

# Control Systems Lab – SC4070

Lecture 2, February 8, 2012

**dr.ir. Alessandro Abate**

*Delft Center for Systems and Control  
Delft University of Technology  
The Netherlands*

e-mail: [a.abate@tudelft.nl](mailto:a.abate@tudelft.nl)

tel: 015 27 85606

(slides modified from the original, drafted by Robert Babuška)

# Lecture outline

1. Parameter optimization in nonlinear models
2. Linear system identification in time domain
3. Experiment design and identification procedure
4. Additional topics, demos

# Refs on parameter and model identification

- L. Ljung, *System Identification: Theory for the User*, 2nd edition, Prentice-Hall, 1999
- L. Ljung, *System Identification*, Chapter 58 of the *Control Engineering Handbook*, pp. 1033–1054, CRC Press, 1996
- K.J. Aström & B. Wittenmark, *Computer-Controlled Systems*, Prentice-Hall, 1997, Chapter 13
- Lecture notes of the courses “Filtering & identification” (SC4040) or “System identification” (SC4110)
- Manual of MATLAB System Identification Toolbox

# General overview on models

1 – Physical models in state-space form:

$$\begin{aligned}\frac{d}{dt}x(t) &= f(x(t), u(t), \theta) \\ y(t) &= h(x(t), u(t), \theta)\end{aligned}$$

2 – Black-box models in frequency:

• 1st order & 2nd order linear models (in continuous-time):

$$G_1(s) = \frac{1}{\tau s + 1} \quad G_2(s) = \frac{\omega^2}{s^2 + 2\zeta\omega s + \omega^2}$$

• linear models: ARX, ARMAX, OE, BJ (in discrete-time):

$$y(k) = G(q, \theta)u(k) + H(q, \theta)e(k)$$

• nonlinear models (neural networks, fuzzy models, ...)

# Physical models in state-space form

Model:

$$\frac{d}{dt}x(t) = f(x(t), u(t), \theta)$$
$$y(t) = h(x(t), u(t), \theta)$$

What is the optimal value of  $\theta$ ? We want to “estimate” it

- Given model, guess  $\theta \rightarrow$ , simulate output  $\hat{y}(t_k|\theta)$  at  $t = t_k = k \cdot h$
- Given measured signal  $y(t_k)$ ,  
define prediction error:  $\varepsilon(k, \theta) = y(t_k) - \hat{y}(t_k|\theta)$

- Define performance index:

$$J(\theta) = \frac{1}{N} \sum_{k=1}^N g(\varepsilon(k, \theta)) \quad \text{with } N: \# \text{ data points}$$

- Compute optimal value:

$$\hat{\theta}_N = \arg \min_{\theta} J(\theta)$$

# Physical models in state-space form

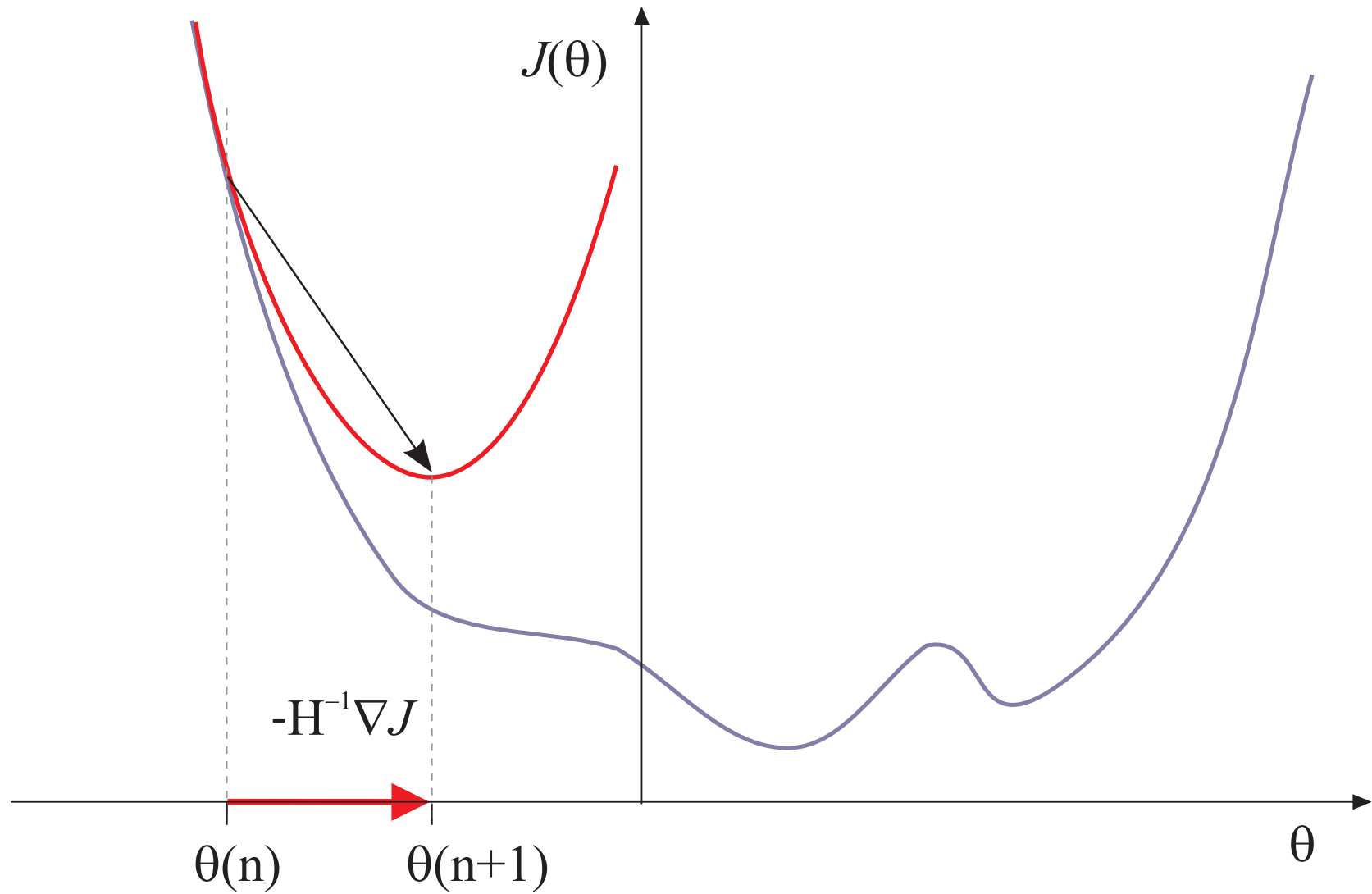
Usually:  $g(\varepsilon) = \varepsilon^2$  :

$$\hat{\theta}_N = \arg \min_{\theta} \frac{1}{N} \sum_{k=1}^N \varepsilon^2(k, \theta)$$

In general: **numerical, nonlinear optimization** algorithms  
see course Optimization in Systems and Control (SC4090)

- Matlab optimization toolbox (`lsqnonlin`)
- (also used in Matlab NCD toolbox (nonlinear control design))

# Nonlinear least squares



## In Matlab: `lsqnonlin`

Syntax: `x=lsqnonlin('fun',x0,lb,ub,options)` with

- `fun` : m-file function returning  $e$  and its Jacobian  $\nabla J$  (optional)

```
[e,J]=fun(x)
```

```
e=...
```

```
% Compute Jacobian if required.
```

```
if ( nargout > 1 )
```

```
    J=...
```

```
end;
```

- `x0` : initial starting point
- `lb, ub` : lower and upper bounds for `x` (optional)
- `options` : options structure (optional)

# Most important options

- `LargeScale` : use a large-scale algorithm ('on') or medium-scale algorithm ('off').
- `Display` : controls display of (intermediate) values. Possible values: 'off', 'iter', and 'final'
- `Jacobian` : indicates whether Jacobian is defined by user
- `MaxIter` : maximum number of iterations allowed
- `TolFun`, `TolX` : termination tolerance on the function value and on  $x$
- `LevenbergMarquardt` : Choose Levenberg-Marquardt over Gauss-Newton algorithm (default: 'on')

