IDENTIFICATION of ARMAX SYSTEMS

First the X, then the AR, finally the MA

Jan C. Willems

K.U. Leuven

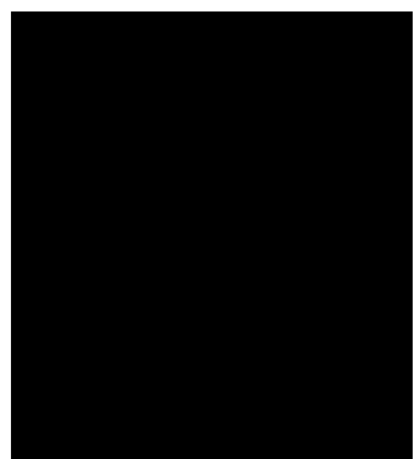
Oberwolfach Tagung Regelungstheorie **March 3, 2005**

On-going joint research with

Ivan Markovsky (K.U. Leuven)

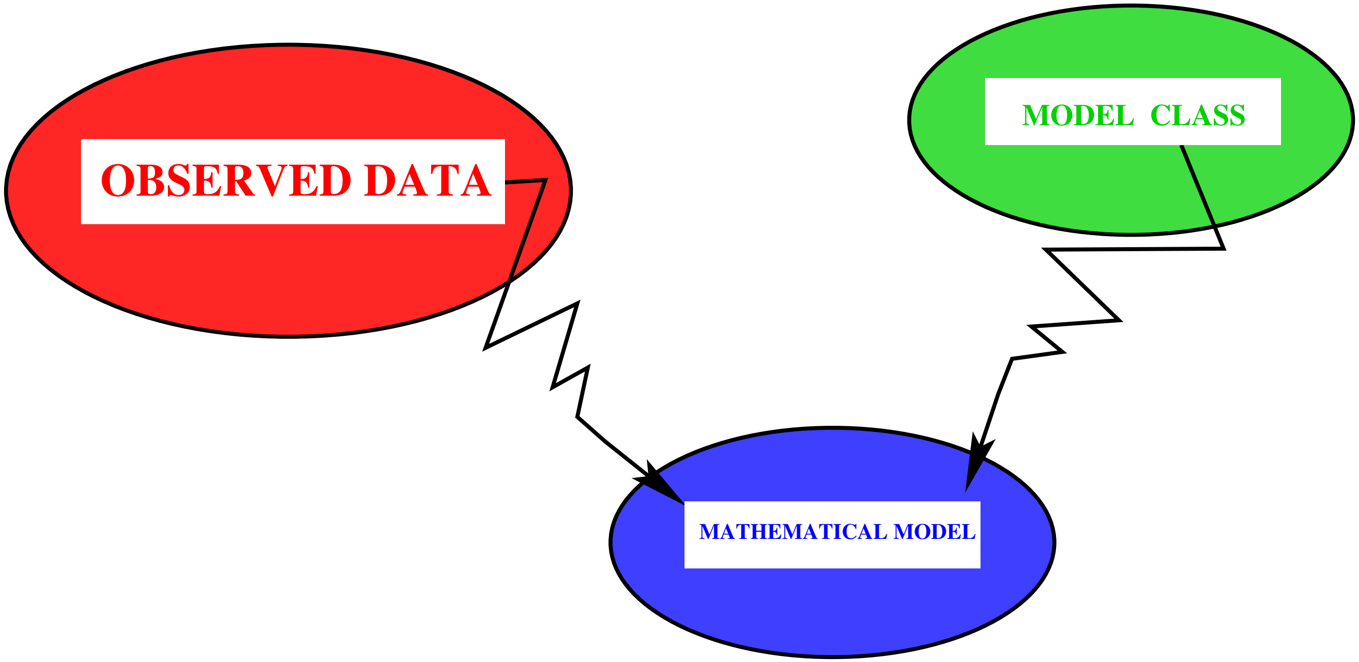
Paolo Rapisarda (Un. Maastricht)

& Bart De Moor (K.U. Leuven)



Problem

System ID



Case of interest today

Data: an 'observed' vector time-series

$$\tilde{w}(1), \tilde{w}(2), \ldots, \tilde{w}(T)$$

$w(t) \in \mathbb{R}^{\mathtt{w}}$, T finite or infinite

$$\Downarrow$$

A **dynamical model** from a model class,

e.g. a difference equation

$$R_0 w(t) + R_1 w(t+1) + \cdots + R_L w(t+L)$$

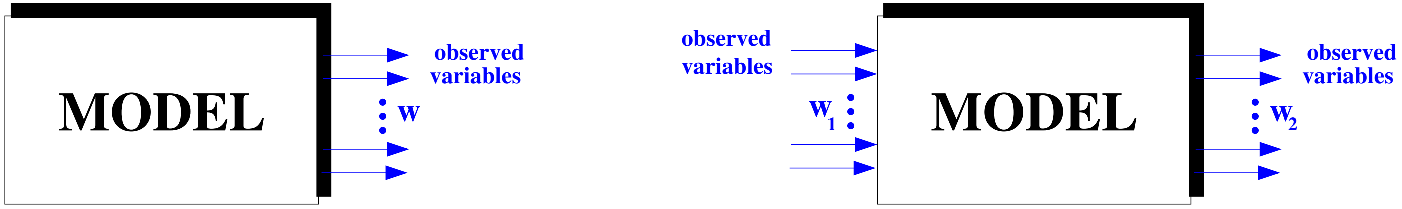
$$= 0$$

or $$= M_0 \varepsilon(t) + M_1 \varepsilon(t+1) + \cdots + M_L \varepsilon(t+L)$$

Case of interest today

We discuss 2 cases:

'deterministic' ID



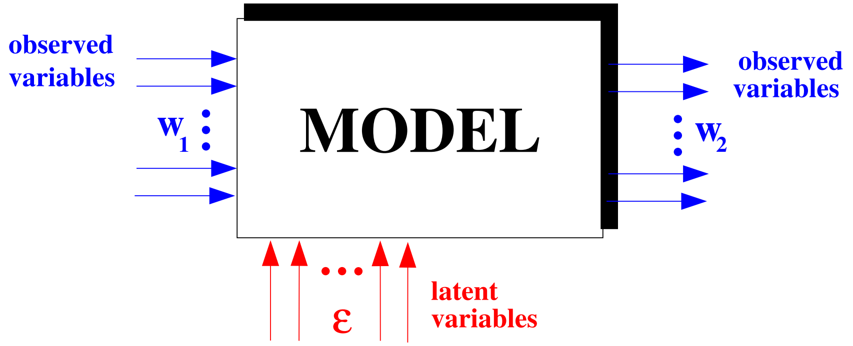
$$R_0 w(t) + R_1 w(t+1) + \cdots + R_L w(t+L) = 0$$

$$\tilde{w}(1), \tilde{w}(2), \ldots, \tilde{w}(T) \mapsto \hat{R}(\xi) = \hat{R}_0 + \hat{R}_1 \xi + \cdots + \hat{R}_{\hat{L}} \xi^{\hat{L}}$$

Case of interest today

We discuss 2 cases:

ID with latent inputs



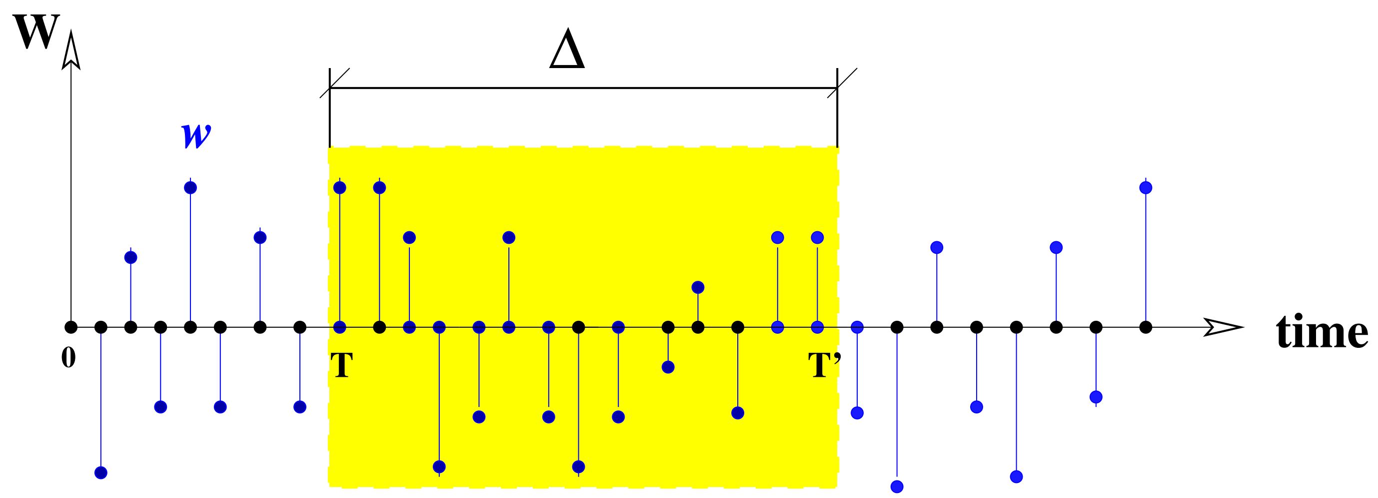
$$R_0 w(t) + R_1 w(t+1) + \cdots + R_L w(t+L)$$
$$= M_0 \varepsilon(t) + M_1 \varepsilon(t+1) + \cdots + M_L \varepsilon(t+L)$$

