Computational Control

Feedback Optimization

Saverio Bolognani

Automatic Control Laboratory (IfA)
ETH Zurich

Optimal steady-state regulation



Many *automation* tasks:

- **Stable** or **pre-stabilized** nonlinear plant
- **Steady-state specifications** in terms of optimality
  - yield
  - economic cost
  - CO2 emissions
- Multiple **constrained inputs** to be set
- Multiple **steady-state constraints** to be satisfied
- **Exogenous factors** (disturbances) affect the plant steady state
  - disturbances
  - unknown parameters

Differences from Economic MPC?

Optimal steady-state regulation

- **Stable** or **pre-stabilized** nonlinear plant
- **Lack of accurate dynamic model**
- **Steady-state specifications** in terms of optimality (efficiency, cost, ...)
- Multiple **constrained inputs** to be set
- Multiple **steady-state constraints** to be satisfied
- **Exogenous factors** (disturbances) affect the plant steady state

Grid frequency control

- Activation of reserves (generation ramp up/down)
- Exogenous time-varying demand
- Economic cost
- Exact steady-state power balance

Processes (compressors)

- Complex steady-state mapping
- Tunable parameters and set-points
- Efficiency goals
- Safety constraints
- Control limits

Fast-stable plant

Assumption

The plant is an exponentially stable, time invariant, system

$$\dot{x} = Ax + Bu + Ew$$
$$y = Cx + Du$$

with **steady-state map**

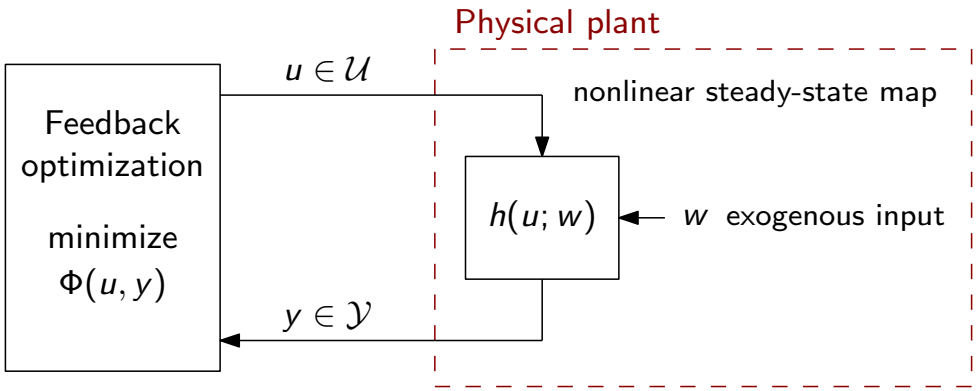
$y = Hu + Lw$ $\qquad$ $y = h(u; w)$ in the general nonlinear case

What is H? What are the conditions on A?

$$0 = Ax + Bu + Ew \quad \Rightarrow \quad x = -A^{-1}Bu - A^{-1}Ew$$
$$y = \underbrace{(-CA^{-1}B + D)}_{H} u \ \underbrace{-CA^{-1}E}_{L} w$$

Therefore A must be invertible (it is needed for exponential stability anyway).

"Certainty-equivalence" design



Steady state specifications

$$\begin{aligned} \min_{u,y} \quad & \Phi(u, y) \\ \text{subject to} \quad & y \in \mathcal{Y} \\ & u \in \mathcal{U} \\ & y = h(u; w) \end{aligned}$$

"Feedforward solution":

Solve the optimization problem and compute $u^* \in \mathcal{U}$ that minimizes

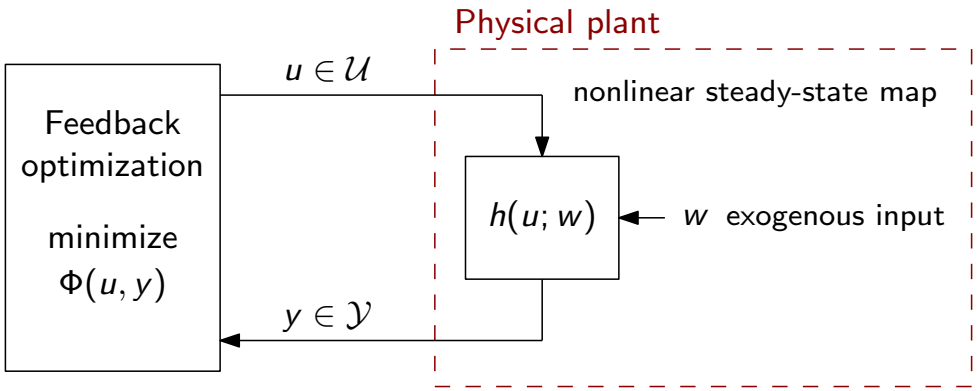
$$\begin{aligned} u^* = \quad \underset{u \in \mathcal{U}}{\text{argmin}} \quad & \tilde{\Phi}(u) := \Phi(u, h(u; w)) \\ \text{subject to} \quad & h(u; w) \in \mathcal{Y} \end{aligned}$$

The usual feedforward problems...

What can go wrong?