Black-box models in frequency

# ARX and ARMAX models

- ARX:

$$y(k) + a_1y(k-1) + \dots a_ny(k-n) =$$
$$u(k) + b_1u(k-1) + b_mu(k-m) + e(k)$$
$$y(k) = \frac{B(q)}{A(q)}u(k) + \frac{1}{A(q)}e(k)$$

- ARMAX:

$$y(k) + a_1y(k-1) + \dots a_ny(k-n) =$$
$$u(k) + b_1u(k-1) + \dots + b_mu(k-m) +$$
$$e(k) + c_1e(k-1) + \dots + c_l e(k-l)$$
$$y(k) = \frac{B(q)}{A(q)}u(k) + \frac{C(q)}{A(q)}e(k)$$

# Output-Error and Box-Jenkins models

- Output-Error:

$$y(k) = \frac{B(q)}{F(q)}u(k) + e(k)$$

- Box-Jenkins:

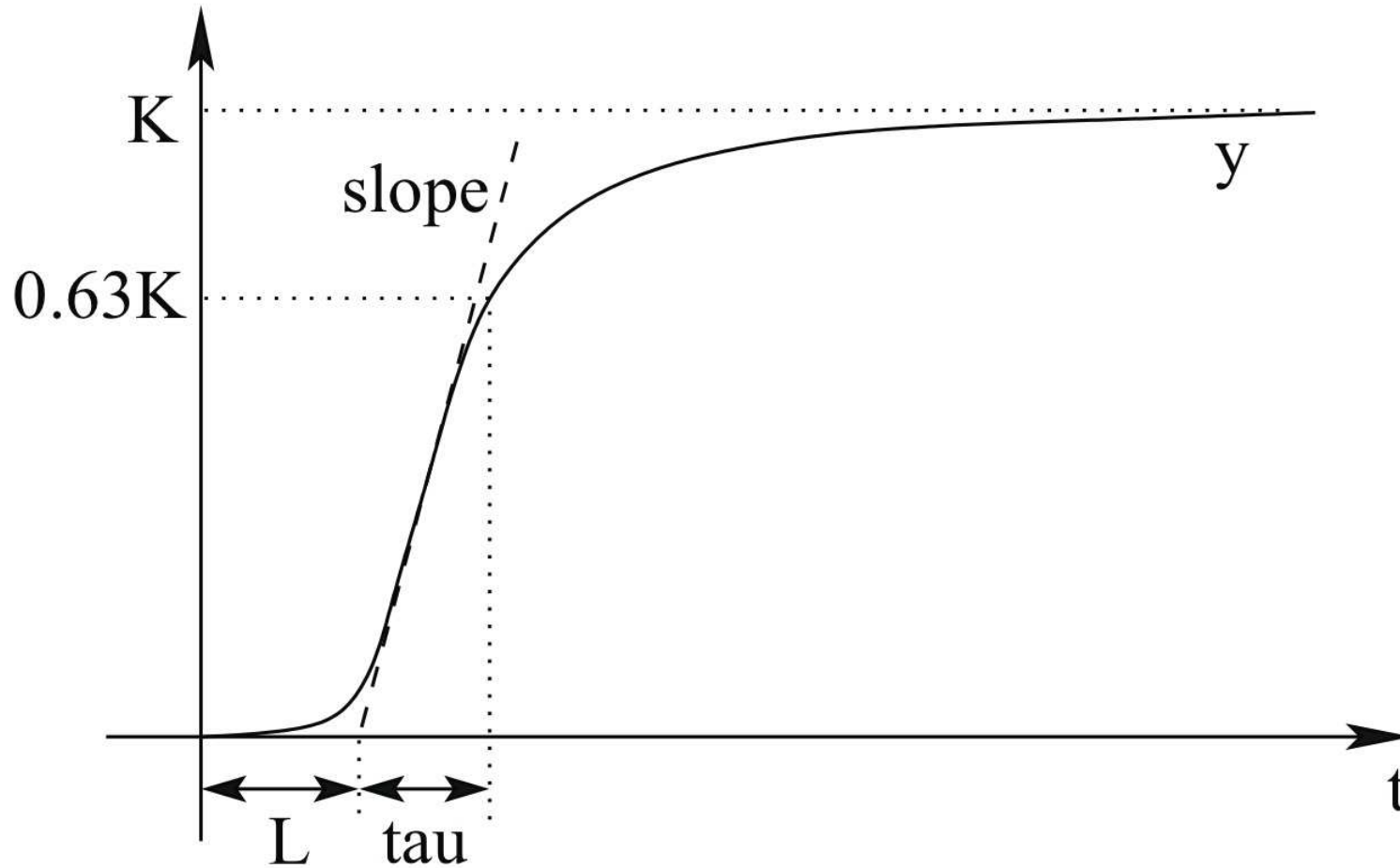
$$y(k) = \frac{B(q)}{F(q)}u(k) + \frac{C(q)}{D(q)}e(k)$$

# ARX, ARMAX, OE, BJ Models

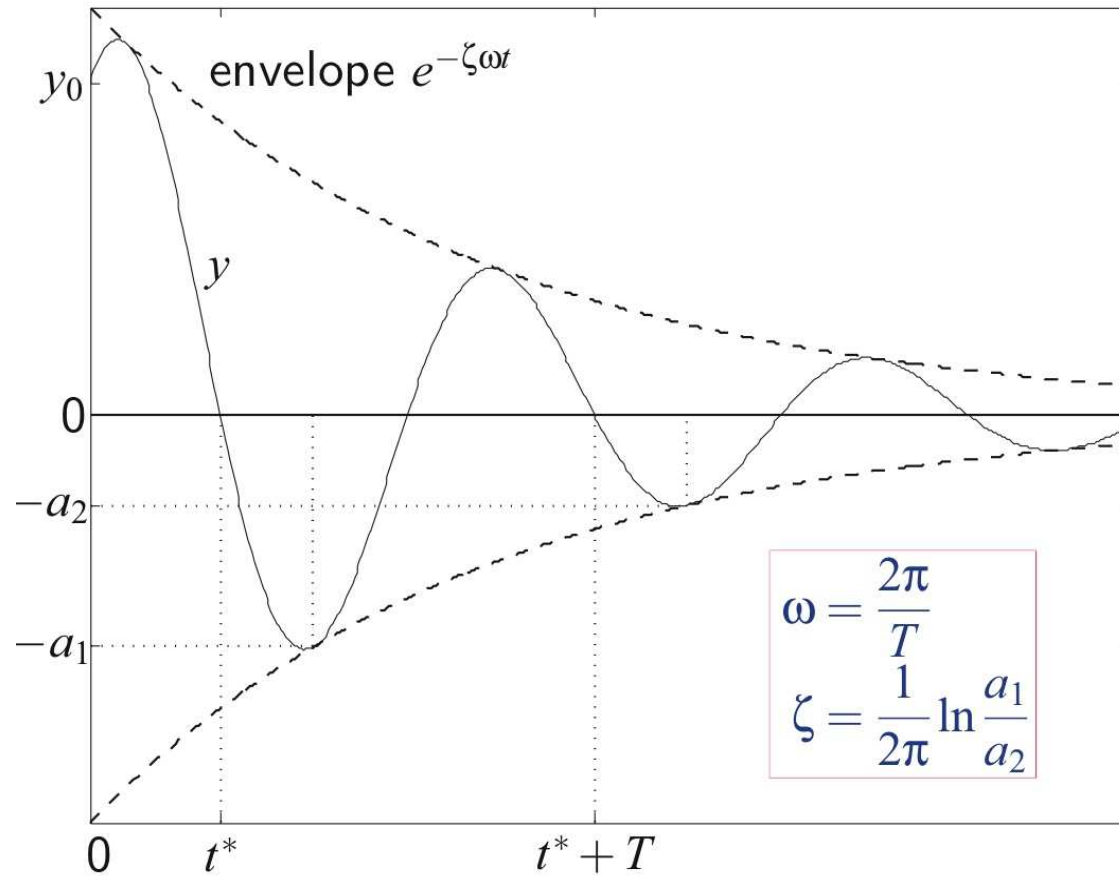
- Simulate parameterized model (parameters:  $a_i, b_i, c_i, \dots$ )
- Obtain real data from experiments
- Optimize over parameters