$$\tilde{w}(1), \tilde{w}(2), \ldots, \tilde{w}(T) \mapsto \left(\hat{R}(\xi), \hat{M}(\xi)\right)$$

Deterministic System ID

$\tilde{w} \mapsto R$

Basic ideas: look through the window in order to discover the laws.



Data: $\tilde{w}(1), \tilde{w}(2), \ldots, \tilde{w}(T)$

Consider

$$\mathcal{H} = \begin{bmatrix} \tilde{w}(1) & \tilde{w}(2) & \cdots & \tilde{w}(T-\Delta+1) \\ \tilde{w}(2) & \tilde{w}(3) & \cdots & \tilde{w}(T-\Delta+2) \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{w}(\Delta) & \tilde{w}(\Delta+1) & \cdots & \tilde{w}(T) \end{bmatrix}$$

Compute left kernel of $\mathcal{H}$.

Structure of a polynomial module

$\rightsquigarrow$ **efficient computation, recursive in 'depth' Δ.**

Identifiability

Assume $\tilde{w} = (\tilde{u}, \tilde{y})$ generated by behavior $\mathfrak{B}$. Then

$$\begin{bmatrix} \tilde{u}(1) & \tilde{u}(2) & \tilde{u}(3) & \cdots & \tilde{u}(T-\Delta+1) \\ \tilde{y}(1) & \tilde{y}(2) & \tilde{y}(3) & \cdots & \tilde{y}(T-\Delta+1) \\ \tilde{u}(2) & \tilde{u}(3) & \tilde{u}(4) & \cdots & \tilde{u}(T-\Delta+2) \\ \tilde{y}(2) & \tilde{y}(3) & \tilde{y}(4) & \cdots & \tilde{y}(T-\Delta+2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \tilde{u}(\Delta) & \tilde{u}(\Delta+1) & \tilde{u}(\Delta+2) & \cdots & \tilde{u}(T) \\ \tilde{y}(\Delta) & \tilde{y}(\Delta+1) & \tilde{y}(\Delta+2) & \cdots & \tilde{y}(T) \end{bmatrix}$$

has 'correct' kernel & image if

1. $\Delta > \mathtt{L}(\mathfrak{B})$ 2. $\mathfrak{B}$ controllable
3. $\tilde{u}$ is persistently exciting of order $> \Delta + \mathtt{n}(\mathfrak{B})$

From the data to the state trajectory

$$\tilde{w} \mapsto \tilde{x} \mapsto \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

If it is possible to pass from the data

$$\tilde{w}(1), \tilde{w}(2), \ldots, \tilde{w}(T)$$

directly to the state trajectory

$$\tilde{x}(1), \tilde{x}(2), \ldots, \tilde{x}(T)$$

Then we can identify the model by solving

$$\begin{bmatrix} \tilde{x}(2) & \tilde{x}(3) & \cdots & \tilde{x}(T) \\ \tilde{y}(1) & \tilde{y}(2) & \cdots & \tilde{y}(T-1) \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} \tilde{x}(1) & \tilde{x}(2) & \cdots & \tilde{x}(T-1) \\ \tilde{u}(1) & \tilde{u}(2) & \cdots & \tilde{u}(T-1) \end{bmatrix}$$

$$\tilde{w} \mapsto \tilde{x} \mapsto \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

How does this work?

$$\tilde{\boldsymbol{w}}(1), \tilde{\boldsymbol{w}}(2), \ldots, \tilde{\boldsymbol{w}}(\boldsymbol{T})$$

$$\Downarrow$$

$$\tilde{\boldsymbol{x}}(1), \tilde{\boldsymbol{x}}(2), \ldots, \tilde{\boldsymbol{x}}(\boldsymbol{T})$$

Several algorithms. We give 3 of them.
Assume $\Delta > L(\mathfrak{B})$, and pers. of exc. as needed.

$$\tilde{w} \mapsto \tilde{x} \mapsto \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

1. Compute ‘the’ left annihilators of $\mathcal{H}$:

$$\begin{bmatrix} N_1 & N_2 & N_3 & \cdots & N_\Delta \end{bmatrix} \begin{bmatrix} \tilde{w}(1) & \tilde{w}(2) & \cdots & \tilde{w}(T-\Delta+1) \\ \tilde{w}(2) & \tilde{w}(3) & \cdots & \tilde{w}(T-\Delta+2) \\ \tilde{w}(3) & \tilde{w}(4) & \cdots & \tilde{w}(T-\Delta+3) \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{w}(\Delta) & \tilde{w}(\Delta+1) & \cdots & \tilde{w}(T) \end{bmatrix} = 0$$

$$\tilde{w} \mapsto \tilde{x} \mapsto \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

1. Compute ‘the’ left annihilators of $\mathcal{H}$:

$$\begin{bmatrix} N_1 & N_2 & N_3 & \cdots & N_\Delta \end{bmatrix} \begin{bmatrix} \tilde{w}(1) & \tilde{w}(2) & \cdots & \tilde{w}(T-\Delta+1) \\ \tilde{w}(2) & \tilde{w}(3) & \cdots & \tilde{w}(T-\Delta+2) \\ \tilde{w}(3) & \tilde{w}(4) & \cdots & \tilde{w}(T-\Delta+3) \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{w}(\Delta) & \tilde{w}(\Delta+1) & \cdots & \tilde{w}(T) \end{bmatrix} = 0$$

Then

$$\begin{bmatrix} \tilde{x}(1) & \tilde{x}(2) & \cdots & \tilde{x}(T-\Delta+1) \end{bmatrix}$$

$$= \begin{bmatrix} N_2 & N_3 & \cdots & N_\Delta & 0 \\ N_3 & N_4 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ N_{\Delta-1} & N_\Delta & \cdots & 0 & 0 \\ N_\Delta & 0 & \cdots & 0 & 0 \end{bmatrix} \begin{bmatrix} \tilde{w}(1) & \tilde{w}(2) & \cdots & \tilde{w}(T-\Delta+1) \\ \tilde{w}(2) & \tilde{w}(3) & \cdots & \tilde{w}(T-\Delta+2) \\ \tilde{w}(3) & \tilde{w}(4) & \cdots & \tilde{w}(T-\Delta+3) \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{w}(\Delta) & \tilde{w}(\Delta+1) & \cdots & \tilde{w}(T) \end{bmatrix}$$

$$\tilde{w} \mapsto \tilde{x} \mapsto \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

$$\left[\begin{array}{c} \mathcal{H}_- \\ \hline \mathcal{H}_+ \end{array}\right] = \left[\begin{array}{cccc} \tilde{w}(1) & \tilde{w}(2) & \cdots & \tilde{w}(T-2\Delta+1) \\ \tilde{w}(2) & \tilde{w}(3) & \cdots & \tilde{w}(T-2\Delta+2) \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{w}(\Delta) & \tilde{w}(\Delta+1) & \cdots & \tilde{w}(T-\Delta) \\ \hline \tilde{w}(\Delta+1) & \tilde{w}(\Delta+2) & \cdots & \tilde{w}(T-\Delta+1) \\ \tilde{w}(\Delta+2) & \tilde{w}(\Delta+3) & \cdots & \tilde{w}(T-\Delta+2) \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{w}(2\Delta) & \tilde{w}(2\Delta+1) & \cdots & \tilde{w}(T) \end{array}\right]$$