- we need accurate knowledge of w
- we need an accurate model h

"Certainty-equivalence" design



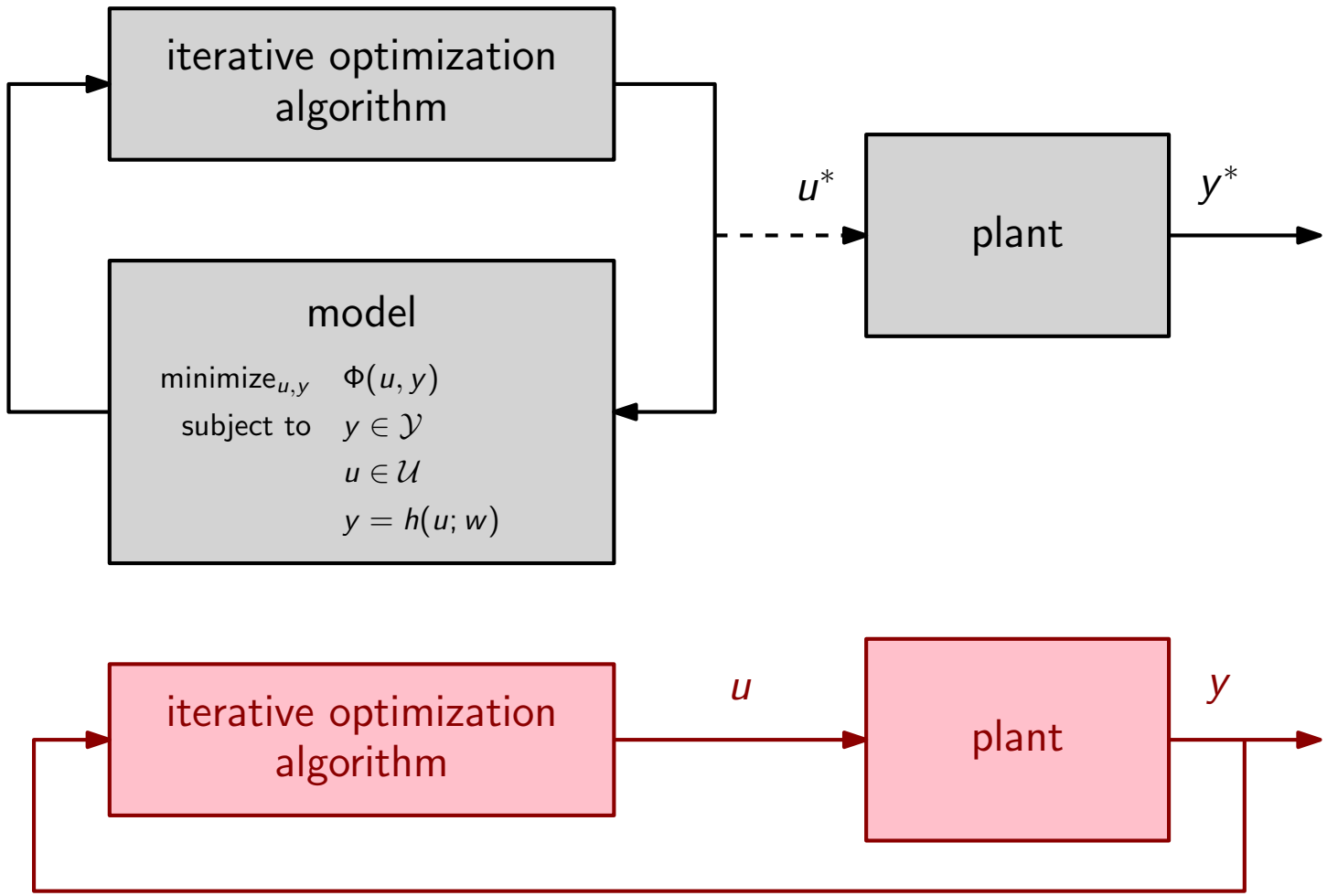
Steady state specifications

$$\begin{aligned} \min_{u,y} \quad & \Phi(u, y) \\ \text{subject to} \quad & y \in \mathcal{Y} \\ & u \in \mathcal{U} \\ & y = h(u; w) \end{aligned}$$

How to design **feedback controllers** that yield the desired **(autonomous) closed-loop behavior**?

Algebraic **input/output steady-state map** $y = h(u; w)$
(with constant $w \approx$ slowly changing)
+
Classical **iterative nonlinear optimization algorithms**

Interconnecting plant and algorithms



Online Feedback Optimization

First-order optimization algorithms

$$\begin{aligned} \min_{u,y} \quad & \Phi(u,y) \\ \text{subject to} \quad & y \in \mathcal{Y} \\ & u \in \mathcal{U} \\ & y = h(u;w) \end{aligned}$$

Equivalent:

$$\begin{aligned} \min_{u \in \mathcal{U}} \quad & \Phi(u, \underbrace{h(u;w)}_{y}) \\ \text{subject to} \quad & \underbrace{h(u;w)}_{y} \in \mathcal{Y} \end{aligned}$$

Many iterative algorithms available to solve this class of problems:

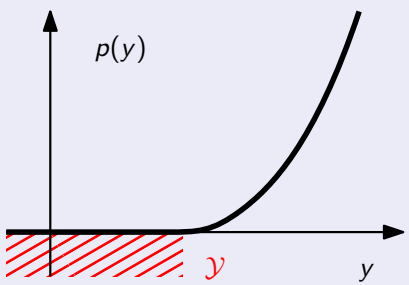
- Gradient descent with penalty / barrier
- Projected gradient descent
- Primal-dual flows

Gradient descent with penalty / barrier

$$\begin{aligned} \min_{u,y} \quad & \Phi(u,y) \\ \text{subject to} \quad & y \in \mathcal{Y} \\ & u \in \mathcal{U} \\ & y = h(u;w) \end{aligned}$$

Penalty function

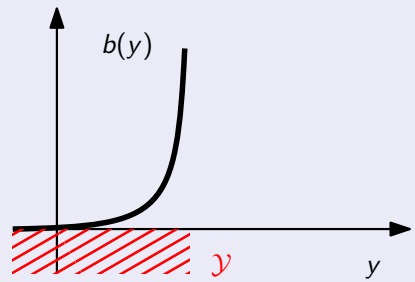
$$\min \quad \Phi(u,y) + p(y)$$



→ small violations are acceptable

Barrier function

$$\min \quad \Phi(u,y) + b(y)$$



→ conservative, never at boundary
→ undefined outside of feasible region

Unconstrained gradient descent

Convergence of gradient descent iterations

Let $\phi(x)$ have bounded second derivative, that is

$$\nabla^2 \phi(x) \leq MI \qquad \forall x.$$

Then the gradient descent iteration

$$x_{k+1} = x_k - \alpha \nabla \phi(x_k)$$

converges to a local minimum of $\min_x \phi(x)$ if $\alpha < 2/M$.