# Identification of 1st-order models

→ step response ( $y(t) \approx K(1 - \exp(-(t - L)/\tau))$ )

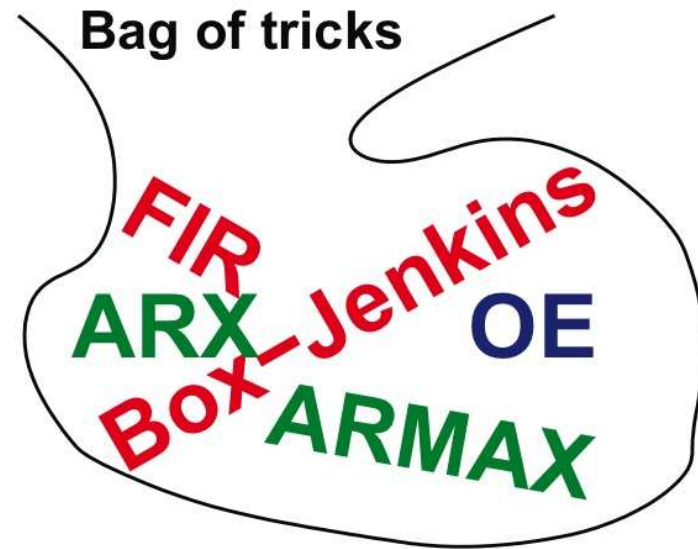


# 2nd-order models



$$\omega = \omega_N \sqrt{1 - \zeta^2}$$

# Trouble with prediction error methods



Alternative: state-space (subspace) identification

# Physical versus black-box models

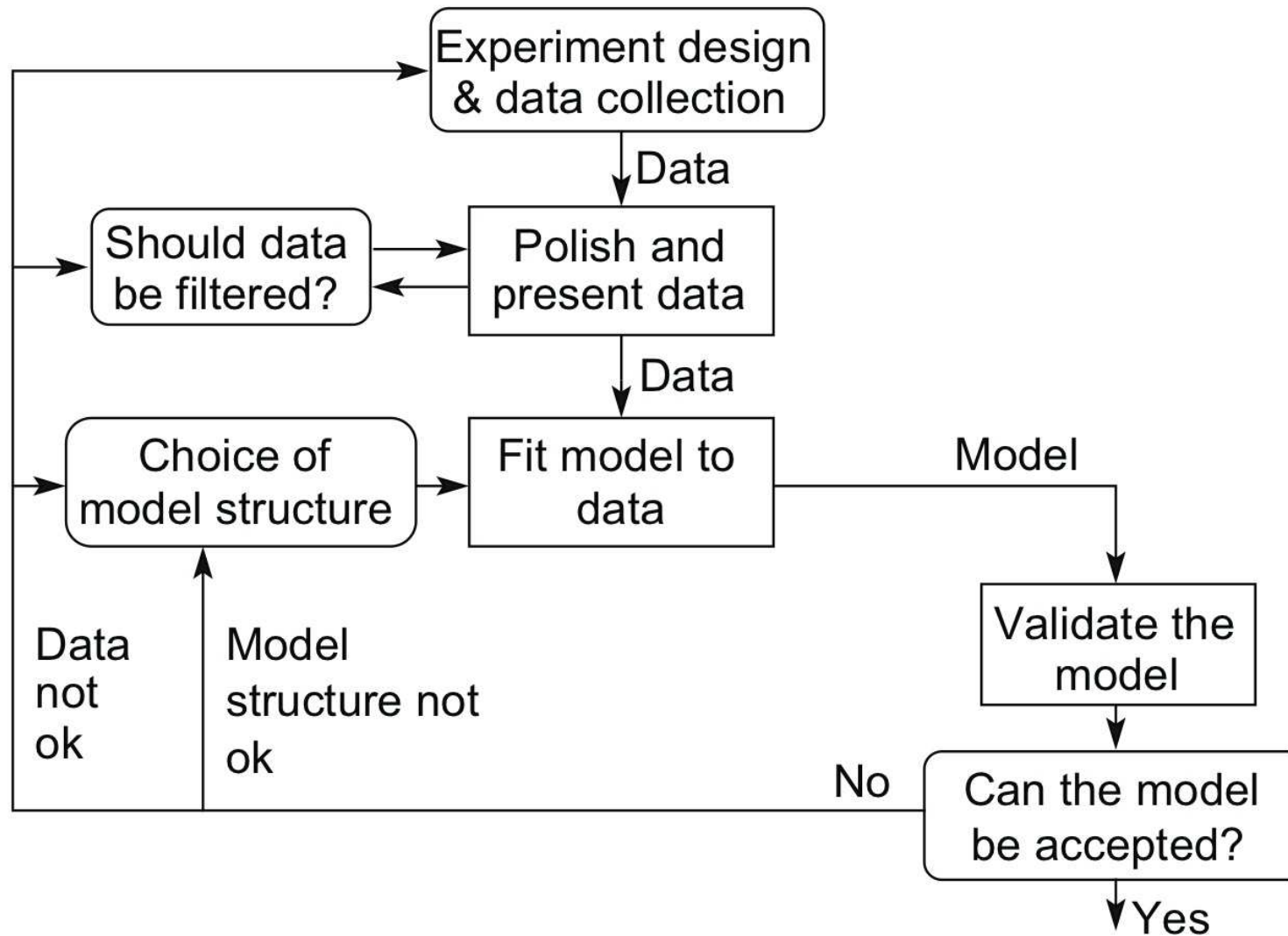
- **Physical models**

- + general (Nonlinear models)
- + based on physical considerations
  - nonlinear optimization → local minima, time-consuming

