ALGORITHMS FOR EXACT AND APPROXIMATE IDENTIFICATION FROM FINITE TIME SERIES

Jan C. Willems
K.U. Leuven

Kyoto University

May 17, 2005
On-going joint research with

Ivan Markovsky (K.U. Leuven)
Paolo Rapisarda (Un. Maastricht)
& Bart De Moor (K.U. Leuven)
This is a very rich area. It involves

- Algorithms:
  - Numerical data $\rightarrow$ model parameters

- ‘Philosophical’ issues:
  - How to deal with uncertainty
  - Role of stochasticity
  - How to deal with ‘open’ systems, etc.

- Important area for applications, because of its relevance in modeling
Case of interest today

**Data:** an ‘observed’ vector time-series

$$\tilde{w}(1), \tilde{w}(2), \ldots, \tilde{w}(T) \quad \tilde{w}(t) \in \mathbb{R}^w, \ T \text{ finite}$$

$$\downarrow$$

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 mainly the case:

**‘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

- Exact
- Deterministic
- Approximate
- Stochastic
- Approximate
- Deterministic
- Exact
- Stochastic
Towards the end, some remarks on **ID with latent inputs**

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

\[ \tilde{w}(1), \tilde{w}(2), \ldots, \tilde{w}(T) \mapsto (\hat{R}(\xi), \hat{M}(\xi)) \]
Basic idea: look through the window (with $\Delta > L$) in order to discover the system laws.

$\tilde{w}(1), \tilde{w}(2), \ldots, \tilde{w}(T)$
Basic idea: look through the window (with $\Delta > L$) in order to discover the system laws.

$\tilde{w}(1), \tilde{w}(2), \ldots, \tilde{w}(T)$
Basic idea: look through the window (with $\Delta > L$) in order to discover the system laws.

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

Is there a recursion, same for all these windows?
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

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

\[
\begin{bmatrix}
\tilde{w}(1) & \tilde{w}(2) & \cdots & \tilde{w}(t) & \cdots & \tilde{w}(t - \Delta) \\
\tilde{w}(2) & \tilde{w}(3) & \cdots & \tilde{w}(t + 1) & \cdots & \tilde{w}(t - \Delta + 1) \\
\tilde{w}(3) & \tilde{w}(4) & \cdots & \tilde{w}(t + 2) & \cdots & \tilde{w}(t - \Delta + 2) \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
\tilde{w}(\Delta + 1) & \tilde{w}(\Delta + 2) & \cdots & \tilde{w}(t + \Delta) & \cdots & \tilde{w}(T)
\end{bmatrix}
\]
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**

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

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

Are there **left annihilitors**, or approximate, or up to a stochastic interpretation, same for all these columns?
Basic idea: look through the window (with $\Delta > L$) in order to discover the system laws.

But first, some language: What do we mean by a model, a model class, an unfalsified model, etc.?
The MPUM
A model := a subset $\mathcal{B} \subseteq (\mathbb{R}^w)^N$, the `behavior'

A family of (vector) time series

Recall notation $\mathcal{B}_{[1,T]}$

$:= \text{all} \text{ `prefixes'} \quad w(1), w(2), \cdots, w(T) \quad \text{of} \quad w \in \mathcal{B}$
A model := a subset $\mathcal{B} \subseteq (\mathbb{R}^w)^N$, the ‘behavior’

$\mathcal{B}$ is unfalsified by $\tilde{w} := \tilde{w}(1), \tilde{w}(2), \ldots, \tilde{w}(T)$

$\iff \tilde{w} \in \mathcal{B}_{[0,t]}$
The MPUM

- A model := a subset $\mathcal{B} \subseteq (\mathbb{R}^w)^N$, the ‘behavior’
- $\mathcal{B}$ is unfalsified by $\tilde{w}$ : $\iff \tilde{w} \in \mathcal{B}|[0,t]$
- $\mathcal{B}_1$ is more powerful than $\mathcal{B}_2$ : $\iff \mathcal{B}_1 \subseteq \mathcal{B}_2$

Every model is prohibition.
The more a model forbids, the better it is.

