

Part 6

Tranformed inputs

MPC

RTO,

Summary , Challenges,



More on nonlinear feedforward, decoupling and linearization

- **Tranformed inputs:** Extremely simple and effective way of achieving feedforward, decoupling and linearization

Feedforward control

- Feedforward control relies on model
 - as opposed to feedback which relies mostly on data
- Feedback control: Linear model is often OK
- Feedforward control: Much less likely that linear model is OK
 - Process changes and disturbances
 - This presentation: Use nonlinear static model

Tuning rules for feedforward control from measurable disturbances combined with PID control: a review

J. L. Guzmán ^a and T. Hägglund ^b

^a Department of Informatics, Universidad de Almería, ceiA3, CIESOL, Almería, Spain ^b Department of Automatic Control, Lund University, Lund, Sweden

5.1. Classical solutions

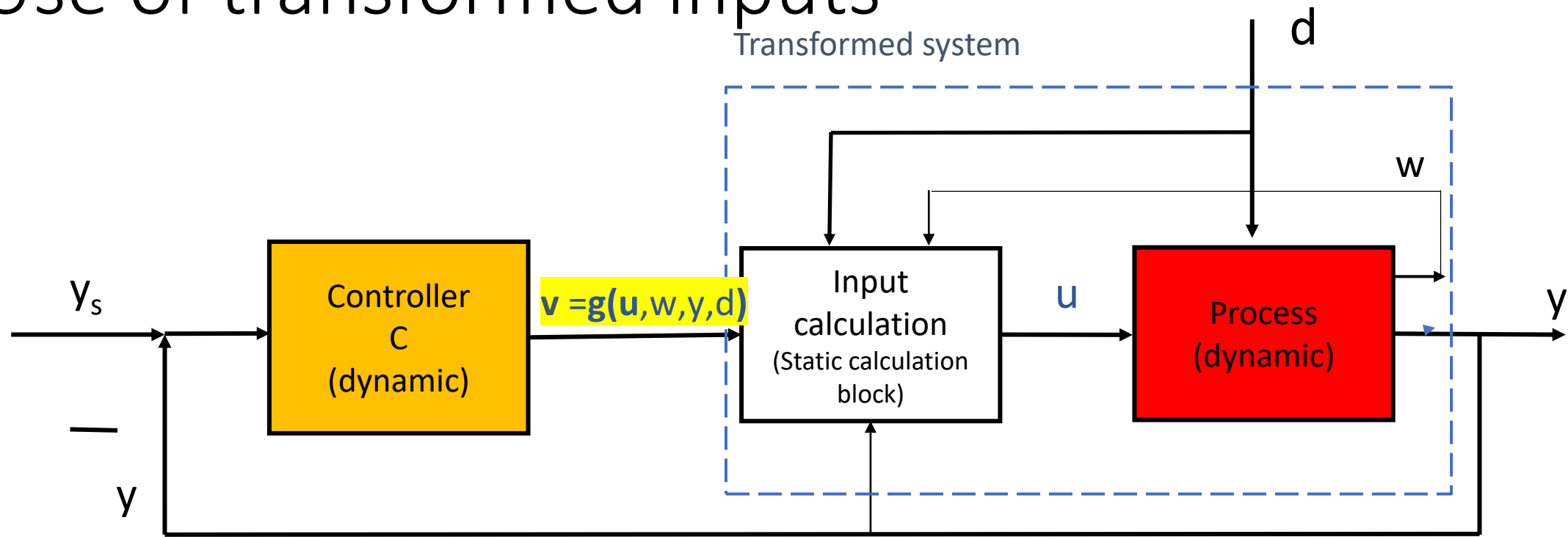
5.1.1. Static feedforward compensator

A static feedforward compensator is a solution widely used in industry, which is given by:

$$C_{ff} = \frac{K_d}{K_v} \quad (15)$$

The reason to use this simple solution is that drastic improvements can be obtained compared with pure feedback control by using just this simple compensator. Moreover, it can be used to account for any non-realizable problem. However, the resulting performance is also limited because of its simplicity.

Use of transformed inputs



Transformed input $v = g(u, w, y, d)$

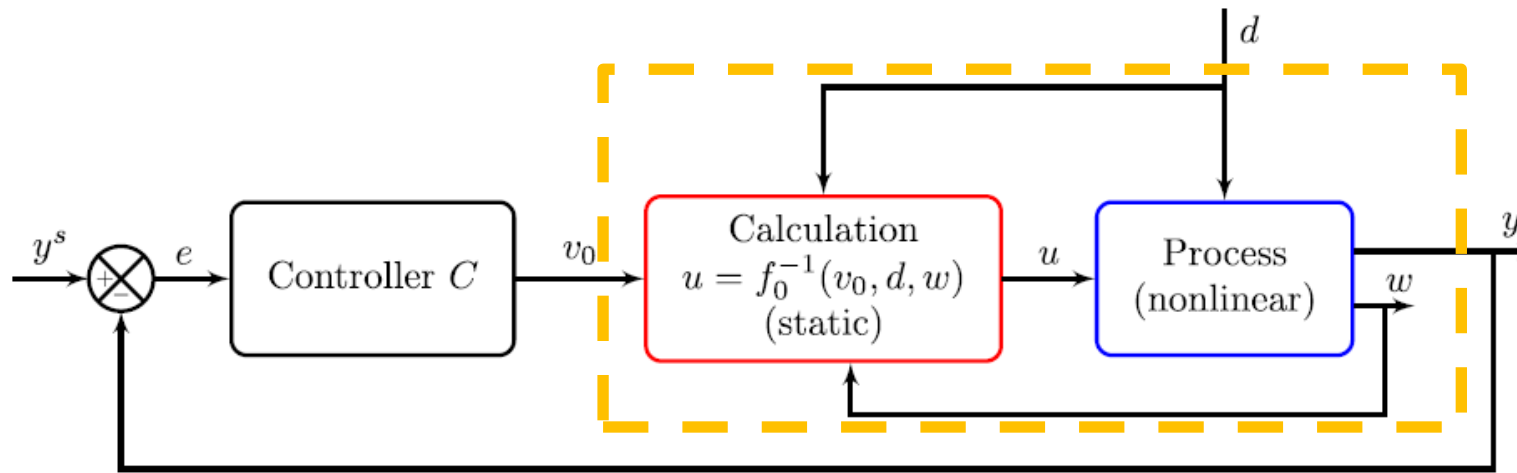
- Replaces the physical input u for control of y .
- Aim: **Transformed system** is easier to control
 - May include:
 - Decoupling
 - Linearization
 - Feedforward

Examples

- $v = u/d$ (ratio control for feedforward)
- $v = u_1/u_2$ (ratio control for decoupling)
- $v = u_1 + u_2$ (from mass balance)
- $v = w(u) = F \rightarrow$ Cascade flow control

General approach based on static model

- Static model: $y = f_0(u, d, w)$
- Invert by solving with respect to u for given $y = v_0$: $u = f_0^{-1}(v_0, d, w)$ (14)
- $v_0 =$ Transformed input



Transformed system:

$$y = v_0$$

(decoupled, linear and independent of d)

Fig. 29. Feedforward, decoupling and linearization (red calculation block) using transformed inputs $v_0 = f_0(u, d, w)$ based on static model $y = f_0(u, d, w)$. In 1 model error, the transformed system from v_0 to y (as seen from the controller C) becomes $y = Iv_0$ at steady state.

The method in (14) and Fig. 29 was published only recently (Skogestad et al., 2023), but it is not new. Industry frequently makes use of nonlinear static model-based “calculation blocks”, “function blocks”, or “ratio elements” to provide feedforward action, decoupling or linearization (adaptive gain), and Shinskey (1981) and Wade (2004) provide examples. In particular, Wade (2004) (pages 217, 225 and 288) presents similar ideas. However, the generality of the method is new.

The method is based on a static model, so it may be necessary to “fine tune” the implementation by adding dynamic compensation (typically lead-lag with delay) on the measured variables (d or w) to improve the dynamic response. Alternatively, there is also a dynamic variant of the method based on using a first-order model, which turns out to be a special case of the nonlinear control method called “feedback linearization” (Skogestad et al., 2023).

Example: Blending process

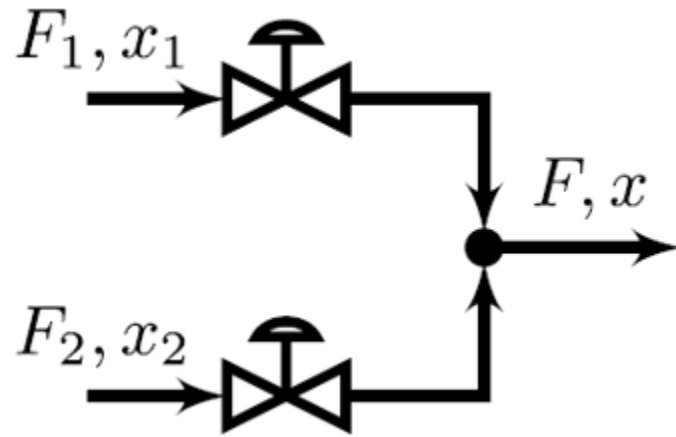


Fig. 28. Flowsheet of in-line blending process (mixer) where F is the flowrate [kg/s] and x is the mass fraction of component A [kg A/kg].

$$u = \begin{pmatrix} F_1 \\ F_2 \end{pmatrix}; \quad y = \begin{pmatrix} x \\ F \end{pmatrix}; \quad d = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

Static material balances, $y = f_0(u, d, w)$:

$$x = \underbrace{(F_1 x_1 + F_2 x_2)}_{v_{0,1}} / (F_1 + F_2)$$

$$F = \underbrace{F_1 + F_2}_{v_{0,2}}$$

Solve for $u = f_0^{-1}(v_0, d, w)$:

$$\begin{aligned} F_1 &= \frac{v_{0,2}(v_{0,1} - x_2)}{x_1 - x_2} \\ F_2 &= \frac{v_{0,2}(x_1 - v_{0,1})}{x_1 - x_2} \end{aligned} \quad (17)$$

This can be implemented directly and give $y=v_0$.