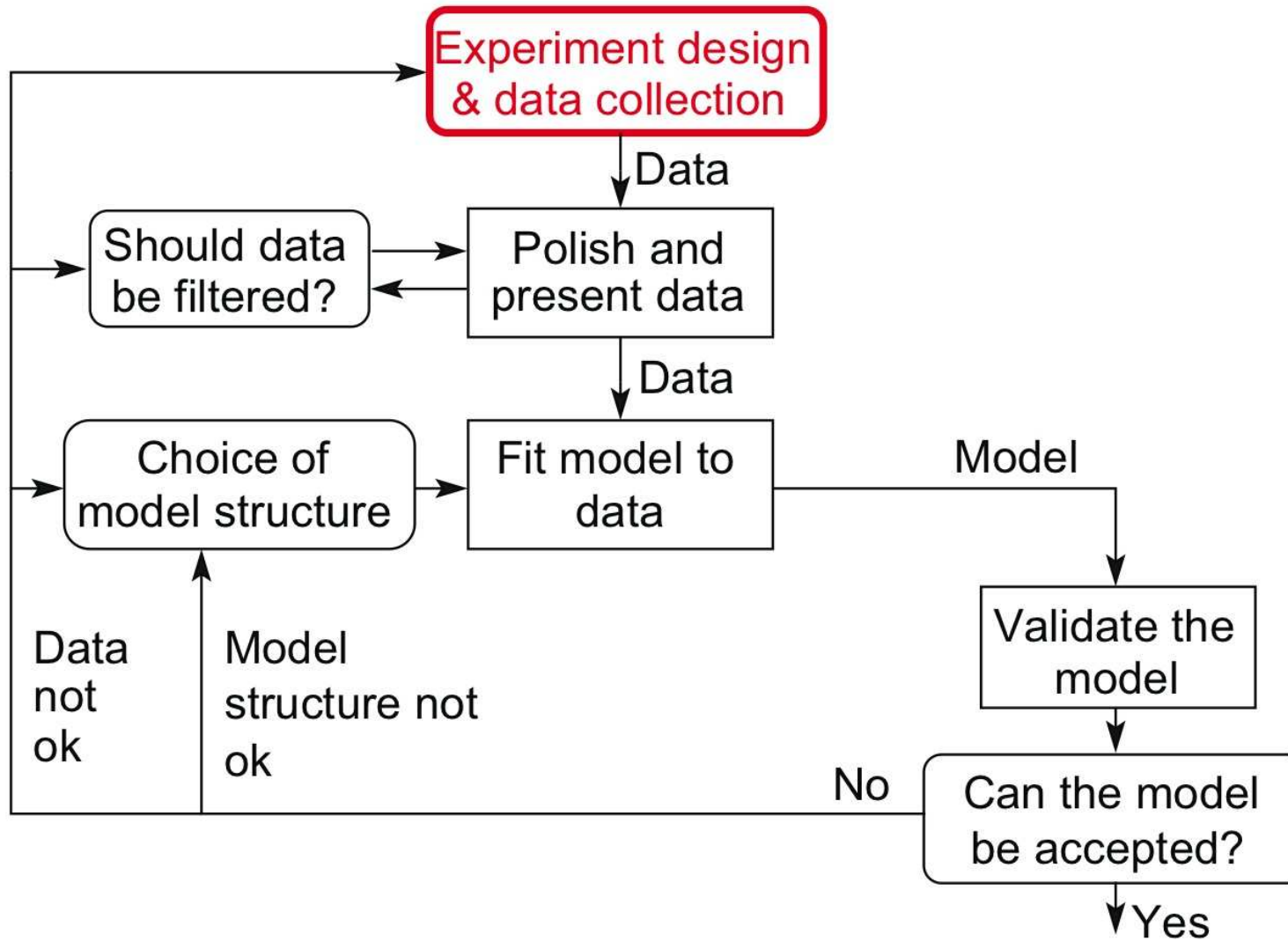
- **black-box models**

- + often very effective
- + intuitive behavior (in particular for 1st & 2nd order models)
- + identification is fast
- + many control design methods for linear models
  - black-box → relation with physical system?
  - linear → not general, only locally valid

# The identification procedure



# Experiment design



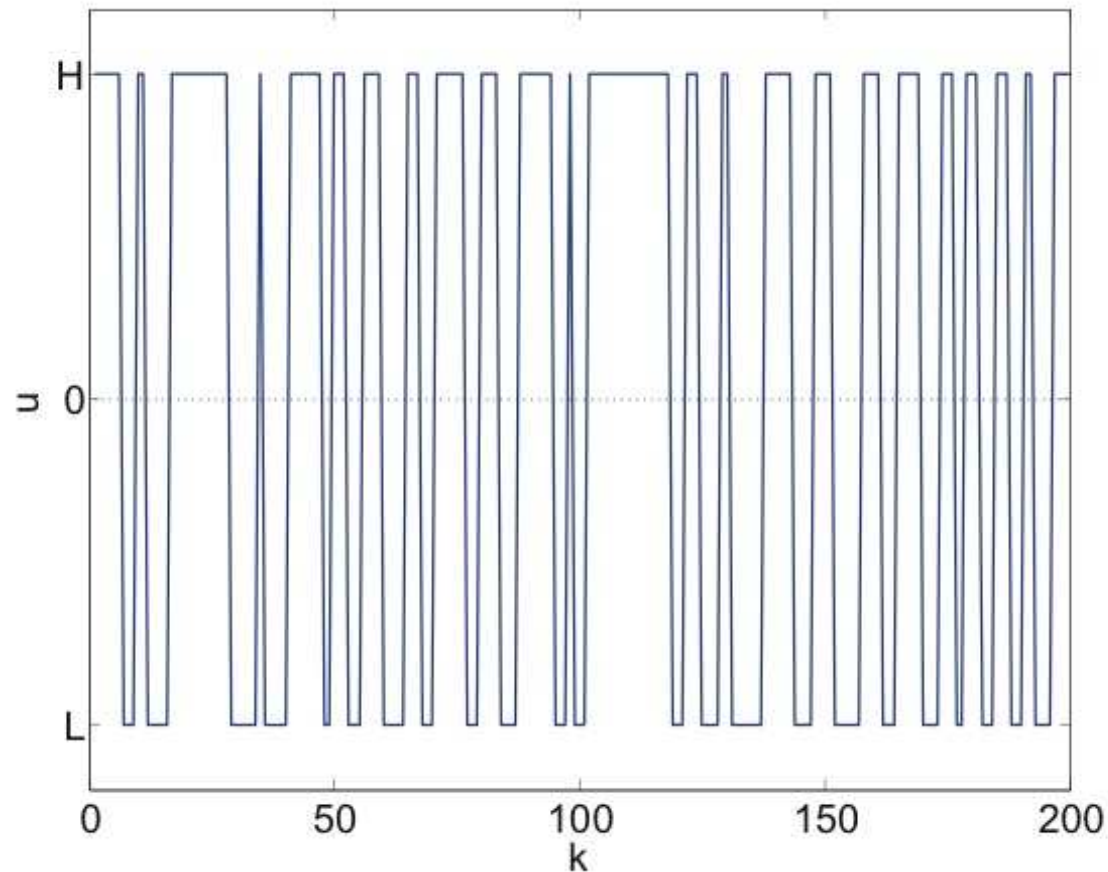
# Experiment design

Experiment design comprises a.o.:

- experiment duration:  $N$  data points
- selection of frequency band of interest
- selection of sampling frequency  $\frac{1}{h}$  Hz
- use of anti-aliasing filters

# Choice of input signal

- Input signal must excite system, i.e., contain enough frequencies
- Telegraph signal  $\rightarrow$  random shift between two levels



# Choice of input signal

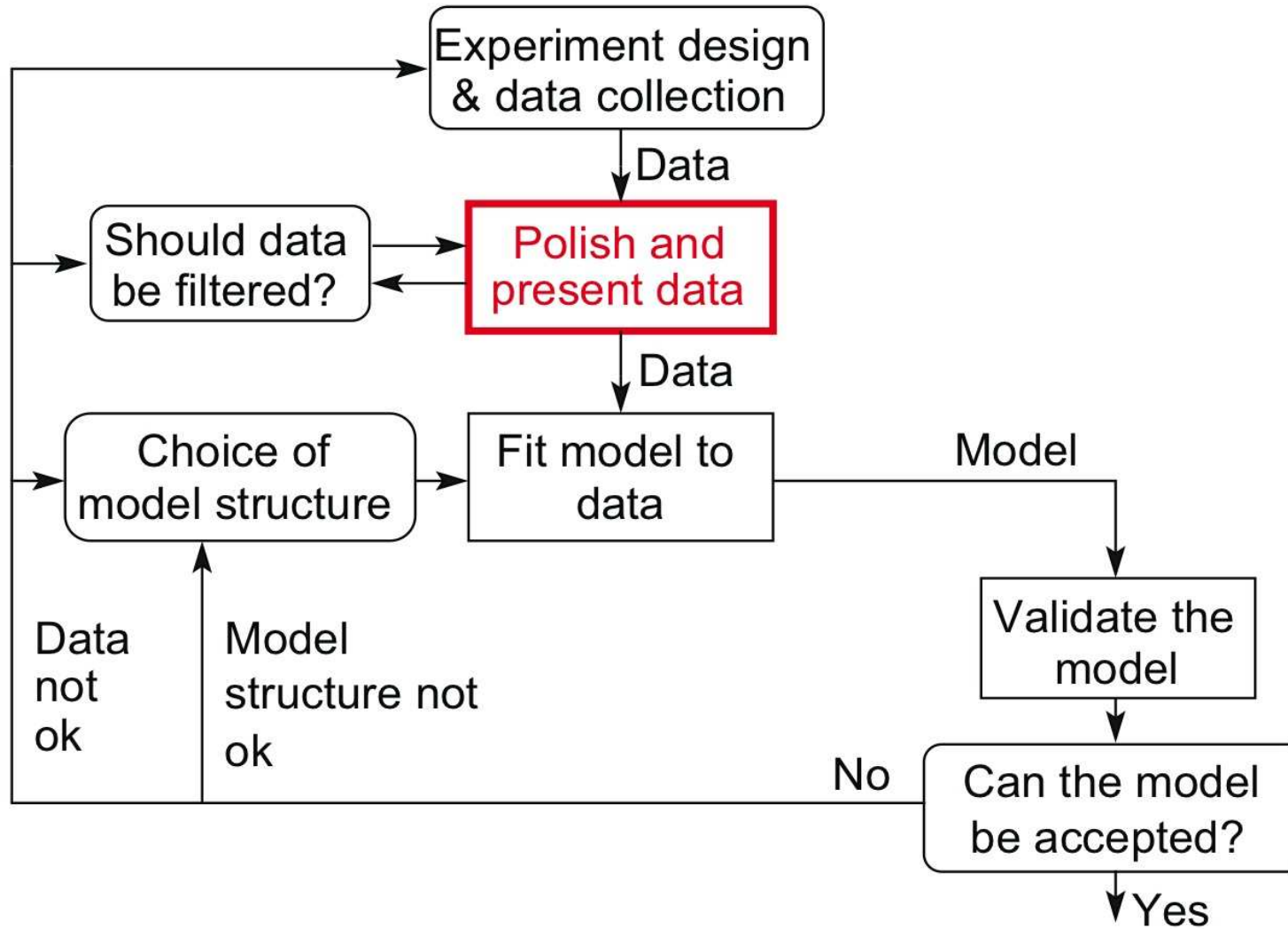
- For linear identification: do not use full range of input signals!  
→ relatively small amplitude around an operating point
- For nonlinear identification: two levels (H, L) may not be sufficient  
→ use multiple levels or smoothly varying signals (multi-sine)!
- Take care that output signal amplitude is significantly larger than noise amplitude → signal-to-noise ratio should be large enough  
Check: measure output with output under zero input to set output noise level

