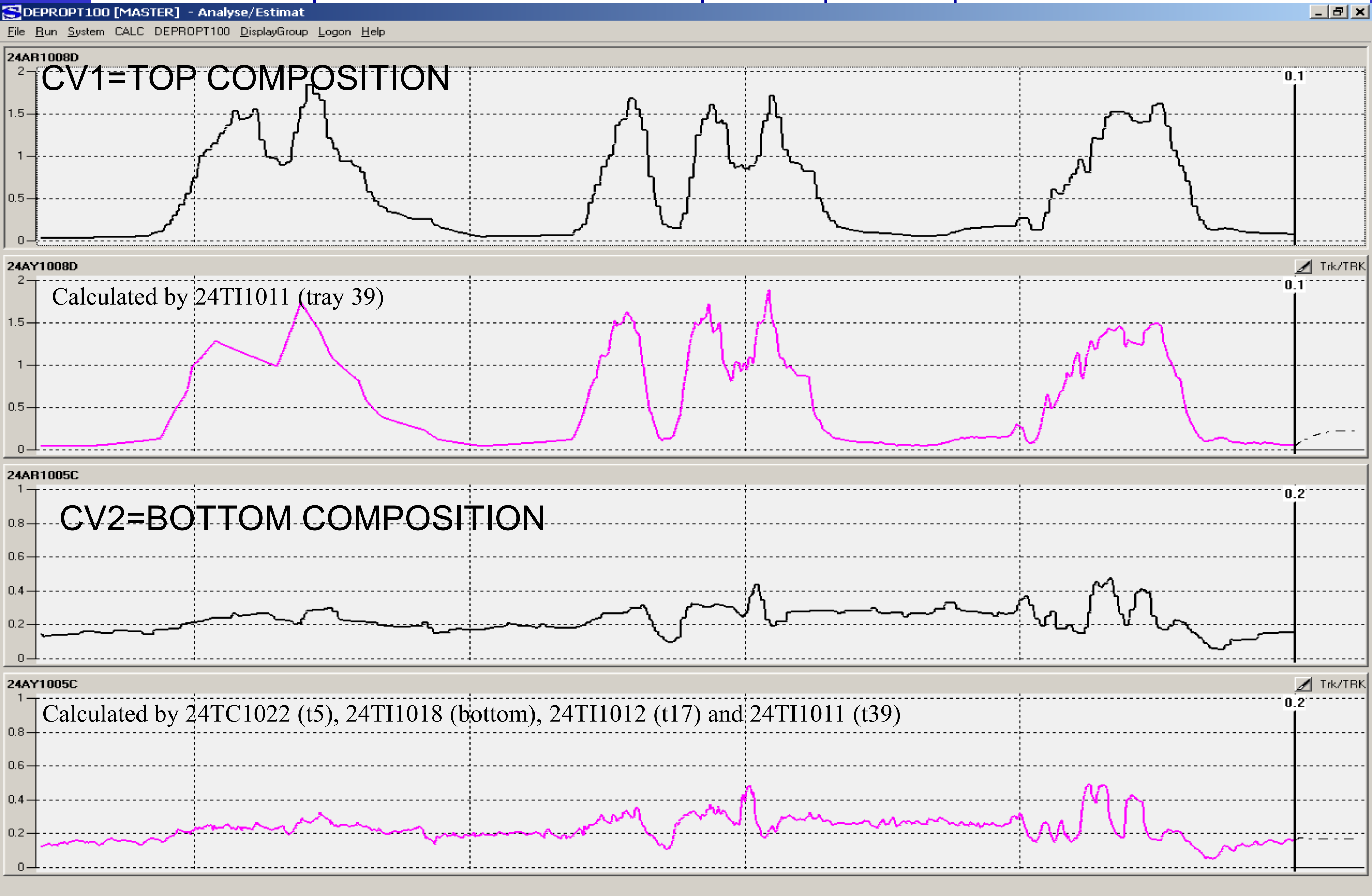
# Depropaniser Train100 step testing

3 days – normal operation during night



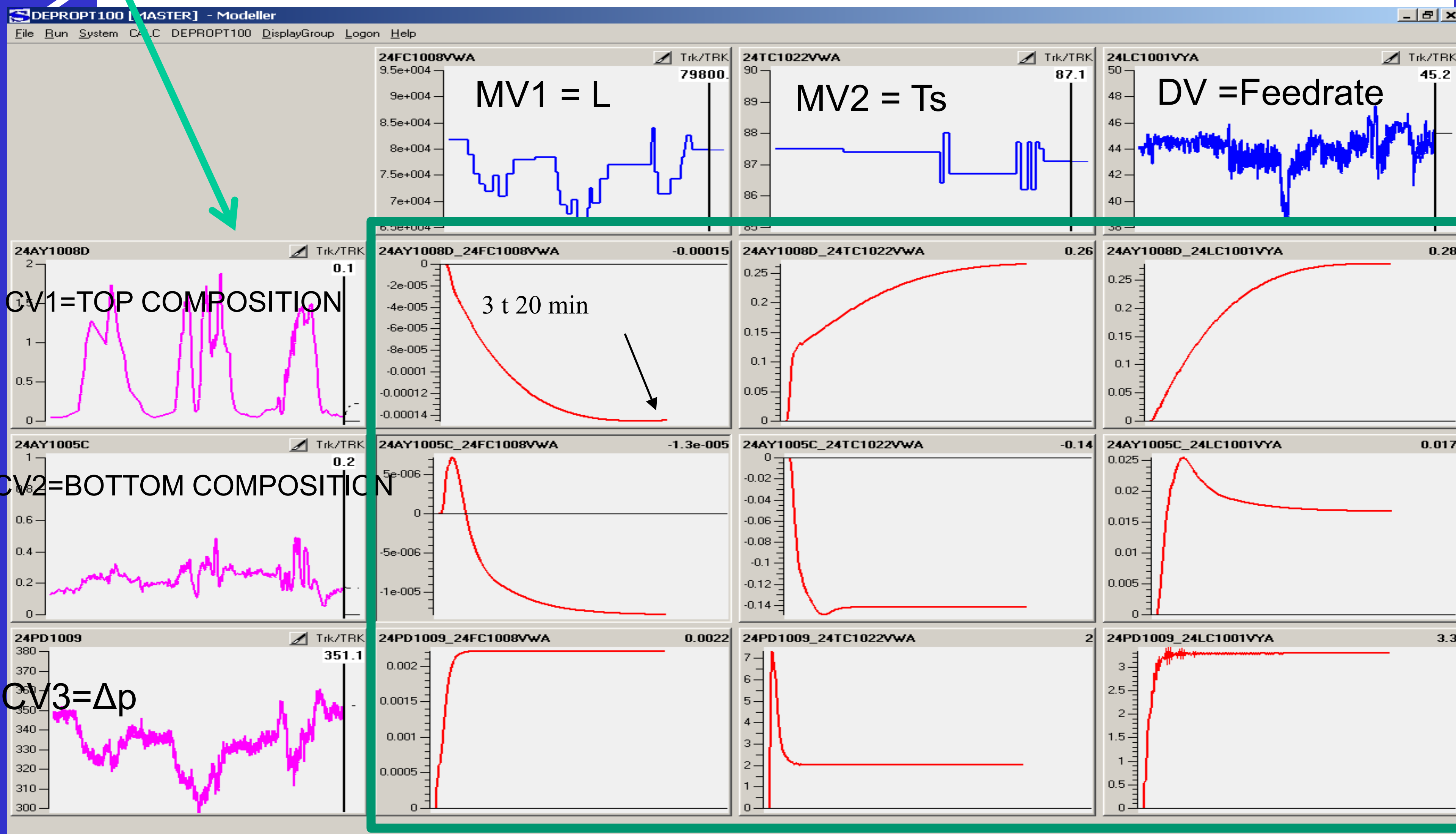
# Estimator: inferential models

