SYSTEM IDENTIFICATION
via
STATE CONTRUCTION

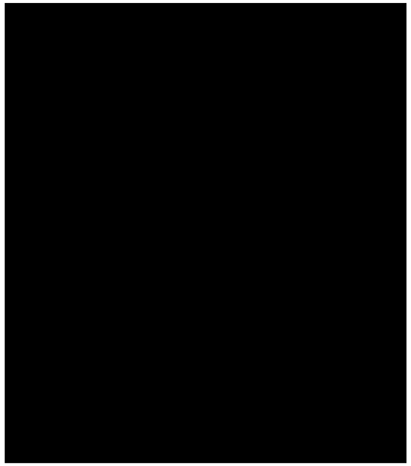
Jan C. Willems
K.U. Leuven, Belgium

Part of a research project with

Ivan Markovsky (K.U. Leuven)

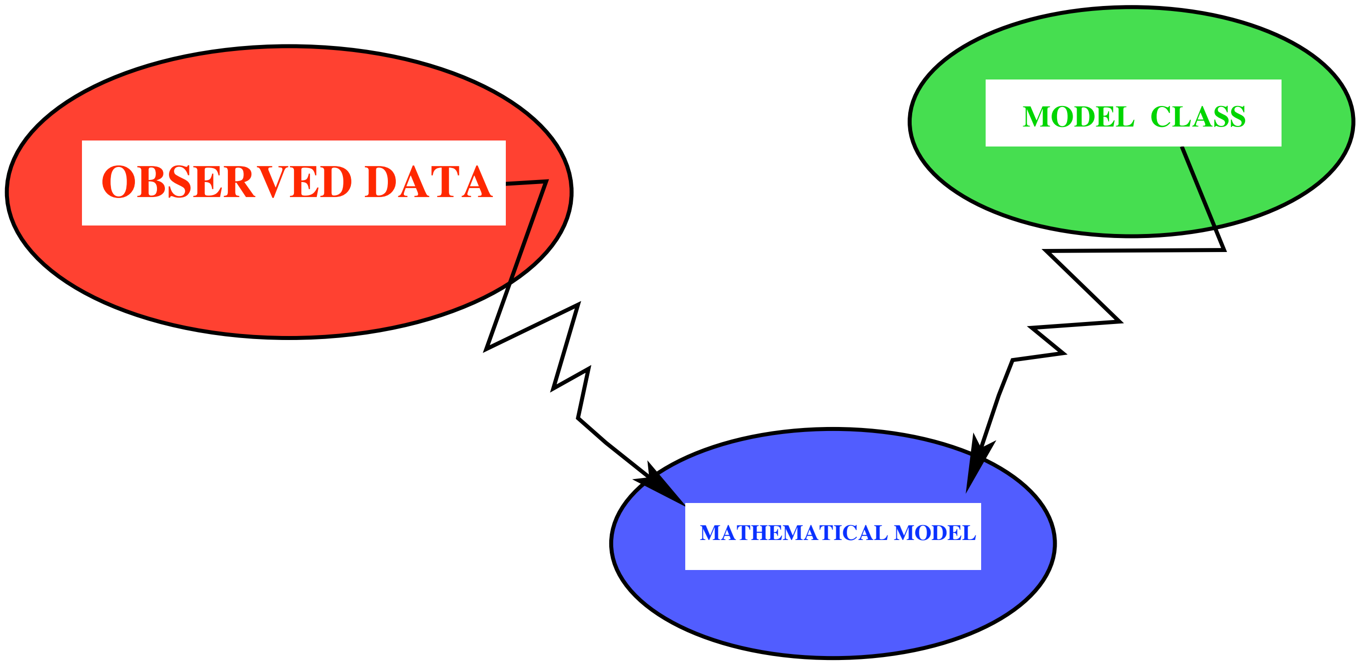
Paolo Rapisarda (Un. of Southampton)

& Bart De Moor (K.U. Leuven)



Problem

SYSID



SYSID

Data: an 'observed' vector time-series

$$\tilde{w}(1), \tilde{w}(2), \ldots, \tilde{w}(T) \qquad w(t) \in \mathbb{R}^{\mathtt{w}}$$

T finite, infinite, or $T \to \infty$

$$\Downarrow$$

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

e.g. a difference equation

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

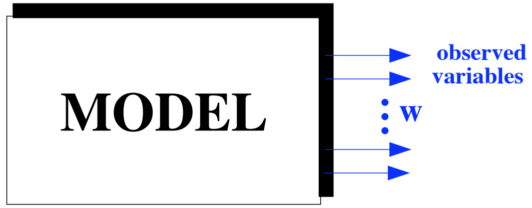
$$= 0$$

or

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

SYSID

‘deterministic’ ID



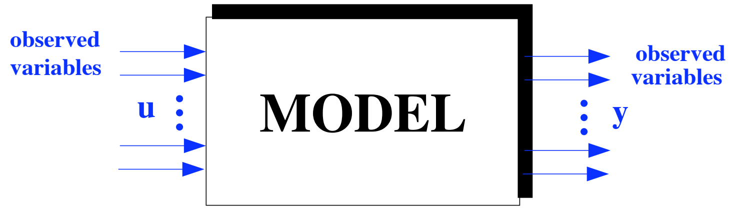
Model class:

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

SYSID algorithm:

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

‘deterministic’ ID: I/O form



Model class:

$$P_0 y(t) + \cdots + P_L y(t+L) = Q_0 u(t) + \cdots + Q_L u(t+L),$$

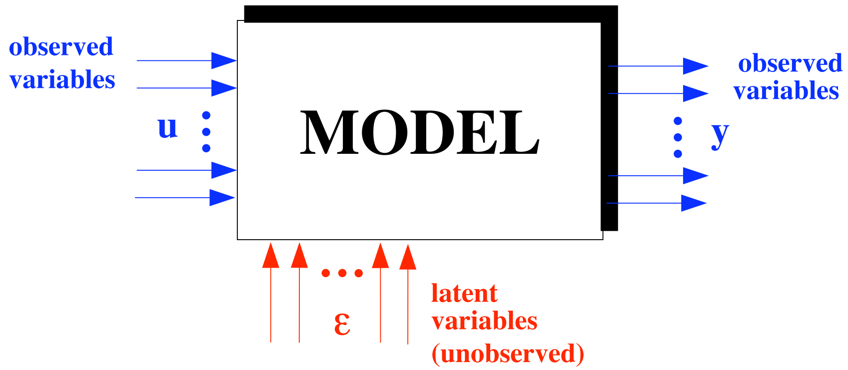
$$w = \Pi \begin{bmatrix} u \\ y \end{bmatrix}, \Pi \text{ permutation }, P(\xi)^{-1} Q(\xi) \text{ proper}$$

SYSID algorithm:

$$\tilde{w}(1), \tilde{w}(2), \ldots, \tilde{w}(T) \mapsto \hat{P}_0, \hat{P}_1, \cdots, \hat{P}_{\hat{L}}; \; \hat{Q}_0, \hat{Q}_1, \cdots, \hat{Q}_{\hat{L}}$$

SYSID

ID with unobserved latent inputs



Model class:

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

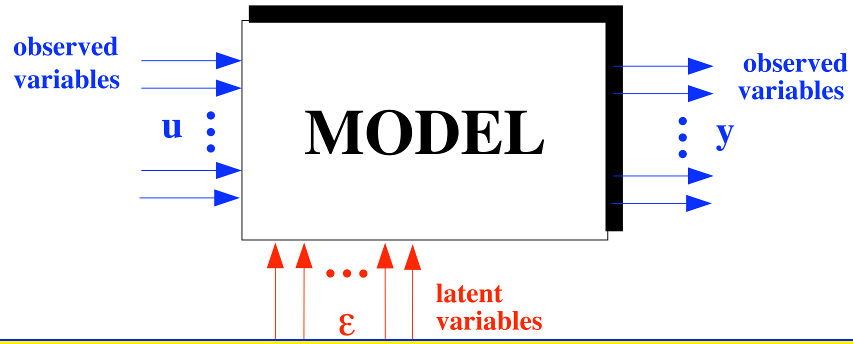
SYSID algorithm (e.g. PEM):

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

Usual assumption: w, ε stochastic. Main contributors: Deistler, Ljung. etc.

SYSID

ID with unobserved latent inputs



Why unobserved stochastic inputs?

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

SYSID algorithm (e.g. PEM):

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

Usual assumption: w, ε stochastic. Main contributors: Deistler, Ljung. etc.

Case of interest today

Assumptions:

- **Data:**

$$\tilde{w}(1), \tilde{w}(2), \ldots, \tilde{w}(t), \ldots \qquad w(t) \in \mathbb{R}^{\mathtt{w}} \qquad T \text{ infinite}$$

Case of interest today

Assumptions:

- **Data:**

  $$\tilde{w}(1), \tilde{w}(2), \ldots, \tilde{w}(t), \ldots \qquad w(t) \in \mathbb{R}^{\mathtt{w}} \qquad T \text{ infinite}$$

- **Deterministic SYSID**

Case of interest today

Assumptions:

- **Data:**

  $$\tilde{w}(1), \tilde{w}(2), \ldots, \tilde{w}(t), \ldots \qquad w(t) \in \mathbb{R}^{\mathtt{w}} \qquad T \text{ infinite}$$

- **Deterministic SYSID**
- **I/O partition known if advantageous**

Case of interest today

Assumptions:

- **Data:**

$$\tilde{w}(1), \tilde{w}(2), \ldots, \tilde{w}(t), \ldots \qquad w(t) \in \mathbb{R}^{\mathtt{w}} \qquad T \text{ infinite}$$

- **Deterministic SYSID**
- **I/O partition known if advantageous**
- **Exact modeling with an eye towards approximations**

Case of interest today

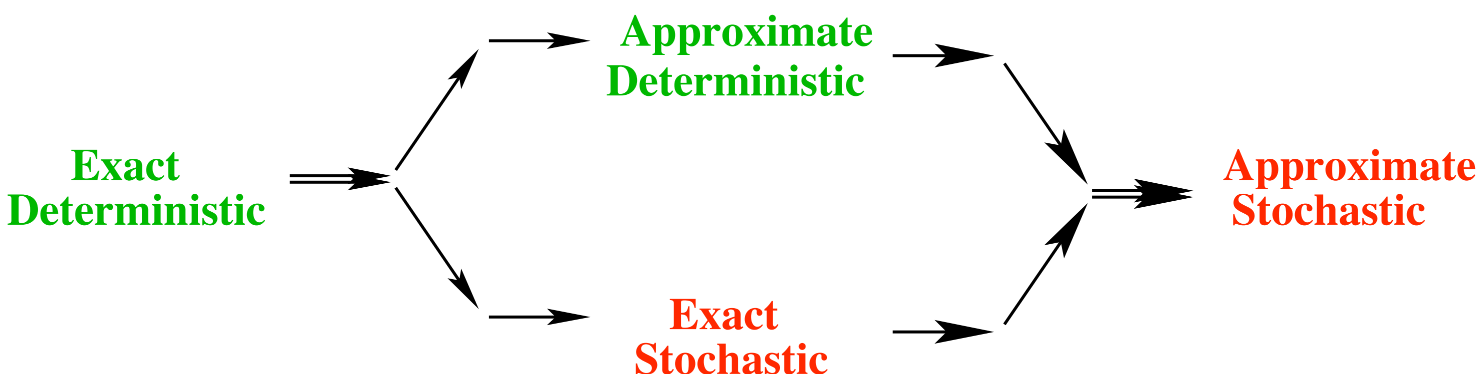
Assumptions:

- **Data:**

$$\tilde{w}(1), \tilde{w}(2), \ldots, \tilde{w}(t), \ldots \qquad w(t) \in \mathbb{R}^{\mathtt{W}} \qquad T \text{ infinite}$$

- **Deterministic SYSID**
- **I/O partition known if advantageous**
- **Exact modeling with an eye towards approximations**

From the simple to the complex!



