

# System Identification

Control Engineering EN, 3<sup>rd</sup> year B.Sc.  
Technical University of Cluj-Napoca  
Romania

Lecturer: Lucian Buşoniu



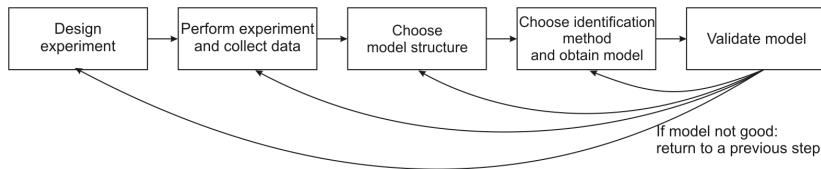
# Part X

## Model validation and practical issues

# Table of contents

- 1 Model validation with correlation tests
  - Introduction
  - Correlation tests
  - Matlab example
- 2 Structure selection and avoiding overparametrization
- 3 Other practical issues

# Recall: Importance of validation



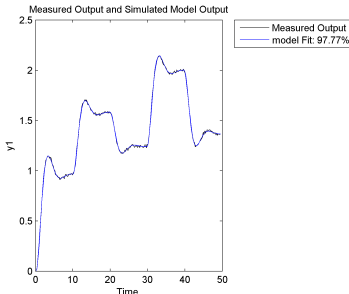
Model validation is a crucial step: the model must be good enough (for its intended usage).

If validation is unsuccessful, previous steps in the workflow must be redone, for instance:

- Rerun the identification algorithm with different parameters (e.g.  $\delta$  in recursive methods).
- Change the model structure: e.g. orders of polynomials  $na$ ,  $nb$  in ARX, or even the model type entirely, e.g. IV instead of ARX
- Design and run a new experiment (e.g. more data, different input signal)

# Motivation

So far, we validated and selected models mostly informally, by examining plots or comparing errors – using *common sense*.



Next, some mathematically well-founded tests will be given.

However, common sense remains indispensable – mathematical tests work under assumptions that may not always be satisfied.

# Focus: Prediction error methods

We focus on *single-output* models obtained by *prediction error methods*.

Some of the tests can be extended to other settings.

# Table of contents

- 1 **Model validation with correlation tests**
  - Introduction
  - **Correlation tests**
  - Matlab example
- 2 Structure selection and avoiding overparametrization
- 3 Other practical issues

# Whiteness: Intuition

Recall general PEM model structure:

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

where  $e(k)$  is *assumed* to be zero-mean white noise.

PEM are derived so that the prediction error  $\varepsilon(k) = y(k) - \hat{y}(k) = e(k)$ . If the system satisfies the model structure (so the assumption holds), and moreover if the model is accurate, then  $\varepsilon(k)$  is also zero-mean white noise.

## Whiteness hypothesis

(W) The prediction errors  $\varepsilon(k)$  are zero-mean white noise.



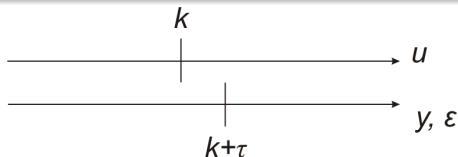
# Independence of past inputs: Intuition

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

If the model  $G$  is accurate, it entirely explains the influence of inputs  $u(k)$  on current and future outputs  $y(k + \tau)$ . Therefore, the errors  $\varepsilon(k + \tau) = y(k + \tau) - \hat{y}(k + \tau)$  are only influenced by the disturbances  $v$ , and are *independent* of inputs  $u(k)$ .

## Independence hypothesis 1

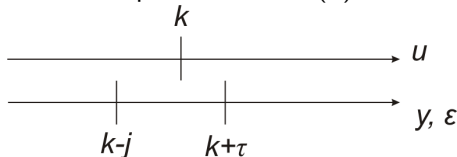
(I1) The prediction errors  $\varepsilon(k + \tau)$  are independent of inputs  $u(k)$  for  $\tau \geq 0$  (i.e., current and future errors are independent of current inputs).