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



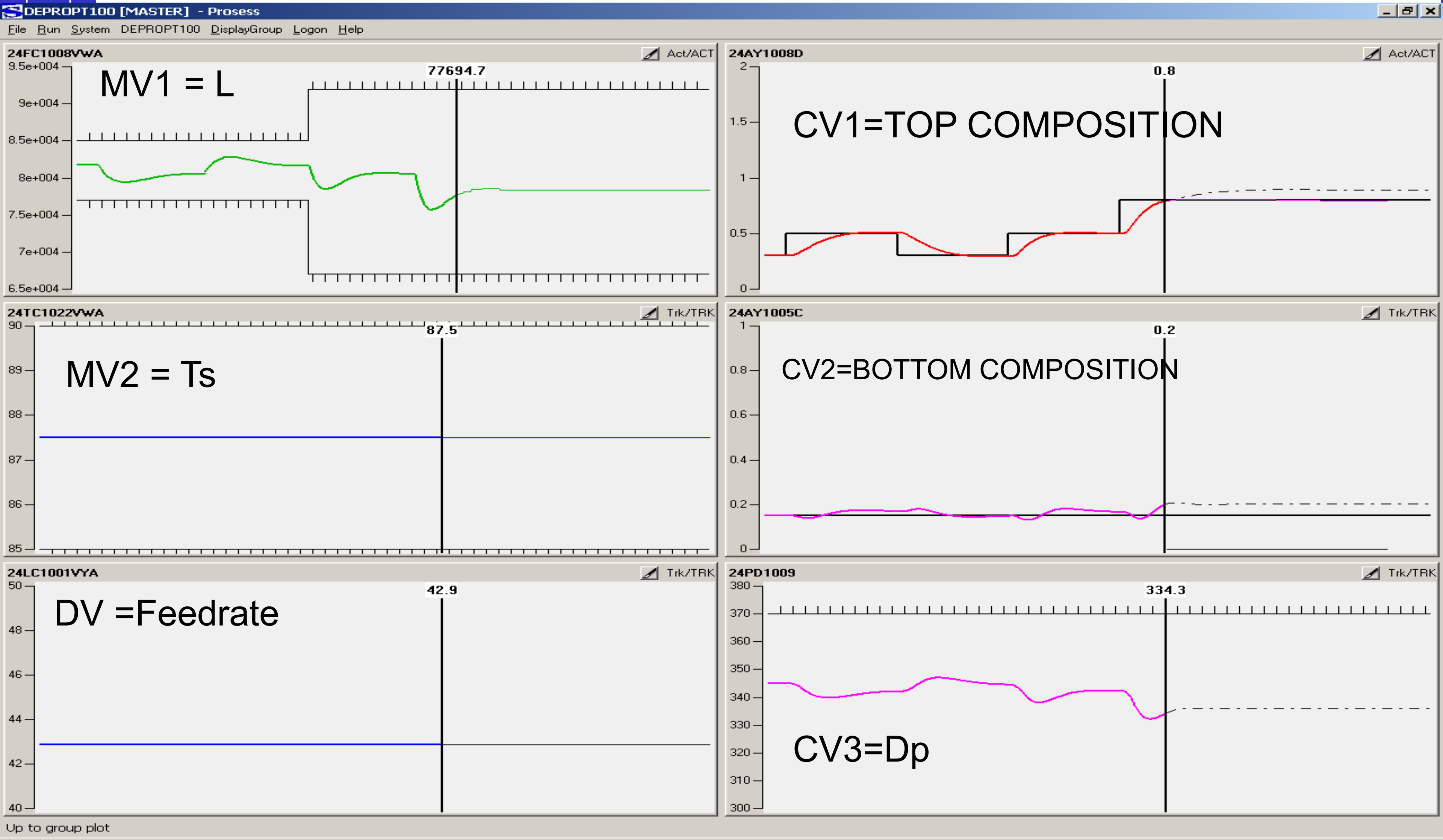
# Depropaniser Train100 step testing – Final model

- 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%.