Alternative simple implementation. Note that

$$F_1 = F_2 \frac{v_{0,1} - x_2}{x_1 - v_{0,1}} \quad (18)$$

$$F_2 = v_{0,2} - F_1$$

$v_{01}, v_{02} =$ Transformed inputs

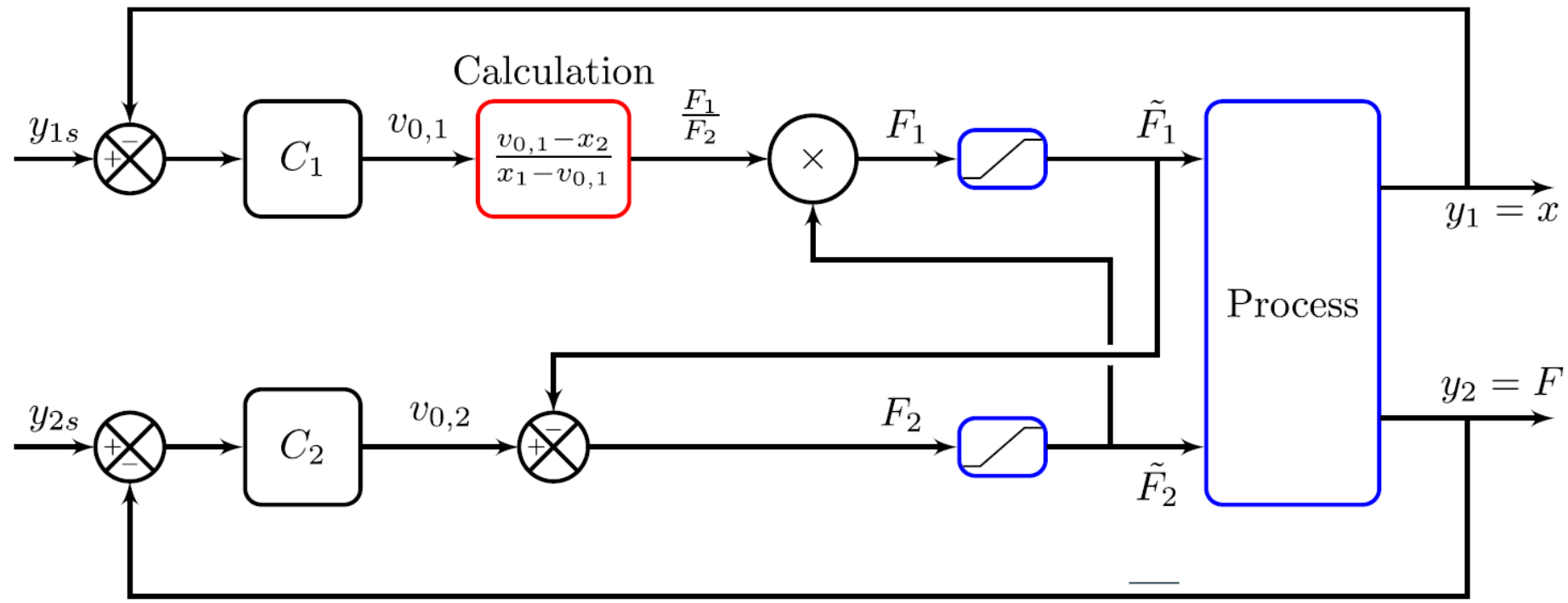


Fig. 30. Simple control structure which provides decoupling, feedforward control and linearization for the mixing process (blending system) in Fig. 28.

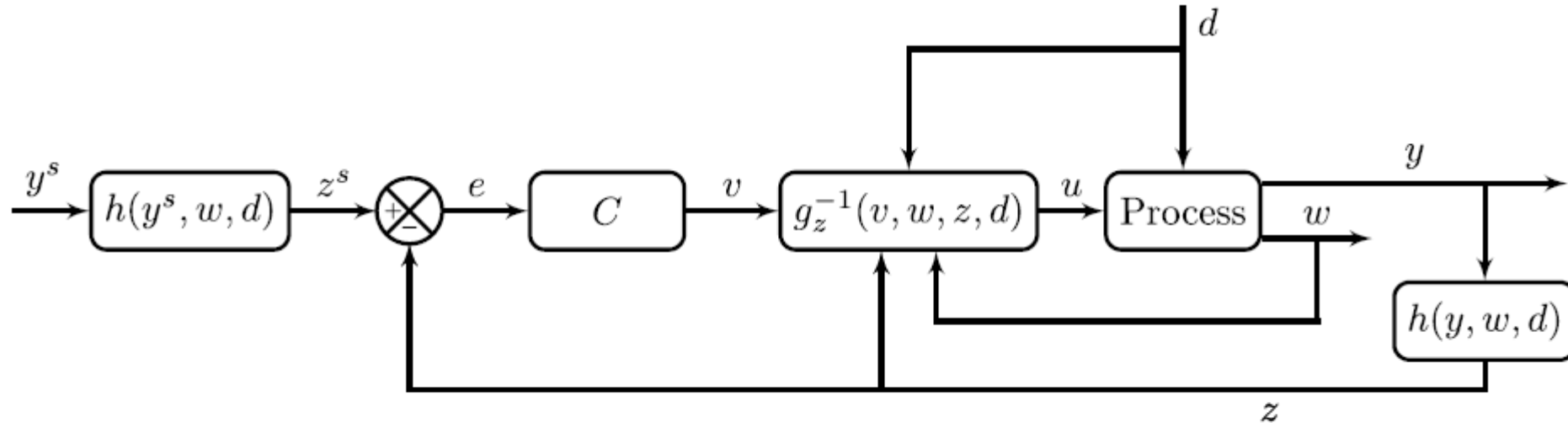
The output from the feedback controller C_1 is the ideal transformed input $v_{0,1}$. From this and measured disturbances (inlet compositions x_1 and x_2), the feedforward calculation element (red) uses (18a) to compute F_1/F_2 . The decoupling is given by one multiplication element and one subtraction element. To work also in the case of input saturation, it uses the actual measured flowrates (\tilde{F}_1, \tilde{F}_2) a The resulting transformed system as seen from the feedback controllers (C_1, C_2) is simply $\underbrace{y_1 = v_{0,1} \text{ and } y_2 = v_{0,2}}_{(11)}$ (with no model error). Note that we need two inner flow controllers (for F_1 and F_2) which are not shown in the figure.

Besides being simple to understand and implement, the advantage with the implementation in (18) and Fig. 30, compared to an inversion using (17), is that it provides partial decoupling and disturbance rejection also when F_1 or F_2 saturate. That is, when F_2 saturates, we will maintain control of $y_1 = x$ but lose control of $y_2 = F$. Similarly, when F_1 saturates, we will maintain control of y_2 but lose control of y_1 .

Based on (11), Seborg et al. (2016) (page 343) write about the choice of transformed manipulated variables in (10): “This means that the controlled variables are identical to the manipulated variables! Thus, the gain matrix is the identity matrix, and the two control loops do not interact at all. This situation is fortuitous, and also unusual, because it is seldom possible to choose manipulated variables that are, in fact, the controlled variables”.

As shown next, the statement that this is “fortuitous, and also unusual” is not correct. If we assume that the disturbances are measured, then it is always possible to derive ideal transformed manipulated variables (inputs) v_0 which are equal to the controlled variables y , simply by choosing v_0 as the right-hand side of the steady-state model equations (Skogestad et al., 2023).

Also: Transformed outputs z



(a) General implementation of transformed output z

- No fundamental advantage, but can simplify input transformation
- For example, $y=T$, $z=H$ (enthalpy)

More on transformed inputs

Journal of Process Control 122 (2023) 113–133



Contents lists available at [ScienceDirect](#)

Journal of Process Control

journal homepage: www.elsevier.com/locate/jprocont



Review

Transformed inputs for linearization, decoupling and feedforward control

Sigurd Skogestad^{a,*}, Cristina Zotică^a, Nicholas Alsop^b

^a Department of Chemical Engineering, Norwegian University of Science and Technology (NTNU), 7491, Trondheim, Norway

^b Senior Process Control Engineer, Borealis AB, Stenungsund, Sweden



What about MPC?