# Choice of input signal

## Summary

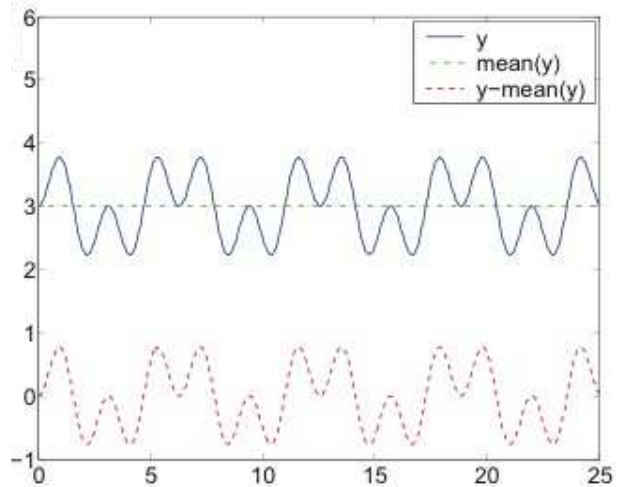
- It is good idea to first generate the input sequence off-line, and to examine its properties before applying it to system  
→ first try it with the simulation model!
- Binary signals (telegraph, PRBN) often suitable to identify linear systems  
For nonlinear models → multi-level telegraph signal
- Choose frequency range of input signal so most of its energy is situated in frequency bands of interest, i.e., let input signal contain pulses that
  - occasionally allow step response to more or less settle, and
  - also clearly excite interesting fast modes in the system.

# Post-treatment of data

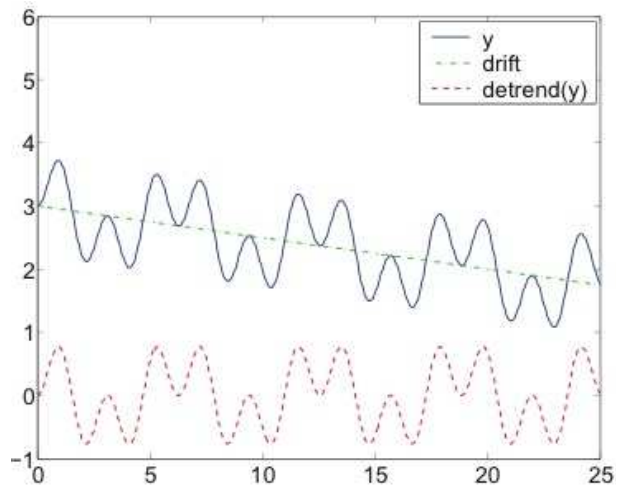


# Post-treatment of data

- Subtract mean (for identification around operating point)

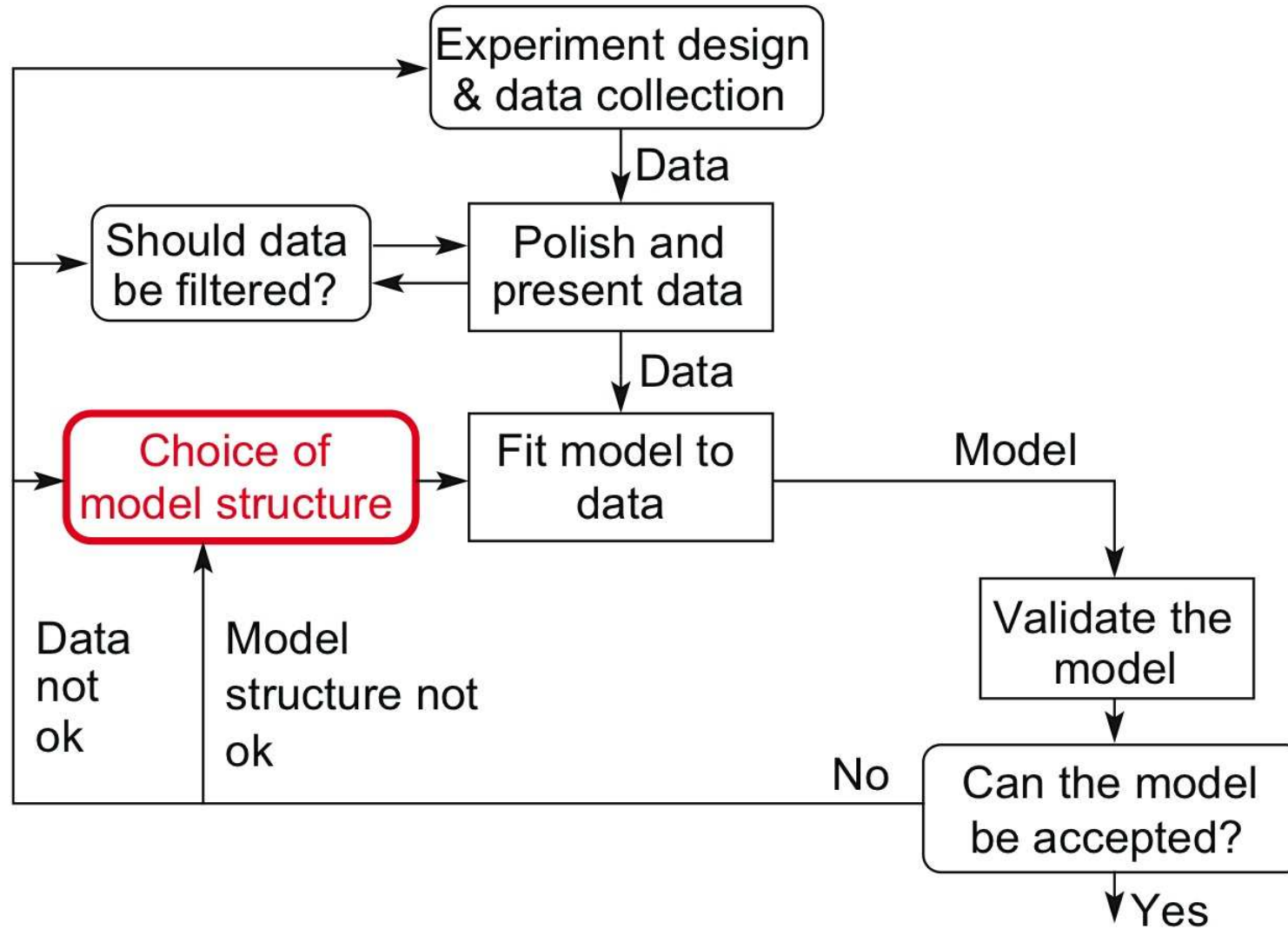


- Removing drift: `detrend(y)`



- If a filter for identification used  $\rightarrow$  use it also in control design

# Choice of model class and structure



# Choice of model class and structure

To answer selection issues prior knowledge is required:

- from “crude” first-principles models
- from simple pre-identification experiments (step response, etc.)