The Taylor expansion of ϕ around x_k is

$$\phi(x_{k+1}) = \phi(x_k) + \nabla \phi(x_k)^\top (x_{k+1} - x_k) + \frac{1}{2}(x_{k+1} - x_k)^\top \nabla^2 \phi(z)(x_{k+1} - x_k)$$

where z is a point between x_k and y. By plugging in the GD iteration and bounding the last term we get

$$\phi(x_{k+1}) \leq \phi(x_k) - \alpha \|\nabla \phi(x_k)\|^2 + \alpha^2 \frac{M}{2} \|\nabla \phi(x_k)\|^2,$$

which is smaller than $\phi(x_k)$ if $\alpha < 2/M$, guaranteeing a decreasing $\phi(x_k)$ as long as $\nabla \phi(x_k) \neq 0$ ($\rightarrow$ La Salle's theorem implies convergence to the set where $\nabla \phi = 0$).

Gradient descent with penalty

$$\min_{u \in \mathcal{U}} \quad \Phi(u, \underbrace{h(u;w)}_{y}) + p(\underbrace{h(u;w)}_{y})$$

Gradient descent algorithm

$$u_{k+1} = u_k - \alpha\Big(\nabla_u \Phi(u_k, h(u_k;w)) + \nabla_u h(u_k;w)^\top \nabla_y \Phi(u_k, h(u_k;w)) + \\ + \nabla_u h(u_k;w)^\top \nabla_y p(h(u_k;w))\Big)$$

Gradient descent controller

$$u_{k+1} = u_k - \alpha\Big(\nabla_u \Phi(u_k, y_k) + \nabla_u h(u_k;w)^\top \nabla_y \Phi(u_k, y_k) + \nabla_u h(u_k;w)^\top \nabla_y p(y_k)\Big)$$

where: measurement, model, and with u constrained to $\mathcal{U}$ (saturation).

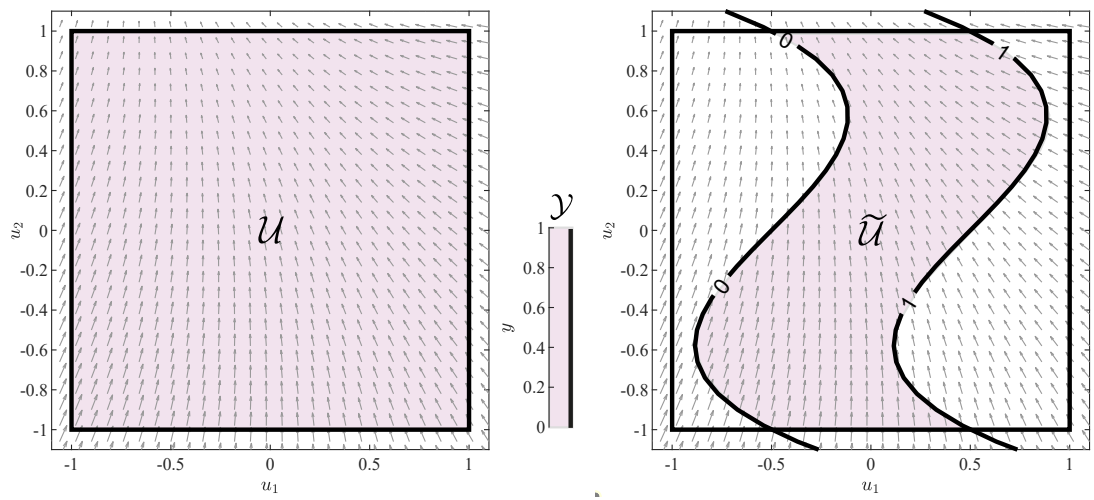
Steady-state sensitivities

What is $\nabla_u h(u;w)^\top$? Make practical examples for plants you know.

Projected gradient descent

$$\begin{aligned} \min_{u,y} \quad & \Phi(u,y) \\ \text{subject to} \quad & y \in \mathcal{Y} \\ & u \in \mathcal{U} \\ & y = h(u;w) \end{aligned}$$

$$\tilde{\mathcal{U}} = \mathcal{U} \cap h^{-1}(\mathcal{Y};w)$$



$$\min_{u \in \tilde{\mathcal{U}}} \quad \Phi(u, \underbrace{h(u;w)}_{y}) \qquad \rightarrow \text{nasty projection of the input...}$$