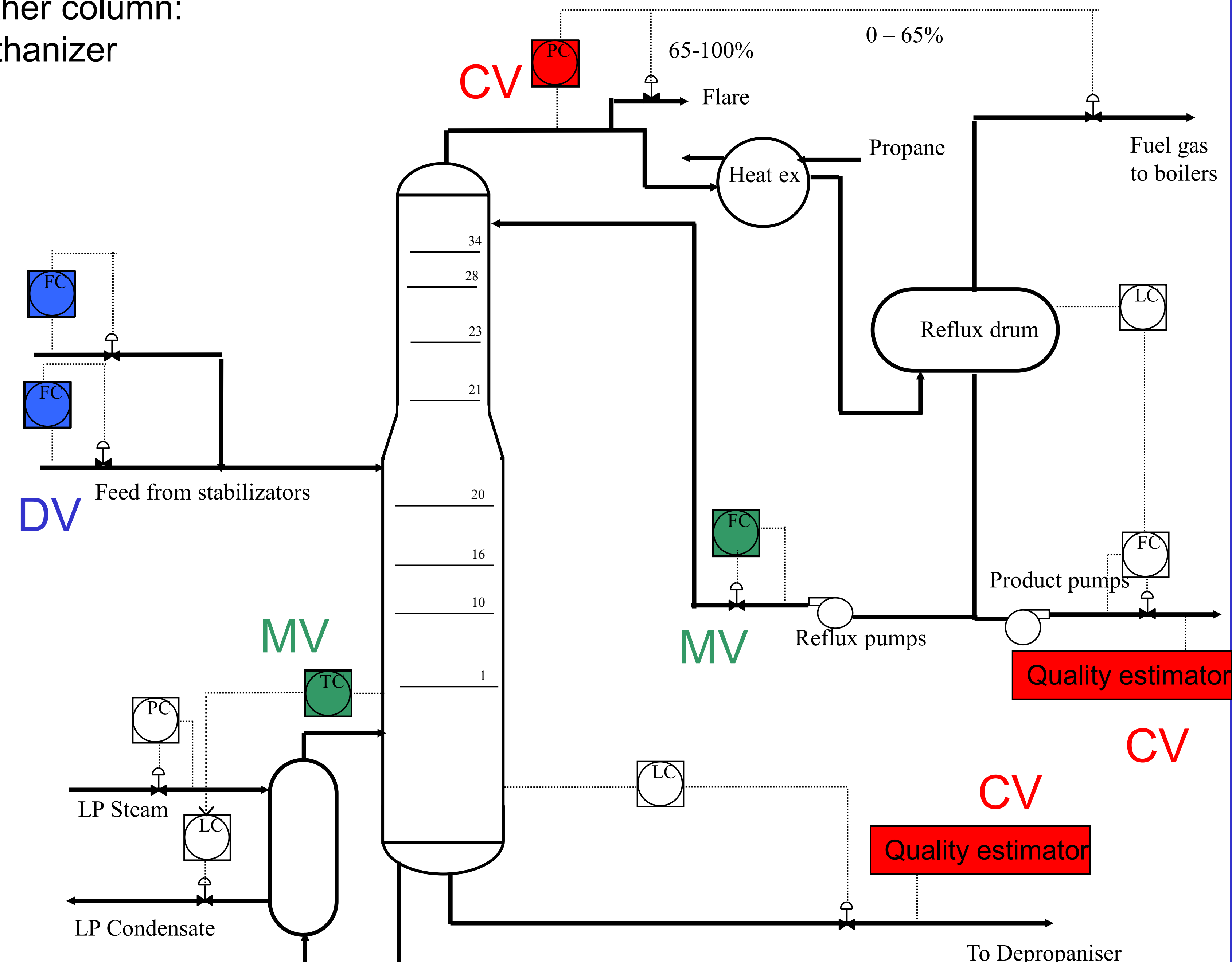


# Depropaniser Train100 MPC – controller activation

- 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)

