# Lecture 6: Model based control

*Jens Kober* Robert Babuška

Knowledge-Based Control Systems (SC42050)
Delft Center for Systems and Control
3mE, Delft University of Technology, The Netherlands

01-03-2017



## 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 → TS Controller

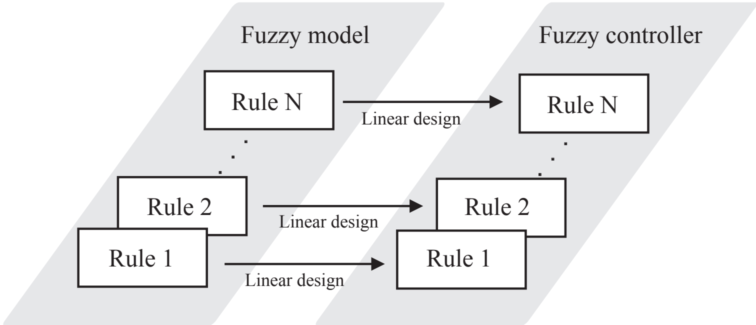
Model:

**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:

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

Controller:

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

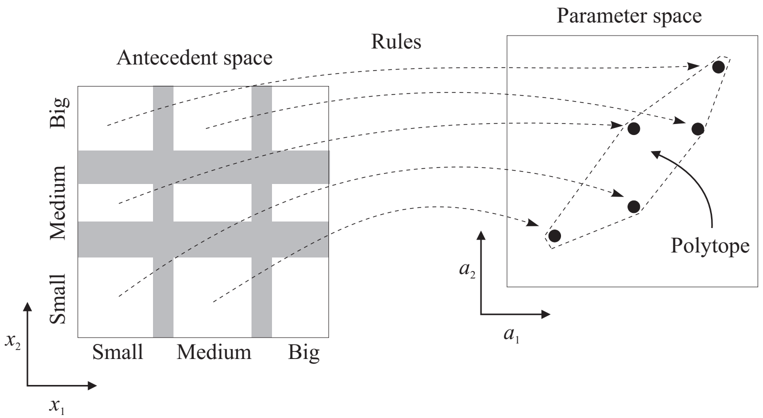
Stability guaranteed if $\exists\, \mathbf{P} > \mathbf{0}$ such that:

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



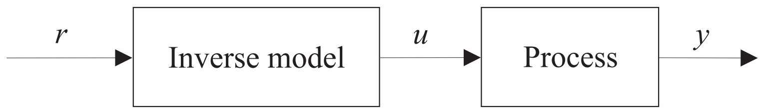
## TS Model is a Polytopic System

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






## Inverse Control (Feedforward)




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

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

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

## 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):

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

   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:

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

express $u(k)$:

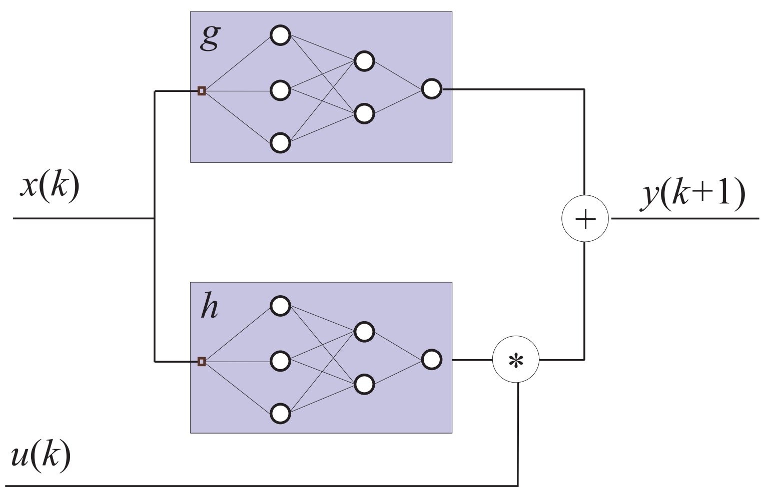
$$u(k) = \frac{y(k+1) - g\big(\mathbf{x}(k)\big)}{h\big(\mathbf{x}(k)\big)}$$