- time constants
- which relationships can be described as static?
- characteristics (oscillatory, poorly damped, monotone, ...)

Also useful for model validation.

# Choice of linear model class/structure

Try Simple Things First

---

ARX simple to determine (prediction error is linear in parameters)

high order necessary to model noise, unrealistic

---

OE nonlinear optimization

realistic for (white) sensor noise

---

BJ nonlinear optimization

most general, most difficult to compute

---

# Rules of thumb used in Matlab Toolbox

1. FIR model for estimation of delay  $L$ :

$$y(k) = G(q^{-1})q^{-L}u(k) + e(k)$$

2. Large-order ARX model:

$$\hat{y}(k) = \frac{\hat{B}(q^{-1})}{\hat{A}(q^{-1})}u(k) + \frac{1}{\hat{A}(q^{-1})}e(k)$$

to approximate

$$y(k) = \frac{B(q^{-1})}{A(q^{-1})}u(k) + \frac{C(q^{-1})}{D(q^{-1})}e(k) = \frac{B(q^{-1})}{A(q^{-1})}u(k) + \frac{1}{\frac{D(q^{-1})}{C(q^{-1})}}e(k)$$

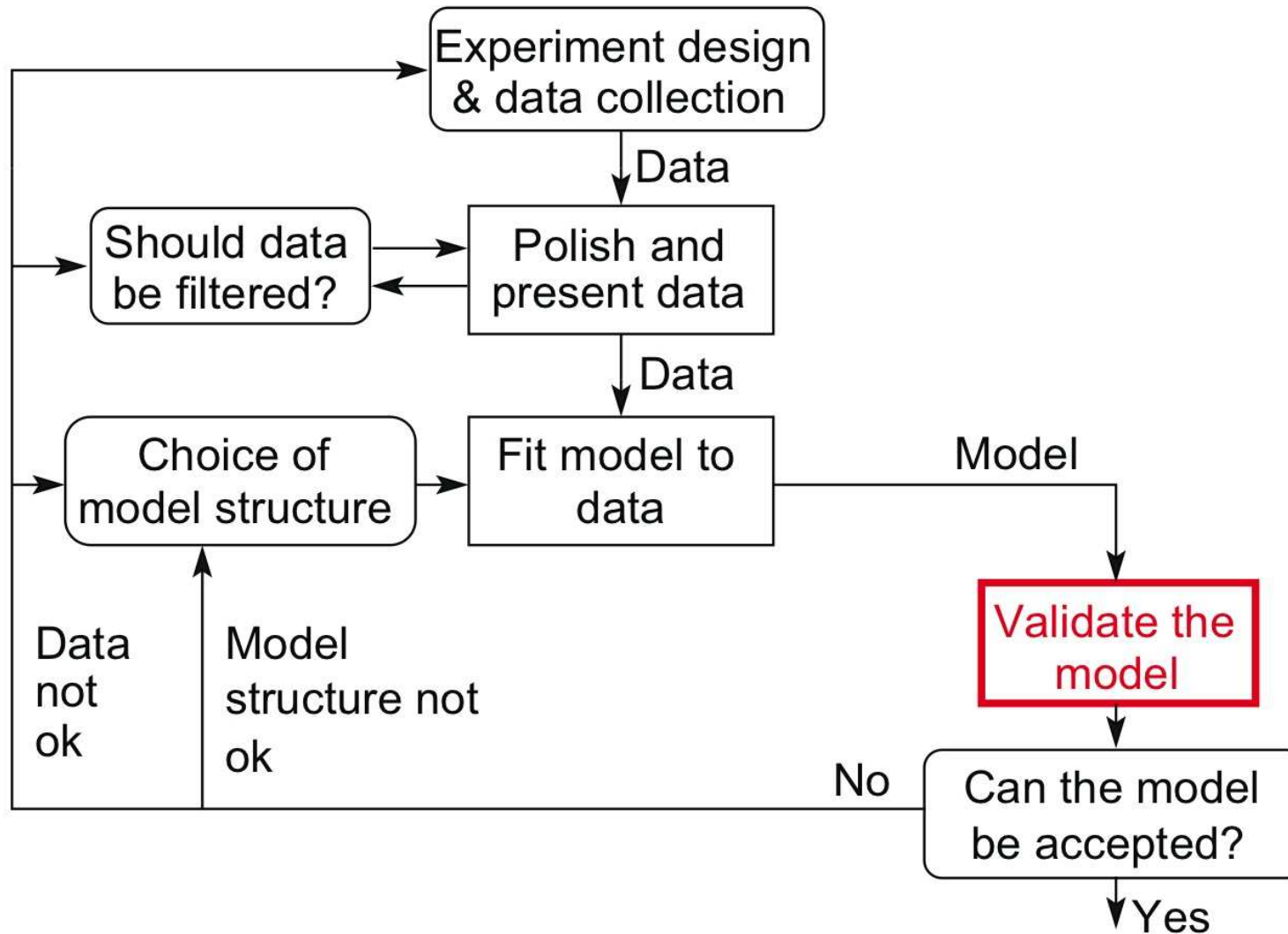
pole-zero cancellation in  $A(q^{-1})$  and  $B(q^{-1})$

3. Estimate OE (ARMAX, BJ) for the order determined in 2.

One can also test a number of model structures (selstruc)

# Validation

## Validation



# Model errors

- Variance error

- due to noise influence

- can be reduced by using longer measurement sequences:

$$E[(\hat{\theta}_N - \theta_0)(\hat{\theta}_N - \theta_0)^T] \propto \frac{\lambda}{N}$$

- with  $\lambda$  : noise variance

- $N$  : # data samples

- $\theta_0$  : “true” parameters

- Bias error

- due to deficiencies in model structure  
(present even if noise-free or  $N \rightarrow \infty$ )

# Over-fitting

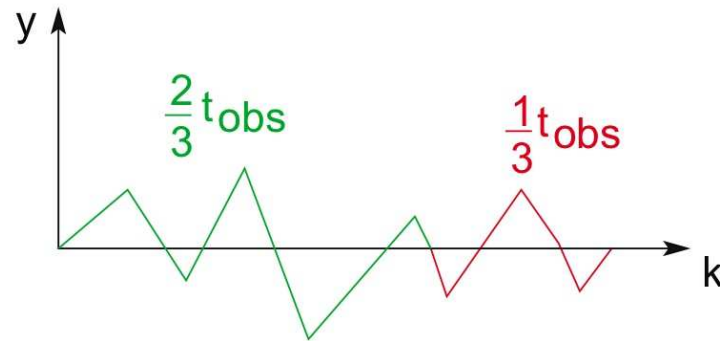
Example: Consider  $\{y_0, y_1, y_2, y_3\}$  from system  $y_k = ay_{k-1} + e_k$ . Then we can exactly fit model  $y_k = \theta y_{k-1} + \gamma y_{k-2}$  to the data.