↑ ↑ ↑

PAST

FUTURE

↓ ↓ ↓

$$\tilde{w} \mapsto \tilde{x} \mapsto \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

$$\left[\frac{\mathcal{H}_-}{\mathcal{H}_+}\right] = \left[\begin{array}{cccc} \tilde{w}(1) & \tilde{w}(2) & \cdots & \tilde{w}(T-2\Delta+1) \\ \tilde{w}(2) & \tilde{w}(3) & \cdots & \tilde{w}(T-2\Delta+2) \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{w}(\Delta) & \tilde{w}(\Delta+1) & \cdots & \tilde{w}(T-\Delta) \\ \hline \tilde{w}(\Delta+1) & \tilde{w}(\Delta+2) & \cdots & \tilde{w}(T-\Delta+1) \\ \tilde{w}(\Delta+2) & \tilde{w}(\Delta+3) & \cdots & \tilde{w}(T-\Delta+2) \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{w}(2\Delta) & \tilde{w}(2\Delta+1) & \cdots & \tilde{w}(T) \end{array}\right]$$

↑ ↑ ↑ PAST

FUTURE ↓ ↓ ↓

2. The intersection of the span of the rows of $\mathcal{H}_-$ with the span of the rows of $\mathcal{H}_+$ equals

$$\begin{bmatrix} \tilde{x}(\Delta) & \tilde{x}(\Delta+1) & \cdots & \tilde{x}(T-\Delta) \end{bmatrix}$$

PRESENT STATE

Nice num. impl. (e.g. via left kernel) $\rightsquigarrow$ subspace ID

$$\tilde{w} \mapsto \tilde{x} \mapsto \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

3. Solve for G

$$\left[\begin{array}{ccc} \tilde{w}(1) & \cdots & \tilde{w}(T-2\Delta+1) \\ \vdots & \vdots & \vdots \\ \tilde{w}(\Delta) & \cdots & \tilde{w}(T-\Delta) \\ \hline \tilde{u}(\Delta+1) & \cdots & \tilde{u}(T-\Delta+1) \\ \vdots & \vdots & \vdots \\ \tilde{u}(2\Delta) & \cdots & \tilde{u}(T) \end{array}\right] G = \left[\begin{array}{ccc} \tilde{w}(1) & \cdots & \tilde{w}(T-2\Delta+1) \\ \vdots & \vdots & \vdots \\ \tilde{w}(\Delta) & \cdots & \tilde{w}(T-\Delta) \\ \hline 0 & \cdots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & 0 \end{array}\right]$$

$$\begin{bmatrix} \tilde{y}(\Delta+1) & \cdots & \tilde{y}(T-\Delta+1) \\ \vdots & \vdots & \vdots \\ \tilde{y}(2\Delta) & \cdots & \tilde{y}(T) \end{bmatrix} G = \begin{bmatrix} \tilde{x}(\Delta) & \cdots & \tilde{x}(T-\Delta) \end{bmatrix}$$

Computes $\tilde{x}$!

$\cong$ **'oblique projection**

$\tilde{w} \mapsto R$ **or** $\begin{bmatrix} A & B \\ C & D \end{bmatrix}$

These algorithms, compute the left kernel of $\mathcal{H}$, etc. allow approximate implementations. For the state algorithms, this is worked out very well (subspace ID).

SVD $$\tilde{X} = \begin{bmatrix} \tilde{x}(1) & \tilde{x}(2) & \cdots & \tilde{x}(T) \end{bmatrix}$$

$$\rightsquigarrow \quad \tilde{X}^{\text{red}} = \begin{bmatrix} \tilde{x}^{\text{red}}(1) & \tilde{x}^{\text{red}}(2) & \cdots & \tilde{x}^{\text{red}}(T) \end{bmatrix}$$

followed by LS solution of

$$\begin{bmatrix} \tilde{x}^{\text{red}}(2) & \tilde{x}^{\text{red}}(3) & \cdots & \tilde{x}^{\text{red}}(T) \\ \tilde{y}(1) & \tilde{y}(2) & \cdots & \tilde{y}(T-1) \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} \tilde{x}^{\text{red}}(1) & \tilde{x}^{\text{red}}(2) & \cdots & \tilde{x}^{\text{red}}(T-1) \\ \tilde{u}(1) & \tilde{u}(2) & \cdots & \tilde{u}(T-1) \end{bmatrix}$$

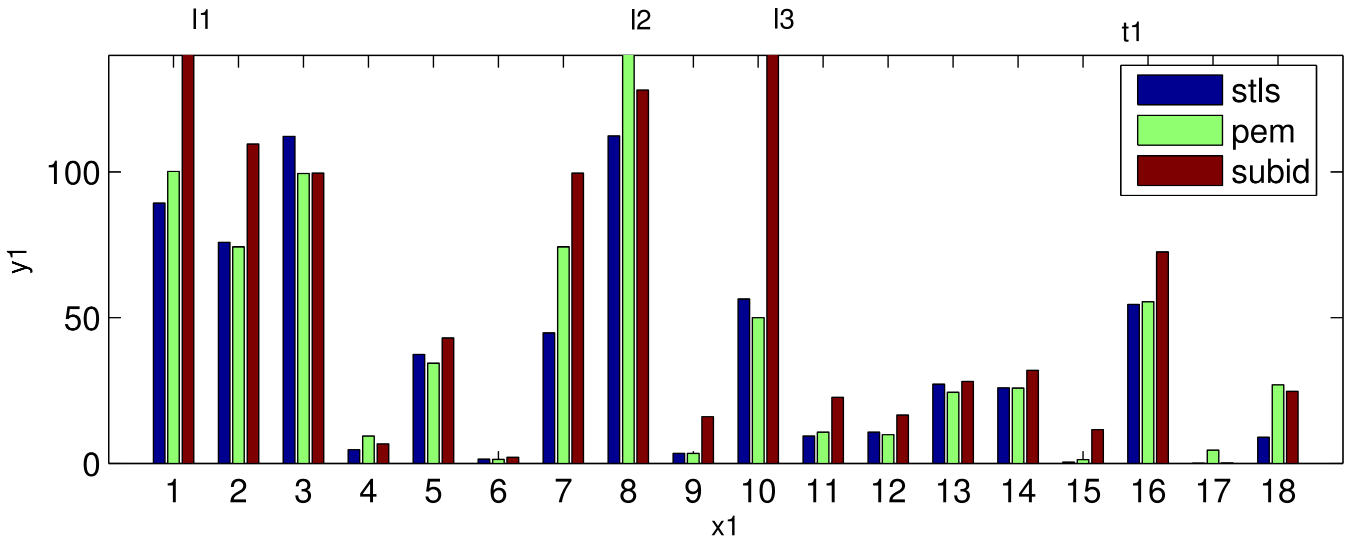
Performance

#	Data set name	T	m	p	l
1	Data of the western basin of Lake Erie	57	5	2	1
2	Data of Ethane-ethylene column	90	5	3	1
3	Data of a 120 MW power plant	200	5	3	2
4	Heating system	801	1	1	2
5	Data from an industrial dryer	867	3	3	1
6	Data of a hair dryer	1000	1	1	5
7	Data of the ball-and-beam setup in SISTA	1000	1	1	2
8	Wing flutter data	1024	1	1	5
9	Data from a flexible robot arm	1024	1	1	4
10	Data of a glass furnace (Philips)	1247	3	6	1
11	Heat flow density through a two layer wall	1680	2	1	2
12	Simulation of a pH neutralization process	2001	2	1	6
13	Data of a CD-player arm	2048	2	2	1
14	Data from an industrial winding process	2500	5	2	2
15	Liquid-saturated heat exchanger	4000	1	1	2
16	Data from an evaporator	6305	3	3	1
17	Continuous stirred tank reactor	7500	1	2	1
18	Model of a steam generator	9600	4	4	1

