Part 2. Decomposition and optimal operation

- Hierarchical decomposition. Control layers.
- Design of overall control system for economic process control
- CV selection

Optimal operation and control of process

- Given process plant
- Want to Maximize profit P => Minimize economic cost $J_\$=-P$ [$/s]
  - $J_\$$ = cost feed + cost energy – value products
    - Excluding fixed costs (capital costs, personell costs, etc)
- Subject to satisfying constraints on
    - Products (quality)
    - Inputs (max, min)
    - States = Internal process variables (pressures, levels, etc)
      - Safety
      - Environment
      - Equipment degradation
- Degrees of freedom = manipulated variables (MVs) = inputs u

Economic motivation for better control: Squeeze and shift rule

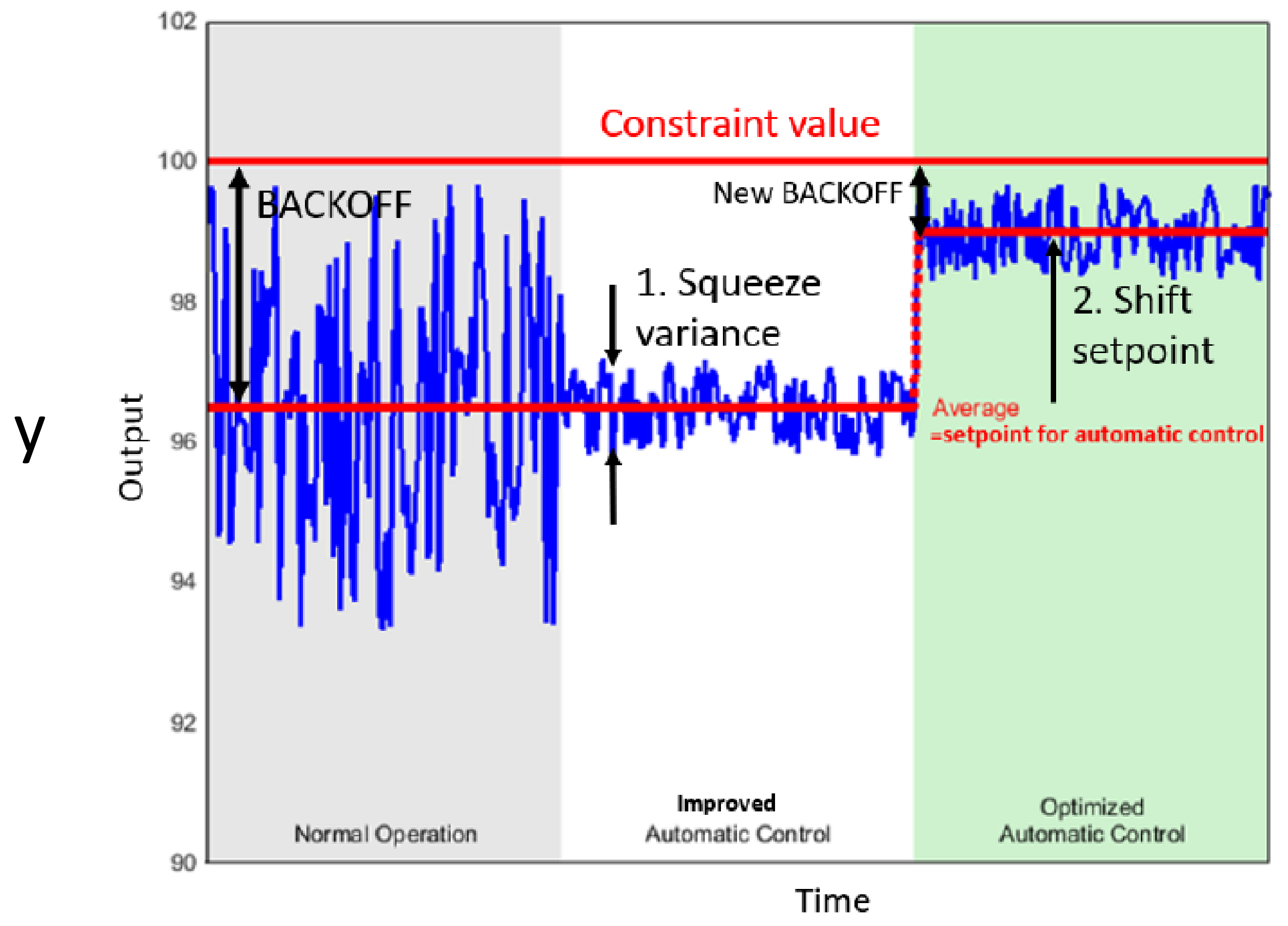
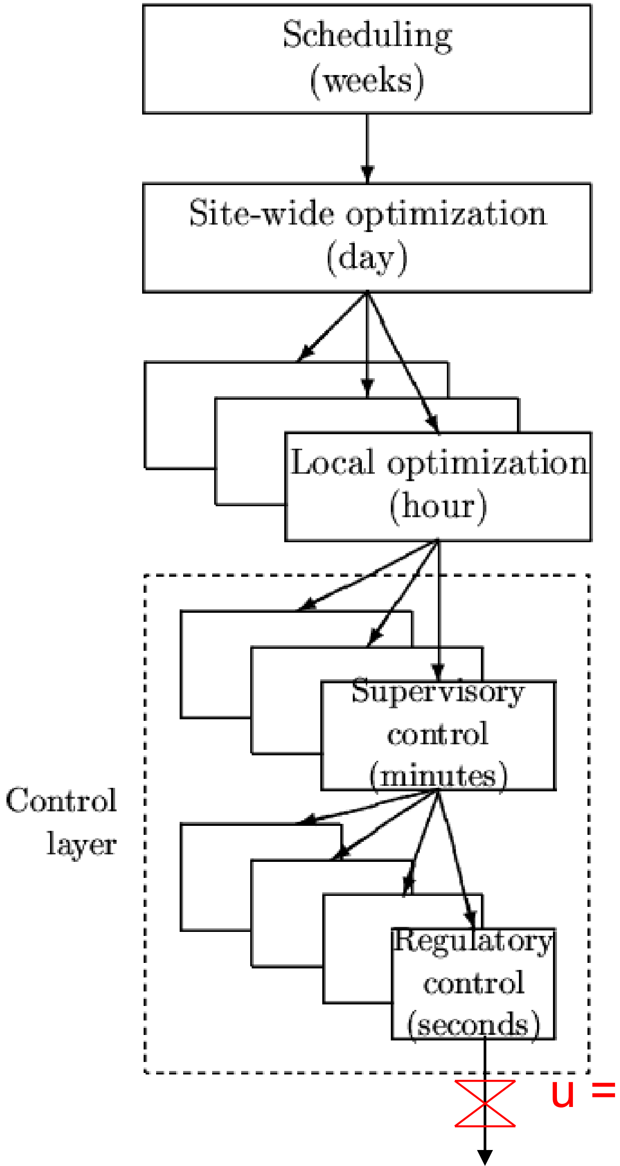


