Lecture 6: Model-based control

Jens Kober Robert Babuška

Knowledge-Based Control Systems (SC42050)

Cognitive Robotics
3mE, Delft University of Technology, The Netherlands

21-02-2018

Considered Settings

- Fuzzy or neural model of the process available (many of the presented techniques apply to other types of models as well)
- Based on the model, design a controller (off line)
- Use the model explicitly within a controller
- Model fixed or adaptive

Outline

1. Local design using Takagi–Sugeno models
2. Inverse model control
3. Model-based predictive control
4. Feedback linearization
5. Adaptive control

TS Model $\rightarrow$ TS Controller

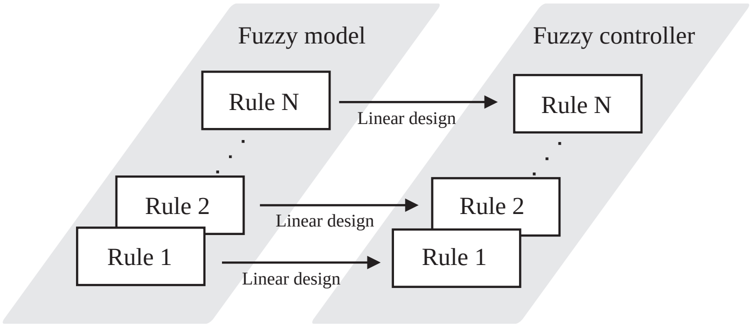
Model:

If $y(k)$ is Small **then** $x(k+1) = a_s x(k) + b_s u(k)$
If $y(k)$ is Medium **then** $x(k+1) = a_m x(k) + b_m u(k)$
If $y(k)$ is Large **then** $x(k+1) = a_l x(k) + b_l u(k)$

Controller:

If $y(k)$ is Small **then** $u(k) = -L_s x(k)$
If $y(k)$ is Medium **then** $u(k) = -L_m x(k)$
If $y(k)$ is Large **then** $u(k) = -L_l x(k)$

Design Using a Takagi–Sugeno Model



Apply classical synthesis and analysis methods locally.

Control Design via Lyapunov Method

Model:

$$\textbf{If } \boldsymbol{x}(k) \text{ is } \Omega_i \qquad \textbf{then} \quad \boldsymbol{x}_i(k+1) = \boldsymbol{A}_i \boldsymbol{x}(k) + \boldsymbol{B}_i \boldsymbol{u}(k)$$

Controller:

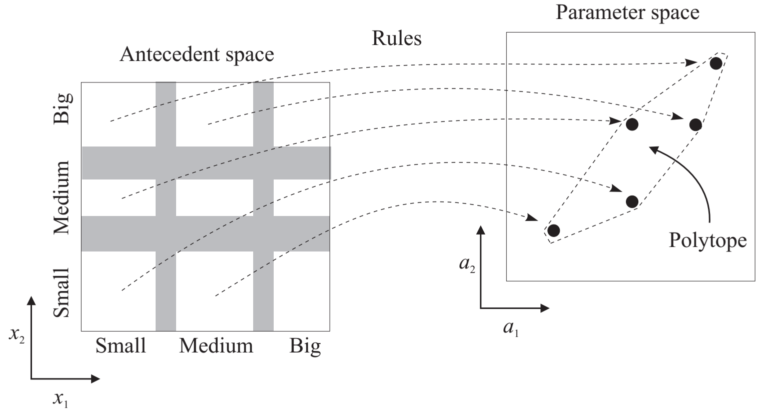
$$\textbf{If } \boldsymbol{x}(k) \text{ is } \Omega_i \qquad \textbf{then} \quad \boldsymbol{u}_i(k) = -\boldsymbol{L}_i \boldsymbol{x}(k)$$

Stability guaranteed if $\exists\, \boldsymbol{P} > \boldsymbol{0}$ such that:

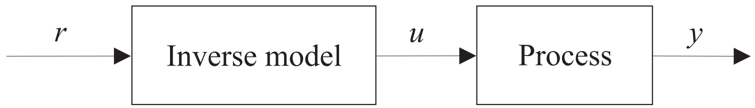
$$(\boldsymbol{A}_i - \boldsymbol{B}_i \boldsymbol{L}_j)^T \boldsymbol{P} (\boldsymbol{A}_i - \boldsymbol{B}_i \boldsymbol{L}_j) - \boldsymbol{P} < \boldsymbol{0}, \quad i, j = 1, \ldots, K$$