Cures to avoid problem of over-fitting:

1. **Cross-validation:** e.g.,

$$\theta^\dagger = \arg \min J_{[0, \frac{2}{3}t_{obs}]}(\theta)$$

while model quality is given by  $J_{[\frac{2}{3}t_{obs}, t_{obs}]}(\theta)$



# Over-fitting

2. Penalize the number of parameters (e.g. Akaike):

$$J_N(\theta) = \left(1 + \frac{2d}{N}\right) \sum_{k=1}^N \varepsilon^2(k, \theta)$$

with  $d$  the total number of model parameters

# Useful Matlab commands

- Model estimation: `arx`, `oe`, `bj`, `armax`
- Filter design: `butter`  
Filtering: `filter`
- `d2c` → poles, gain  
(compare with white-box model)
- `step`, `bode` → validation, check with prior knowledge / experiments

# Linearization of nonlinear models

$$\dot{x}_{nl} = f(x_{nl}, u_{nl})$$

$$y_{nl} = g(x_{nl}, u_{nl})$$

Choose  $x_0$ ,  $y_0$  or  $u_0$  such that  $0 = f(x_0, u_0)$  and  $y_0 = g(x_0, u_0)$  (equilibrium). Linearize the above model:

$$\dot{x} = \left. \frac{\partial f}{\partial x_{nl}} \right|_{x_{nl}=x_0} (x_{nl} - x_0) + \left. \frac{\partial f}{\partial u_{nl}} \right|_{u_{nl}=u_0} (u_{nl} - u_0)$$

$$y_{nl} - y_0 = \left. \frac{\partial g}{\partial x_{nl}} \right|_{x_{nl}=x_0} (x_{nl} - x_0) + \left. \frac{\partial g}{\partial u_{nl}} \right|_{u_{nl}=u_0} (u_{nl} - u_0)$$

# Linearization of nonlinear models

$$\dot{x} = \left. \frac{\partial f}{\partial x_{nl}} \right|_{x_{nl}=x_0} (x_{nl} - x_0) + \left. \frac{\partial f}{\partial u_{nl}} \right|_{u_{nl}=u_0} (u_{nl} - u_0)$$

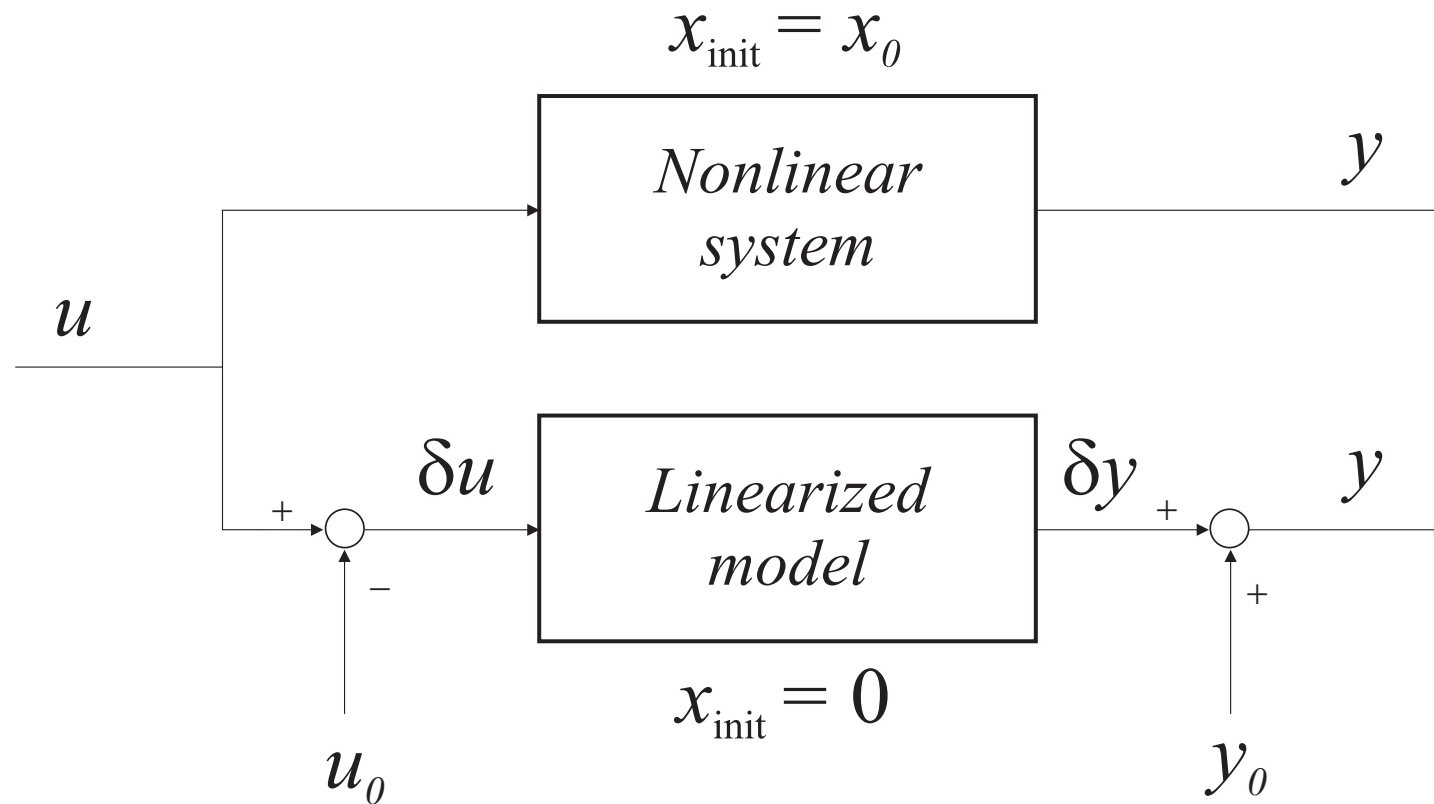
$$y_{nl} - y_0 = \left. \frac{\partial g}{\partial x_{nl}} \right|_{x_{nl}=x_0} (x_{nl} - x_0) + \left. \frac{\partial g}{\partial u_{nl}} \right|_{u_{nl}=u_0} (u_{nl} - u_0)$$

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

with  $x = x_{nl} - x_0$ ,  $u = u_{nl} - u_0$ ,  $y = y_{nl} - y_0$ , etc.

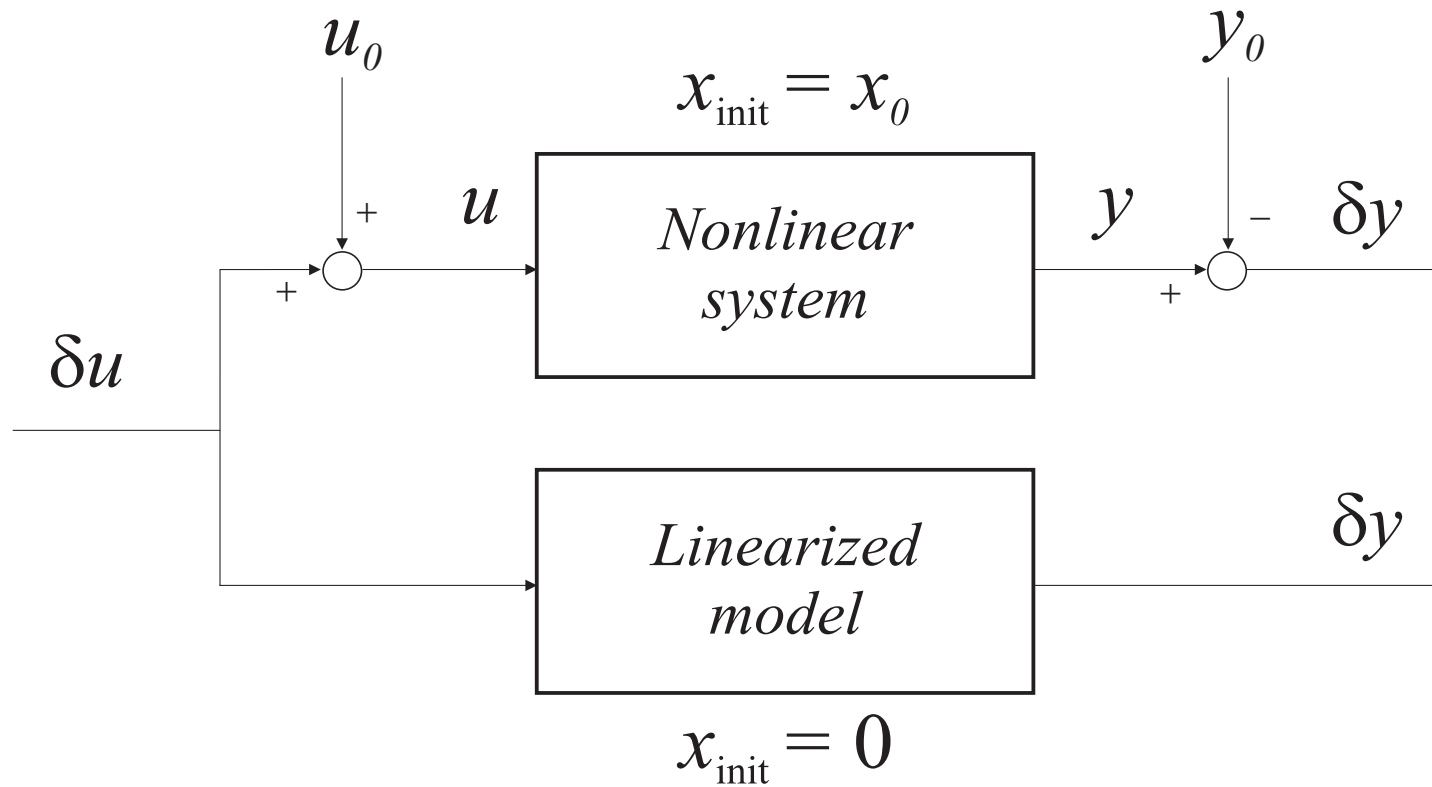
# Comparison of models through simulation