The MPUM

The MPUM

- A model:= a subset $\mathfrak{B} \subseteq (\mathbb{R}^{\mathtt{w}})^{\mathbb{N}}$, the 'behavior'

  A family of (vector) time series

The MPUM

- A **model**:= a subset $\mathfrak{B} \subseteq (\mathbb{R}^{\mathtt{w}})^{\mathbb{N}}$, the **'behavior'**
- $\mathfrak{B}$ is **unfalsified by $\tilde{w}$** $:\Leftrightarrow \tilde{w} \in \mathfrak{B}$

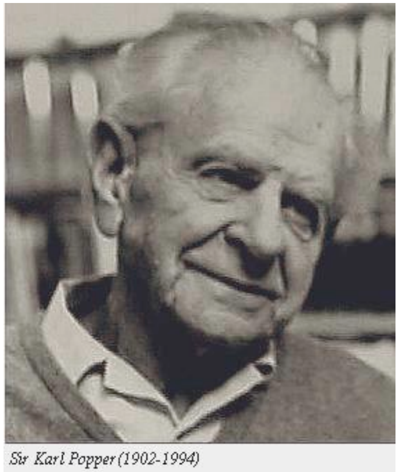
$$\tilde{w} = (\tilde{w}(1), \tilde{w}(2), \ldots, \tilde{w}(t), \ldots)$$

The MPUM

- A model:= a subset $\mathfrak{B} \subseteq (\mathbb{R}^{\mathtt{w}})^{\mathbb{N}}$, the 'behavior'
- $\mathfrak{B}$ is unfalsified by $\tilde{w}$ $:\Leftrightarrow \tilde{w} \in \mathfrak{B}$
- $\mathfrak{B}_1$ is more powerful than $\mathfrak{B}_2$ $:\Leftrightarrow \mathfrak{B}_1 \subset \mathfrak{B}_2$

Every model is prohibition.

The more a model forbids, the better it is.



Karl Popper
(1902-1994)

The MPUM

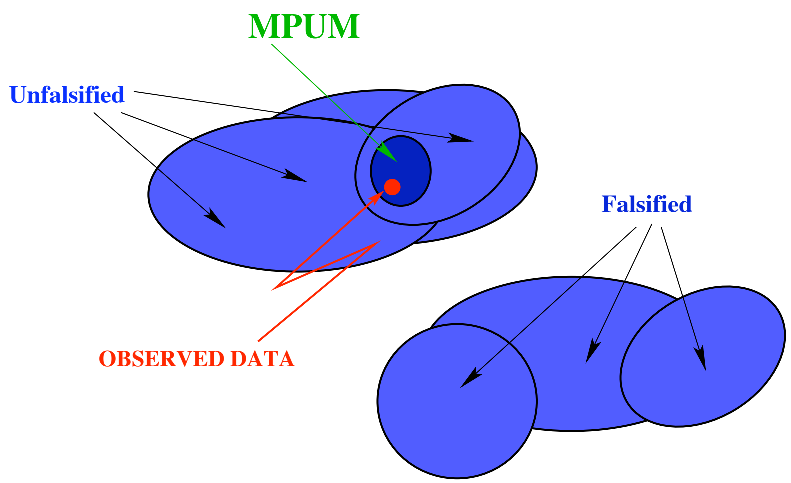
- A model:= a subset $\mathfrak{B} \subseteq (\mathbb{R}^{\mathtt{w}})^{\mathbb{N}}$, the 'behavior'
- $\mathfrak{B}$ is unfalsified by $\tilde{w}$ $:\Leftrightarrow \tilde{w} \in \mathfrak{B}$
- $\mathfrak{B}_1$ is more powerful than $\mathfrak{B}_2$ $:\Leftrightarrow \mathfrak{B}_1 \subset \mathfrak{B}_2$
- A model class: a family, $\mathbb{B}$, of models

The MPUM

- A **model**:= a subset $\mathfrak{B} \subseteq (\mathbb{R}^{\mathtt{w}})^{\mathbb{N}}$, the **'behavior'**
- $\mathfrak{B}$ is **unfalsified by** $\tilde{w}$ $:\Leftrightarrow \tilde{w} \in \mathfrak{B}$
- $\mathfrak{B}_1$ **is more powerful than** $\mathfrak{B}_2$ $:\Leftrightarrow \mathfrak{B}_1 \subset \mathfrak{B}_2$
- A **model class:** a family, $\mathbb{B}$, of models
- The **MPUM** **'most powerful unfalsified model'** in $\mathbb{B}$ for $\tilde{w}$, denoted $\mathfrak{B}^*_{\tilde{w}}$:
  1. $\mathfrak{B}^*_{\tilde{w}} \in \mathbb{B}$
  2. $\tilde{w} \in \mathfrak{B}^*_{\tilde{w}}$
  3. $\mathfrak{B} \in \mathbb{B}$ and $\tilde{w} \in \mathfrak{B} \Rightarrow \mathfrak{B}^*_{\tilde{w}} \subseteq \mathfrak{B}$

The MPUM

- A **model**:= a subset $\mathfrak{B} \subseteq (\mathbb{R}^{\mathtt{w}})^{\mathbb{N}}$, the **'behavior'**
- $\mathfrak{B}$ is **unfalsified by $\tilde{w}$** $:\Leftrightarrow \tilde{w} \in \mathfrak{B}$
- **$\mathfrak{B}_1$ is more powerful than $\mathfrak{B}_2$** $:\Leftrightarrow \mathfrak{B}_1 \subset \mathfrak{B}_2$
- A **model class:** a family, $\mathbb{B}$, of models
- The **MPUM** **'most powerful unfalsified model'** in $\mathbb{B}$ for $\tilde{w}$, denoted $\mathfrak{B}^*_{\tilde{w}}$