# Independence of all inputs: Intuition

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

If the experiment is closed-loop,  $u(k)$  depends on past outputs and this will lead to a correlation of past errors  $\varepsilon(k + \tau)$ ,  $\tau < 0$  with  $u(k)$  (note the independence for  $\tau \geq 0$  is not affected). If open-loop, then  $\varepsilon(k + \tau)$ ,  $\tau < 0$  is also independent from  $u(k)$ .



## Independence hypothesis 2

(I2) The prediction errors  $\varepsilon(k + \tau)$  are independent of  $u(k)$  for any  $\tau$  (i.e., all the errors are independent of all the inputs).

# All hypotheses

- (W) The prediction errors  $\varepsilon(k)$  are zero-mean white noise.
- (I1) The prediction errors  $\varepsilon(k + \tau)$  are independent of  $u(k)$  for  $\tau \geq 0$  (current and future errors are independent of current inputs).
- (I2) The prediction errors  $\varepsilon(k + \tau)$  are independent of  $u(k)$  for *any*  $\tau$  (all the errors are independent of all the inputs).

A good model should satisfy (W) and (I1), and if there is no feedback, also (I2).

We will develop tests that allow to either accept or reject these hypotheses for a given model, and therefore validate or reject the model.

# Whiteness: Correlations

Recall the correlation function (equal to the covariance in zero-mean case):

$$r_{\varepsilon}(\tau) = \mathbb{E} \{ \varepsilon(k + \tau) \varepsilon(k) \}$$

If  $\varepsilon(k)$  is zero-mean white noise:

- The correlation function is zero,  $r_{\varepsilon}(\tau) = 0$  for any nonzero  $\tau$ .
- At zero,  $r_{\varepsilon}(0)$  is the variance  $\sigma^2$  of the white noise.

# Whiteness: Correlations from data

Estimating correlations from data:

$$\hat{r}_\varepsilon(\tau) = \frac{1}{N} \sum_{k=1}^{N-\tau} \varepsilon(k + \tau) \varepsilon(k)$$

Normalization by the (estimated) variance:

$$x(\tau) = \frac{\hat{r}_\varepsilon(\tau)}{\hat{r}_\varepsilon(0)}$$

# Whiteness test

In practice,  $x(\tau)$  will never be zero for finite data, so instead we check if it is small for nonzero  $\tau$ . For statistical reasons we do not go into, we impose a cutoff at  $\frac{1.96}{\sqrt{N}}$

## Whiteness test

If  $|x(\tau)| \leq \frac{1.96}{\sqrt{N}}$  for all  $\tau \neq 0$  supported by the data, then the whiteness hypothesis (W) is accepted. Otherwise, (W) is rejected.

# Independence: Correlations

To verify independence of  $\varepsilon$  from  $u$ , use *cross-correlation* function:

$$r_{\varepsilon u}(\tau) = \mathbf{E} \{ \varepsilon(k + \tau) u(k) \}$$

- 1 If (I1) is true, then  $r_{\varepsilon u}(\tau) = 0$  for  $\tau \geq 0$ .
- 2 If (I2) is true, then  $r_{\varepsilon u}(\tau) = 0$  for any  $\tau$ .

Estimation from data and normalization:

$$\hat{r}_{\varepsilon u}(\tau) = \begin{cases} \frac{1}{N} \sum_{k=1}^{N-\tau} \varepsilon(k + \tau) u(k) & \text{if } \tau \geq 0 \\ \frac{1}{N} \sum_{k=1-\tau}^N \varepsilon(k + \tau) u(k) & \text{if } \tau < 0 \end{cases}$$

$$x(\tau) = \frac{\hat{r}_{\varepsilon u}(\tau)}{\sqrt{\hat{r}_{\varepsilon}(\tau) \hat{r}_u(\tau)}}$$

# Independence test at $\tau$

## Independence tests

If  $|x(\tau)| \leq \frac{1.96}{\sqrt{N}}$ ,  $\forall \tau \geq 0$  supported by the data, then the independence hypothesis (I1) is accepted.