- First industrial use in the 1970s
- Became common in the refining and petrochemical industry in the 1980s
- In the 1990s a bright future was predicted for MPC in all industries (chemical, thermal power, ...)
- 30 years later: We know that this did not happen
- Why? First, the performance benefits of MPC compared to ARC are often minor (if any)
- In addition, MPC has some limitations
 1. Expensive to obtain model
 2. Cannot easily handle integral action, cascade and ratio control
 3. Normally, cannot be used at startup (so need ARC anyway)
 4. Often difficult to tune
 5. Slow, time consuming for large problems
 6. Robustness (e.g., gain margin) handled indirectly
- Advantages of MPC
 1. Interactive multivariable dynamic processes
 2. Coordinate feedforward and feedback
 3. Coordinate use of many inputs
 4. Make use of information about future disturbances, setpoints and prices (predictive capabilities of MPC)
 5. Nonlinear dynamic processes (nonlinear MPC)
- What about constraints
 - Not really a major advantage with MPC; can be handled well also with ARC

7.6.7. Summary of MPC shortcomings

Some shortcomings of MPC are listed below, in the expected order of importance as seen from the user's point of view:

1. MPC requires a “full” dynamic model involving all variables to be used by the controller. Obtaining and maintaining such a model is costly.
2. MPC can handle only indirectly and with significant effort from the control engineer (designer), the three main inventions of process control; namely integral control, ratio control and cascade control (see above).
3. Since a dynamic model is usually not available at the startup of a new process plant, we need initially a simpler control system, typically based on advanced regulatory control elements. MPC will then only be considered if the performance of this initial control system is not satisfactory.
4. It is often difficult to tune MPC (e.g., by choosing weights or sometimes adjusting the model) to give the engineer the desired response. In particular, since the control of all variables is optimized simultaneously, it may be difficult to obtain a solution that combines fast and slow control in the desired way. For example, it may be difficult to tune MPC to have fast feedforward control for disturbances because it may affect negatively the robustness of the feedback part (Pawlowski et al., 2012).
5. The solution of the online optimization problem is complex and time-consuming for large problems.
6. Robustness to model uncertainty is handled in an ad hoc manner, for example, through the use of the input weight R . On the other hand, with the SIMC PID rules, there is a direct relationship between the tuning parameter τ_c and robustness margins, such as the gain, phase and delay margin Grimholt and Skogestad (2012), e.g., see (C.13) for the gain margin.

7.6.8. Summary of MPC advantages

The above limitations of MPC, for example, with respect to integral action, cascade control and ratio control, do not imply that MPC will not be an effective solution in many cases. On the contrary, MPC should definitely be in the toolbox of the control engineer. First, standard ratio and cascade control elements can be put into the fast regulatory layer and the setpoints to these elements become the MVs for MPC. More importantly, MPC is usually better (both in terms of performance and simplicity) than advanced regulatory control (ARC) for:

1. Multivariable processes with (strong) dynamic interactions.
2. Pure feedforward control and coordination of feedforward and feedback control.
3. Cases where we want to dynamically coordinate the use of many inputs (MVs) to control one CV.
4. Cases where future information is available, for example, about future disturbances, setpoint changes, constraints or prices.
5. Nonlinear dynamic processes (nonlinear MPC).

The handling of constraints is often claimed to be a special advantage of MPC, but it can in most cases also be handled well by ARC (using selectors, split-range control solutions, anti-windup, etc.). Actually, for the Tennessee Eastman Challenge Process, Ricker (1996) found that ARC (using decentralized PID control) was better than MPC. Ricker (1996) writes in the abstract: “There appears to be little, if any, advantage to the use of NMPC (nonlinear MPC) in this application. In particular, the decentralized strategy does a better job of handling constraints – an area in which NMPC is reputed to excel”. In the discussion section he adds: “The reason is that the TE problem has too many competing goals and special cases to be dealt with in a conventional MPC formulation”.

Real-time optimization

- We have presented effective approach for constraint switching (MV-MV, CV-CV, MV-CV).
- Most important is CV-CV switching
 - CV = constraint or self-optimizing variable (ideal = gradient = J_u)
 - Each CV is paired with one MV
 - MV-CV switching covers (some) cases where MV may saturate and we need to pair with another MV.
- Optimal in many cases, but not in general
- For example, may not be able to cover cases with more than one unconstrained region \Rightarrow More than one self-optimizing variable

Economic real-time optimization(RTO)

General approaches

- I. Separate RTO layer (online steady-state optimization)
- II. Feedback-optimizing control (put optimization into the feedback layer)
 - Alt.1. (Most general): Based on dual decomposition (iterate on Lagrange multipliers λ)
 - Alt.2 (Tighter constraint control): Region-based with reduced gradient,
- III. Data-based approaches (model free)
 - Hill-climbing methods = Extremum-seeking control

Computers and Chemical Engineering 161 (2022) 107723



Contents lists available at [ScienceDirect](#)

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

Review

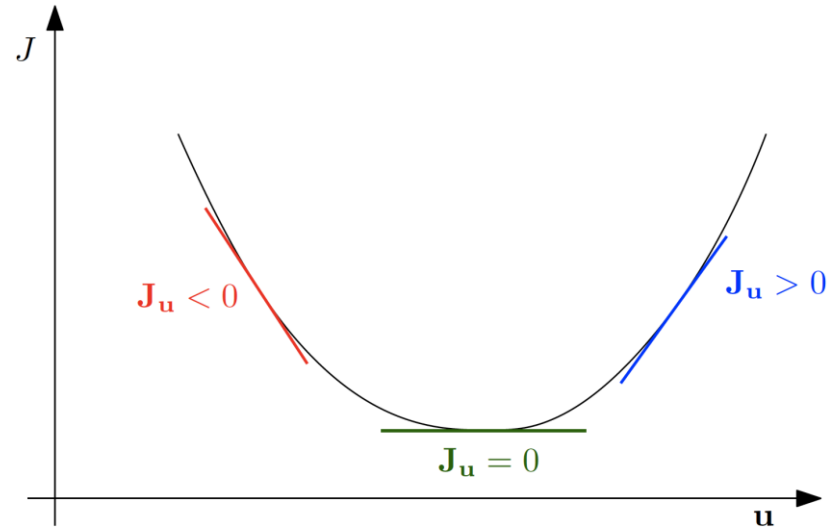
Real-Time optimization as a feedback control problem – A review

Dinesh Krishnamoorthy^{a,b,*}, Sigurd Skogestad^b

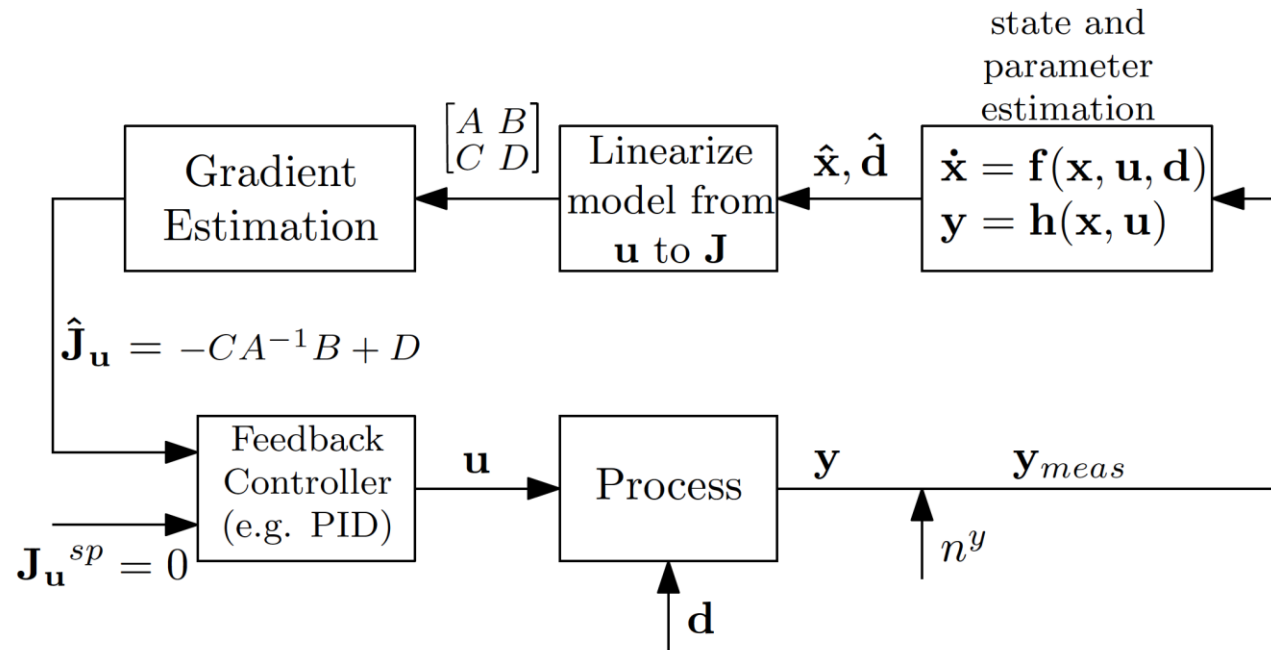
Unconstrained optimization.