TS Model is a Polytopic System

$$\boldsymbol{x}(k+1) = \left(\sum_{i=1}^{K} \sum_{j=1}^{K} \gamma_i(\boldsymbol{x}) \gamma_j(\boldsymbol{x}) (\boldsymbol{A}_i - \boldsymbol{B}_i \boldsymbol{L}_j) \right) \boldsymbol{x}(k)$$



Inverse Control (Feedforward)



Process model: $y(k+1) = f(\boldsymbol{x}(k), u(k))$, where

$$\boldsymbol{x}(k) = [y(k), \ldots, y(k-n_y+1), u(k-1), \ldots, u(k-n_u+1)]^T$$

Controller: $u(k) = f^{-1}(\boldsymbol{x}(k), r(k+1))$

When is Inverse-Model Control Applicable?

1. Process (model) is stable and invertible
2. The inverse model is stable
3. Process model is accurate (enough)
4. Little influence of disturbances
5. In combination with feedback techniques

How to invert $f(\cdot)$?

1. Numerically (general solution, but slow):

$$J(u(k)) = \left[r(k+1) - f(\mathbf{x}(k), u(k))\right]^2$$

   minimize w.r.t. $u(k)$

2. Analytically (for some special forms of $f(\cdot)$ only):
   - affine in $u(k)$
   - singleton fuzzy model
3. Construct inverse model directly from data

Inverse of an Affine Model

affine model:

$$y(k+1) = g(\mathbf{x}(k)) + h(\mathbf{x}(k)) \cdot u(k)$$

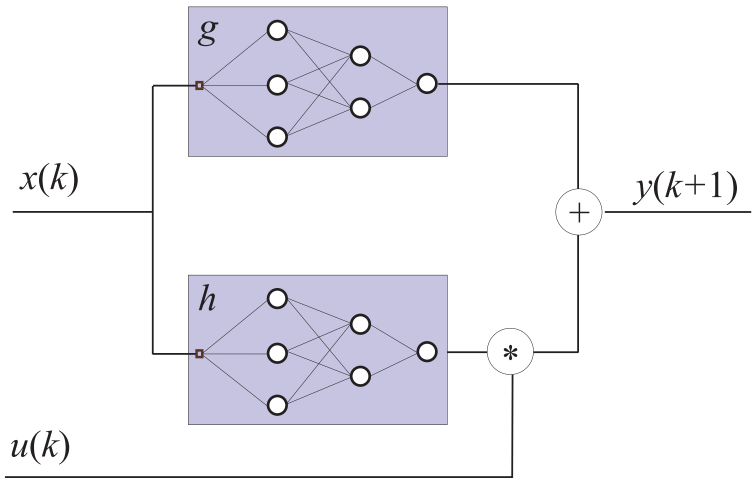
express $u(k)$:

$$u(k) = \frac{y(k+1) - g(\mathbf{x}(k))}{h(\mathbf{x}(k))}$$

substitute $r(k+1)$ for $y(k+1)$

necessary condition $h(\mathbf{x}) \neq 0$ for all $\mathbf{x}$ of interest

Example: Affine Neural Network



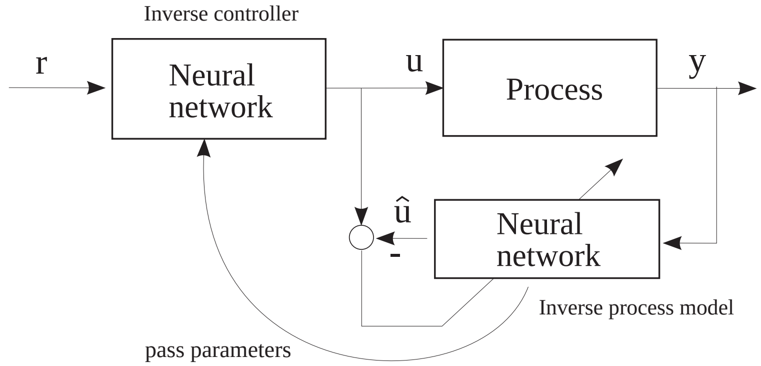
Example: Affine TS Fuzzy Model

$\mathcal{R}_{\rangle}$: **If** $y(k)$ **is** A_{i1} **and** ... **and** $y(k-n_y+1)$ **is** A_{in_y} **and** $u(k-1)$ **is** B_{i2} **and** ... **and** $u(k-n_u+1)$ **is** B_{in_u} **then**