Performance

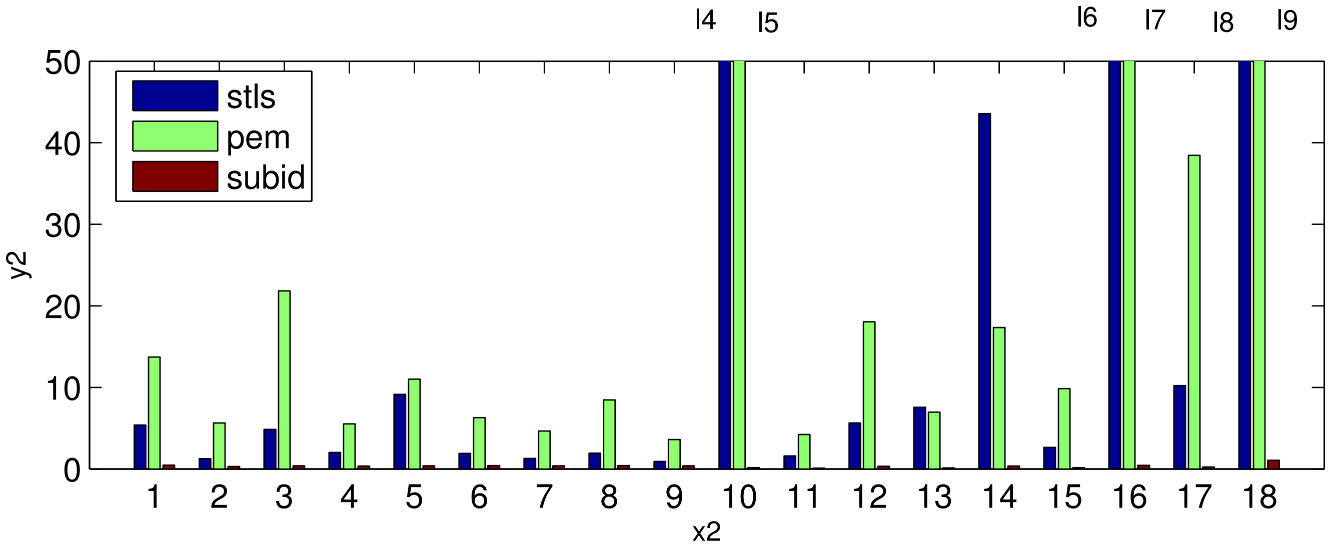
Compare the misfit on the last 30% of the outputs and the execution time for computing the ID model from the first 70% of the data.

Misfit

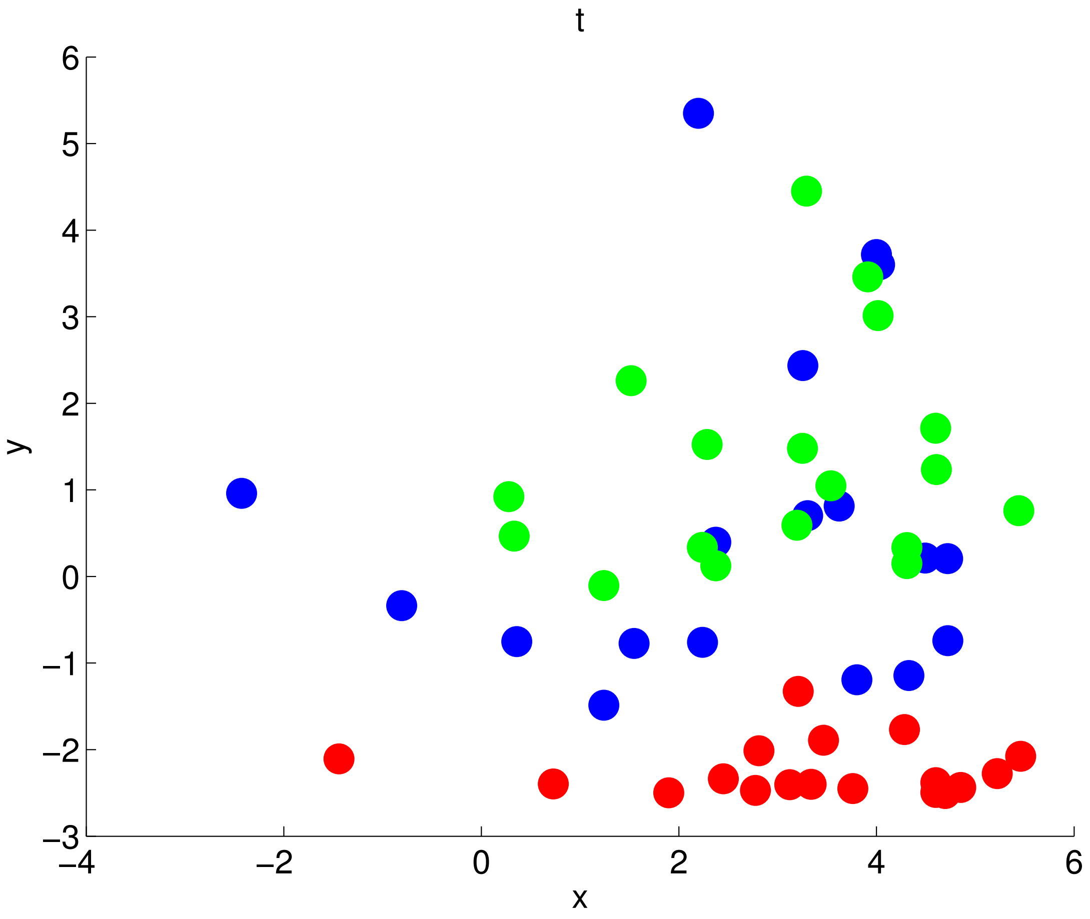


Performance

Execution time

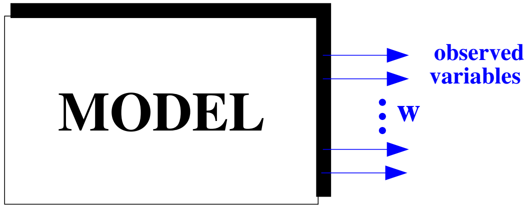


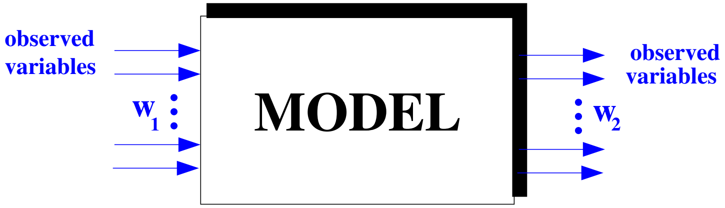
Performance



Latency minimization

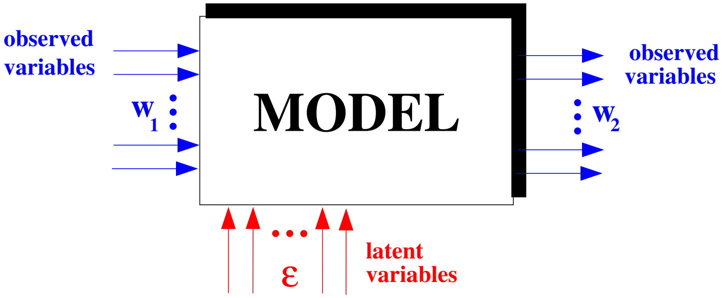
Why latent variables?





$$R_0 w(t) + R_1 w(t+1) + \cdots + R_L w(t+L) = 0$$

versus



$$R_0 w(t) + R_1 w(t+1) + \cdots + R_L w(t+L)$$
$$= M_0 \varepsilon(t) + M_1 \varepsilon(t+1) + \cdots + M_L \varepsilon(t+L)$$

Why latent variables?

As far as the w-behavior is concerned, this gives nothing new (⇐ **elimination theorem**).

So, what is the rationale for using latent variables ε ?

Why latent variables?

Data $\tilde{w}(t_1), \tilde{w}(t_1+1), \ldots, \tilde{w}(t_2)$ **with** $\tilde{w}(t) \in \mathbb{R}$

The model

$$R_0 w(t) + R_1 w(t+1) + \cdots + R_L w(t+L) = 0$$

$\rightsquigarrow$ **either** w **= input, free,** $\mathfrak{B} = \mathbb{R}^{\mathbb{T}}$

or w **= output,** $\rightsquigarrow \mathfrak{B} \cong$ **sums of 'exponentials'**

$\rightsquigarrow$ **very restrictive.**

Assuming unobserved inputs:

$$R_0 w(t) + \cdots + R_L w(t+L) = M_0 \varepsilon(t) + \cdots + M_L \varepsilon(t+L)$$

gives better possibilities, e.g. for prediction.