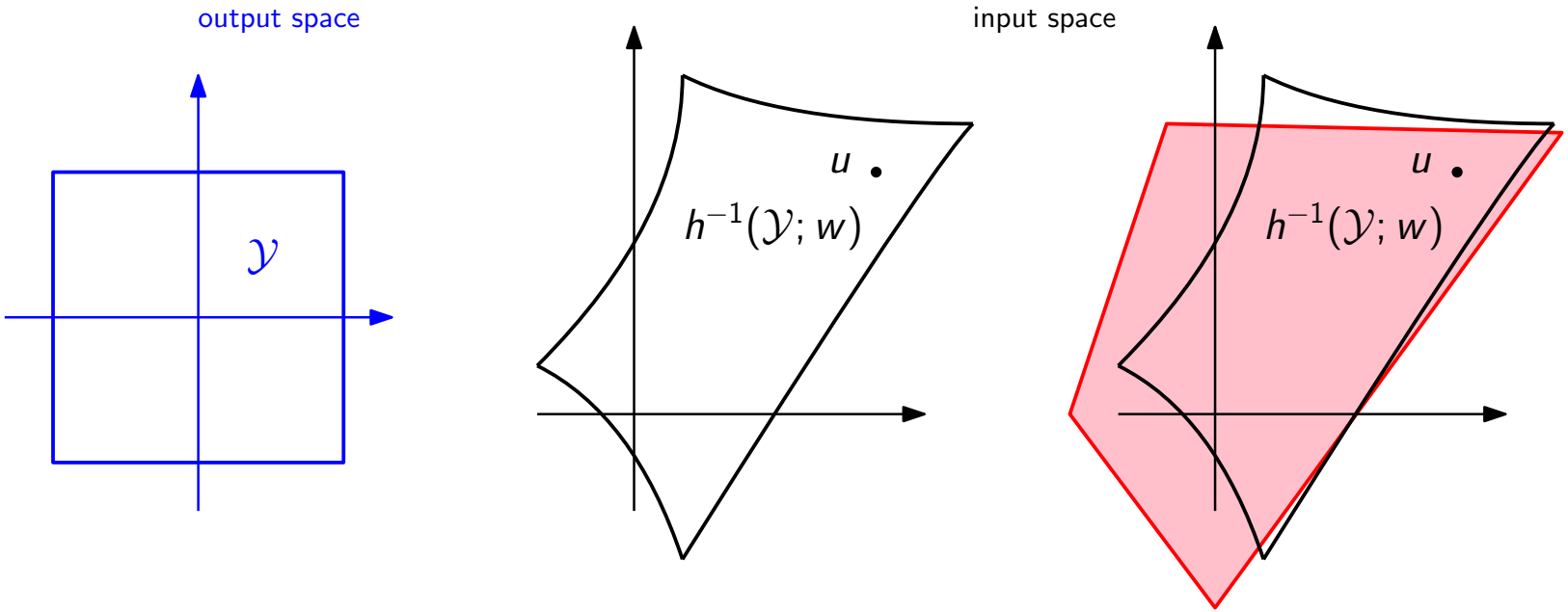
Projected gradient descent

$$\min_{u \in \tilde{\mathcal{U}}} \quad \Phi(u, \underbrace{h(u; w)}_{y})$$

$$\tilde{\mathcal{U}} = \mathcal{U} \cap h^{-1}(\mathcal{Y}; w)$$

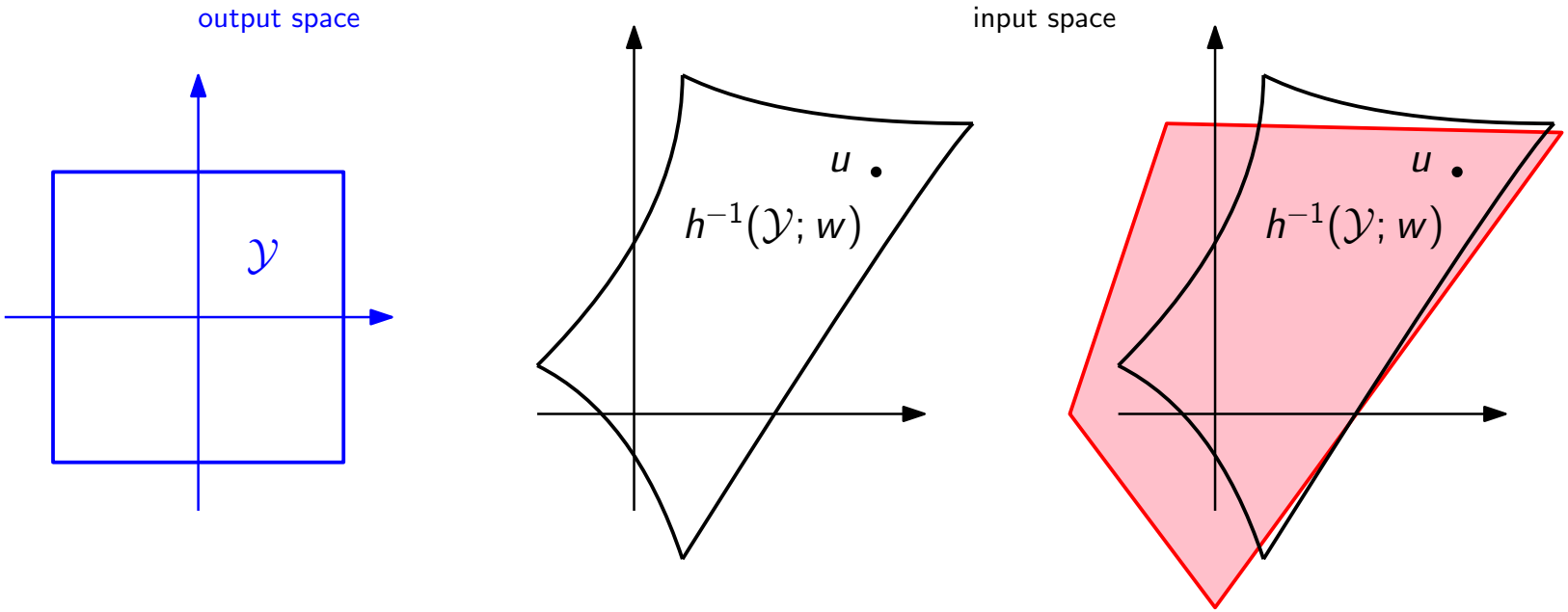
Assumption

$$\mathcal{U} := \{u \in \mathbb{R}^p \mid Au \leq b\}$$
$$\mathcal{Y} := \{y \in \mathbb{R}^n \mid Cy \leq d\}$$



What is a good approximation of $h^{-1}(\mathcal{Y})$ that

- is easy to project to (polytope)
- is accurate (especially when y is at the boundary of $\mathcal{Y}$)?