The MPUM

- A **model**:= a subset $\mathfrak{B} \subseteq (\mathbb{R}^{\mathtt{w}})^{\mathbb{N}}$, the **'behavior'**
- $\mathfrak{B}$ is **unfalsified by $\tilde{w}$** $:\Leftrightarrow \tilde{w} \in \mathfrak{B}$
- **$\mathfrak{B}_1$ is more powerful than $\mathfrak{B}_2$** $:\Leftrightarrow \mathfrak{B}_1 \subset \mathfrak{B}_2$
- A **model class:** a family, $\mathbb{B}$, of models
- The **MPUM** **'most powerful unfalsified model'** in $\mathbb{B}$ for $\tilde{w}$, denoted $\mathfrak{B}^*_{\tilde{w}}$
- **Given $\tilde{w}$ and $\mathbb{B}$, does $\mathfrak{B}^*_{\tilde{w}}$ exist?**
- 'Exact' SYSID: Construct algorithms $\tilde{w} \mapsto \mathfrak{B}^*_{\tilde{w}}$

The model class

The model class $\mathfrak{L}^{\mathtt{W}}$

We now define our model class (a family of subsets of $\left(\mathbb{R}^{\mathtt{w}}\right)^{\mathbb{N}}$).

It is an exceedingly familiar one: $\mathfrak{L}^{\mathtt{W}}$.

$\mathfrak{B} \subseteq \left(\mathbb{R}^{\mathtt{w}}\right)^{\mathbb{N}}$ belongs to $\mathfrak{L}^{\mathtt{W}}$ $:\Leftrightarrow$

The model class $\mathfrak{L}^{\mathtt{w}}$

$\mathfrak{B} \subseteq (\mathbb{R}^{\mathtt{w}})^{\mathbb{N}}$ **belongs to** $\mathfrak{L}^{\mathtt{w}}$ $:\Leftrightarrow$

- $\mathfrak{B}$ **is linear, shift-invariant, and closed**

  shift-invariant $:\Leftrightarrow \sigma\mathfrak{B} \subseteq \mathfrak{B}$

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

- $\mathfrak{B}$ **is linear, time-invariant, and complete** $:\Leftrightarrow$ **'prefix determined'**

The model class $\mathfrak{L}^{\mathtt{w}}$

$\mathfrak{B} \subseteq (\mathbb{R}^{\mathtt{w}})^{\mathbb{N}}$ belongs to $\mathfrak{L}^{\mathtt{w}}$ $:\Leftrightarrow$

- **$\mathfrak{B}$ is linear, shift-invariant, and closed**
- **$\mathfrak{B}$ is linear, time-invariant, and complete**
- **$\exists$ matrices $R_0, R_1, \ldots, R_L$ such that $\mathfrak{B}$ consists of all w that satisfy**

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

In the obvious polynomial matrix notation

$$R(\sigma) w = 0$$

The model class $\mathfrak{L}^{\mathtt{w}}$

$\mathfrak{B} \subseteq (\mathbb{R}^{\mathtt{w}})^{\mathbb{N}}$ **belongs to** $\mathfrak{L}^{\mathtt{w}}$ $:\Leftrightarrow$

- $\mathfrak{B}$ **is linear, shift-invariant, and closed**
- $\mathfrak{B}$ **is linear, time-invariant, and complete**
- $$R(\sigma)w = 0$$
- **Including input/output partition**

$$P(\sigma)y = Q(\sigma)u, \quad w \cong \begin{bmatrix} u \\ y \end{bmatrix}$$

$\det(P) \neq 0$, $\mathtt{m}$ **inputs,** $\mathtt{p}$ **outputs (= # of equations)**

The model class $\mathfrak{L}^{\mathtt{w}}$

$\mathfrak{B} \subseteq (\mathbb{R}^{\mathtt{w}})^{\mathbb{N}}$ belongs to $\mathfrak{L}^{\mathtt{w}}$ $:\Leftrightarrow$

- $\mathfrak{B}$ is linear, shift-invariant, and closed
- $\mathfrak{B}$ is linear, time-invariant, and complete
-

$$R(\sigma)w = 0$$

-

$$P(\sigma)y = Q(\sigma)u, \quad w \cong \begin{bmatrix} u \\ y \end{bmatrix}$$

- $\exists$ matrices A, B, C, D such that
  $\mathfrak{B}$ consists of all $w's$ generated by
  $\sigma x = Ax + Bu, \; y = Cx + Du, \quad w \cong \begin{bmatrix} u \\ y \end{bmatrix}$

The lag

$\mathtt{L} : \mathfrak{L}^{\mathtt{w}} \to \mathbb{Z}_+,$

$\mathtt{L}(\mathfrak{B}) =$ **smallest** L **such that there is a kernel representation:**

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

Polynomial matrix in

$$R(\sigma)w = 0$$

has $\text{degree}(R) \leq \mathtt{L}$**.**

The MPUM in $\mathfrak{L}^{\mathtt{w}}$

Theorem: For infinite observation interval, $T = \infty$ (our case), the MPUM for $\tilde{w}$ in $\mathfrak{L}^{\mathtt{w}}$ exists.