If the condition holds  $\forall \tau$  supported by the data (including negative  $\tau$ ), then (I2) is also accepted.

If the model is accurate (I1 holds), then checking the condition at  $\tau < 0$  (I2) verifies the presence of feedback.



# Correlation tests: Overall interpretation

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

- If  $W$  and  $I1$  hold, then the entire model ( $G$  and  $H$ ) is correct
- If  $I1$  holds but  $W$  fails, then  $G$  is correct but  $H$  is incorrect
- If  $I1$  holds and  $I2$  fails, there is feedback in the data. If  $I2$  also holds then there is no feedback

# Table of contents

- 1 Model validation with correlation tests
  - Introduction
  - Correlation tests
  - Matlab example
- 2 Structure selection and avoiding overparametrization
- 3 Other practical issues

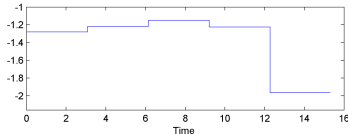
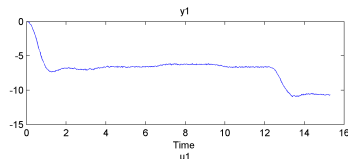
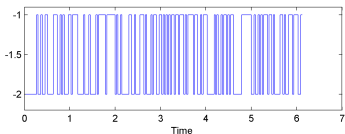
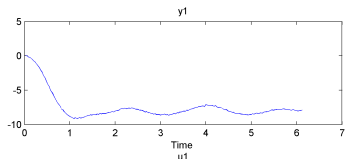
# Matlab example: Experimental data

The real system is in output-error form:

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

and has order  $n = 3$ .

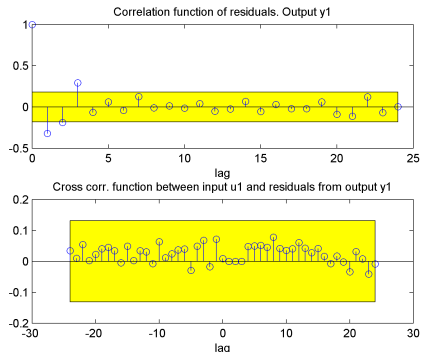
`plot(id);` and `plot(val);`



# Matlab: ARX model

First, we try an ARX model:

```
mARX = arx(id, [3, 3, 1]); resid(mARX, id);
```

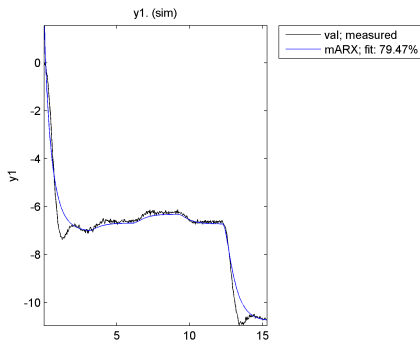


The whiteness test (W) fails, and the model is rejected. This is because the system is not within the model class.

As I1 holds, we conclude that the input-output model  $G$  is good, but the noise model  $H$  is wrong and we should work to improve that part.

# Matlab: ARX model (continued)

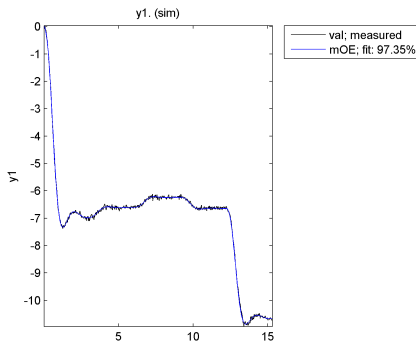
Simulating on the validation data confirms the fact that the model is poor.





# Matlab: OE model (continued)

Simulating the model on the validation data confirms the model has good quality.