Figure 8: Squeeze and shift rule: Squeeze the variance by improving control and shift the setpoint closer to the constraint (i.e., reduce the backoff) to optimize the economics (Richalet et al., 1978).

Practical operation: Hierarchical (cascade) structure based on time scale separation

NOTE: Control system is decomposed both
- Hierarhically (in time)
- Horizontally (in space)

Status industry:
- RTO is rarely used.
- MPC is used in the petrochemical and refining industry, but in general it is much less common than was expected when MPC «took off» around 1990
- ARC is common
- Manual control still common…



Manager

Process engineer

Operator/RTO

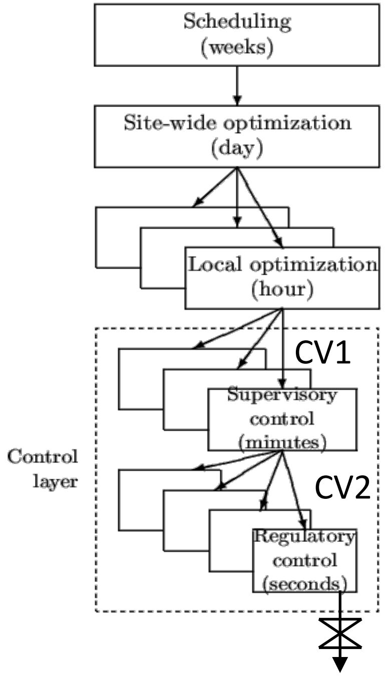
Operator/Advanced regulatory control (ARC)/MPC

PID-control
May include some ratio/feedforward and cascade control

u = valves

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

CV = controlled variable
MV = manipulated variable

Main objectives operation

1. **Economics**: Implementation of acceptable (near-optimal) operation

2. **Regulation:** Stable operation around given setpoint

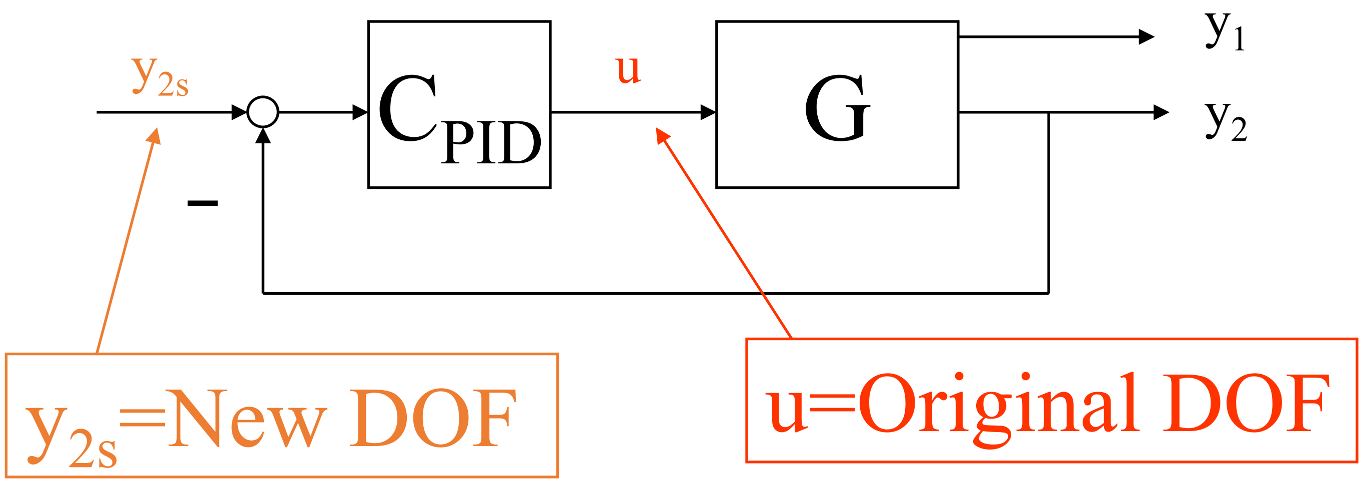
ARE THESE OBJECTIVES CONFLICTING?
IS THERE ANY LOSS IN ECONOMICS?

- Usually NOT
  - Different time scales
    - Stabilization fast time scale
  - Stabilization doesn’t “use up” any degrees of freedom
    - Reference value (setpoint) available for layer above
    - But it “uses up” part of the time window

Hierarchical structure: Degrees of freedom unchanged

- No degrees of freedom lost as setpoints y_{2s} replace inputs u as new degrees of freedom for control of y_1

Cascade control:



Systematic procedure for economic process control

Start "top-down" with economics (steady state):