In fact,

$$\mathfrak{B}^*_{\tilde{w}} = \text{span}(\{\tilde{w}, \sigma\tilde{w}, \sigma^2\tilde{w}, \ldots\})^{\text{closure}}$$

We are looking for effective computational algorithms to go from $\tilde{w}$ to (a representation of) $\mathfrak{B}^*_{\tilde{w}}$,

e.g., a kernel repr. $\rightsquigarrow$ the corresponding R;

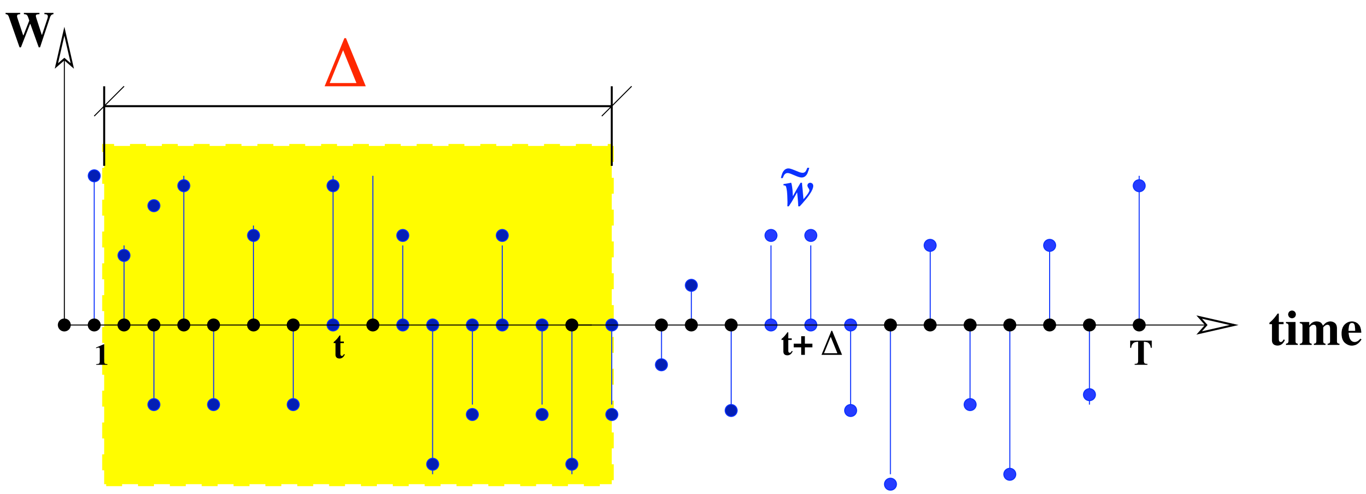
e.g., the matrices $\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$ of an i/s/o representation of $\mathfrak{B}^*_{\tilde{w}}$.

From data to kernel representation

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

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

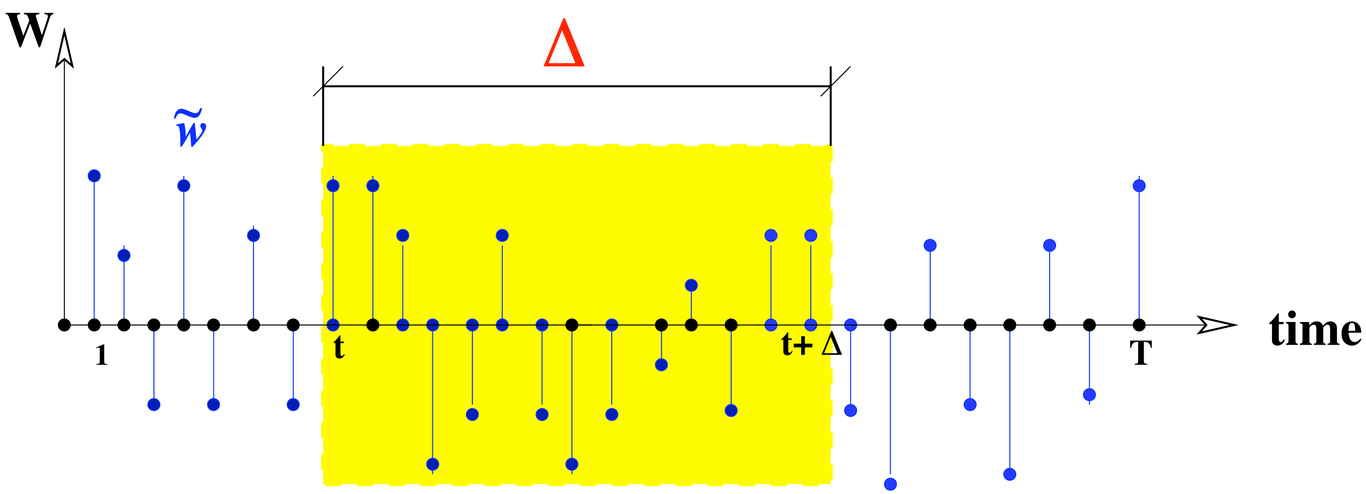
Basic idea: look through the window (with $\Delta > L$) in order to discover the system laws.



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

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

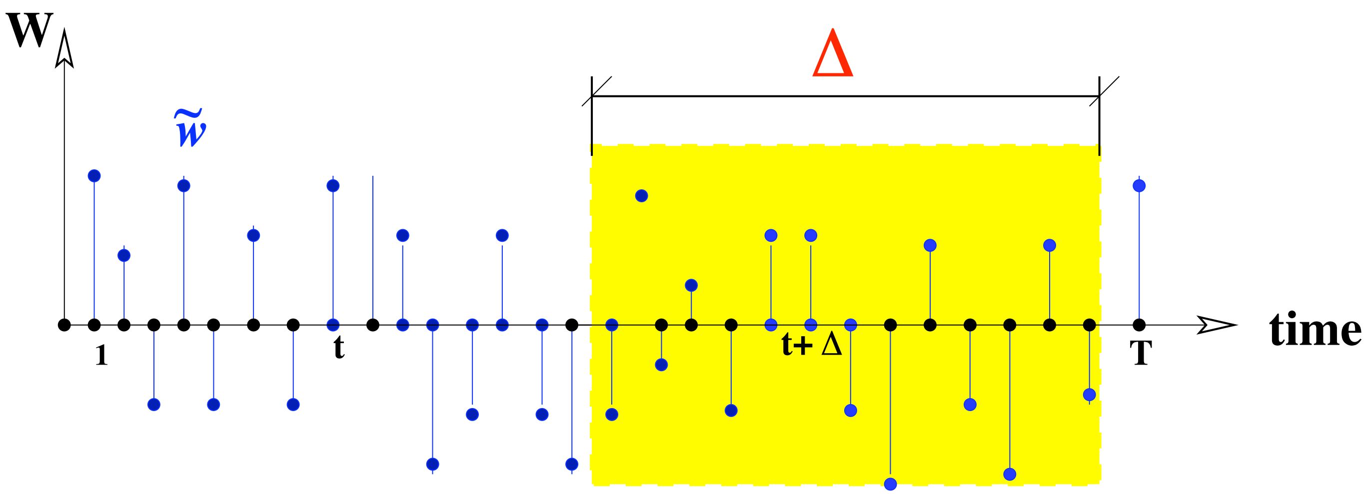
Basic idea: look through the window (with $\Delta > L$) in order to discover the system laws.