Taylor expansion of $h(u; w)$ around u

$$h(u'; w) \approx h(u; w) + \nabla_u h(u; w)^\top (u' - u)$$

First-order approximation of $h^{-1}(\mathcal{Y}; w)$ around u

$$Cy \leq d \quad \rightarrow \quad \underbrace{C\big(h(u; w) + \nabla_u h(u; w)^\top (u' - u)\big)}_{Ch(u'; w)} \leq d$$

Projected gradient flow via repeated Quadratic Programming

Polytope constraints on **input** and **output** (local approximation)

$$\mathcal{U} := \{u \in \mathbb{R}^p \mid Au \leq b\}$$
$$\mathcal{Y} := \{y \in \mathbb{R}^n \mid Cy \leq d\} \qquad \rightarrow \qquad C\big(\underbrace{h(u;w)}_{y} + \nabla_u h(u;w)^\top (u' - u)\big) \leq d$$

Projected gradient descent

$$u_{k+1} = u_k + \alpha \delta u$$

where δu is the solution of a simple quadratic program

$$\delta u := \underset{v}{\text{argmin}} \quad \Big\| v - \underbrace{\Big(-\nabla_u \Phi(u_k, y_k) - \nabla h(u_k;w)^\top \nabla_y \Phi(u_k, y_k)\Big)}_{-\text{ gradient}} \Big\|^2$$

$$\text{subject to} \quad A(u_k + \alpha v) \leq b$$
$$\underbrace{C(y_k + \alpha \nabla h(u_k;w)^\top v) \leq d}_{\text{1st order approx of } h^{-1}(\mathcal{Y};w) \text{ centered at the measurement } y}$$

Dual problem

Convex optimization notes

Read the short notes available on Moodle on convexity and duality.

Consider the constrained optimization problem

$$\begin{aligned} \min_x \quad & \phi(x) \\ \text{subject to} \quad & g(x) \leq 0 \end{aligned}$$

Consider the Lagrangian

$$\mathcal{L}(x, \lambda) := \phi(x) + \lambda^\top g(x)$$

Under **convexity assumptions** and **constraint qualifications**, we have

$$\begin{aligned} \min \quad & \phi(x) \\ \text{subject to} \quad & g(x) \leq 0 \end{aligned} \quad = \quad \min_x \max_{\lambda \geq 0} \mathcal{L}(x, \lambda) \quad = \quad \max_{\lambda \geq 0} \min_x \mathcal{L}(x, \lambda)$$

Proof of the first equality: $\max_{\lambda \geq 0} \mathcal{L}(x, \lambda)$ is $+\infty$ unless $g(x) \leq 0$, in which case $\lambda = 0$ is a solution. Then $\min_x \mathcal{L}(x, 0)$ is precisely the original optimization problem with the constraint replaced by an indicator function.

Primal/dual iterations

To solve the dual problem

$$\max_{\lambda \geq 0} \min_{x} \mathcal{L}(x, \lambda)$$

with

$$\mathcal{L}(x, \lambda) := \phi(x) + \lambda^\top g(x)$$

one can use

- **primal descent / dual ascent**

$$x_{k+1} = x_k - \alpha \nabla_x \mathcal{L}(x_k, \lambda_k)$$
$$\lambda_{k+1} = \lambda_k + \beta \nabla_\lambda \mathcal{L}(x_k, \lambda_k)$$

- **dual ascent**

$$x_k = \arg\min_x \mathcal{L}(x, \lambda_k)$$
$$\lambda_{k+1} = \lambda_k + \beta \nabla_\lambda \mathcal{L}(x_k, \lambda_k) \qquad \text{what is } \nabla_\lambda \mathcal{L}(x_k, \lambda_k)?$$

(similar to saddle flow with $\beta << \alpha$).

Dual ascent

$$\mathcal{Y} := \{y \in \mathbb{R}^n \,|\, g(y) \leq 0\}, \qquad \mathcal{L}(u, \lambda) = \Phi(u, \underbrace{h(u; w)}_{y}) + \lambda^\top g(\underbrace{h(u; w)}_{y})$$

Primal descent / dual ascent

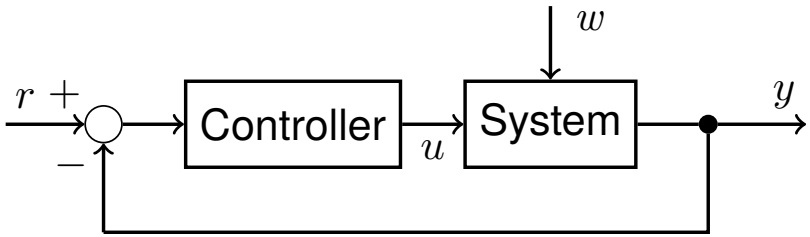
$$u_{k+1} = u_k - \alpha\Big(\nabla_u \Phi(u_k, y) + \underbrace{\nabla_u h(u_k; w)^\top}_{\text{model}} \nabla_y \Phi(u_k, y_k) + \underbrace{\nabla_u h(u_k; w)^\top}_{\text{model}} \nabla g(y_k)^\top \lambda_k\Big)$$

$$\lambda_{k+1} = \Pi_{\geq 0}\left[\lambda_k + \beta g(y_k)\right]$$

Once again, we only need steady-state sensitivities and measurements (feedback)!

What is the role of the internal controller memory λ_k?

An old friend



$$\begin{aligned} \min_{u,y} \quad & \frac{1}{2}\|u\|^2 + \frac{\rho}{2}\|Hu + w - r\|^2 \\ \text{subject to} \quad & y = r \quad \text{(tracking)} \\ & y = Hu + w \quad \text{(steady-state)} \end{aligned}$$

$$\text{Lagrangian } \mathcal{L}(u, \lambda) = \|u\|^2/2 + \lambda^\top (Hu + w - r) + \frac{\rho}{2}\|Hu + w - r\|^2$$

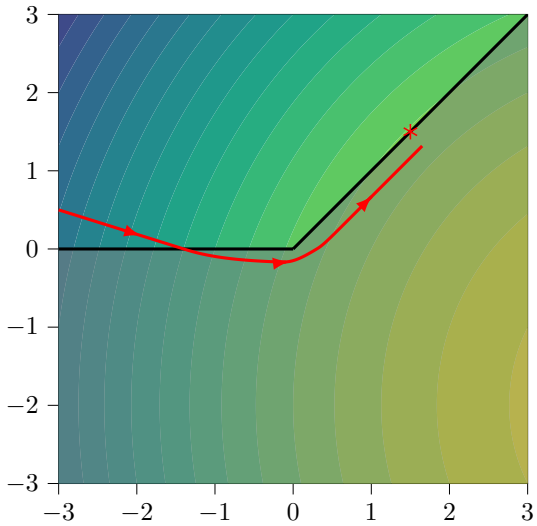
- Primal minimization $\nabla_u \mathcal{L}(u_k, \lambda_k) = 0$
- Dual ascent $\lambda_{k+1} = \lambda_k + \beta \nabla_\lambda \mathcal{L}(u_k, \lambda_k)$

Dual ascent = PI controller!