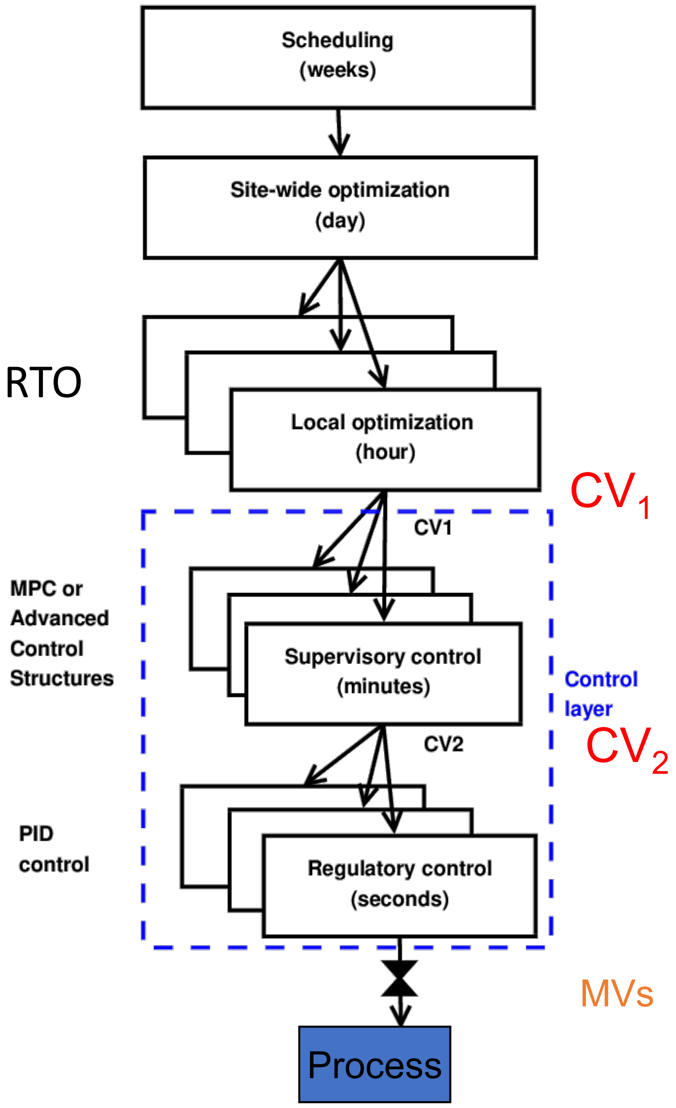
- Step 1: Define operational objectives (J) and constraints
- Step 2: Optimize steady-state operation
- Step 3: Decide what to control (CVs)
  - Step 3A: Identify active constraints = primary CV1.
  - Step 3B: Remaining unconstrained DOFs: Self-optimizing CV1 (find H)
- Step 4: Where do we set the throughput? TPM location

Then bottom-up design of control system (dynamics):

- Step 5: Regulatory control
  - Control variables to stop "drift" (sensitive temperatures, pressures, ...)
  - Inventory control radiating around TPM

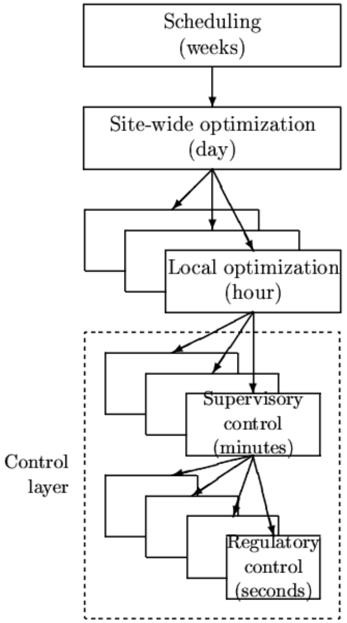
Finally: Make link between "top-down" and "bottom up"

- Step 6: "Advanced/supervisory control"
    - Control economic CVs: Active constraints and self-optimizing variables
    - Look after variables in regulatory layer below (e.g., avoid saturation)
- **Step 7: Real-time optimization** (Do we need it?)



S. Skogestad, ``Control structure design for complete chemical plants'',
Computers and Chemical Engineering, **28** (1-2), 219-234 (2004).

Example: Bicycle riding
Design of control system

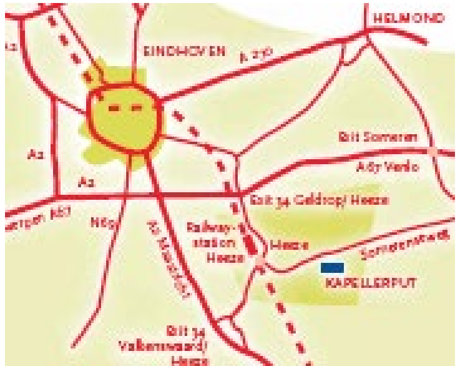


Note: design starts from the bottom

- *Regulatory control (step 5)*:
  - First need to learn to stabilize the bicycle
    - CV = y_2 = tilt of bike
    - MV = body position

- *Supervisory control (step 6)*:
  - Then need to follow the road.
    - CV = y_1 = distance from right hand side
    - MV=y_{2s}
  - Usually a constant setpoint policy is OK, e.g. y_{1s}=0.5 m

- *Optimization (step 7)*:
  - Which road should you follow?
  - Temporary (discrete) changes in y_{1s}



Step 1. Define optimal operation (economics)

NTNU

Usually steady state

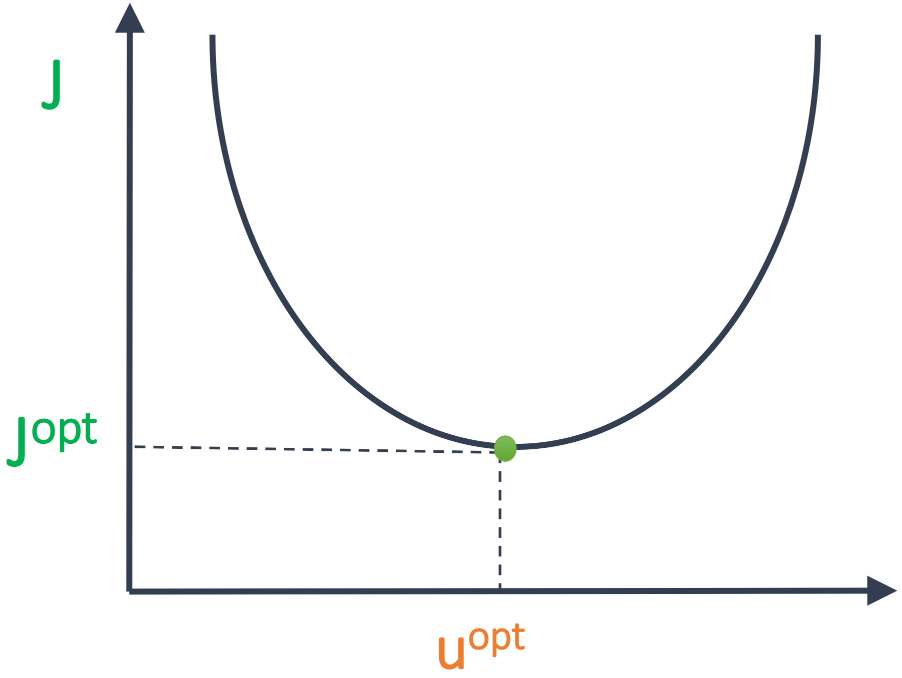
Minimize cost $J = J(u,x,d)$

subject to:

Model equations: $f(u,x,d) = 0$

Operational constraints: $g(u,x,d) < 0$

- u = degrees of freedom
- x = states (internal variables)
- d = disturbances



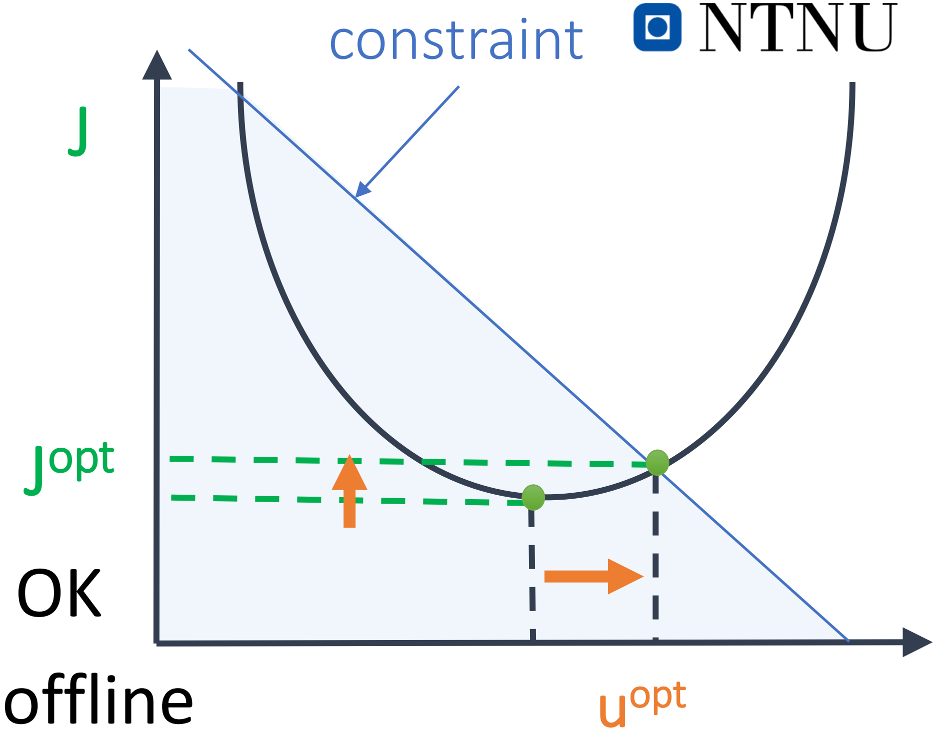
Typical cost function in process control:

J = cost feed + cost energy – value of products

Step 2. Optimize

(a) Identify degrees of freedom
(b) Optimize for expected disturbances

- Need good model, usually steady-state is OK
- Optimization is time consuming! But it is offline
- Main goal: Identify **ACTIVE CONSTRAINTS**
- A good engineer can often guess the active constraints



NTNU

Step 3. Decide what to control (Economic CV1=Hy)

“Move optimization into the control layer by selecting the right CV1”

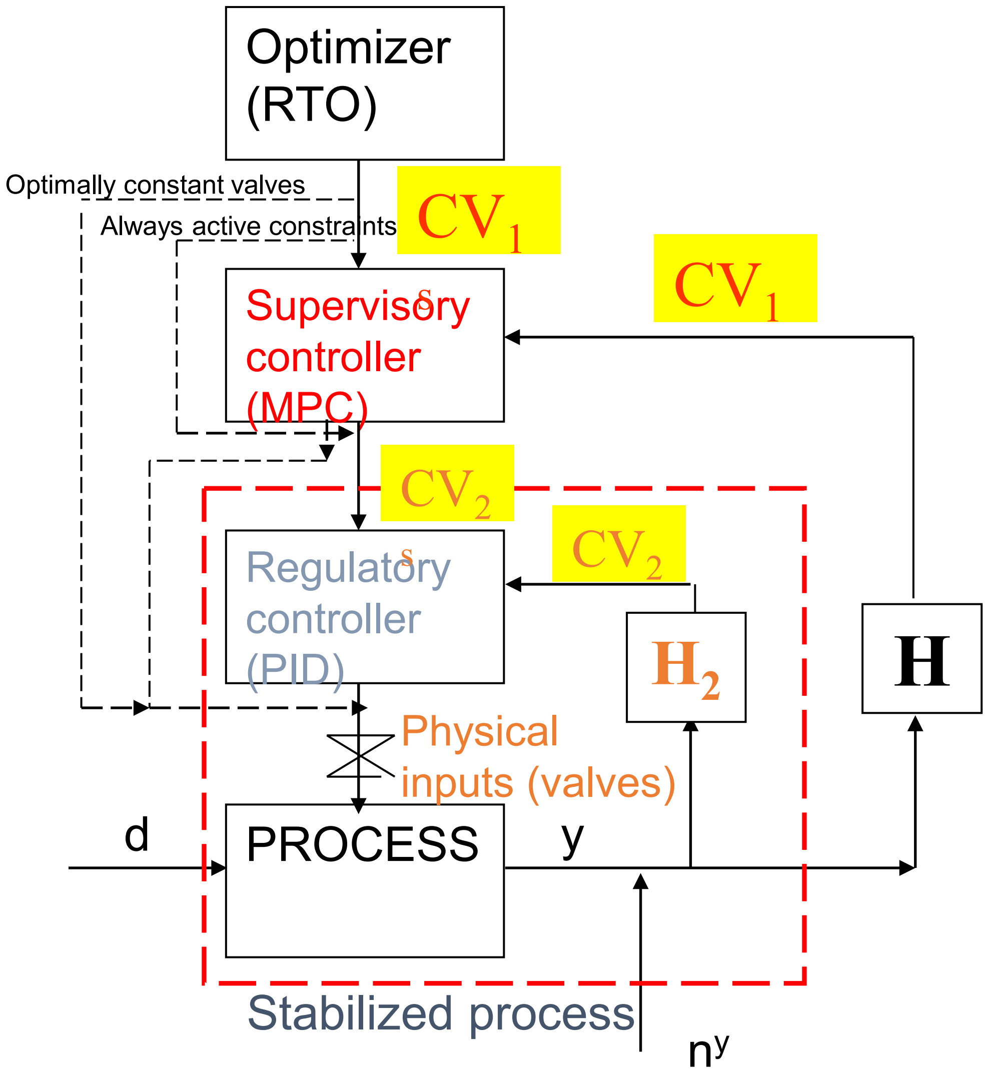
(Morari *et al.*, 1980): *“We want to find a function c of the process variables which when held constant, leads automatically to the optimal adjustments of the manipulated variables, and with it, the optimal operating conditions.”*