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

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

Basic idea: look through the window (with $\Delta > L$) in order to discover the system laws.



Is there a recursion, same for all these windows?

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

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

Basic idea: look through the window (with $\Delta > L$) in order to discover the system laws.

The windows lead linea recta to the Hankel matrix

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

and finding the vectors $[a_0 \; a_1 \; \cdots \; a_{\Delta-1}]$ in its left kernel

$\tilde{w} \mapsto R$

The windows lead linea recta to the Hankel matrix

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

The problem of computing the left kernel of this Hankel matrix has been studied in many aspects

- Recursively in T (Berlekamp-Massey, Antoulas, Kuijper, Polderman, e.a)
- Recursively in Δ
- A set of generators for the module generated by the left kernel
- Approximately
- Consistency aspects and persistency of excitation

From data to state representation

$$\tilde{w} \mapsto \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$$

Of course, once we have $\mathfrak{B}^*_{\tilde{w}}$, we can analyze it, make an input/output partition, make an observable state representation

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t), \qquad w(t) \cong \begin{bmatrix} u(t) \\ y(t) \end{bmatrix} \end{aligned}$$

and compute the state trajectory

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

corresponding to

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

$$\tilde{w} \mapsto \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$$

Of course, once we have $\mathfrak{B}^*_{\tilde{w}}$, we can analyze it, make an input/output partition, make an observable state representation

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t), \qquad w(t) \cong \begin{bmatrix} u(t) \\ y(t) \end{bmatrix} \end{aligned}$$

Of course,

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

$$\tilde{w} \mapsto \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$$

Of course,

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

But if we could go the other way:

first compute the state trajectory, directly from the data,
then this equation provides a way of

identifying the parameters $\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$ of $\mathfrak{B}^*_{\tilde{w}}$!

$$\tilde{w} \mapsto \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$$

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

This yields indeed a very attractive SYSID procedure:

- **Truncation at some sufficiently large t**
- **Model reduce using SVD or one of its friends by lowering the row dimension of**

$$\begin{bmatrix} \tilde{x}(1) & \tilde{x}(2) & \cdots & \tilde{x}(t) & \cdots \end{bmatrix}$$

- **Solving for $\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$ using Least Squares**

$\rightsquigarrow$ **'Subspace ID', oblique projection, etc.,: championed by Bart De Moor and Peter Van Overschee.**

$$\tilde{w} \mapsto \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$$

$$\begin{bmatrix} \tilde{\boldsymbol{x}}(\mathbf{2}) & \tilde{\boldsymbol{x}}(\mathbf{3}) & \cdots & \tilde{\boldsymbol{x}}(\boldsymbol{t}+\mathbf{1}) & \cdots \\ \tilde{\boldsymbol{y}}(\mathbf{1}) & \tilde{\boldsymbol{y}}(\mathbf{2}) & \cdots & \tilde{\boldsymbol{y}}(\boldsymbol{t}) & \cdots \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{x}}(\mathbf{1}) & \tilde{\boldsymbol{x}}(\mathbf{2}) & \cdots & \tilde{\boldsymbol{x}}(\boldsymbol{t}) & \cdots \\ \tilde{\boldsymbol{u}}(\mathbf{1}) & \tilde{\boldsymbol{u}}(\mathbf{2}) & \cdots & \tilde{\boldsymbol{u}}(\boldsymbol{t}) & \cdots \end{bmatrix}$$

Note that classical realization theory is a special case: data is impulse response.

$$\tilde{w} \mapsto \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$$

How does this work?

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

$$\Downarrow$$

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

This is a very nice system theoretic question.

From data to state representation

$\tilde{w} \mapsto \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$ by past/future intersection

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

$\tilde{w} \mapsto \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$ by past/future intersection

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

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

$$\left[\begin{array}{ccccc} \tilde{x}(\Delta) & \tilde{x}(\Delta+1) & \cdots & \tilde{x}(t+\Delta-1) & \cdots \end{array}\right] \quad \leftarrow \text{PRESENT STATE}$$

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

$\tilde{w} \mapsto \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$ by oblique projection

Solve for G

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

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

$\cong$ ‘oblique projection

Computes $\tilde{x}$!

$$\tilde{w} \mapsto \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$$

These algorithms do not make use of the Hankel structure.

Recent development: uses the Hankel structure, together with shift-and-cut state construction algorithm.

$\tilde{w} \mapsto \begin{bmatrix} A & B \\ C & D \end{bmatrix}$ via left annihilators

Implementation. Compute 'the' left annihilators of $\mathcal{H}$:

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

$\tilde{w} \mapsto \begin{bmatrix} A & B \\ C & D \end{bmatrix}$ via left annihilators

Implementation. Compute 'the' left annihilators of $\mathcal{H}$:

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

Then

$$\begin{bmatrix} \tilde{x}(1) & \tilde{x}(2) & \cdots & \tilde{x}(t) & \cdots \end{bmatrix}$$

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

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

It actually suffices to compute a set of generators for the module generated by the left kernel.