Latency minimization

Define the **'latency'**:

$$\text{latency}\ (\tilde{w}, \mathfrak{B}) := \ \text{minimum}\ ||\tilde{\varepsilon}||_{\ell^2}$$

with the minimum taken over all $\tilde{\varepsilon}$ such that

$$R_0\tilde{w}(t) + \cdots + R_L\tilde{w}(t+L) = M_0\tilde{\varepsilon}(t) + \cdots + M_L\tilde{\varepsilon}(t+L)$$

i.e. min. over all $\tilde{\varepsilon}$ that 'explain' $\tilde{w}(1), \tilde{w}(2), \ldots, \tilde{w}(T)$.

$\rightsquigarrow$ **system ID:** search for the optimal model, *in the sense of minimal latency*, in a given model class.

Latency minimization

- How do we compute the latency, the optimal $\tilde{\varepsilon}$'s?
- Algorithms for minimization over the R's, M's in the model class.

The latency minimization is a deterministic Kalman filtering problem

The latency is actually equal to the prediction error!

$\leadsto$ deterministic interpretation, system ID toolbox, etc.

Stochastic System ID

Why stochastic interpretation?

$$R_0 w(t) + \cdots + R_L w(t+L) = M_0 \varepsilon(t) + \cdots + M_L \varepsilon(t+L)$$

In this model we can, of course, consider ε as a stochastic disturbance. If we consider also u as a stochastic process, then also w becomes stochastic.

This has the virtue to make the system ID problem to a statistical one, leading to questions of maximum likelihood estimation (very related to prediction error). It allows evaluation of the algorithms in terms of their behavior as $T \to \infty$. Nice statistical questions emerge, as consistency, asymptotic efficiency, etc.

$\rightsquigarrow$ deep theory of ARMAX systems.

Why stochastic interpretation?

It is difficult to argue that stochastic unobserved disturbances offer a realistic explanation of the lack of fit between observations and the deterministic part.

This lack of fit is more likely a result of low order, linear models for nonlinear systems, neglected dynamics, approximation, in addition to unmeasured inputs, which may or may not be stochastic.

Stochastic methods offer the user a 'certificate' under which the algorithms work well.

ARMAX Systems

Processes

In the remainder of this talk **‘process’** means:

a vector of real stoch. processes on $\mathbb{Z}$ (or $\mathbb{N}$),
(jointly) gaussian, zero mean, stationary, and **ergodic.**

‘White noise’ means: a process ε with

the $\varepsilon(t)$’s i.i.d. and $\mathbb{E}\Big(\varepsilon(0)\varepsilon^\top(0)\Big) = I$

$\perp\!\!\!\perp$ means: ‘independence’.

Processes

In the remainder of this talk **‘process’** means:

a vector of real stoch. processes on $\mathbb{Z}$ (or $\mathbb{N}$),
(jointly) gaussian, zero mean, stationary, and **ergodic.**

A **(stochastic) system** means:

:= a collection of processes = the **‘behavior’**

ARMAX

Consider the difference eq'ns

$$W(\sigma)w = E(\sigma)\varepsilon \qquad \textbf{(ARMAX)}$$

with E, E **real polynomial matrices;**

$\sigma =$ **the 'shift':** $\quad (\sigma f)(t) := f(t+1)$.

Consider the difference eq'ns

$$W(\sigma)w = E(\sigma)\varepsilon \qquad \text{(ARMAX)}$$

The stochastic system consisting of all processes w satisfying (ARMAX) with ε white noise

is called the **ARMAX system (W, E).**

Consider the difference eq'ns

$$\boxed{W(\sigma)w = E(\sigma)\varepsilon} \qquad \text{(ARMAX)}$$

Example: the difference eq'ns

$$\boxed{Y(\sigma)y + U(\sigma)u = E(\sigma)\varepsilon} \qquad \text{(ARMAX)}$$

with Y, U, E real polynomial matrices, Y square, $\det(Y) \neq 0$;

Under 'generic' conditions, u is free: $\forall u \ \exists\, y$ ... ,
u is an 'exogeneous' input; y an 'endogenous' output.

AR-MA-X

Refine the ARMAX notation, by factoring out A, to:

$$A(\sigma)\Big(R(\sigma)w\Big) = M(\sigma)\varepsilon$$

A, R, M real polynomial matrices,

A square, $\det(A) \neq 0$,

R left prime.

AR-MA-X

Refine the ARMAX notation, by factoring out A, to:

$$A(\sigma)\Big(R(\sigma)w\Big) = M(\sigma)\varepsilon$$

We call

A	the	AR-part
M	the	MA-part
R	the	X-part

$G = P^{-1}Q \;\; (R = [\,P \;\; Q\,])$ = tf f'n of the 'deterministic part'.

$\exists$ a 'classification up to equivalence issue' for (R, A, M).

Estimate R, A, M **from observed**

$$\tilde{w}(1), \tilde{w}(2) \ldots, \tilde{w}(T)$$

AR-MA-X system ID

In the stochastic case, the subspace algorithms

$$\begin{bmatrix} \mathcal{H}_- \\ \hline \mathcal{H}_+ \end{bmatrix} = \left[\begin{array}{cccc} \tilde{w}(1) & \tilde{w}(2) & \cdots & \tilde{w}(T-2\Delta+1) \\ \tilde{w}(2) & \tilde{w}(3) & \cdots & \tilde{w}(T-2\Delta+2) \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{w}(\Delta) & \tilde{w}(\Delta+1) & \cdots & \tilde{w}(T-\Delta) \\ \hline \tilde{w}(\Delta+1) & \tilde{w}(\Delta+2) & \cdots & \tilde{w}(T-\Delta+1) \\ \tilde{w}(\Delta+2) & \tilde{w}(\Delta+3) & \cdots & \tilde{w}(T-\Delta+2) \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{w}(2\Delta) & \tilde{w}(2\Delta+1) & \cdots & \tilde{w}(T) \end{array}\right] \begin{array}{c} \uparrow \\ \uparrow \\ \uparrow \\ \text{PAST} \\ \hline \text{FUTURE} \\ \downarrow \\ \downarrow \\ \downarrow \end{array}$$

require, e.g. for consistency, $T \to \infty$, which is fine, but also $\Delta \to \infty$, which is unfortunate!

Estimate R, A, M **from observed**

$$\tilde{w}(1), \tilde{w}(2) \ldots, \tilde{w}(T)$$

Divide et impera **algorithm:**

$$\tilde{w} \mapsto R \mapsto A \mapsto M$$

Today, we explain the 'X-part': how to compute

$$\tilde{w} \mapsto R$$

$$\boxed{\tilde{w} \mapsto R}$$

Assume that $\tilde{w} = \begin{bmatrix} \tilde{u} \\ \tilde{y} \end{bmatrix}$, and $\tilde{u} \perp \varepsilon$. Then

$$A(\sigma)\Big(P(\sigma)\tilde{y} + Q(\sigma)\tilde{u}\Big) = R(\sigma)\tilde{w} = M(\sigma)\varepsilon$$

$\Rightarrow$

$$R(\sigma)\tilde{w} = \sum_{t=-\infty}^{+\infty} H(t)\ \sigma^t \varepsilon$$

$\Rightarrow$ (since $\varepsilon \perp \tilde{u}$)

$$\boxed{R(\sigma)\tilde{w}\ \perp\ \tilde{u}.}$$

$\tilde{w} \mapsto R$

Basic idea: the linear combinations of the rows of the observed

$$\begin{bmatrix} \tilde{w}(1) & \tilde{w}(2) & \tilde{w}(3) & \cdots & \tilde{w}(t) & \cdots \\ \tilde{w}(2) & \tilde{w}(3) & \tilde{w}(4) & \cdots & \tilde{w}(t+1) & \cdots \\ \tilde{w}(3) & \tilde{w}(4) & \tilde{w}(5) & \cdots & \tilde{w}(t+2) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

that are orthogonal to the rows of the observed

$$\begin{bmatrix} \tilde{u}(1) & \tilde{u}(2) & \tilde{u}(3) & \cdots & \tilde{u}(t) & \cdots \\ \tilde{u}(2) & \tilde{u}(3) & \tilde{u}(4) & \cdots & \tilde{u}(t+1) & \cdots \\ \tilde{u}(3) & \tilde{u}(4) & \tilde{u}(5) & \cdots & \tilde{u}(t+2) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

determine R.