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

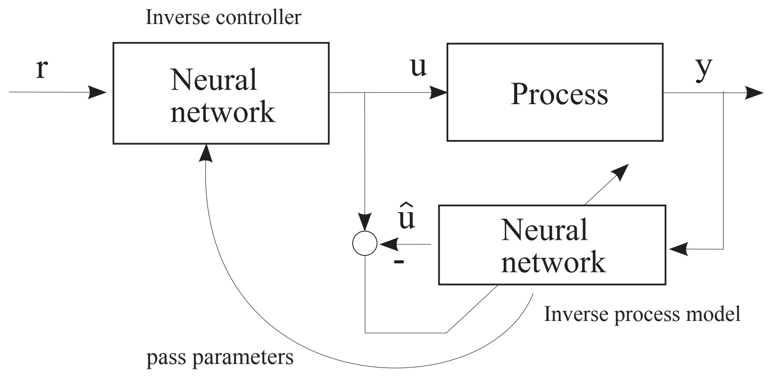
$\mathcal{R}:$ **If** $y(k)$ **is** $A_{i1}$ **and** ... **and** $y(k-n_y+1)$ **is** $A_{in_y}$ **and** $u(k-1)$ **is** $B_{i2}$ **and** ... **and** $u(k-n_u+1)$ **is** $B_{in_u}$ **then**