$$u = u_{nl} - u_0$$

$$y_{nl} = y + y_0$$

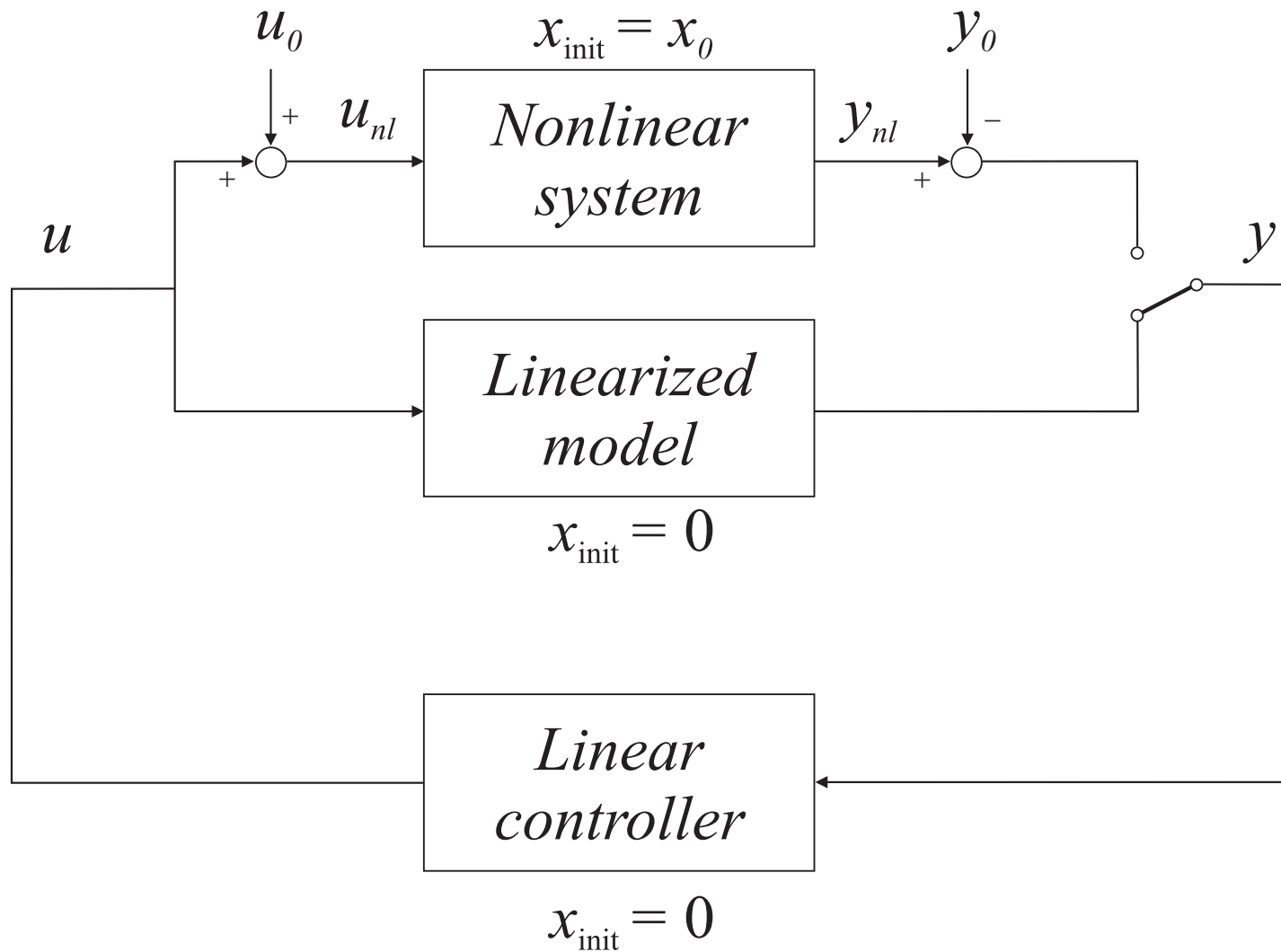
# Alternatively ...



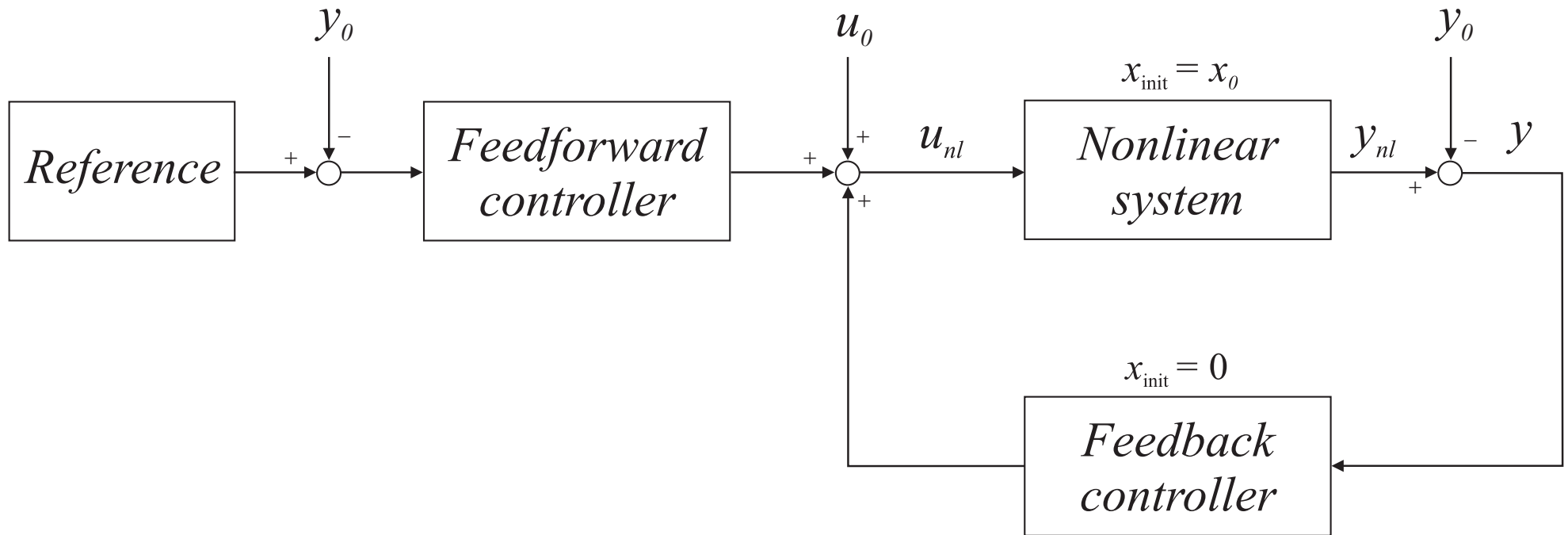
$$u_{nl} = u + u_0$$

$$y = y_{nl} - y_0$$

# Control by local linear controller



# Two-degree-of-freedom control



Linear controllers must use signals  $u$ ,  $y$ ,  $x$ !