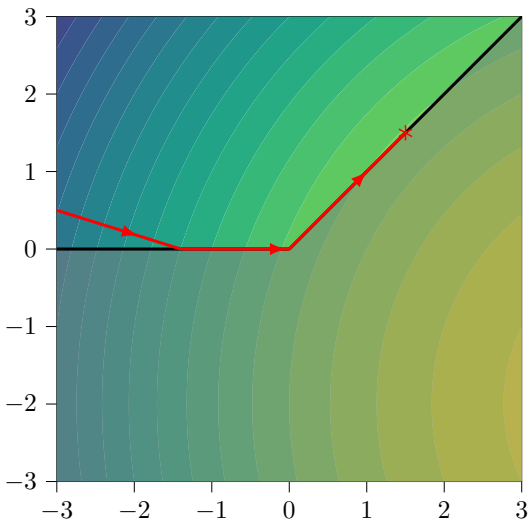
$$u_k = -H^\top \lambda + \rho H^\top (\underbrace{Hu_k + w}_{y_k} - r) \qquad \lambda_{k+1} = \lambda_k + \beta(\underbrace{Hu_k + w}_{y_k} - r)$$

Design guideline and tradeoffs

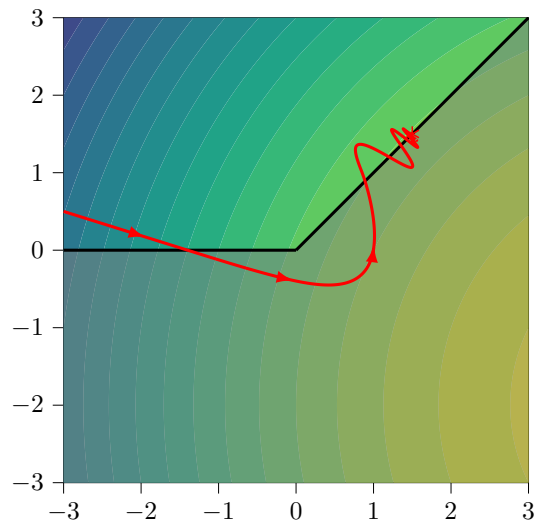
Penalty
arbitrarily small violation
'proportional"
steep penalty limits speed



Projected gradient
any-time feasibility
quadratic programming
simple gain limit



Saddle flow
asymptotic feasibility
"(proportional)-integral"
oscillations



In all cases

- **assumption:** fast stable system
- **model information:** steady-state sensitivities $\nabla_u h(u; w)$
- **feedback measurement:** output measurement y

Online Feedback Optimization in the wild

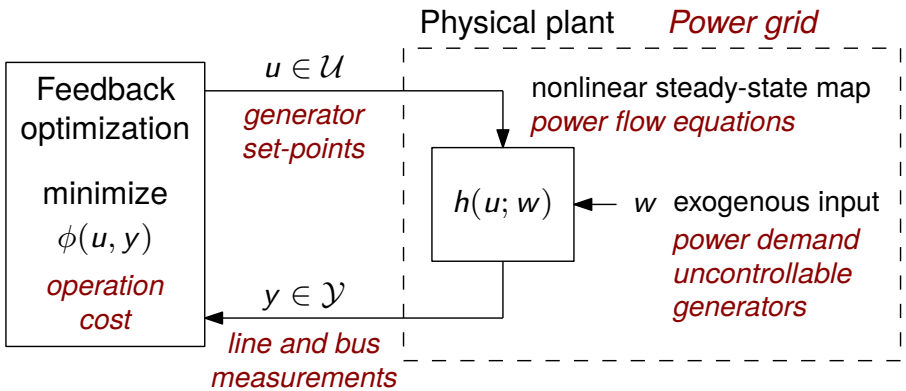
NCCR Automation team win Watt d'Or for efficient and sustainable power supply with photovoltaics

SUSTAINABLE AUTOMATION | 11-01-2024

NCCR Automation researchers from the Automatic Control Laboratory, ETH Zurich and the electricity supplier AEW Energie AG have won the 2024 Watt d'Or award in the Energy Technologies category. Their algorithm, which was implemented at AEW Energie, makes it possible to optimise electricity grid operations.



Online Feedback Optimization is currently running on the Swiss distribution grid in Aargau!

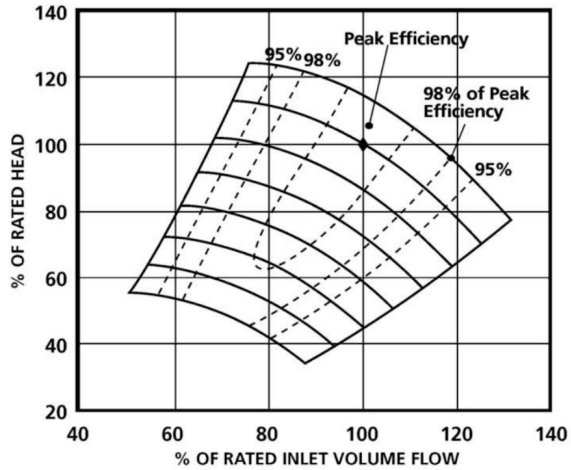


$\rightarrow$ More info...

Extremum seeking

In many applications, **steady-state sensitivities** may be unknown.