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

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



## Learning Inverse (Neural) Model






## How to obtain x?

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

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

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

   Open-loop feedforward control

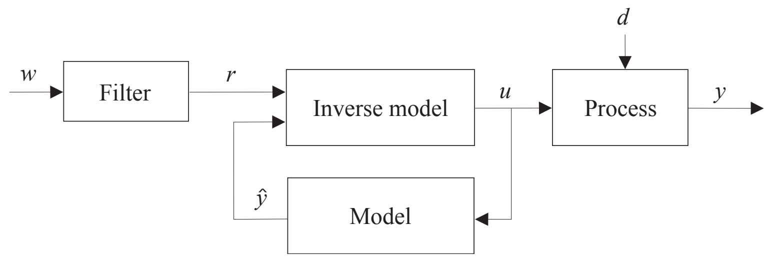
2. Use measured process output

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

   Open-loop feedback control

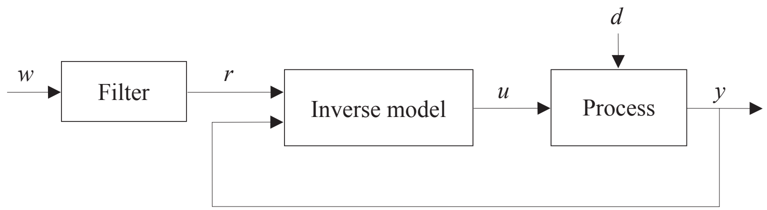


## Open-Loop Feedforward Control




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

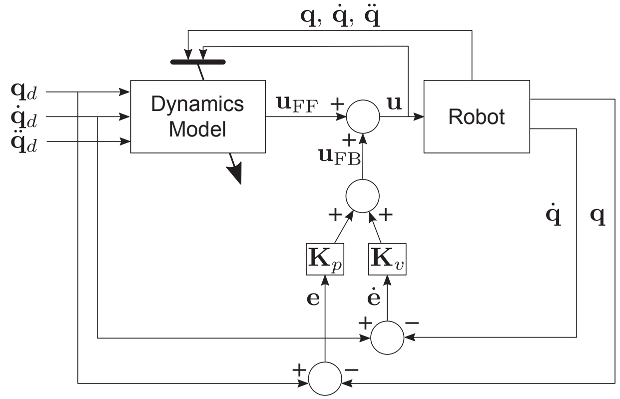
# Open-Loop Feedback Control




- Can to some degree compensate disturbances
- Can become unstable



# Example: Computed Torque Control[1]

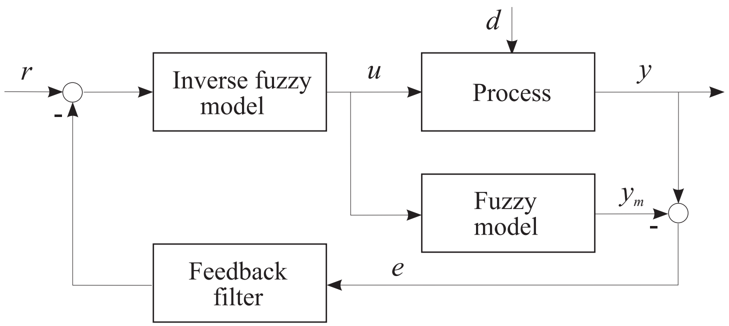



- Video: RBD model vs. learned model
- Video: adaptive model

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

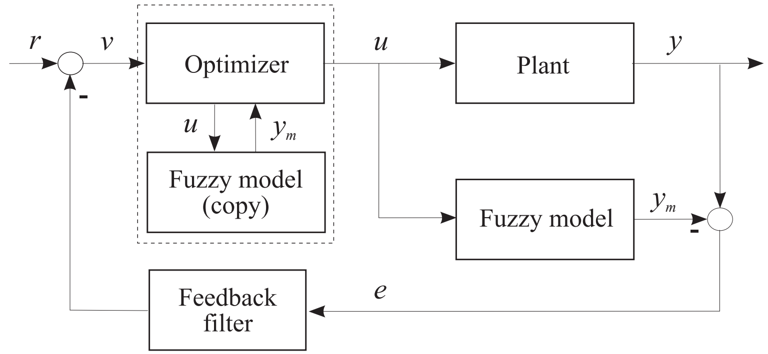


# Internal Model Control

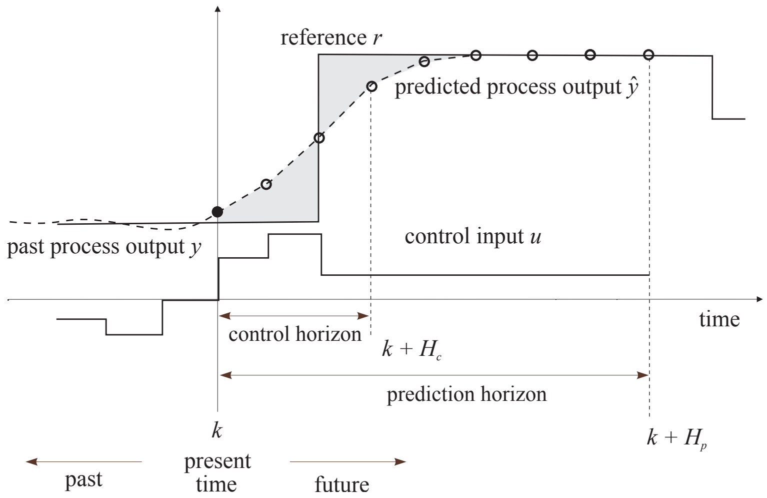





# Model-Based Predictive Control



## Model-Based Predictive Control






## Objective Function and Constraints

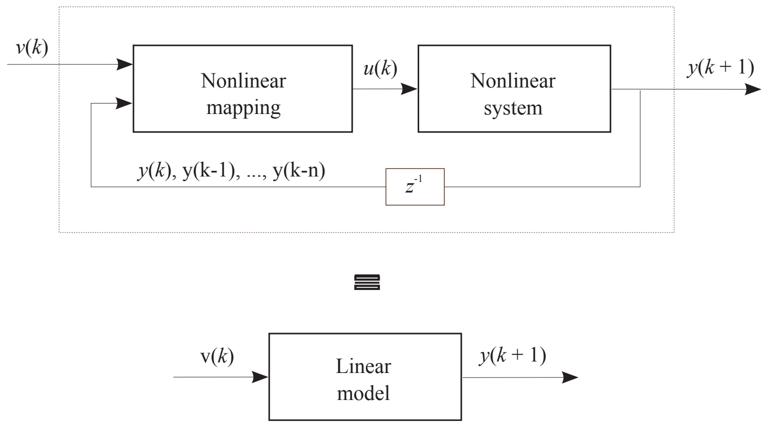
$$J = \sum_{i=1}^{H_p} \left\| \mathbf{r}(k+i) - \hat{\mathbf{y}}(k+i) \right\|_{P_i}^2 + \sum_{i=1}^{Hc} \left\| \mathbf{u}(k+i-1) \right\|_{Q_i}^2$$