Necessary condition of optimality (NCO):

- Gradient of cost function = 0
- $J_u = dJ/du = 0$



Feedback RTO (unconstrained case)



Linearize the dynamic model

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{d}) & \Rightarrow & \dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ J &= \mathbf{g}(\mathbf{x}, \mathbf{u}) & & J = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \end{aligned}$$

$$\begin{aligned} A &= \left. \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right|_{\mathbf{x}=\hat{\mathbf{x}}} & B &= \left. \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \right|_{\mathbf{x}=\hat{\mathbf{x}}} \\ C &= \left. \frac{\partial \mathbf{g}}{\partial \mathbf{x}} \right|_{\mathbf{x}=\hat{\mathbf{x}}} & D &= \left. \frac{\partial \mathbf{g}}{\partial \mathbf{u}} \right|_{\mathbf{x}=\hat{\mathbf{x}}} \end{aligned}$$

Trick, set $\dot{\mathbf{x}} = 0$:

$$J = \underbrace{\left(-\mathbf{C}\mathbf{A}^{-1}\mathbf{B} + \mathbf{D} \right)}_{\hat{\mathbf{J}}_{\mathbf{u}}} \mathbf{u}$$

Including constraints

Constrained optimization problem

$$\begin{aligned} \min_{\mathbf{u}} \quad & f(\mathbf{u}, \mathbf{y}, \mathbf{d}) \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{u}, \mathbf{y}, \mathbf{d}) \leq \mathbf{J} \end{aligned}$$

Solution: Turn into **unconstrained** optimization problem using **Lagrange multipliers**

$$\mathcal{L}(\mathbf{u}, \mathbf{y}, \mathbf{d}, \boldsymbol{\lambda}) = \mathbf{J}(\mathbf{u}, \mathbf{y}, \mathbf{d}) + \boldsymbol{\lambda}^T \mathbf{g}(\mathbf{u}, \mathbf{y})$$

$$\min_{\mathbf{u}, \boldsymbol{\lambda}} \mathcal{L}$$

\mathbf{u} = primal variables = inputs

$\boldsymbol{\lambda} \geq 0$ = dual variables = Lagrange multipliers = shadow prices

Necessary conditions of optimality (KKT-conditions)

$$\nabla_{\mathbf{u}} \mathcal{L} = 0, \quad \boldsymbol{\lambda} \geq 0, \quad \mathbf{g} \cdot \boldsymbol{\lambda} = 0 \quad (\text{complementary condition})$$

Alt.1: Feedback implementation using dual decomposition
(Krishnamoorthy, Dirza, Skogestad)

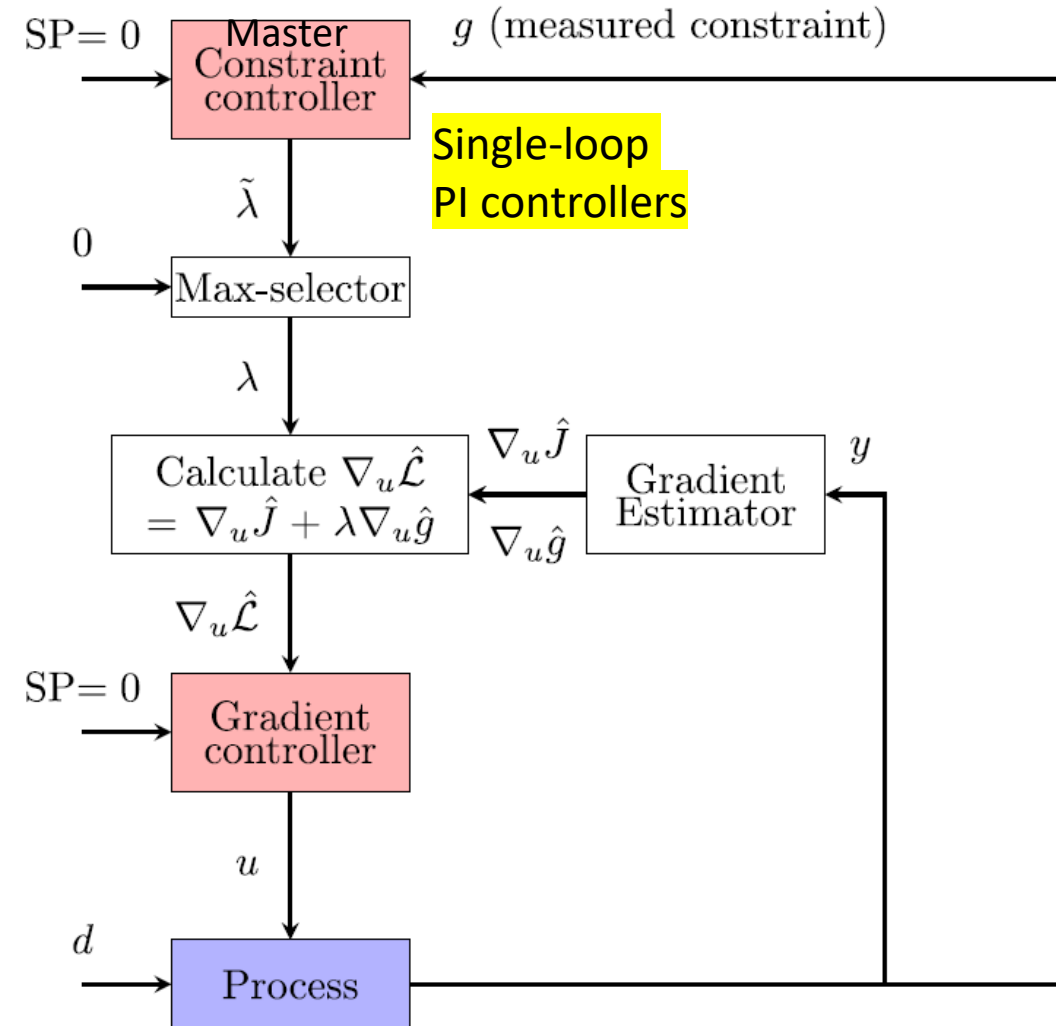
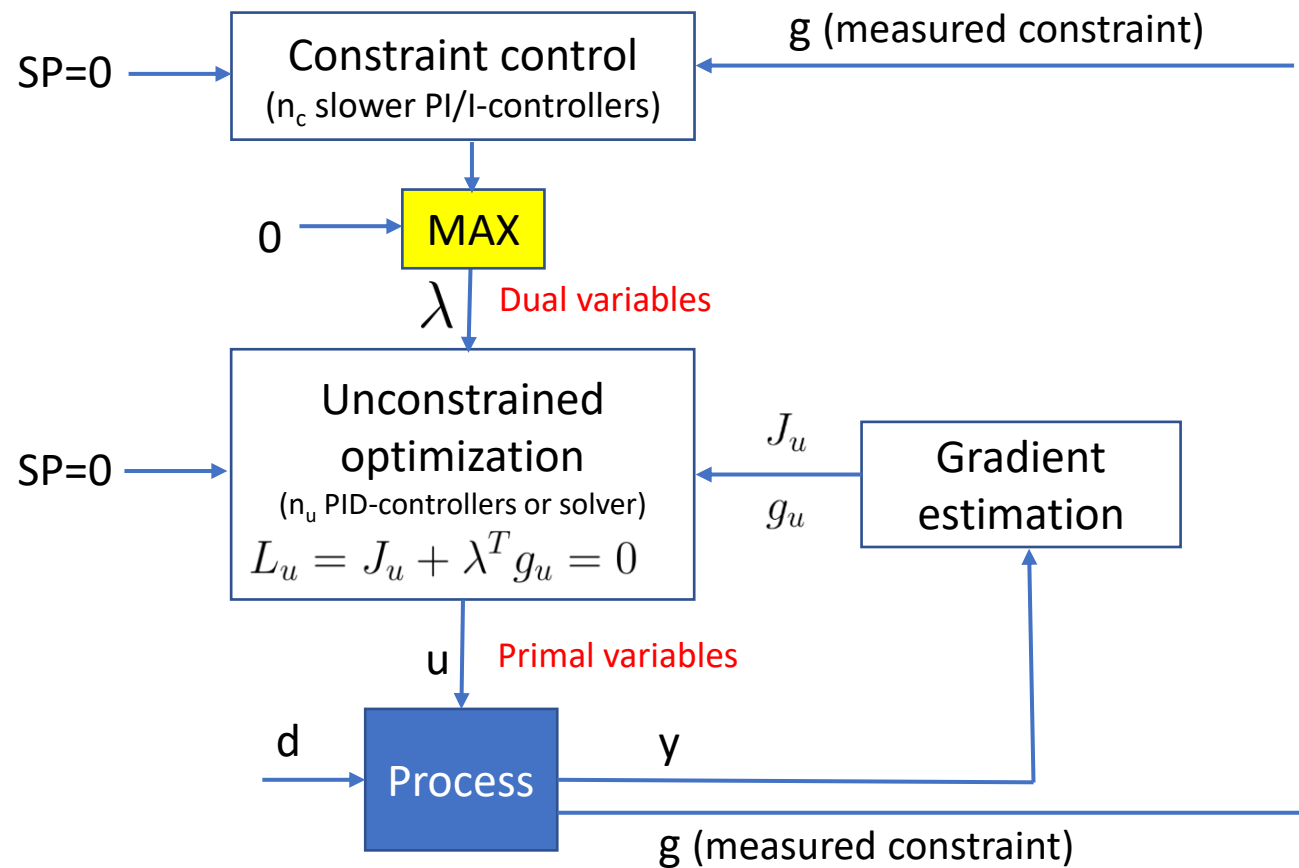


Fig. 40. Dual decomposition of constrained optimization with upper (slow) constraint controller and max-selector on the dual variable λ (Lagrange multiplier).