Economic CV1:

1. Control active constraints

2. Control Self-optimizing variables

- Look for a variable c that can be kept constant



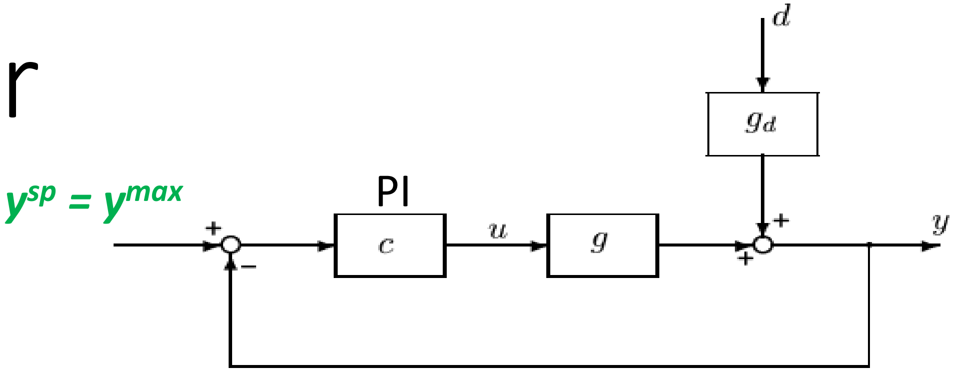
Sigurd's rules for CV selection

1. Always control active constraints! (almost always)
2. Purity constraint on expensive product always active (no overpurification):
   (a) "Avoid product give away" (e.g., sell water as expensive product)
   (b) Save energy (costs energy to overpurify)

Unconstrained optimum:

3. Look for "self-optimizing" variables. They should
   - Be sensitive to the MV
   - have close-to-constant optimal value
4. **NEVER try to control a variable that reaches max or min at the optimum**
   - In particular, never try to control directly the cost J
   - Assume we want to minimize J (e.g., J = V = energy) - and we make the stupid choice os selecting CV = V = J
     - Then setting J < Jmin: Gives infeasible operation (cannot meet constraints)
     - and setting J > Jmin: Forces us to be nonoptimal (which may require strange operation)

Optimization with PI-controller



max y

s.t. $y \le y^{max}$

$u \le u^{max}$

Example: Drive as fast as possible to airport (u=power, y=speed, y^{max} = 110 km/h)

- **Optimal solution has two active constraint regions:**
  1. $y = y^{max}$ → **speed limit**
  2. $u = u^{max}$ → **max power**
- Note: Positive gain from MV (u) to CV (y)
- **Solved with PI-controller**
  - $y^{sp} = y^{max}$
  - Anti-windup: I-action is off when $u = u^{max}$

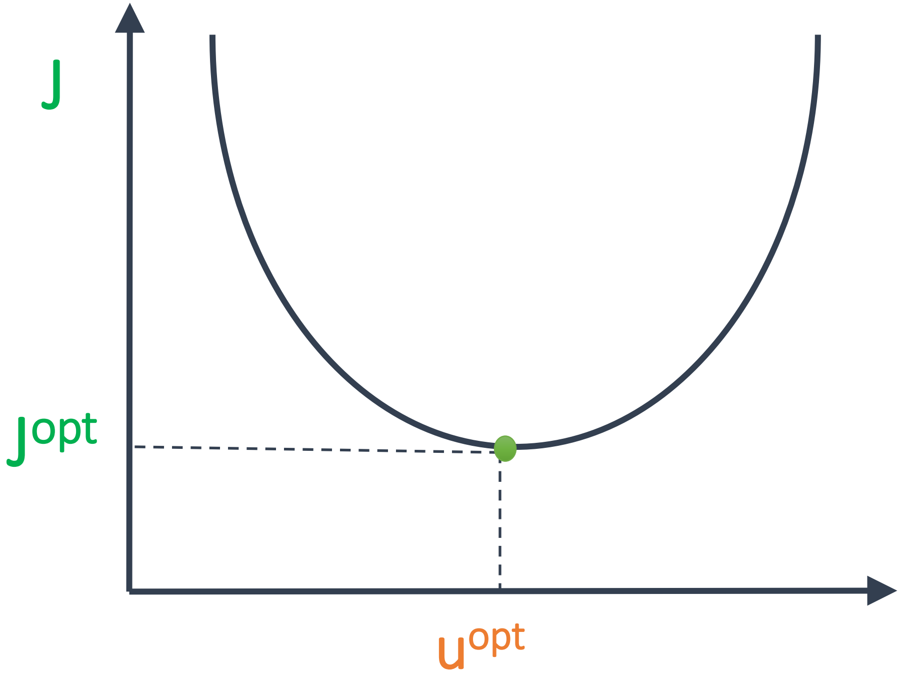
s.t. = subject to

y = CV = controlled variable

NTNU

The less obvious case: Unconstrained optimum

- u = unconstrained MV
- What to control? y=CV=?



Example: Optimal operation of runner

- Cost to be minimized, J=T
- One degree of freedom (u=power)
- What should we control?