$\tilde{w} \mapsto R$

$$\begin{bmatrix} \tilde{w}(1) & \tilde{w}(2) & \tilde{w}(3) & \cdots & \tilde{w}(t) & \cdots \\ \tilde{w}(2) & \tilde{w}(3) & \tilde{w}(4) & \cdots & \tilde{w}(t+1) & \cdots \\ \tilde{w}(3) & \tilde{w}(4) & \tilde{w}(5) & \cdots & \tilde{w}(t+2) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$\begin{bmatrix} \tilde{u}(1) & \tilde{u}(2) & \tilde{u}(3) & \cdots & \tilde{u}(t) & \cdots \\ \tilde{u}(2) & \tilde{u}(3) & \tilde{u}(4) & \cdots & \tilde{u}(t+1) & \cdots \\ \tilde{u}(3) & \tilde{u}(4) & \tilde{u}(5) & \cdots & \tilde{u}(t+2) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$\exists$ an ∞ number of such ‘orthogonalizing’ linear combinations.

What special structure do they have, so that they are determined by a finite number of them, $\cong$ R ?

Is there a way to limit the number of rows?

Modules

Modules

A module can be thought of as 'a vector space over a ring'.

A mathematician's favorite ex.:

the module $\mathbb{Z} + \sqrt{2}\,\mathbb{Z} + \sqrt{5}\,\mathbb{Z}$ over the ring $\mathbb{Z}$

A system theorist's favorite ex.:

the module $\mathbb{R}^{\mathtt{n}}[\xi]$ over the ring $\mathbb{R}[\xi]$

$\mathbb{R}^{\mathtt{n}}[\xi]$:= the $\mathtt{n}$-dimensional vectors of polynomials

with real coefficients, in the indeterminate ξ.

Modules

A **submodule** of $\mathbb{R}^{\mathrm{n}}[\xi]$ is a subset that is also a module over $\mathbb{R}[\xi]$.

E.g., for given polynomial vectors $v_1, v_2, \ldots, v_{\mathrm{k}}$, all sums

$$p_1 v_1 + p_2 v_2 + \cdots + p_{\mathrm{k}} v_{\mathrm{k}}$$

p's polynomials. The submodule 'generated by' $v_1, v_2, \ldots, v_{\mathrm{k}}$.

Fact: Every submodule of $\mathbb{R}^{\mathrm{n}}[\xi]$ is of this form: 'finitely generated'.

Fact: Number of generators $\leq \mathrm{n}$. (mimimum =: the dimension)

Modules

A **submodule** of $\mathbb{R}^n[\xi]$ is a subset that is also a module over $\mathbb{R}[\xi]$.

A submodule of $\mathbb{R}^n[\xi]$ is said to be **slim** if it does not contain other submodules of the same dimension

$$\Leftrightarrow V = [\, v_1 \; v_2 \; \cdots \; v_k \,] \text{ right prime.}$$

<u>slim</u>: $\begin{bmatrix} * \\ \vdots \\ * \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ <u>not slim</u>: $p(\xi)\, \mathbb{R}[\xi]$.

Modules

Submodules of $\mathbb{R}^{\mathrm{n}}[\xi]$ are of great importance in system theory:

linear time-inv. diff. systems	$\stackrel{1:1}{\longleftrightarrow}$	submodules of $\mathbb{R}^{\mathrm{n}}[\xi]$
controllable LTIS	$\stackrel{1:1}{\longleftrightarrow}$	slim submodules

Modules

Submodules of $\mathbb{R}^{n}[\xi]$ are of great importance in system theory:

The 'left' or 'right' kernel of any Hankel matrix

$$\begin{bmatrix} H(1) & H(2) & H(3) & \cdots & H(t'') & \cdots \\ H(2) & H(3) & H(4) & \cdots & H(t''+1) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ H(t') & H(t'+1) & H(t'+2) & \cdots & H(t'+t''-1) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$\cong$ a submodule of $\mathbb{R}^{\text{coldim}(H)}[\xi]$ or $\mathbb{R}^{\text{rowdim}(H)}[\xi]$:

$\rightsquigarrow$ effectively not ∞-dimensional, but

$\leq \text{rowdim}(H)$- or $\text{coldim}(H)$-dimensional!

The orthogonalizers

The orthogonalizers

Let $w = \begin{bmatrix} u \\ y \end{bmatrix}$ be a process. Say, $\mathtt{w}$-dimensional.

$n \in \mathbb{R}^{\mathtt{w}}[\xi]$ is an **orthogonalizer** (for w w.r.t u) if

$$n^\top(\sigma)w \perp u.$$

i.e. a linear combination of the components of the process w and its shifts which becomes independent of the process u.

The orthogonalizers

Let $w = \begin{bmatrix} u \\ y \end{bmatrix}$ be a process. Say, $\mathtt{w}$-dimensional.

$n \in \mathbb{R}^{\mathtt{w}}[\xi]$ is an **orthogonalizer** (for w w.r.t u) if

$$n^\top(\sigma)w \perp u.$$

Ex.: the transpose of the rows of R, since

$$A(\sigma)\big(R(\sigma)w\big) = M(\sigma)\varepsilon \quad \perp u.$$

$\Rightarrow$ every element of the module generated by these.

Is this all? Are there no other orthogonalizers?

Let $w = \begin{bmatrix} u \\ y \end{bmatrix}$ be a process. Say, $\mathtt{w}$-dimensional.

Theorem:

1. The orthogonalizers for w w.r.t. u form a submodule of $\mathbb{R}^{\mathtt{w}}[\xi]$.
2. In fact, a slim one.
3. If $w = \begin{bmatrix} u \\ y \end{bmatrix}$ and u is ' persistently exciting', then it is precisely the submodule generated by the transposes of the rows of R.

The orthogonalizers

Let $w = \begin{bmatrix} u \\ y \end{bmatrix}$ be a process. Say, $\mathtt{w}$-dimensional.

Theorem:

1. The orthogonalizers for w w.r.t. u form a submodule of $\mathbb{R}^{\mathtt{w}}[\xi]$.
2. In fact, a slim one.
3. If $w = \begin{bmatrix} u \\ y \end{bmatrix}$ and u is ' persistently exciting', then it is precisely the submodule generated by the transposes of the rows of R.

Proof: 1. is easy. 2. uses ergodicity! 3. a bit of module theory.

$$\boxed{\tilde{\boldsymbol{w}} \mapsto \boldsymbol{R}}$$

$\tilde{w} \mapsto R$

Find (a module basis for) the linear combinations of the rows of

$$\begin{bmatrix} \tilde{w}(1) & \tilde{w}(2) & \tilde{w}(3) & \cdots & \tilde{w}(t) & \cdots \\ \tilde{w}(2) & \tilde{w}(3) & \tilde{w}(4) & \cdots & \tilde{w}(t+1) & \cdots \\ \tilde{w}(3) & \tilde{w}(3) & \tilde{w}(5) & \cdots & \tilde{w}(t+2) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

that are orthogonal to the rows of

$$\begin{bmatrix} \tilde{u}(1) & \tilde{u}(2) & \tilde{u}(3) & \cdots & \tilde{u}(t) & \cdots \\ \tilde{u}(2) & \tilde{u}(3) & \tilde{u}(4) & \cdots & \tilde{u}(t+1) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$\tilde{w} \mapsto R$

Find the linear combinations of the rows of

$$W := \begin{bmatrix} \tilde{w}(1) & \tilde{w}(2) & \tilde{w}(3) & \cdots & \tilde{w}(t) & \cdots \\ \tilde{w}(2) & \tilde{w}(3) & \tilde{w}(4) & \cdots & \tilde{w}(t+1) & \cdots \\ \tilde{w}(3) & \tilde{w}(3) & \tilde{w}(5) & \cdots & \tilde{w}(t+2) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

that are orthogonal to the rows of

$$U := \begin{bmatrix} \tilde{u}(1) & \tilde{u}(2) & \tilde{u}(3) & \cdots & \tilde{u}(t) & \cdots \\ \tilde{u}(2) & \tilde{u}(3) & \tilde{u}(4) & \cdots & \tilde{u}(t+1) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

!! Compute the left kernel of $WU^\top$.

$\tilde{w} \mapsto R$

Find the linear combinations of the rows of

$$W := \begin{bmatrix} \tilde{w}(1) & \tilde{w}(2) & \tilde{w}(3) & \cdots & \tilde{w}(t) & \cdots \\ \tilde{w}(2) & \tilde{w}(3) & \tilde{w}(4) & \cdots & \tilde{w}(t+1) & \cdots \\ \tilde{w}(3) & \tilde{w}(3) & \tilde{w}(5) & \cdots & \tilde{w}(t+2) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

that are orthogonal to the rows of

$$U := \begin{bmatrix} \tilde{u}(1) & \tilde{u}(2) & \tilde{u}(3) & \cdots & \tilde{u}(t) & \cdots \\ \tilde{u}(2) & \tilde{u}(3) & \tilde{u}(4) & \cdots & \tilde{u}(t+1) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

¿¿ Can we limit the number of rows that are needed ??