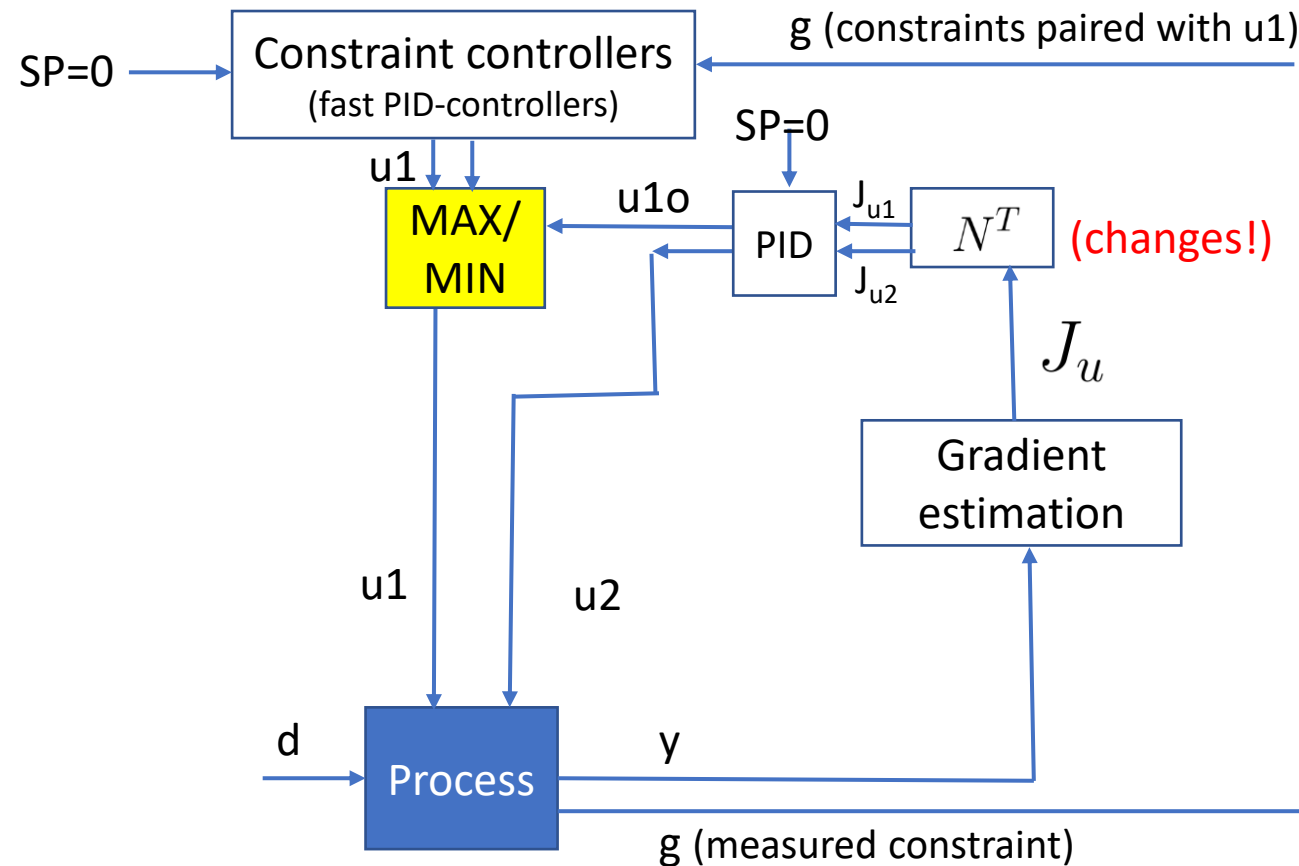
Alternative 1: Feedback-RTO that tracks active constraints by adjusting Lagrange multipliers (= shadow prices = dual variables) λ



«Primal-dual feedback control»

- Makes use of «dual decomposition» of constrained optimization
- Selector on dual variables λ
- Very nice for cases with shared utility g , for example steam plant
 - Local cost: $L_i = J_i + \lambda g_i$
- Problem 1: Constraint control on slow time scale (upper layer)
- Problem 2: Single-loop control in lower layer ($L_u=0$) may not be possible for coupled processes

Alternative 2: Region-based Feedback-RTO with «direct» constraint control



$J_{u1}, J_{u2} = N^T J_u$ reduced gradients
(«self-optimizing variables»)

$$N^T \nabla_{\mathbf{u}} \mathbf{g}_A(\mathbf{u}, \mathbf{d})^T = 0$$

- Selector on primal variables (inputs)
- Similar to selectors in APC
- Limitation: need to pair each constraint with an input u , may not work if many constraints

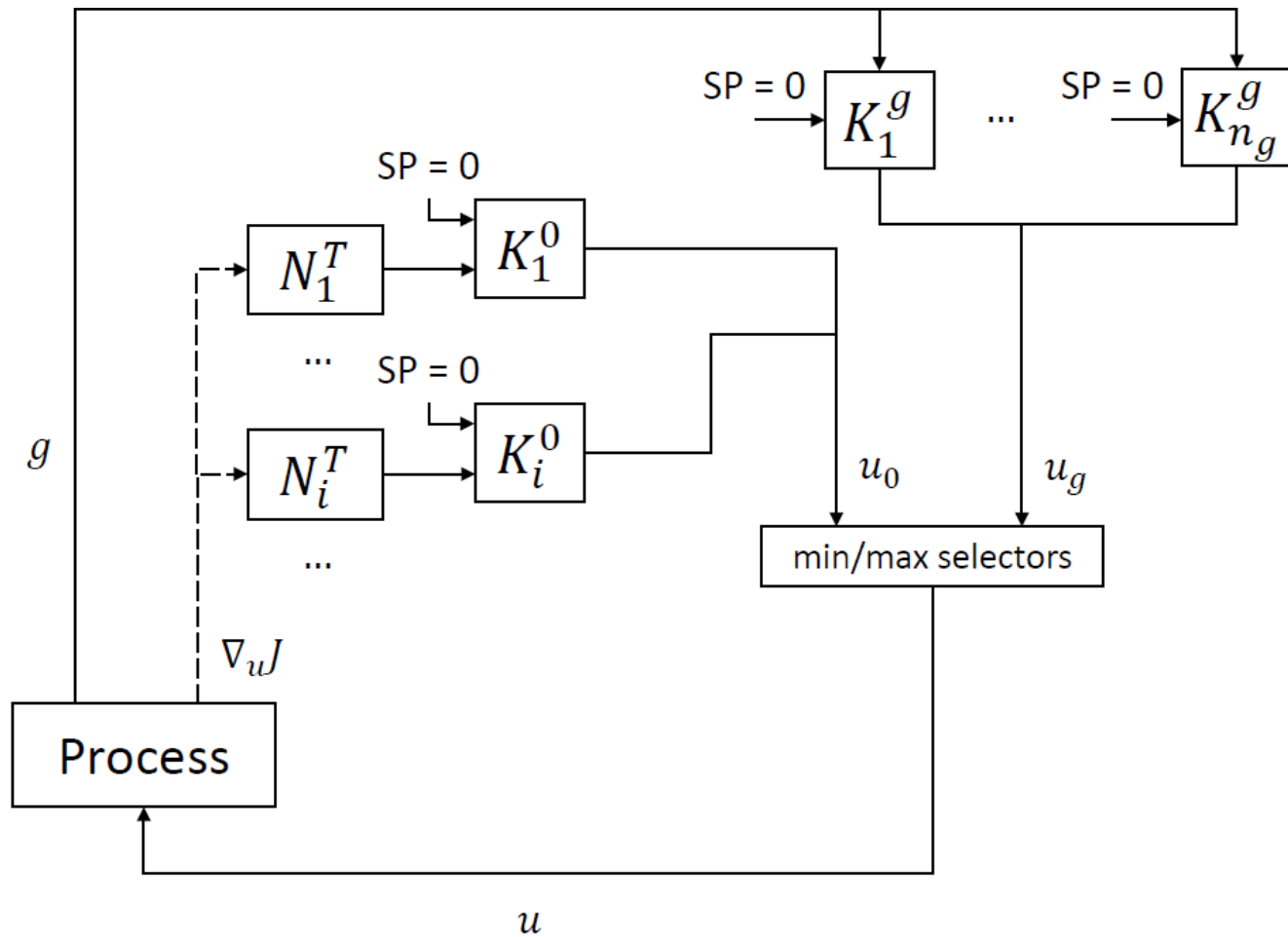


Fig. 1. Region-based control strategy using selectors

Model-free optimization: Extremum Seeking Control (ESC)