$$y_i(k+1) = \sum_{j=1}^{n_y} a_{ij} y(k-j+1) + \sum_{j=1}^{n_u} b_{ij} u(k-j+1) + c_i,$$

$$y(k+1) = \sum_{i=1}^{K} \gamma_i(\boldsymbol{x}(k)) \left[\sum_{j=1}^{n_y} a_{ij} y(k-j+1) + \sum_{j=2}^{n_u} b_{ij} u(k-j+1) + c_i \right] + \sum_{i=1}^{K} \gamma_i(\boldsymbol{x}(k)) b_{i1} u(k)$$

Learning Inverse (Neural) Model



How to obtain $\boldsymbol{x}$?

inverse model: $u(k) = f^{-1}(\boldsymbol{x}(k), r(k+1))$

1. Use the prediction model: $\hat{y}(k+1) = f(\hat{\boldsymbol{x}}(k), u(k))$

   $$\hat{\boldsymbol{x}}(k) = [\hat{y}(k), \ldots, \hat{y}(k-n_y+1), u(k-1), \ldots, u(k-n_u+1)]^T$$

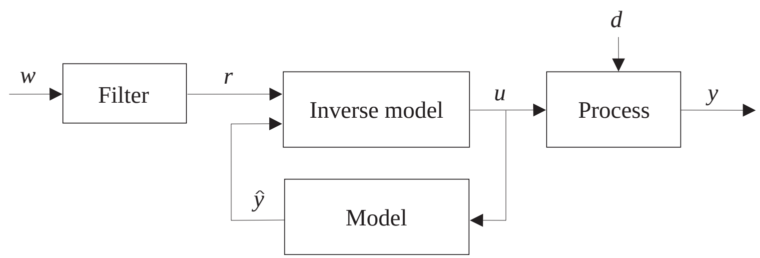
   Open-loop feedforward control

2. Use measured process output

   $$\boldsymbol{x}(k) = [y(k), \ldots, y(k-n_y+1), u(k-1), \ldots, u(k-n_u+1)]^T$$

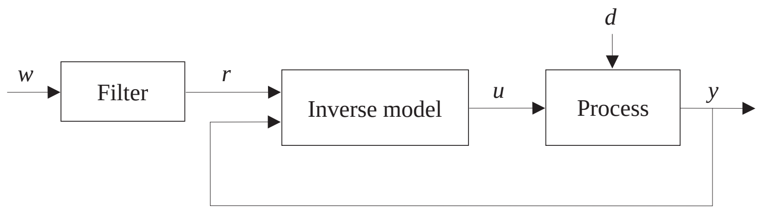
   Open-loop feedback control

Open-Loop Feedforward Control



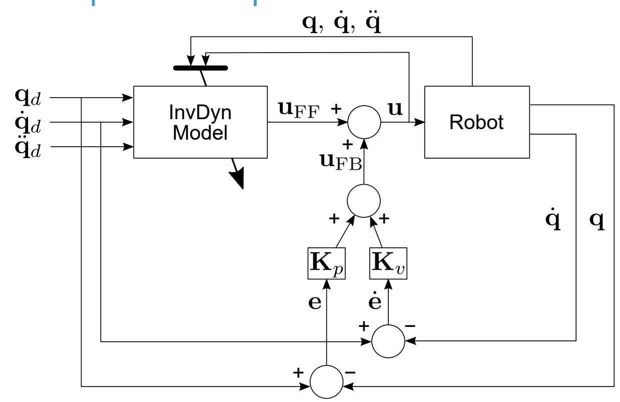
- Always stable (for stable processes)
- No way to compensate for disturbances