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

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



## Feedback linearization






## Feedback Linearization (continued)

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

express $u(k)$:

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

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

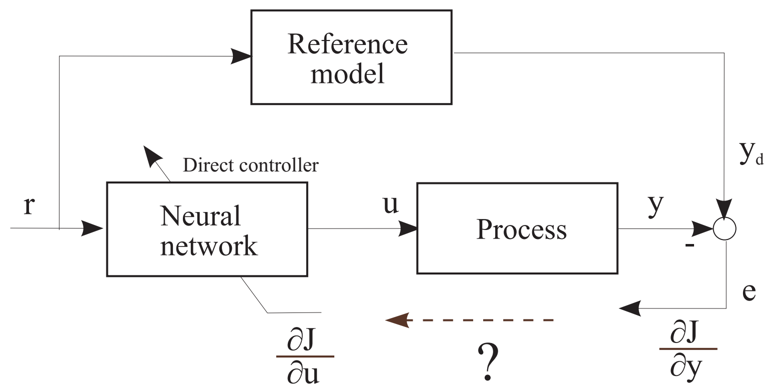
$$u(k) = \frac{A(q)y(k) + B(q)v(k) - g\big(\mathbf{x}(k)\big)}{h\big(\mathbf{x}(k)\big)}$$

## 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

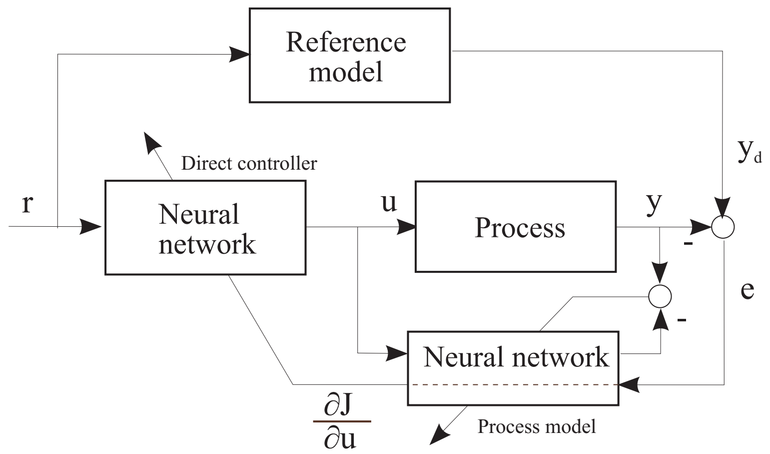


## Model Reference Adaptive Neurocontrol



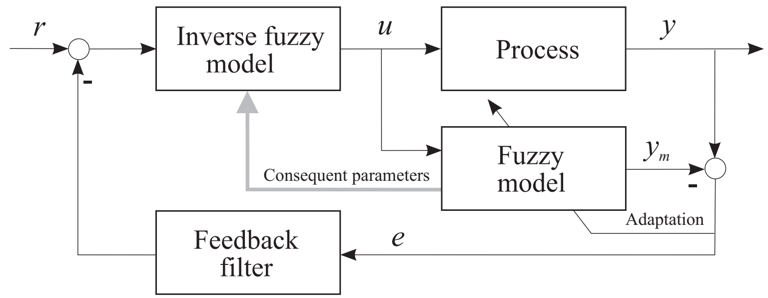



## Model Reference Adaptive Neurocontrol






## Indirect Adaptive Control




no 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.