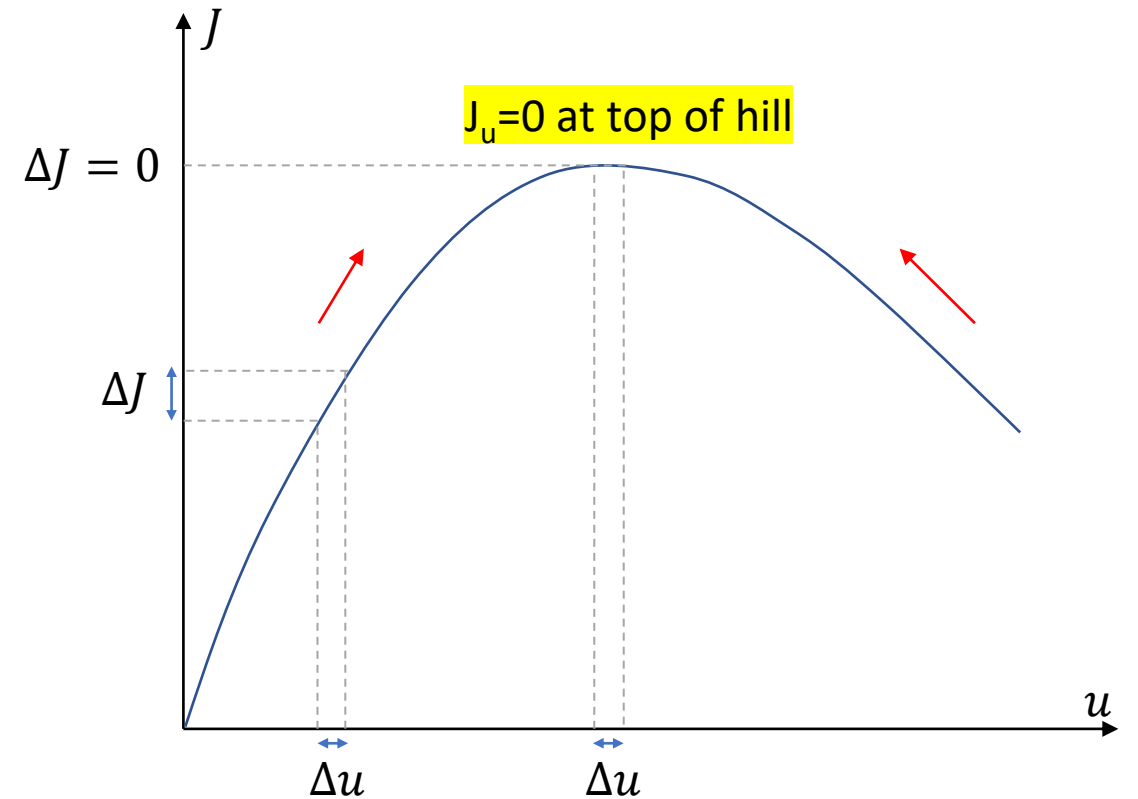
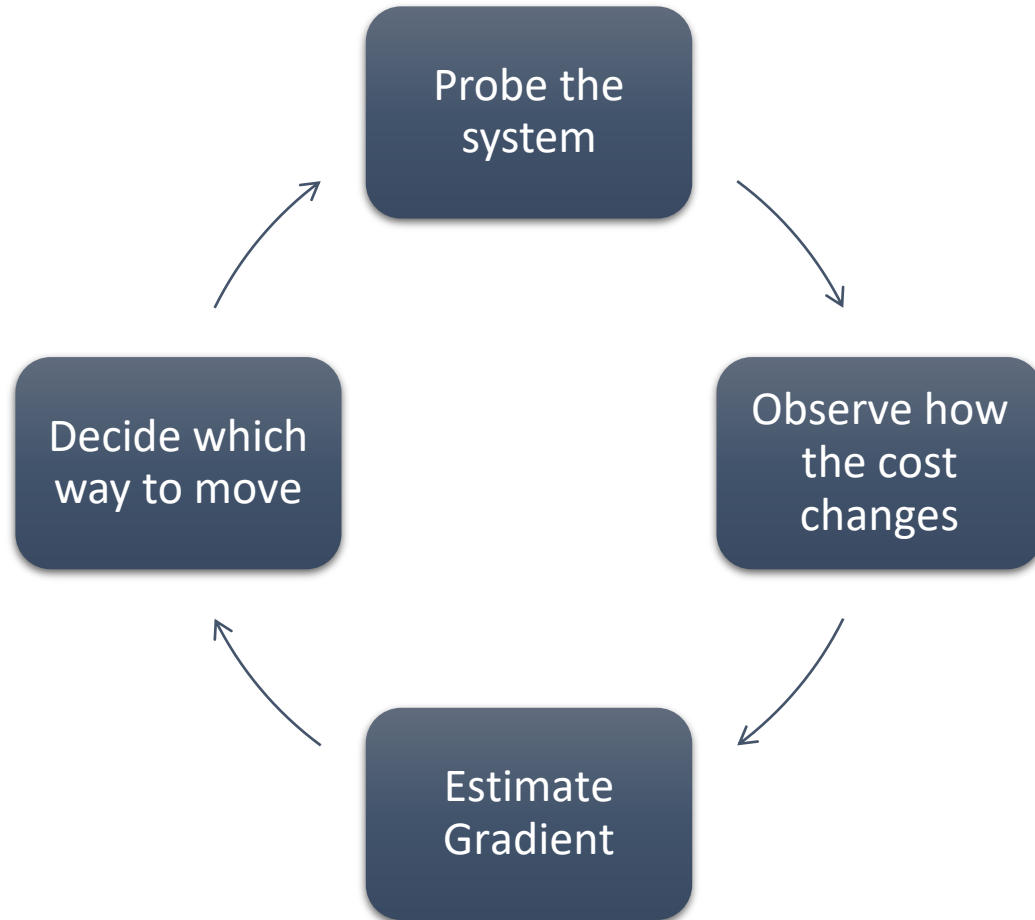
Why?

- Expensive to obtain model
- May be used on top of RTO to correct for model error

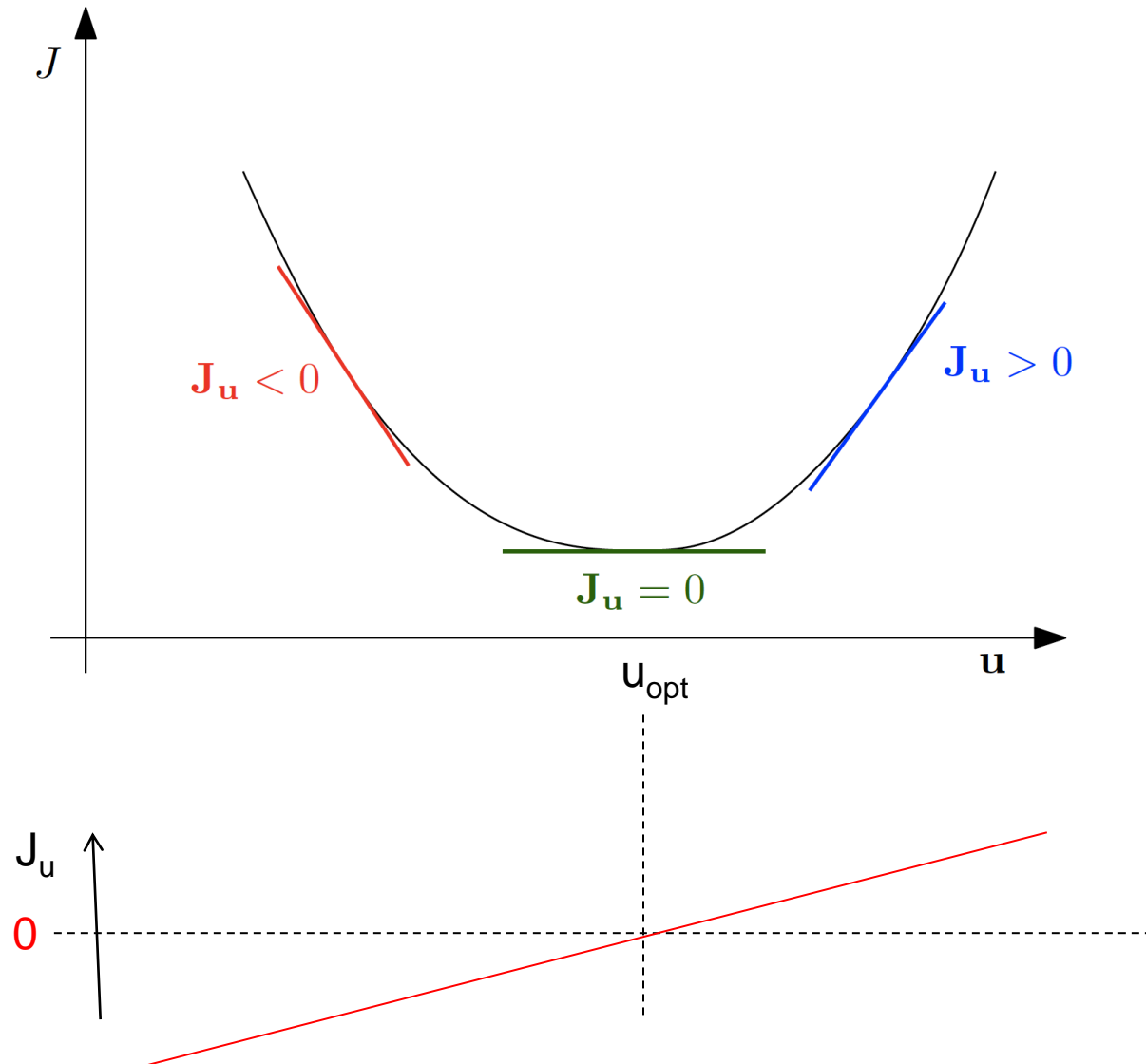
Main problems with model-free optimization:

- Cost function J not measured (need model...)
- Very slow. Typically 100-1000 times slower than process dynamics

Data-based optimization: “Hill-climbing” / “Extremum seeking control”
Drive gradient $J_u = dJ/du$ to zero.