Open-Loop Feedback Control



- Can to some degree compensate disturbances
- Can become unstable

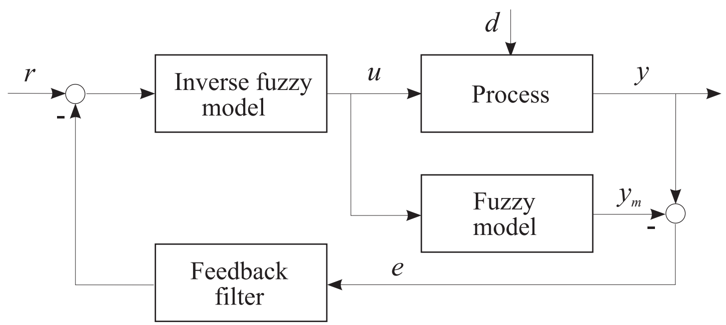
Example: Computed Torque Control[1]



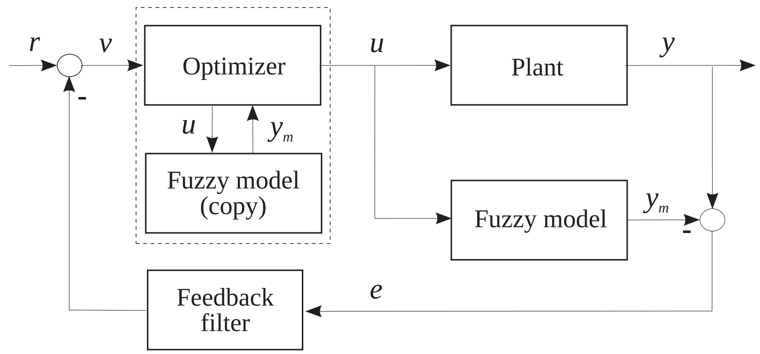
- Open-loop feedforward control + low gain PD controller
- Video: RBD model vs. learned model
- Video: adaptive model

[1]D. Nguyen-Tuong and J. Peters (2011). "Incremental Sparsification for Real-time Online Model Learning". In: *Neurocomputing* 74.11, pp. 1859–1867

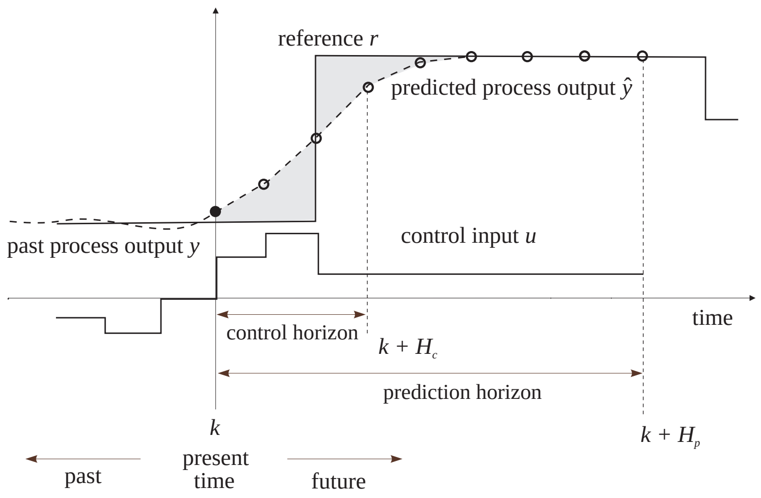
Internal Model Control



Model-Based Predictive Control



Model-Based Predictive Control



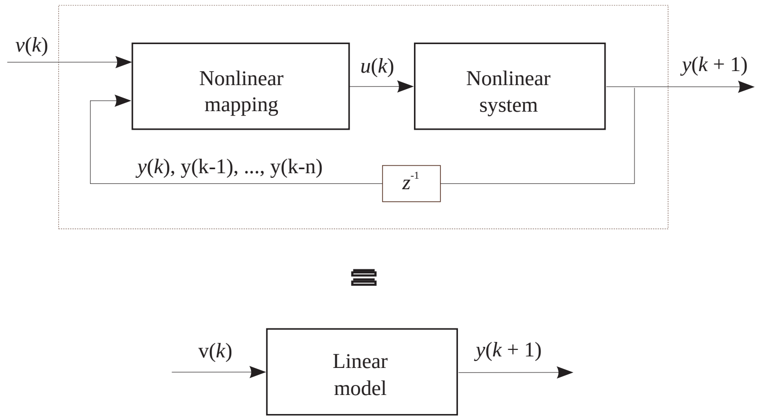
Objective Function and Constraints

$$J = \sum_{i=1}^{H_p} \| \mathbf{r}(k+i) - \hat{\mathbf{y}}(k+i) \|^2_{P_i} + \sum_{i=1}^{Hc} \| \mathbf{u}(k+i-1) \|^2_{Q_i}$$