- **Robust approach:** Feedback optimization works with very inaccurate steady-state sensitivities

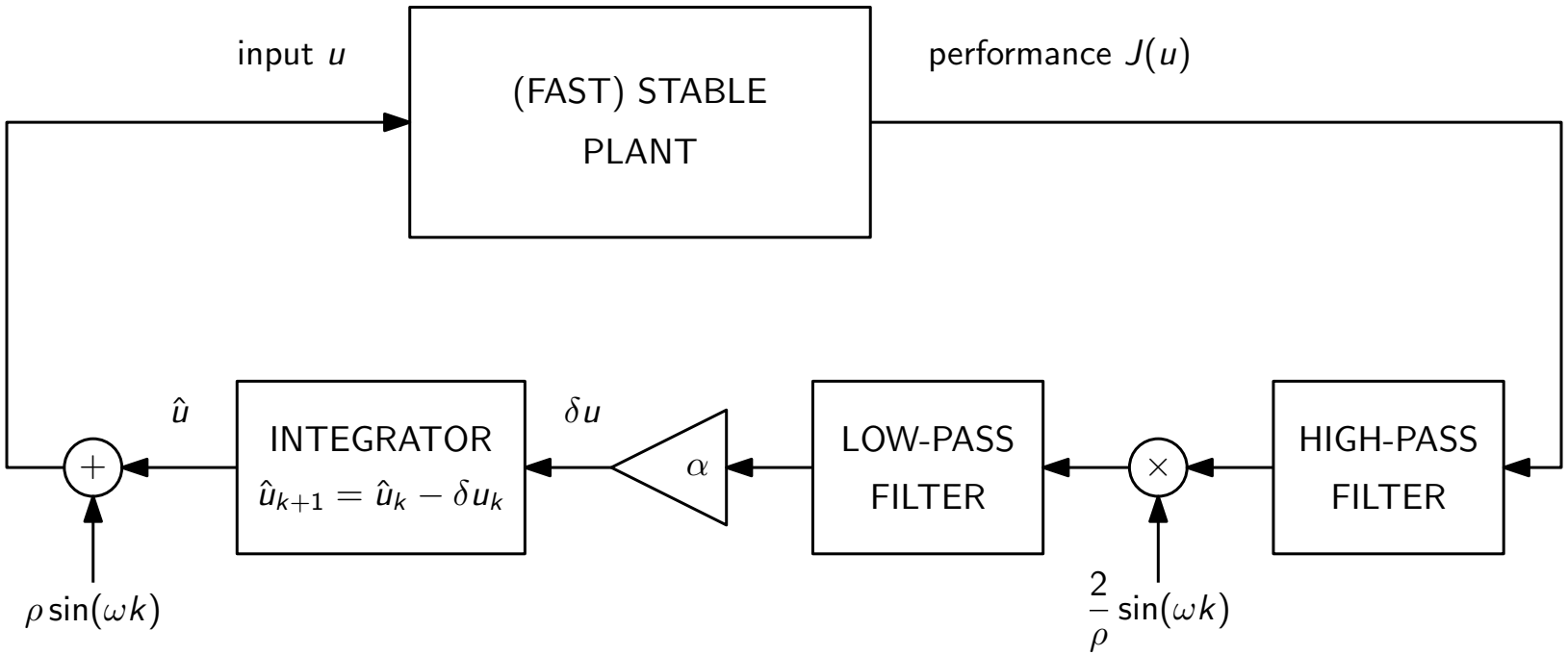
Exercise

Prove that both the projected gradient descent and the saddle flow cannot have an unfeasible equilibrium $y \notin \mathcal{Y}$.

- **Learning:** Sensitivities can be estimated
- **Zeroth order algorithms:** optimize without sensitivites

Extremum seeking

How to optimize a performance metric $J(u)$ without any knowledge of $\nabla_u J(u)$?



Think of

- $\rho \sin(\omega k)$ as an **exploration** signal
- $\frac{2}{\rho} \sin(\omega k)$ as a **demodulation** signal

Extremum seeking is gradient descent (sketch)

Let us look at the signals in the block diagram clockwise, from

$$u_k = \rho \sin(\omega t) + \hat{u}_k.$$

Them using a Taylor expansion of J (for small ρ)

$$J(u_k) \approx J(\hat{u}_k) + \nabla_u J(\hat{u}_k) \rho \sin(\omega t).$$

After the HP filter:

$$\nabla_u J(\hat{u}_k) \rho \sin(\omega t)$$

After the demodulation:

$$\nabla_u J(\hat{u}_k) 2 \sin^2(\omega t)$$

After the LP filter:

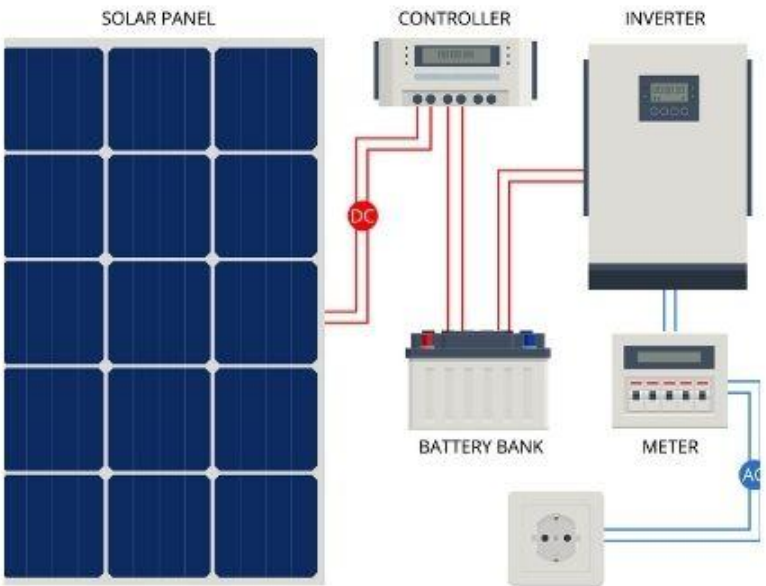
$$\nabla_u J(\hat{u}_k)$$

Therefore

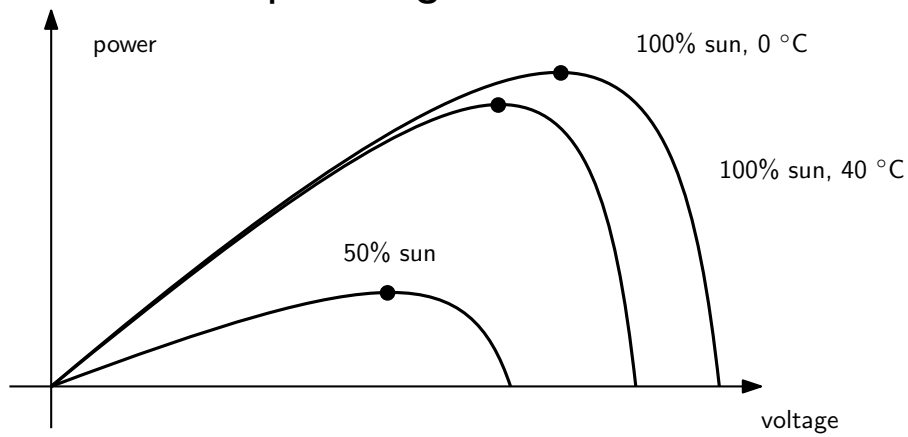
$$\delta u_k = \alpha \nabla_u J(\hat{u}_k)$$

And the integrator implements a gradient descent $\hat{u}_{k+1} = \hat{u}_k - \alpha \nabla_u J(\hat{u}_k)$.

Applications of extremum seeking



Track PV voltage that maximizes active power generation



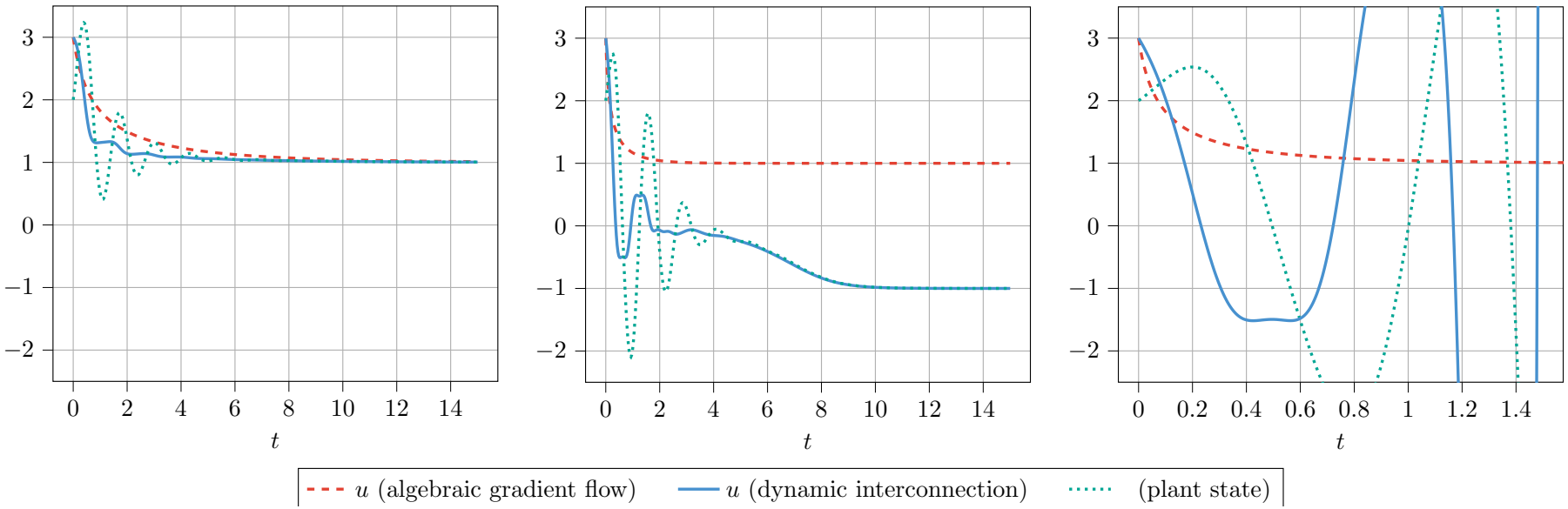
- Time-varying, parameter dependent, poor model
- Cheap actuation
- Low dimensional problem

Other applications: internal combustion engines, chemical processes, meta-optimization (e.g., PI tuning!), and many more

Warning!

We have always assumed that the plant is **fast-stable**.

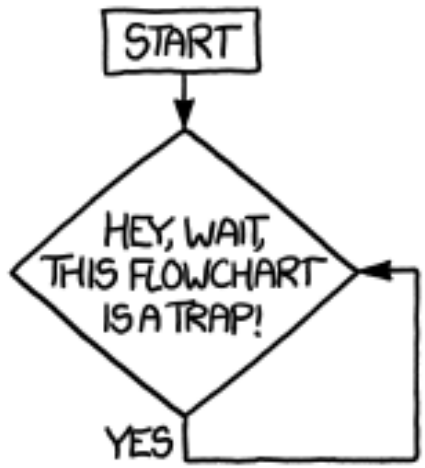
» Increasing gain α »



The interconnection between these controllers and **dynamic plants** can be studied to obtain limits on how fast the controller can be (**timescale separation**):

- Gradient-based feedback optimization (Khalil, "Nonlinear systems", Ch. 9) (check also "Timescale Separation in Autonomous Optimization")
- Extremum seeking (Khalil, "Nonlinear systems", Ch. 8)

The control engineer flowchart





This work is licensed under a
Creative Commons Attribution-NonCommercial 4.0 International License

https://bsaver.io/COCO