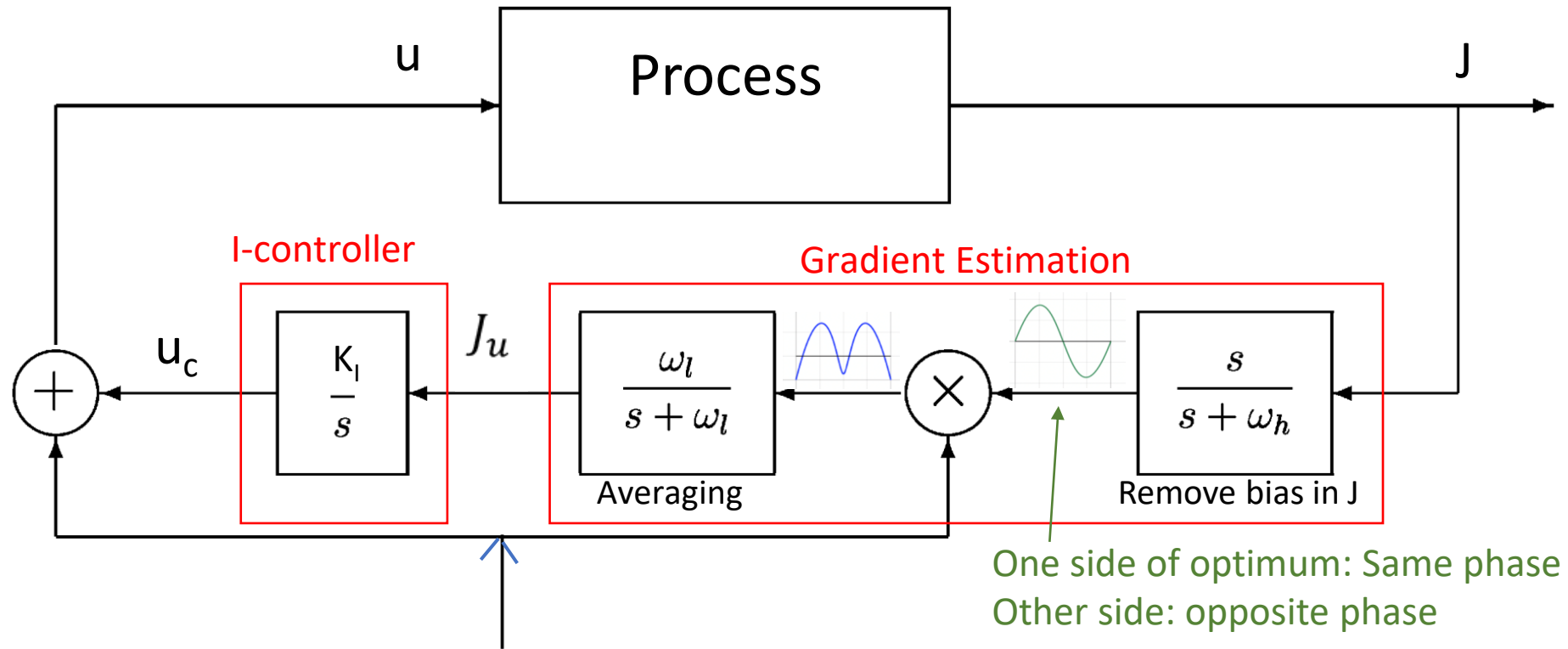


Equivalent: Minimize cost J (go to bottom of valley)

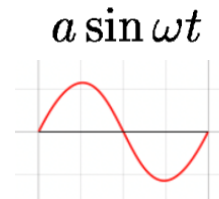


- Optimal setpoint: $J_u = 0$
- If Hessian J_{uu} is constant:
 - J_u as a function of u is a **straight line** with slope J_{uu}
- Nice properties for feedback control of J_u
- No dynamics: Pure I-controller optimal
 - SIMC-rule: $K_I = 1/(J_{uu} \tau_c)$

Classical Extremum seeking control using sinusoids

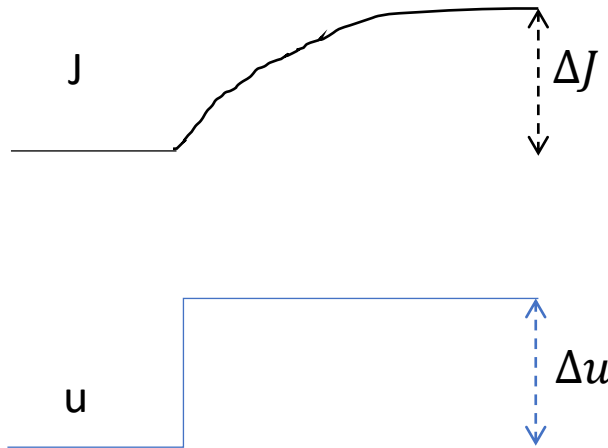


Multiplication trick: Draper & Li (1951)
Theory: Krstic & Wang (Automatica, 2000)



- **Simple to implement (don't need computer)**, but
- Prohibitively **slow convergence** for systems with slow dynamics
- **Typically 100 times slower than the system dynamics !**

More common today: Estimate Steady-state gradient using discrete perturbations (steps)



$$J_u = \frac{\Delta J}{\Delta u}$$

Usually only one input. Simplest: step change in u :

- Hill climbing control (Shinskey, 1967)
- Evolutionary operation (EVOP) (1960's)
- NCO tracking (Francois & Bonvin, 2007)
- "Perturb and observe" = Maximum power point tracking (MPPT) (2010's).

More advanced variants which may also be applied to multivariable systems

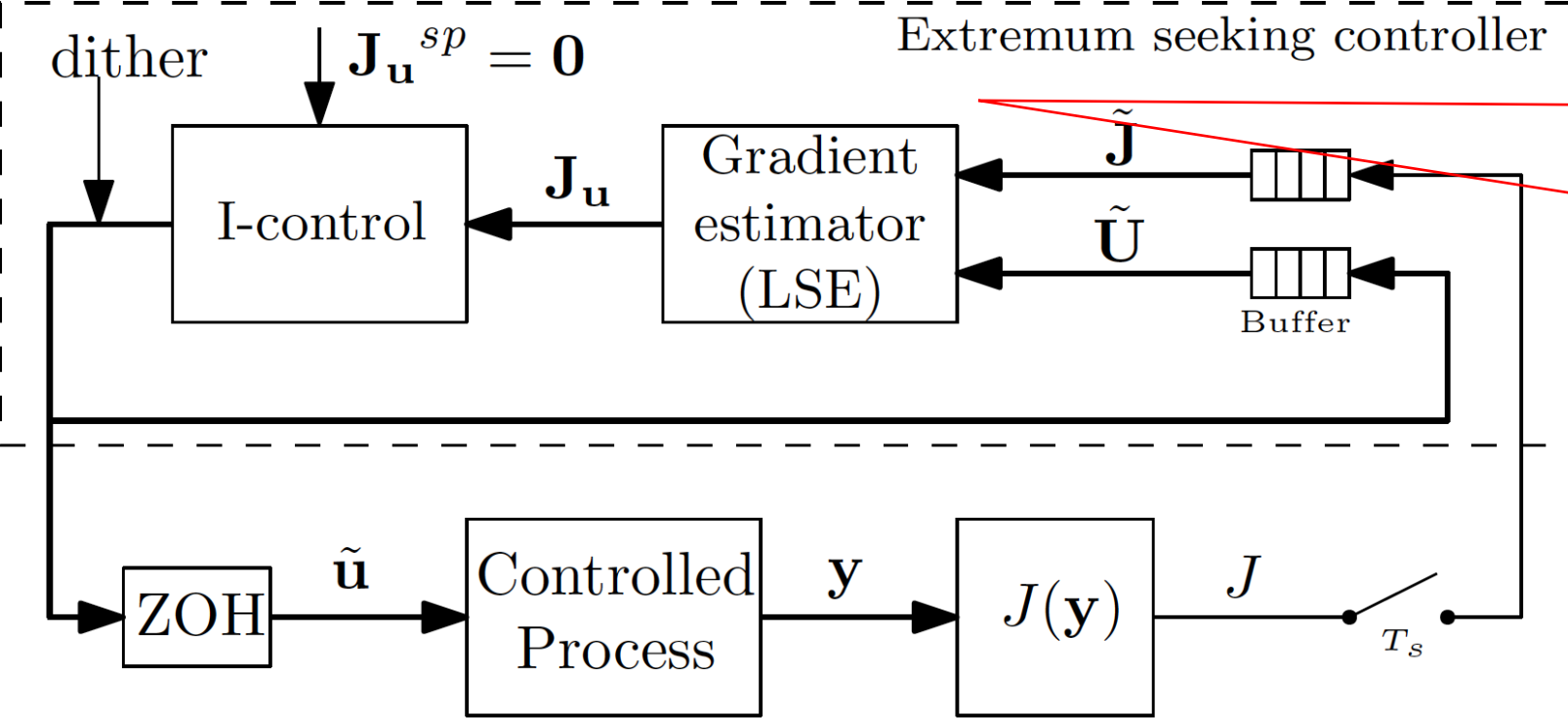
- **Least squares estimation**
- Fast Fourier transform