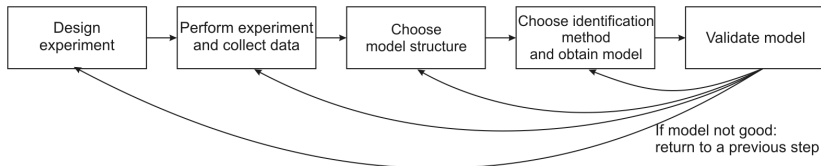


# Table of contents

- 1 Model validation with correlation tests
- 2 Structure selection and avoiding overparametrization**
  - Structure selection
  - Overparametrization
- 3 Other practical issues



# Structure selection in workflow



While we nearly always tuned the model structure (e.g. type, orders, length), the criteria for doing so were often informal.

Next, we discuss structure selection in a formal way.

Consider we are given several model structures  $\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_\ell$ .

**Example:** ARX structures of increasing order.

**How to choose among them?**

First idea: choose  $\mathcal{M}_i$  leading to the smallest mean squared error:

$$V(\hat{\theta}) = \frac{1}{N} \sum_{k=1}^N \varepsilon(k)^2$$

This does not take into account the complexity of the model, so it ignores the computational effort needed to identify and simulate it, and also has a risk of overfitting. We explore other options that do consider model complexity (without going into their derivation).

# Akaike's information criterion (AIC)

$$W_{\text{AIC}} = N \log V(\hat{\theta}) + 2p, \text{ or equivalently: } \log V(\hat{\theta}) + \frac{2p}{N}$$

where  $N$  is the number of data points and  $p$  the number of parameters (e.g.,  $na + nb$  in ARX).

**Choice:** Model with smallest  $W_{\text{AIC}}$ .

**Intuition:**

- The term  $2p$  penalizes the complexity of the model (number of parameters).
- Division by the number  $N$  of data points in  $2p/N$  takes into account that more data allows more parameters to be identified.
- Taking the logarithm of the MSE allows to better differentiate between small values of the MSE.

# Final prediction error (FPE)

$$W_{\text{FPE}} = V(\hat{\theta}) \frac{1 + p/N}{1 - p/N}$$

**Choice:** Model with smallest  $W_{\text{FPE}}$ .

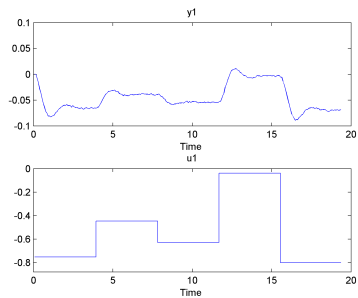
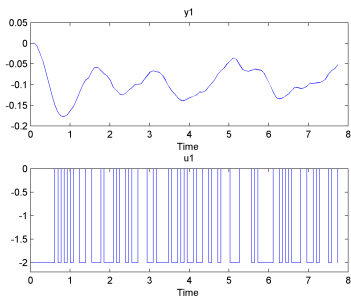
**Intuition:** When  $N$  is large:

$$V(\hat{\theta}) \frac{1 + p/N}{1 - p/N} = V(\hat{\theta}) \left(1 + \frac{2p/N}{1 - p/N}\right) \approx V(\hat{\theta}) \left(1 + \frac{2p}{N}\right)$$

The term  $\frac{2p}{N}$  works like before, but now it *multiplies* the MSE rather than getting added.

# Matlab example

An OE system with  $n = 2$ .



# Matlab: selstruc with AIC

Recall `arxstruc`:

```
Na = 1:15; Nb = 1:15; Nk = 1:5;  
NN = struc(Na, Nb, Nk); V = arxstruc(id, val, NN);
```

- `struc` generates all combinations of orders in `Na`, `Nb`, `Nk`.
- `arxstruc` identifies for each combination an ARX model on the data `id`, simulates it on the data `val`, and returns information about the MSEs, model orders etc. in `V`.

# Matlab: `selstruc` with AIC (continued)

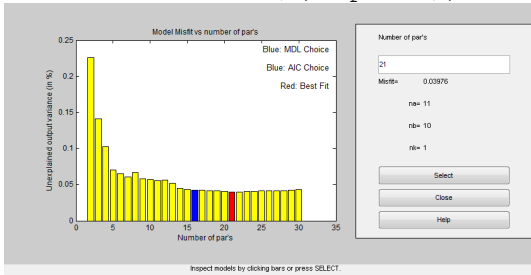
To choose the structure with the Akaike's information criterion:

```
N = selstruc(V, 'aic');
```

For our data,  $N = [8, 8, 1]$ .