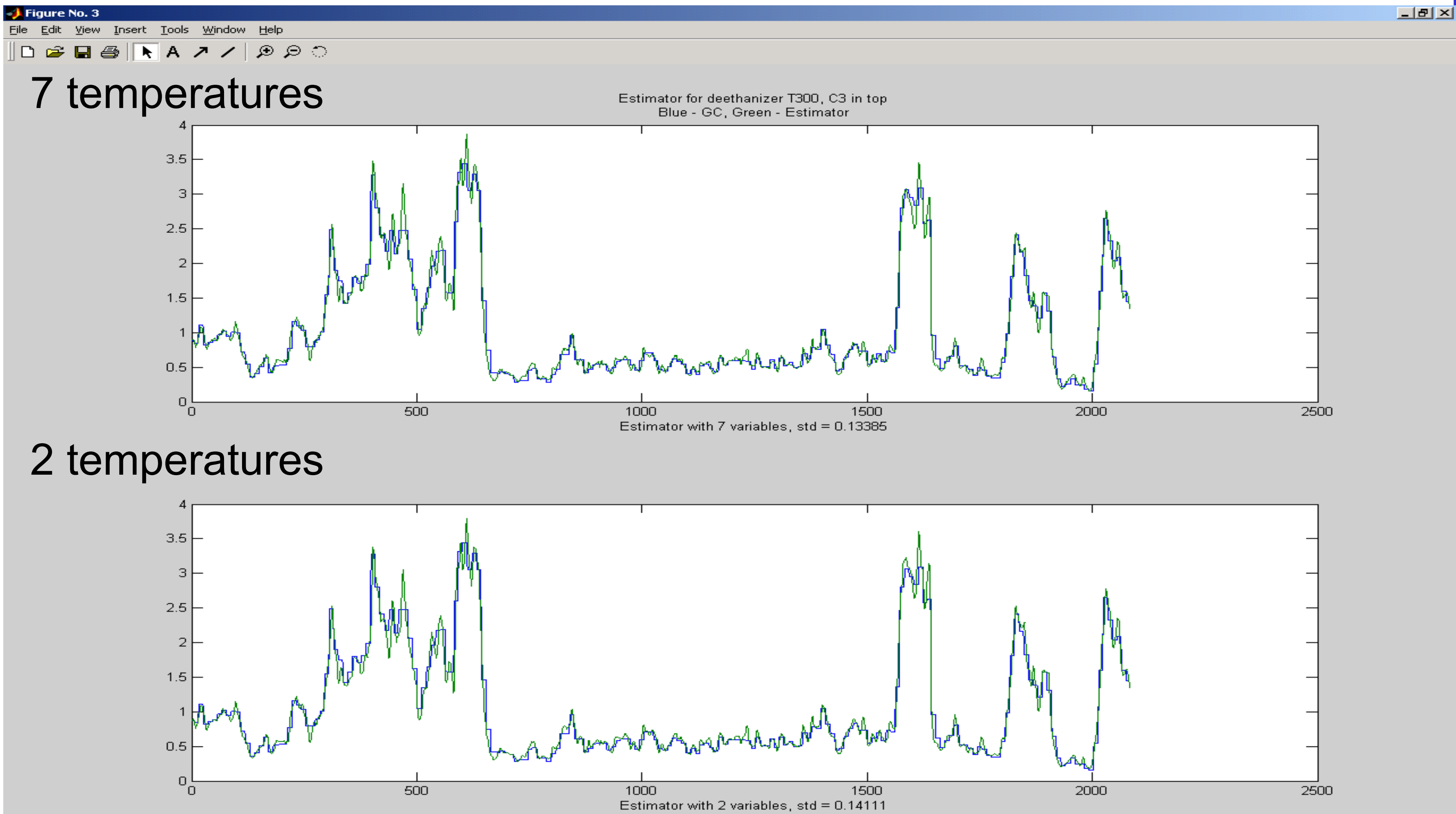


# Another column: Deethanizer



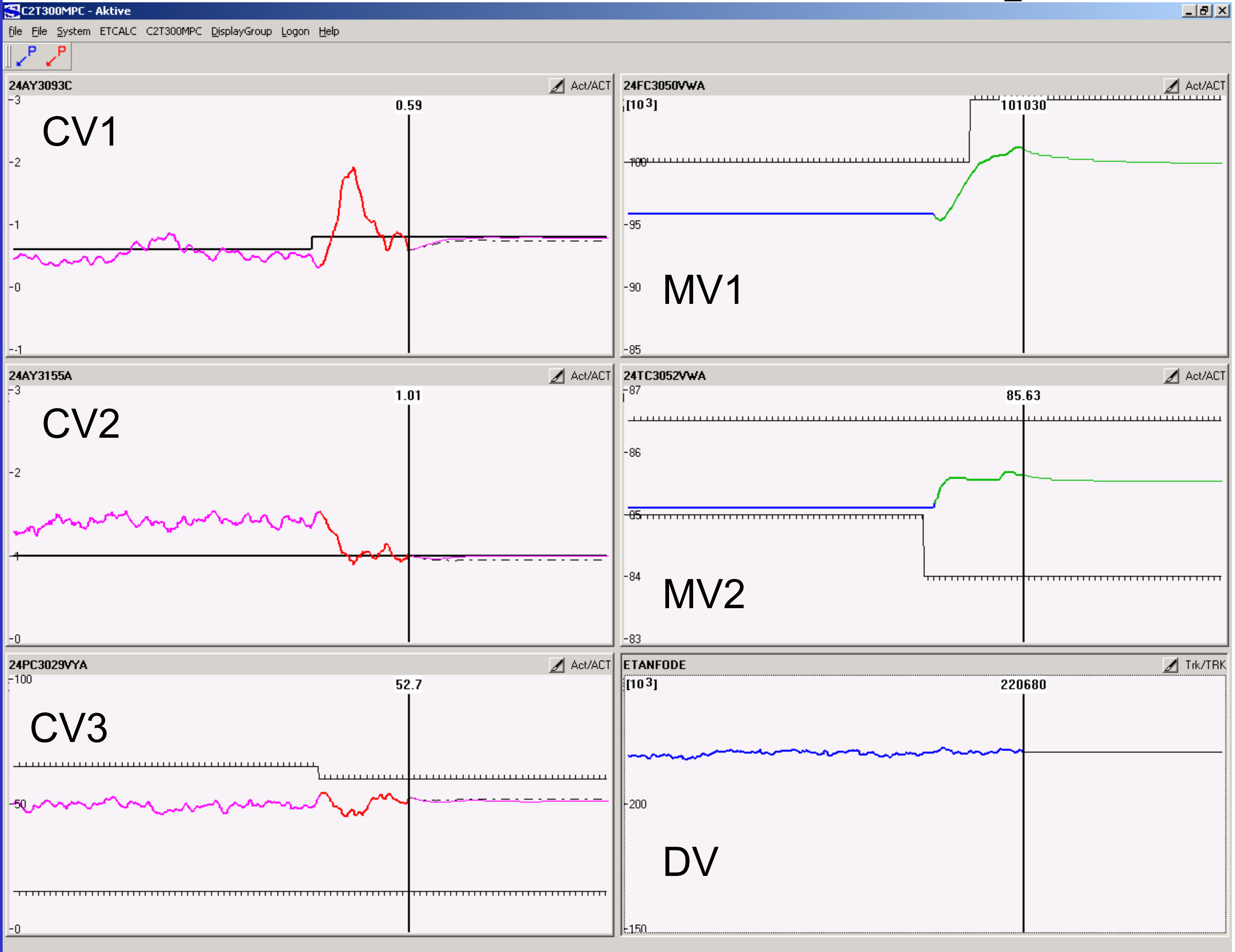
# Top: Binary separation in this case
# Quality estimator vs. gas chromatograph