- **Truncation at some sufficiently large t**
- **Model reduce using SVD or one of its friends by lowering the row dimension of**

$$\begin{bmatrix} \tilde{\boldsymbol{x}}(1) & \tilde{\boldsymbol{x}}(2) & \cdots & \tilde{\boldsymbol{x}}(t) & \cdots \end{bmatrix}$$

- **Solving for $\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$ using Least Squares**

Open question: Construct a balanced state trajectory from data.

Shift-and-cut

State maps

Problem: Given a behavior, for example, a kernel representation

$$R\left(\frac{d}{dt}\right)w = 0, \qquad R(\sigma)w = 0$$

say known i/o $w \cong (u, y)$, find a state repr. $\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$ from R. I.e.

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t), \qquad w(t) \cong \begin{bmatrix} u(t) \\ y(t) \end{bmatrix} \end{aligned}$$

or

$$Gw(t) + Hx(t) + Fx(t+1) = 0$$

with same w-behavior.
This is the classical problem of realization, where the impulse response case has dominated the scene.

State maps

Paolo Rapisarda's Ph.D. thesis: start from any representation.

Key Idea: Construct first a state map : $x = X(\sigma)w$ for a suitable polynomial matrix X, get $\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$ from (R, X).

State maps

Define the **'shift-and-cut'** operator σ on $\mathbb{R}[\xi]$ as follows:

$$\sigma : p_0 + p_1\xi + \cdots + p_{n-1}\xi^{n-1} + p_n\xi^n \mapsto p_1 + p_2\xi + \cdots + p_{n-1}\xi^{n-2} + p_n\xi^{n-1}$$

Extend-able in the obvious term-by-term way to $\mathbb{R}^{\bullet\times\bullet}[\xi]$.

Repeated use of the cut-and-shift on $P \in \mathbb{R}^{\bullet\times\bullet}$ yields the *'stack' operator* Σ_P, defined by

$$\Sigma_P := \begin{bmatrix} \sigma(P) \\ \sigma^2(P) \\ \vdots \\ \sigma^{\text{degree}(P)}(P) \end{bmatrix}$$

State maps

Construction of state map by cut-and-shift and stack operators:

Theorem: Let $R(\sigma)w = 0$ be a kernel representation of $\mathfrak{B} \in \mathfrak{L}^{\mathtt{w}}$. Then $\Sigma_R(\sigma)$ is a state map for $\mathfrak{B}$.

The resulting state representation

$$\boxed{R(\sigma)w = 0\,; \quad x = \Sigma_R(\sigma)w}$$

need not be minimal.

$\exists$ reduction algorithms.

The third algorithm implements this on an observed time-series.