NTNU

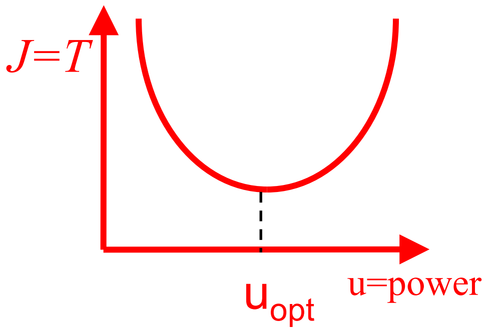
1. Optimal operation of Sprinter

- 100m. J=T
- Active constraint control:
  - Maximum speed ("no thinking required")
  - CV = power (at max)

NTNU

2. Optimal operation of Marathon runner

- 40 km. J=T
- What should we control? CV=?
- Unconstrained optimum

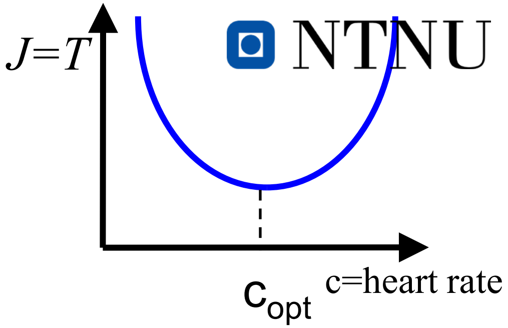


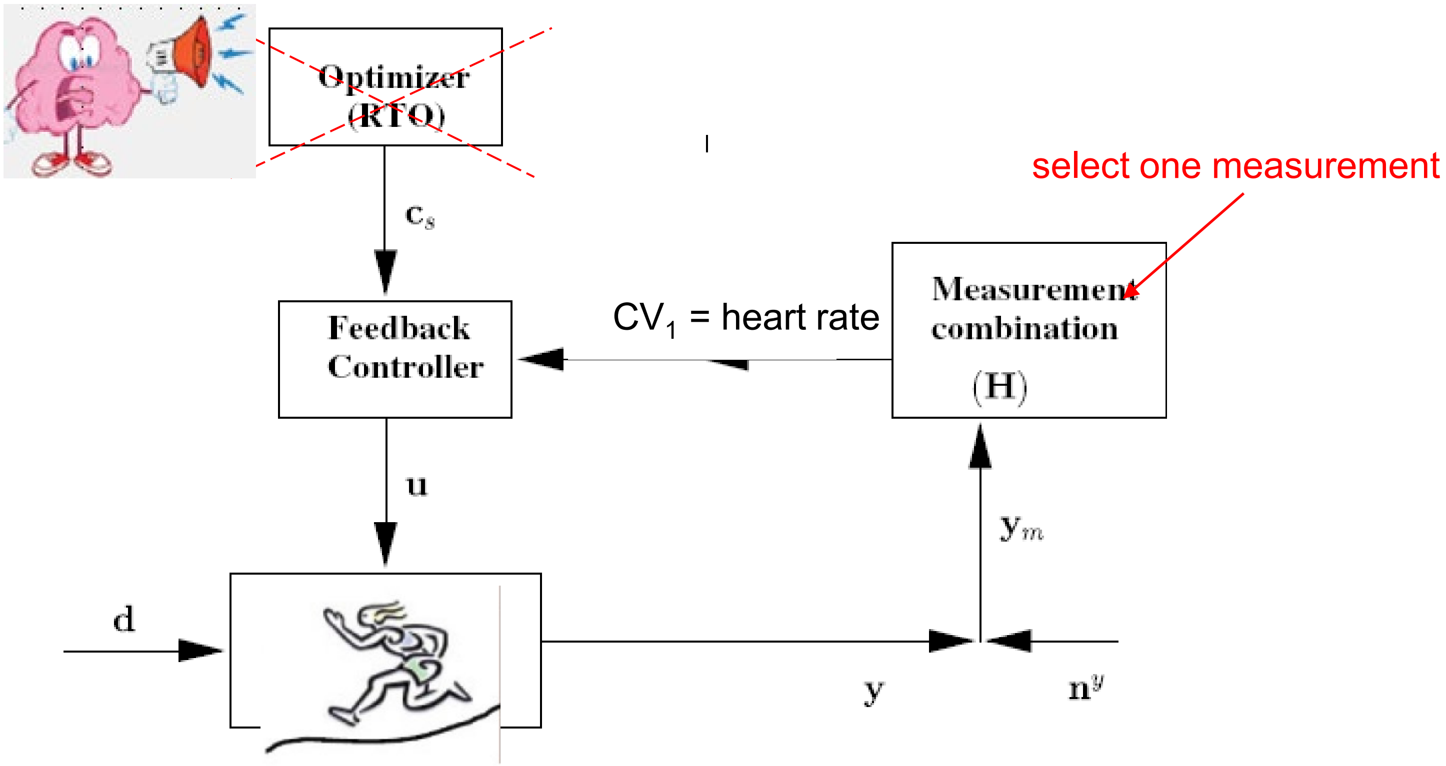
NTNU

Marathon runner (40 km)

- Any self-optimizing variable (to control at constant setpoint)?
  - c_1 = distance to leader of race
  - c_2 = speed
  - **c_3 = heart rate**
  - c_4 = level of lactate in muscles

Conclusion Marathon runner





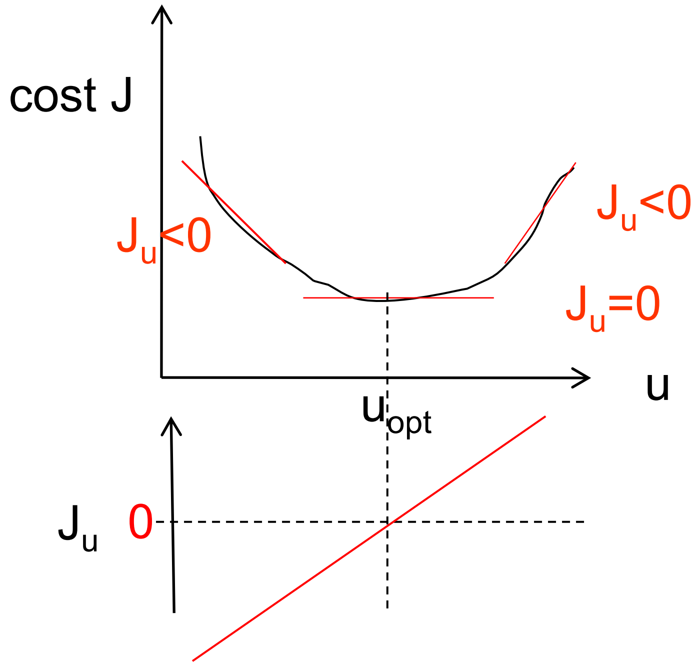
- CV = heart rate is good "self-optimizing" variable
- Simple and robust implementation
- Disturbances are indirectly handled by keeping a constant heart rate
- May have infrequent adjustment of setpoint (c_s)