$$\hat{y}(k+1) = f(\hat{\mathbf{x}}(k), u(k))$$

$$\begin{array}{ccccc} \mathbf{u}^{\min} & \leq & \mathbf{u} & \leq & \mathbf{u}^{\max} \\ \Delta\mathbf{u}^{\min} & \leq & \Delta\mathbf{u} & \leq & \Delta\mathbf{u}^{\max} \\ \mathbf{y}^{\min} & \leq & \mathbf{y} & \leq & \mathbf{y}^{\max} \\ \Delta\mathbf{y}^{\min} & \leq & \Delta\mathbf{y} & \leq & \Delta\mathbf{y}^{\max} \end{array}$$

Feedback linearization



Feedback Linearization (continued)

given affine system: $y(k+1) = g(\mathbf{x}(k)) + h(\mathbf{x}(k)) \cdot u(k)$

express $u(k)$:

$$u(k) = \frac{y(k+1) - g(\mathbf{x}(k))}{h(\mathbf{x}(k))}$$

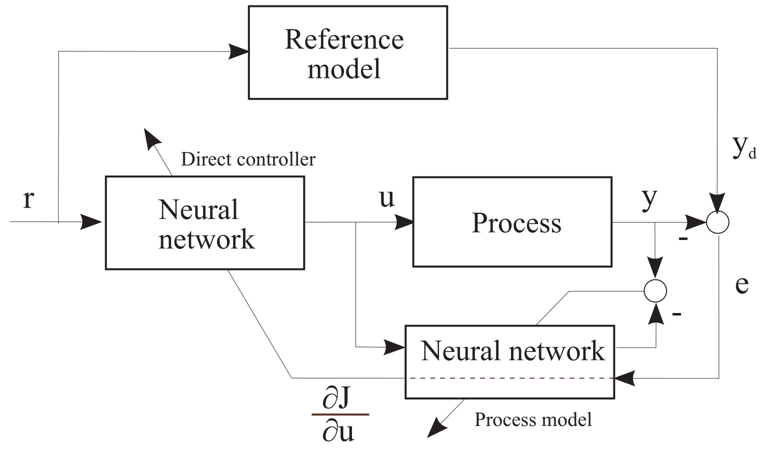
substitute $A(q)y(k) + B(q)v(k)$ for $y(k+1)$:

$$u(k) = \frac{A(q)y(k) + B(q)v(k) - g(\mathbf{x}(k))}{h(\mathbf{x}(k))}$$

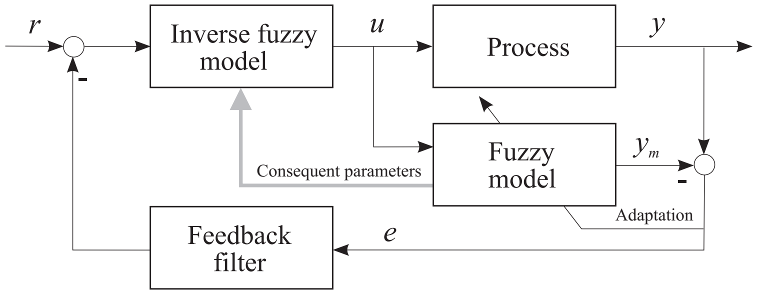
Adaptive Control

- Model-based techniques (use explicit process model):
  - model reference control through backpropagation
  - indirect adaptive control
- Model-free techniques (no explicit model used)
  - reinforcement learning

Model Reference Adaptive Neurocontrol



Indirect Adaptive Control



not only for fuzzy models, but also for affine NNs, etc.

Reinforcement Learning

- Inspired by principles of human and animal learning.
- No explicit model of the process used.
- No detailed feedback, only reward (or punishment).
- A control strategy can be learnt from scratch.