Karl Popper
(1902-1994)
A model := a subset $\mathcal{B} \subseteq (\mathbb{R}^w)^N$, the ‘behavior’

$\mathcal{B}$ is unfalsified by $\tilde{w}$ $\iff \tilde{w} \in \mathcal{B}[0,t]$

$\mathcal{B}_1$ is more powerful than $\mathcal{B}_2$ $\iff \mathcal{B}_1 \subseteq \mathcal{B}_2$

A model class: a family, $\mathcal{B}$, of models
A model := a subset $\mathcal{B} \subseteq (\mathbb{R}^w)^N$, the ‘behavior’

$\mathcal{B}$ is unfalsified by $\tilde{w}$ : $\iff \tilde{w} \in \mathcal{B}_{[0,t]}$

$\mathcal{B}_1$ is more powerful than $\mathcal{B}_2$ : $\iff \mathcal{B}_1 \subset \mathcal{B}_2$

A model class: a family, $\mathcal{B}$, of models

The MPUM ‘most powerful unfalsified model’

in $\mathcal{B}$ for $\tilde{w}$, denoted $\mathcal{B}^*_{\tilde{w}}$ :

1. $\mathcal{B}^*_{\tilde{w}} \in \mathcal{B}$
2. $\tilde{w} \in \mathcal{B}^*_{\tilde{w}} \mid [1,T]$
3. $\mathcal{B} \in \mathcal{B}$ and $\tilde{w} \in \mathcal{B}_{[1,T]} \Rightarrow \mathcal{B}^*_{\tilde{w}} \subset \mathcal{B}$
The MPUM

- A model: a subset \( \mathcal{B} \subseteq (\mathbb{R}^w)^N \), the ‘behavior’
- \( \mathcal{B} \) is unfalsified by \( \tilde{w} \) \( \iff \tilde{w} \in \mathcal{B}_{[0,t]} \)
- \( \mathcal{B}_1 \) is more powerful than \( \mathcal{B}_2 \) \( \iff \mathcal{B}_1 \subseteq \mathcal{B}_2 \)
- A model class: a family, \( \mathcal{B} \), of models
- The MPUM ‘most powerful unfalsified model’
  in \( \mathcal{B} \) for \( \tilde{w} \), denoted \( \mathcal{B}_{\tilde{w}}^* \)
- Given \( \tilde{w} \) and \( \mathcal{B} \), does \( \mathcal{B}_{\tilde{w}}^* \) exist?
The MPUM

MPUM

Unfalsified

Falsified

OBSERVED DATA
The model class
The model class \( \mathcal{L}^w \)

Our model class (a family of subsets of \( (\mathbb{R}^w)^N \)).

It is an exceedingly familiar one. First, \( \mathcal{L}^w \).

\[ B \subseteq (\mathbb{R}^w)^N \text{ belongs to } \mathcal{L}^w : \iff \]
The model class $\mathcal{L}^w$

$\mathcal{B} \subseteq (\mathbb{R}^w)^N$ belongs to $\mathcal{L}^w$ if $\mathcal{B}$ is linear, shift-invariant, and closed.

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

shift-invariant $\iff \sigma \mathcal{B} \subseteq \mathcal{B}$

$\sigma$ = the ‘shift’: $(\sigma f)(t) := f(t + 1)$. 
The model class $\mathcal{L}^w$

$\mathcal{B} \subseteq (\mathbb{R}^w)^N$ belongs to $\mathcal{L}^w$ if:

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

$$R_0w(t) + R_1w(t+1) + \cdots + R_Lw(t+L) = 0$$

In obvious polynomial matrix notation

$$R(\sigma)w = 0$$
The model class $\mathcal{L}^w$

$\mathcal{B} \subseteq (\mathbb{R}^w)^N$ belongs to $\mathcal{L}^w : \iff$

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

$R(\sigma)w = 0$

- Including input/output partition

$$P(\sigma)y = Q(\sigma)u, \quad w \cong [u \ y]$$

$\det(P) \neq 0$, $m$ inputs, $p$ outputs ($= \# \text{ of equations}$)
The model class $\mathcal{L}^w$

$\mathcal{B} \subseteq (\mathbb{R}^w)^N$ belongs to $\mathcal{L}^w$ if

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

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

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

- There exist matrices $A, B, C, D$ such that
  $\mathcal{B}$ consists of all $w$'s generated by
  $$\sigma x = Ax + Bu, \quad y = Cx + Du, \quad w \cong \begin{bmatrix} u \\ y \end{bmatrix}$$
Let $\mathcal{B} \in \mathcal{L}^w$. Define its annihilators by

$$\mathcal{N}_\mathcal{B} := \{ n \in \mathbb{R}^w[\xi] \mid n^\top \left( \frac{d}{dt} \right) \mathcal{B} = 0 \}$$