NTNU

The ideal "self-optimizing" variable is the gradient, J_u

$$c = \partial J/\partial u = J_u$$

- Keep gradient at zero for all disturbances ($c = J_u = 0$)

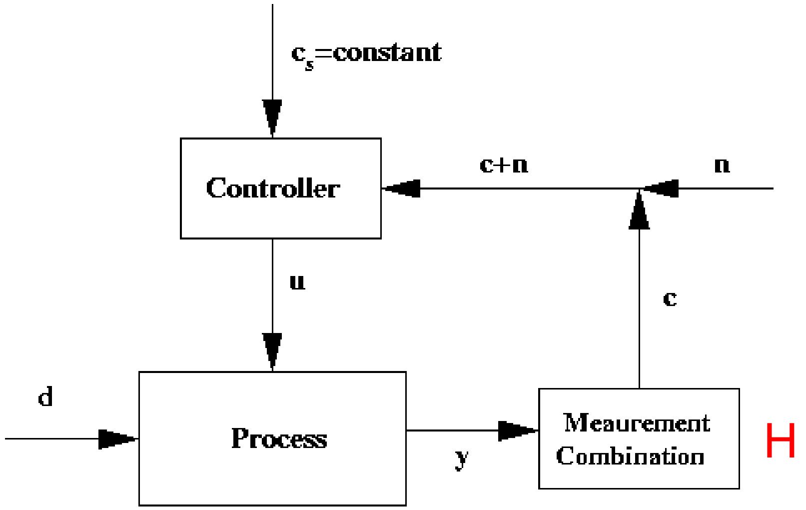


Problem: Usually no measurement of gradient

NTNU

Ideal: $c = J_u$

In practise, use available measurements: c = H y. Task: Select H!



- Single measurements:

$$\mathbf{c} = \mathbf{H}\mathbf{y} \qquad \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

- Combinations of measurements:

$$\mathbf{c} = \mathbf{H}\mathbf{y} \qquad \mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \end{bmatrix}$$

- Combinations of measurements, $c = Hy$

Nullspace method for H (Alstad):

$HF=0$ where $F = dy_{opt}/dd$

Proof: $y_{opt} = F\,d$

$c_{opt} = H\,y_{opt} = HF\,d$

- Proof. Appendix B in: Jäschke and Skogestad, "NCO tracking and self-optimizing control in the context of real-time optimization", *Journal of Process Control*, 1407-1416 (2011)

NTNU

Example. Nullspace Method for Marathon runner

u = power, d = slope [degrees]

y_1 = hr [beat/min], y_2 = v [m/s]

c = Hy, H = [h_1 h_2]]

F = dy_{opt}/dd = [0.25 -0.2]’

HF = 0 -> $h_1 f_1 + h_2 f_2 = 0.25\, h_1 – 0.2\, h_2 = 0$

Choose $h_1 = 1$ -> $h_2 = 0.25/0.2 = 1.25$

Conclusion: **c = hr + 1.25 v**

Control **c = constant** -> hr increases when v decreases (OK uphill!)

NTNU

Exact local method for H

“=0” in nullspace method (no noise)

$$\min_H \left\| J_{uu}^{1/2} (HG^y)^{-1} H [FW_d \quad W_{n^y}] \right\|_2$$

“Minimize” in Maximum gain rule
(maximize S_1 G $J_{uu}^{-1/2}$, G=HG^y)

“Scaling” S_1

Analytical solution:

$H = G^{yT}(YY^T)^{-1}$ where $Y = [FW_d \quad W_{n^y}]$

Advantages compared to nullspace method:

- Can have any number of measurements y
- Includes measurement noise

Step 4: Inventory control and TPM (later!)

Step 5: Design of regulatory control layer

Usually single-loop PID controllers

Choice of CVs (CV2):

- CV2 = «drifting variables»
  - Levels, pressures
  - Some temperatures
- CV2 may also include economic variables (CV1) that need to be controlled on a fast time scale
  - Hard constraints

Choice of MVs and pairings (MV-CV):

- **Main rule: "Pair close**". Want:
  - Large gain
  - Small delay
  - Small time constant
- **Avoid pairing on negative steady-state RGA-elements**
  - It's possible, but then you must be sure that the loops are always working (no manual contriol or MV-saturation)
- **Generally: Avoid MVs that may saturate in regulatory layer**
  - Otherwise, will need logic for re-pairing (MV-CV switching)

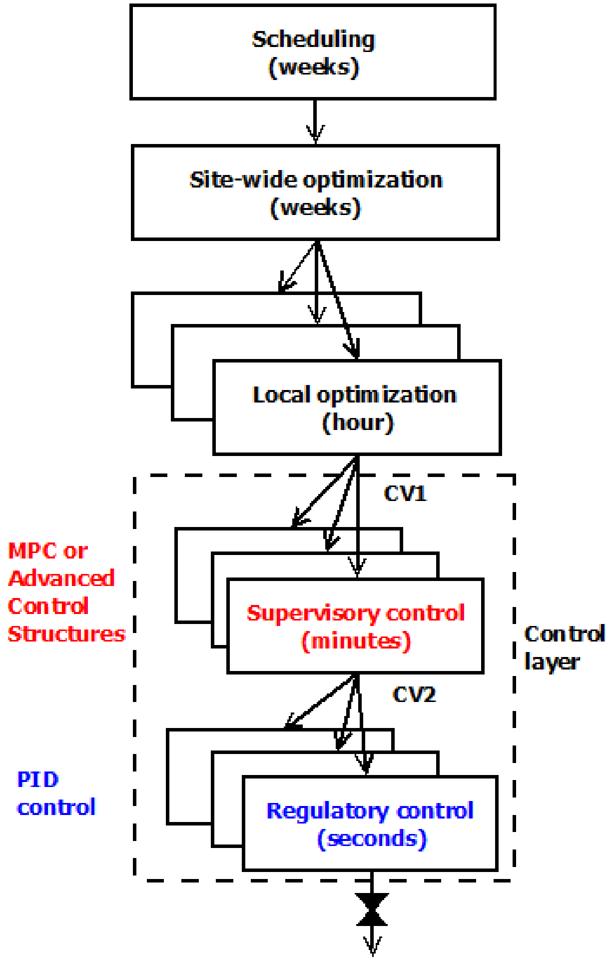
- May include cascade loops (flow control!) and some feedforward, decoupling, linearization