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




=little difference if the right temperatures are chosen

# The final test: MPC in closed-loop



# Conclusion MPC

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

# V. 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

$$dx/dt = A\,x + B\,u$$

  - and use "State feedback" (assuming we can measure all the states)

$$u = K\,x$$

Note

1. SIMC is "pole placement" (p=-1/tauc), but with output feedback (y), and we also place zeros
2. If we cannot measure all the states, then we can estimate x from y using a "Kalman filter".
3. State feedback uses extra measurements – an altertive is cascade control

### 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

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

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

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

NTNU

### 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.

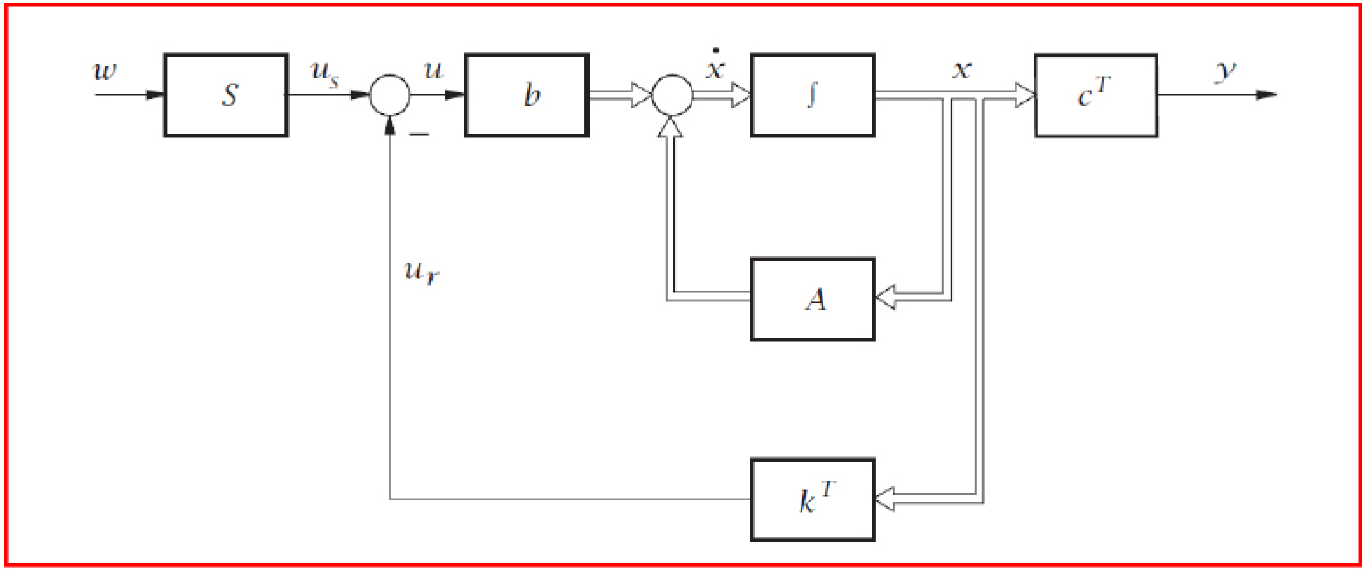



Figure 8-6: SISO-system with state feedback

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