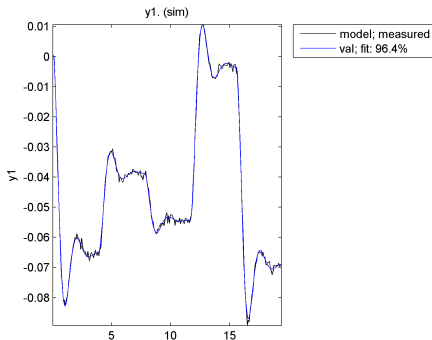
Alternatively, graphical selection also allows using AIC:

```
N = selstruc(V, 'plot');
```



Note that the best-AIC model is not (always) the same as the best-fit model!

# Matlab: Results





# Remarks

AIC, FPE also work if the system is not in the model class.

Matlab offers functions `aic`, `fpe` that compute these criteria for a list of models with any structure.

# Table of contents

- 1 Model validation with correlation tests
- 2 Structure selection and avoiding overparametrization**
  - Structure selection
  - Overparametrization**
- 3 Other practical issues

# Motivation

Consider a case where *the real system* obeys the ARMAX structure:

$$A_0(q^{-1})y(k) = B_0(q^{-1})u(k) + C_0(q^{-1})e(k)$$

where subscript 0 indicates quantities related to the real system.

This is equivalent to any model:

$$W(q^{-1})A_0(q^{-1})y(k) = W(q^{-1})B_0(q^{-1})u(k) + W(q^{-1})C_0(q^{-1})e(k)$$

with  $W(q^{-1})$  some polynomial of order  $nw$ .

So, using ARMAX identification with  $na = na_0 + nw$ ,  $nb = nb_0 + nw$ ,  $nc = nc_0 + nw$  can produce an accurate model. This model is however **too complicated** (overparametrized), and will have some nearly common factors  $W(q^{-1})$  in all polynomials (only “nearly” because of the approximate nature of the identification).

# Pole-zero cancellations

This type of situation can be identified by checking if some poles and zeros of the model (approximately) cancel each other out.

We exemplify using Matlab function `pzmap`, which shows the poles and zeros of  $G$  in the generic model:

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

For the ARMAX example,  $G(q^{-1}) = \frac{W(q^{-1})B_0(q^{-1})}{W(q^{-1})A_0(q^{-1})}$ , so the roots of  $W$  are both poles and zeros and (approximately) cancel each other out.

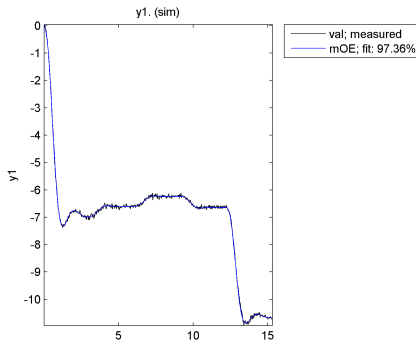
This idea extends to other model types besides ARMAX.

# Matlab: overparameterized OE model

On the same data as for correlation tests (recall system has order  $n = 3$ ):

```
mOE = oe(id, [5, 5, 1]);
```

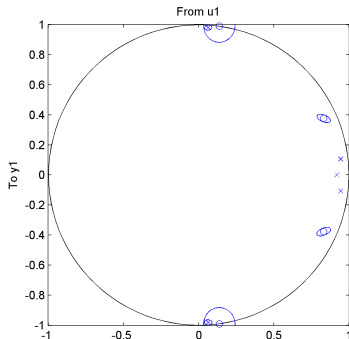
Looking at the validation data, the model is accurate:



# Matlab: testing for pole-zero cancellations

```
pzmap(mOE, 'sd', nsd);
```

Arguments 'sd', nsd specify a statistical confidence region around the poles and zeros. Here we take  $nsd=1.96$ .