•

Step 6: Design of Supervisory layer

Alternative implementations:

1. Model predictive control (MPC)
2. Advanced regulatorty control (ARC)
   - PID, selectors, etc.



Academia: (E)MPC

- MPC
  - General approach, but we need a dynamic model
    - MPC is usually based on experimental model
    - and implemented after some time of operation
  - Not all problems are easily formulated using MPC

Alternative simpler solutions to MPC

- Would like: Feedback solutions that can be implemented without a detailed models

- Machine learning?
  - Requires a lot of data
  - Can only be implemented after the process has been in operation

- But we have "advanced regulatory control“ (ARC) based on simple control elements
  - Goal: Optimal operation using conventional advanced control
  - PID, feedforward, decouplers, selectors, split range control etc.
  - Extensively used by industry
  - Problem for engineers: Lack of design methods
    - Has been around since 1940’s
    - But almost completely neglected by academic researchers
  - Main fundamental limitation: Based on single-loop (need to choose pairing)

How design ARC system based on simple elements?

- Main topic of this workshop

Advanced regulatory control (ARC) = Classical APC = Advanced PID contol

- Industrial literature (e.g., Shinskey).
  Many nice ideas. But not systematic. Difficult to understand reasoning

- Academia: Very little work so far

APC = Advanced process control

Step 7: Do we really need RTO?

- Often not!
- We can usually measure the constraints
- From this we can identify the **active constraints**
  - Example: Assume it's optimal with max. reactor temperature
  - No need for complex model with energy balance to find the optimal cooling
  - Just use a PI-controller
    - CV = reactor temperature (with setpoint=max)
    - MV = cooling
- And for the remaining unconstrained variables
  - Look for good variables to control (where optimal setpoint changes little)
  - «self-optimizing» variables

RTO = real-time optimization

Summary: Systematic procedure for economic process control

Start "top-down" with economics (steady state):

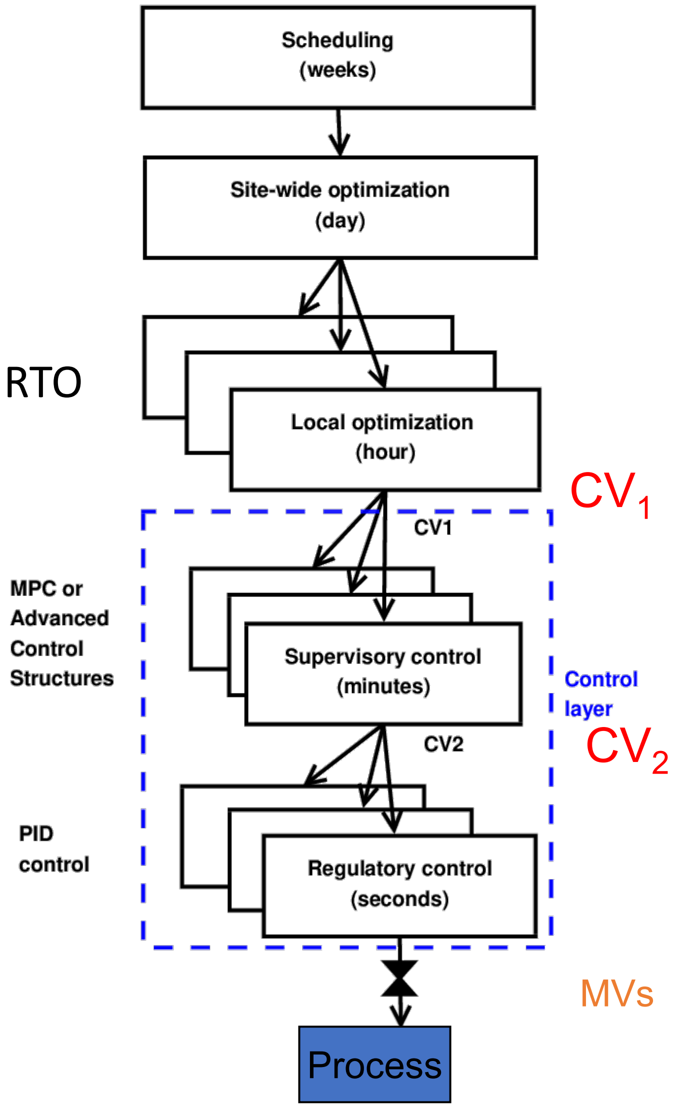
- Step 1: Define operational objectives (J) and constraints
- Step 2: Optimize steady-state operation
- Step 3: Decide what to control (CVs)
  - Step 3A: Identify active constraints = primary CV1.
  - Step 3B: Remaining unconstrained DOFs: Self-optimizing CV1 (find H)
- Step 4: Where do we set the throughput? TPM location

Then bottom-up design of control system(dynamics):

- Step 5: Regulatory control
  - Control variables to stop "drift" (sensitive temperatures, pressures, ...)
  - Inventory control radiating around TPM

Finally: Make link between "top-down" and "bottom up"

- Step 6: "Advanced/supervisory control"
  - Control economic CVs: Active constraints and self-optimizing variables
  - Look after variables in regulatory layer below (e.g., avoid saturation)
- **Step 7: Real-time optimization** (Do we need it?)



S. Skogestad, ``Control structure design for complete chemical plants'', *Computers and Chemical Engineering*, **28** (1-2), 219-234 (2004).