To avoid waiting for steady state

- Fitting of data to ARX model (difficult to make robust)

Note: Assumes steady state -> sampling (step) time > 3-10 time process time constant

Least square Extremum seeking control



LSE: Fit a linear model

$$J = \mathbf{J}_{\tilde{\mathbf{u}}}^T \tilde{\mathbf{u}} + m$$

Using least squares fit

$$\mathbf{Y} = [J_k, J_{k-1}, \dots, J_{k-N+1}]^T$$

$$\mathbf{U} = [\mathbf{u}_k, \dots, \mathbf{u}_{k-N+1}]^T$$

$$\theta = [\hat{\mathbf{J}}_{\mathbf{u}}^T, m]^T$$

$$\hat{\theta} = \arg \min_{\theta} \|\mathbf{Y} - \Phi^T \theta\|_2^2$$

to which the analytical solution is given by

$$\hat{\theta} = [\Phi^T \Phi]^{-1} \Phi^T \mathbf{Y}$$

Note: Assumes no dynamics -> sampling time > 3-10 time process constant

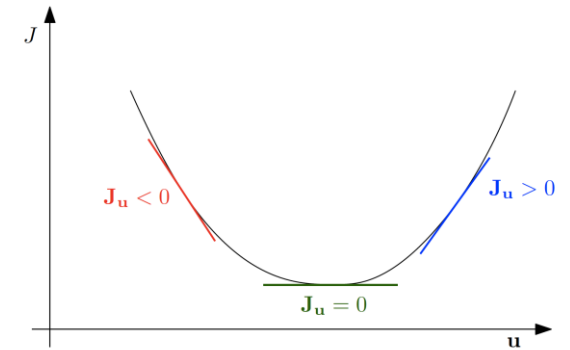
Summary extremum seeking control

Idea: Estimate the cost gradient J_u from data and drive it to zero

- **Common to all methods:**
 - Need measurement of cost J
 - Must wait for steady state (except ARX method which fails frequently)
 - Must assume no «fast» disturbances (while optimizing)

Algorithm needs two layers on top of process:

1. Optimization layer (slowest): Drive J_u to zero (may use I-controller)
 2. Lower estimation layer: Estimate the local gradient J_u using data
 - Must wait for the process to reach steady state
- Need time scale separation between layers.
 - At best this means that the optimization needs to be 10 times slower than the process.
 - Often it needs to be 100 times slower.
 - **Useful for fast processes with settling time a few seconds**
 - Not useful for many chemical processes where time constant typically are several minutes
 - 10 minutes * 100 = 1000 minutes = 16 hours
 - Unlikely with 16 hours without disturbance



ARC: Research tasks

8.1. A list of specific research tasks

Here is a list of some research topics, which are important but have received limited (or no) academic attention:

1. Vertical decomposition including time scale separation in hierarchically decomposed systems (considering performance and robustness)
2. Horizontal decomposition including decentralized control and input/output pairing
3. Selection of variables that link the different layers in the control hierarchy, for example, self-optimizing variables (CV1 in Fig. 4) and stabilizing variables (CV2).
4. Selection of intermediate controlled variables (w) in a cascade control system.⁹
5. Tuning of cascade control systems (Figs. 9 and 10)
6. Structure of selector logic
7. Tuning of anti-windup schemes (e.g., optimal choice of tracking time constant, τ_T) for input saturation, selectors, cascade control and decoupling.
8. How to make decomposed control systems based on simple elements easily understandable to operators and engineers
9. Default tuning of PID controllers (including scaling of variables) based on limited information
10. Comparison of selector on input or setpoint (cascade)

8.2. The harder problem: Control structure synthesis

The above list of research topics deals mainly with the individual elements. A much harder research issue is *the synthesis of an overall decomposed control structure, that is, the interconnection of the simple control elements for a particular application*. This area definitely needs some academic efforts.

One worthwhile approach is case studies. That is, to propose “good” (= effective and simple) control strategies for specific applications, for example, for a cooling cycle, a distillation column, or an integrated plant with recycle. It is here suggested to design also a centralized controller (e.g., MPC) and use this as a benchmark to quantify the performance loss (or maybe the benefit in some cases) of the decomposed ARC solution. A related issue, is to suggest new smart approaches to solve specific problems, as mentioned in item 11 in the list above.

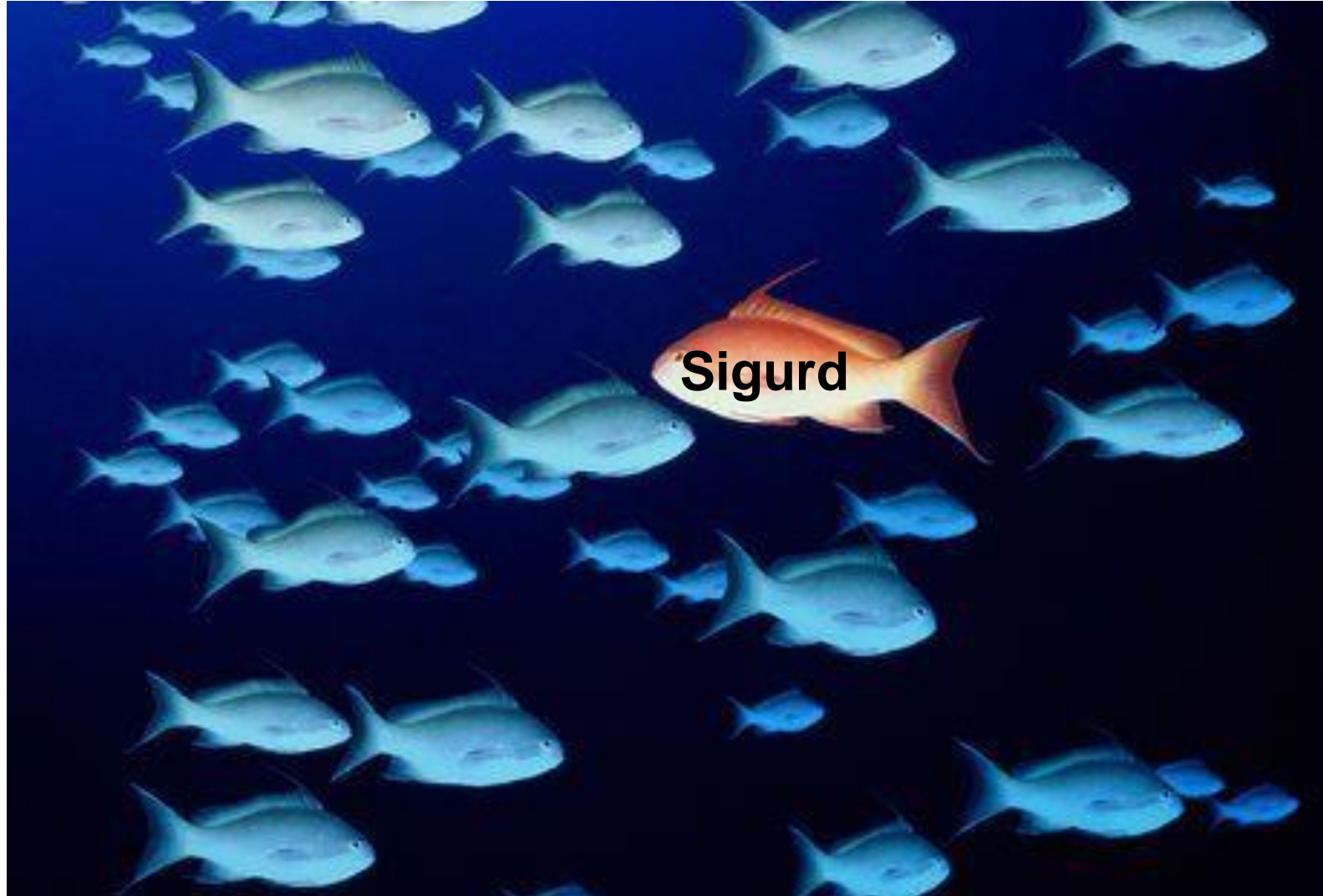
A second approach is mathematical optimization: Given a process model, how to optimally combine the control elements E1–E18 to meet the design specifications. However, even for small systems, this is a very difficult combinatorial problem, which easily becomes prohibitive in terms of computing power. It requires both deciding on the control structure as well as tuning the individual PID controllers.

As a third approach, **machine learning** may prove to be useful. Machine learning has one of its main strength in pattern recognition, in a similar way to how the human brain works. I have observed over the years that some students, with only two weeks of example-based teaching, are able to suggest good process control solutions with feedback, cascade, and feedforward/ratio control for realistic problems, based on only a flowsheet and some fairly general statements about the control objectives. This is the basis for believing that machine learning (e.g., a tool similar to ChatGPT) may provide a good initial control structure, which may later be improved, either manually or by optimization. It is important that such a tool has a graphical interface, both for presenting the problem and for proposing and improving solutions.

Present Academic control community fish pond

Complex optimal centralized
Solution (EMPC, FL)

Simple solutions
that work (ARC, PID)



Future Academic control community fish pond

Complex optimal centralized
Solution (EMPC, FL)

Simple solutions
that work (SRC, PID)