Performance

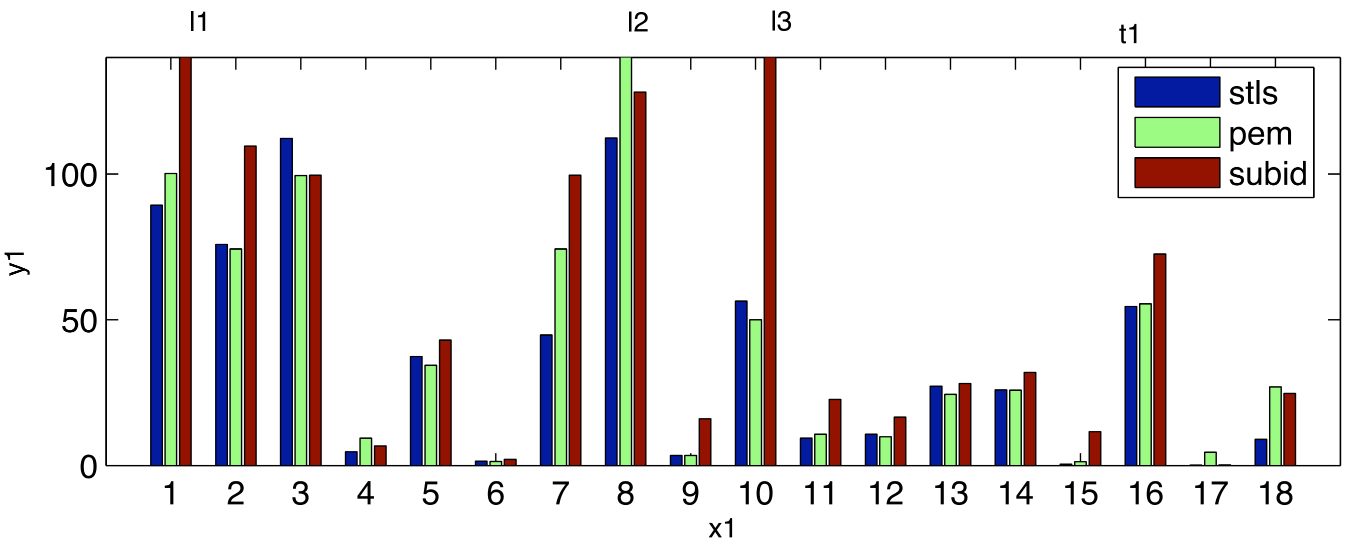
Performance

#	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

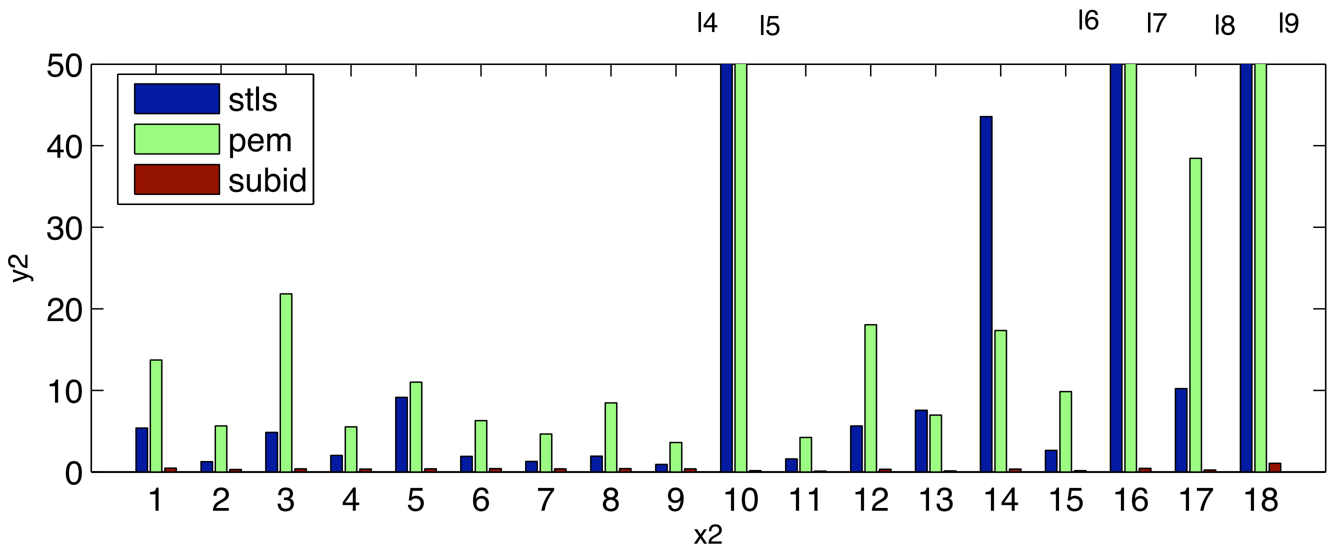
Compare the misfit on the last 30% of the outputs and the execution time for computing the ID model from the first 70% of the data.

Misfit

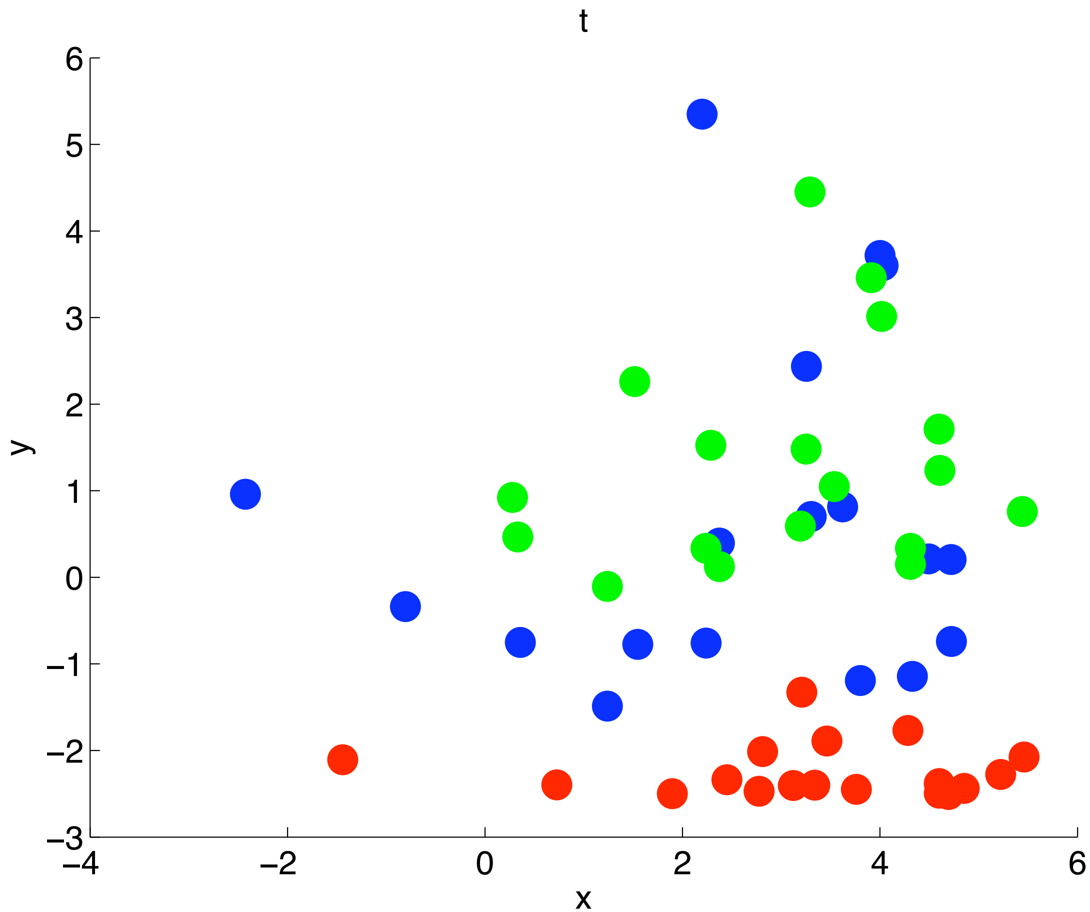


Performance

Execution time



Performance



Conclusions

Conclusions

- **Deterministic SYSID: possible**

Conclusions

- **Deterministic SYSID: possible**
- **MPUM: elegant**

Conclusions

- **Deterministic SYSID: possible**
- **MPUM: elegant**
- **$\exists$ algorithms to compute the MPUM from data: feasible**

Conclusions

- **Deterministic SYSID: possible**
- **MPUM: elegant**
- $\exists$ **algorithms to compute the MPUM from data: feasible**
- **Direct state construction from data: clever and useful**

Details & copies of the lecture frames are available from/at

Jan.Willems@esat.kuleuven.be

http://www.esat.kuleuven.be/∼jwillems

Details & copies of the lecture frames are available from/at

`Jan.Willems@esat.kuleuven.be`

`http://www.esat.kuleuven.be/∼jwillems`

Thank you

Thank you

Thank you

Thank you

Thank you

Thank you

Thank you

Thank you