Yes, provided we assume a known bound on the lags.

It is easy to prove that this, applied to a finite time-series, yields a consistent algorithm.

Note: no stability needed P for $R = [\ P \ \ Q\ ]$.

$\tilde{w} \mapsto A, M$

Once we have R, we can compute the process

$$\tilde{a} = R(\sigma)\tilde{w}$$

This is an ARMA process.

Our thinking in modules proceeds by estimating A.

$\tilde{w} \mapsto A, M$

Once we have R, we can compute the process

$$\tilde{a} = R(\sigma)\tilde{w}$$

This is an ARMA process.
Our thinking in modules proceeds by estimating A.

Then compute

$$\tilde{m} = A(\sigma)$$

There are very effective algorithms for estimating M.

A simulation

A simulation

The system is

$$\underbrace{A(\sigma)P(\sigma)}_{P'(\sigma)}y = \underbrace{A(\sigma)Q(\sigma)}_{Q'(\sigma)}u + M(\sigma)\varepsilon,$$

where the polynomials A, P, Q, and M are selected as follows:

$$A(\xi) = 1+\xi+0.5\xi^2, \qquad Q(\xi) = 1-1.2\xi+0.6\xi^2-0.7\xi^3, \qquad M(\xi) = 1+0.5\xi,$$

$$P(\xi) = 1 - 0.8713\xi - 1.539\xi^2 + 1.371\xi^3 + 0.6451\xi^4 - 0.5827\xi^5.$$

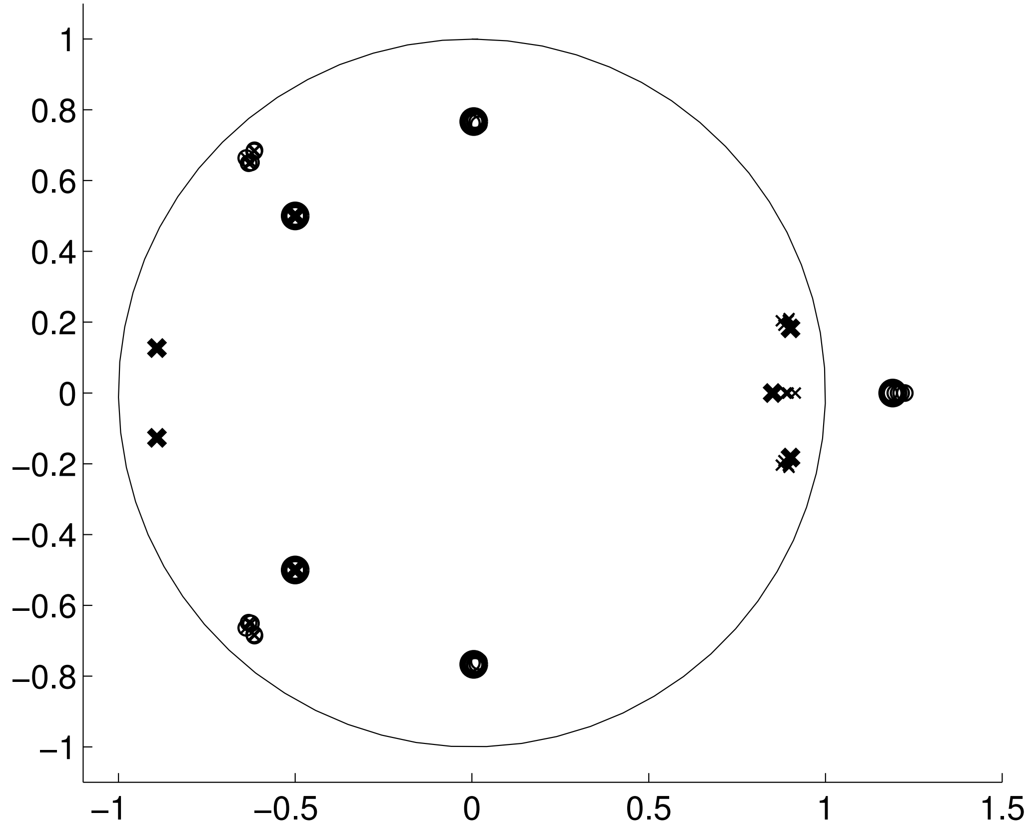
A simulation

The inputs u and ε are zero mean, gaussian, white, with variances 1 and 0.2, respectively. The initial condition, under which y is obtained from u and ε, is a random vector.

The time horizon for the simulation is $T = 1000$ and the simulated time series (u, y) is used for estimation.

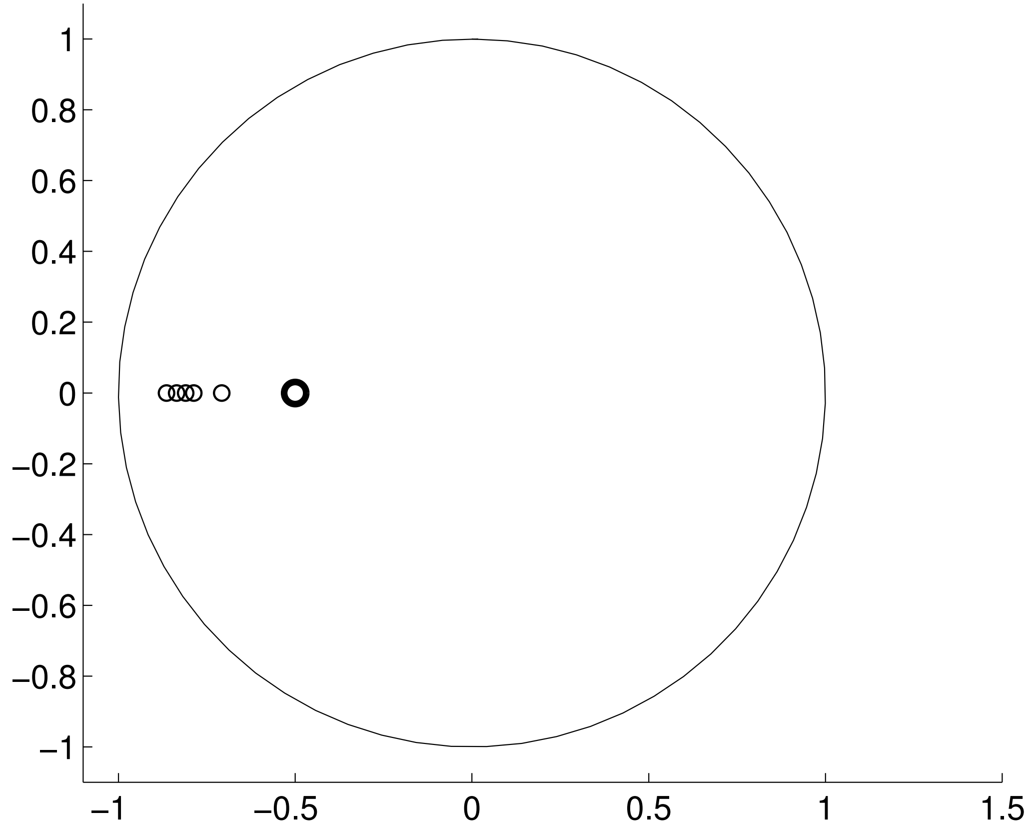
The experiment is repeated $N = 5$ times with different realizations of u and ε in each run.