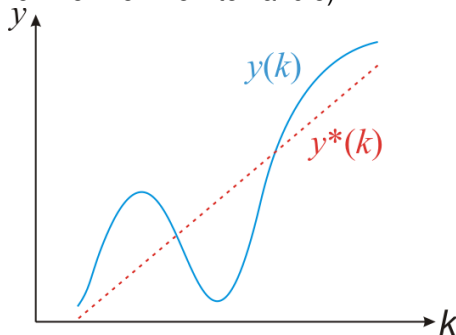
Two pairs of poles and zeros have overlapping confidence regions  $\Rightarrow$  likely they are canceling each other. This indicates the order should be 3 (the true system order).

# Table of contents

- 1 Model validation with correlation tests
- 2 Structure selection and avoiding overparametrization
- 3 Other practical issues**
  - Drifts
  - Time delays
  - Local minima
  - Outliers

# Drifts

Sometimes, the data will contain spurious slow signals called *drifts*, coming e.g. from slow disturbances (as opposed to the fast noise or disturbance, which we know how to handle)



Idea: Treat the drifts as time series, fit them with linear regression, and remove them



# Estimating drifts

- 1 Treat the input and output as separate time series, write drift models:

$$u^*(k) = \theta_1^u + \theta_2^u k + \theta_3^u k^2 + \dots + \theta_n^u k^{n-1}$$

$$y^*(k) = \theta_1^y + \theta_2^y k + \theta_3^y k^2 + \dots + \theta_n^y k^{n-1}$$

- 2 Find the parameter vectors  $\theta^u, \theta^y$  by linear regression on  $u(k), y(k)$  and compute the corresponding drifts  $u^*(k), y^*(k)$
- 3 Subtract the drifts from the data:

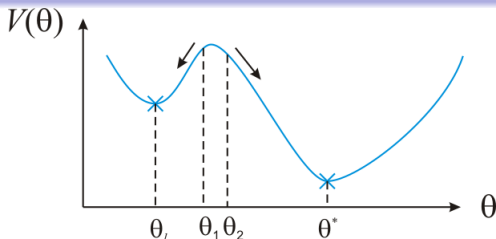
$$\bar{u}(k) = u(k) - u^*(k), \quad \bar{y}(k) = y(k) - y^*(k)$$

- 4 Identify as usual, but with “detrended” signals  $\bar{u}, \bar{y}$

Note: Removing zero-order drifts = removing the means



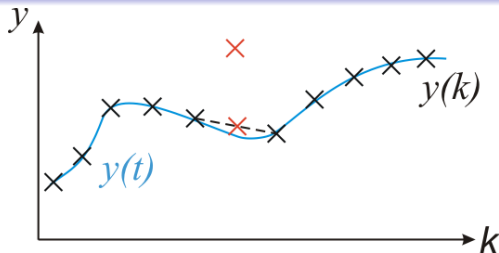
# Local minima



- Iterative optimization (needed for methods that cannot be solved as linear regression, such as ARMAX and OE) may get stuck in local minima
  - E.g. if initialized to  $\theta_1$ , Newton's method likely converges to the local minimum  $\theta_l$ . But from  $\theta_2$  it finds the global optimum  $\theta^*$ !
- ⇒ If result is bad and local minima suspected, restart the optimization from another initial parameter vector

Note: ARMAX usually converges to the global optimum; OE often to local optima except when  $u$  is white noise

# Outliers



- Sometimes, a few measurements will be wildly incorrect, due to e.g. transient malfunctions. These are called outliers
- Best tested via the prediction error  $\varepsilon$ , after finding an initial model: if  $\varepsilon(k)$  is anomalously large at some step  $k$ , an outlier is likely
- **Solution 1:** Fill in the data using e.g. the average of  $y(k-1)$  and  $y(k+1)$  (pictured), or the model prediction  $\hat{y}(k)$
- **Solution 2:** Cap the prediction error at a reasonable maximum  $\varepsilon_{\max}$ , so  $V(\theta) = \sum_{k=1}^N \min\{\varepsilon^2(k), \varepsilon_{\max}\}$