A simulation



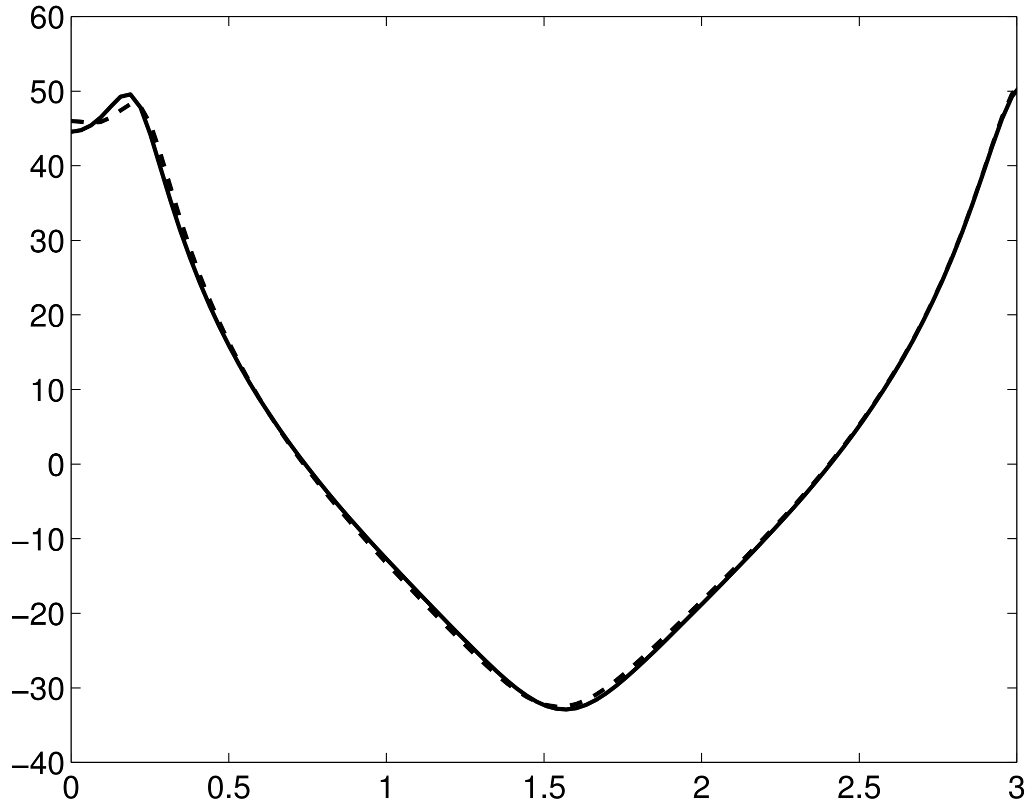
Roots of P', Q', $\hat{P}^{(k)}$, and $\hat{Q}^{(k)}$, for $k = 1, \ldots, N$.

A simulation



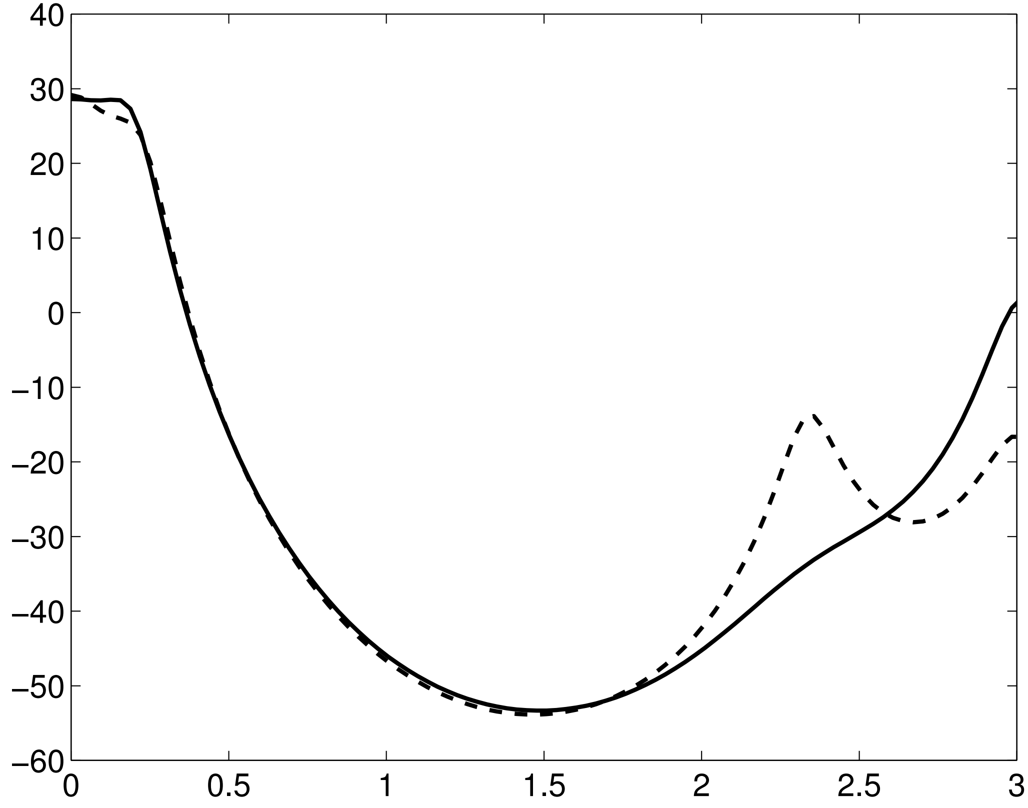
Roots of M and $\hat{M}^{(k)}$, for $k = 1, \ldots, N$.

A simulation



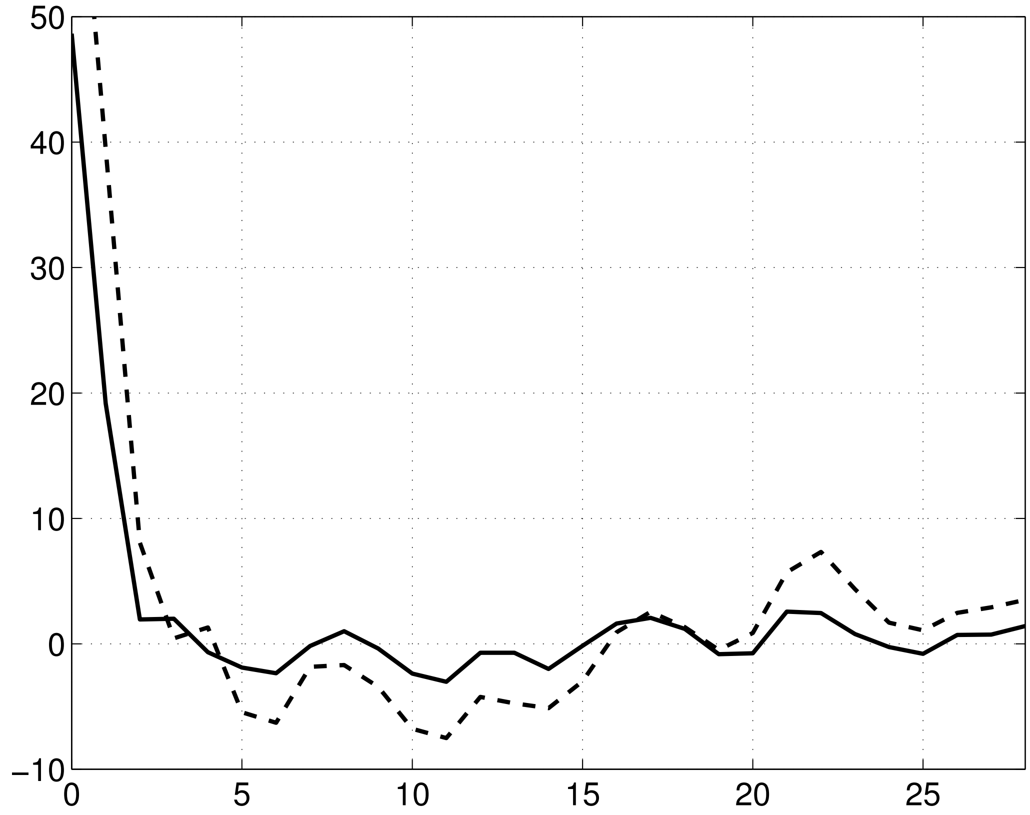
Bode plots of Q/P (solid line) and $\hat{Q}^{(1)}/\hat{P}^{(1)}$ (dashed line).

A simulation



Bode plots of M/P' (solid line) and $\hat{M}^{(1)}/\hat{P}^{(1)}$ (dashed line).

A simulation



Autocorrelation of $P'(\sigma)y - Q'(\sigma)u$ (solid line) and ˆ (dashed line).

Conclusion

For system ID, linear algebra on the Hankel matrix of the data, with a limited depth (Δ), contains the laws of the system.

We have shown this through
state construction directly from data, and
through the identification of the 'X' part
by means of the orthogonalizers.

Thank you

Thank you

Thank you

Thank you

Thank you

Thank you

Thank you

Thank you