Note: $\mathcal{N}_\mathcal{B}$ is a $\mathbb{R}[\xi]$ sub-module of $\mathbb{R}^w[\xi]$. Means:

$$n_1, n_2 \in \mathbb{R}^w[\xi], p \in \mathbb{R}[\xi]$$

$$\Rightarrow n_1 + n_2 \in \mathcal{N}_\mathcal{B}, p n_1 \in \mathcal{N}_\mathcal{B}$$
The module structure

Let $\mathcal{B} \in \mathcal{L}^w$. Define its annihilators by

$$\mathcal{N}_\mathcal{B} := \{ n \in \mathbb{R}^w[\xi] | n^\top \left( \frac{d}{dt} \right) \mathcal{B} = 0 \}$$

Note: $\mathcal{N}_\mathcal{B}$ is a $\mathbb{R}[\xi]$ sub-module of $\mathbb{R}^w[\xi]$. In fact,

$$\mathcal{L}^w \overset{\text{one-to-one}}{\longleftrightarrow} \text{sub-modules of } \mathbb{R}^w[\xi]$$

Consequence: since sub-module is finitely generated, $\mathcal{B}$ is determined by finite number of generators.

For example, the rows of $\mathcal{R}$, but this is non-unique.
The model class $\mathcal{L}^w_L$.

We now define our model class $\mathcal{L}^w_L$. It consists of all $\mathcal{B} \in \mathcal{L}^w$ such that

$\exists$ matrices $R_0, R_1, \ldots, R_L$

with restricted lag: $L \leq L$

such that $\mathcal{B}$ consists of all $w$ that satisfy

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

Polynomial matrix in

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

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

– p.14/33
For infinite observation interval, \( T = \infty \), the MPUM for \( \tilde{w} \) in \( \mathcal{L}_L^w \) always exists.

In fact, it equals

\[
\mathcal{B}_{\tilde{w}}^* = \text{span}(\{\tilde{w}, \sigma \tilde{w}, \sigma^2 \tilde{w}, \ldots\})^{\text{closure}}
\]

\( \exists \) effective computational algorithms to go from \( \tilde{w} \) to the corresponding \( R \).
For finite observation interval, $T < \infty$, the MPUM in $\mathcal{L}_L^w$ is not very useful.

We hence restrict attention to the MPUM in $\mathcal{L}_L^w$.

Also here the MPUM may not exist. Example:

$$\tilde{w} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 2 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

has no MPUM in $\mathcal{L}_2^w$. What is the issue?
The MPUM in $\mathcal{L}_L^w$ is equivalent to the left kernel of the Hankel matrix (‘windows’):

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

This must have a ‘module-like’ structure, i.e.

$$
\begin{bmatrix}
N_0 & N_1 & \cdots & N_{L-1} & 0
\end{bmatrix}
$$

in left kernel

$$
\Rightarrow
\begin{bmatrix}
0 & N_0 & \cdots & N_{L-2} & N_{L-1}
\end{bmatrix}
$$
in left kernel
Proposition: the MPUM in $\mathcal{L}_L^v$ exits if

$$\text{rank} \begin{bmatrix} \tilde{w}(1) & \tilde{w}(2) & \cdots & \tilde{w}(T - L) \\ \tilde{w}(2) & \tilde{w}(3) & \cdots & \tilde{w}(T - L + 1) \\ \tilde{w}(3) & \tilde{w}(4) & \cdots & \tilde{w}(T - L + 2) \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{w}(L) & \tilde{w}(L + 1) & \cdots & \tilde{w}(T - 1) \end{bmatrix}$$

$$= \text{rank} \begin{bmatrix} \tilde{w}(1) & \tilde{w}(2) & \cdots & \tilde{w}(T - L) & \tilde{w}(T - L + 1) \\ \tilde{w}(2) & \tilde{w}(3) & \cdots & \tilde{w}(T - L + 1) & \tilde{w}(T - L + 2) \\ \tilde{w}(3) & \tilde{w}(4) & \cdots & \tilde{w}(T - L + 2) & \tilde{w}(T - L + 3) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \tilde{w}(L) & \tilde{w}(L + 1) & \cdots & \tilde{w}(T - 1) & \tilde{w}(T) \end{bmatrix}$$

We henceforth assume this to be the case.
Computation of this MPUM
Recursive computation

We need to compute the left kernel of

\[
\begin{bmatrix}
\tilde{w}(1) & \tilde{w}(2) & \cdots & \tilde{w}(T-L-1) & \tilde{w}(T-L) \\
\tilde{w}(2) & \tilde{w}(3) & \cdots & \tilde{w}(T-L) & \tilde{w}(T-L+1) \\
\tilde{w}(3) & \tilde{w}(4) & \cdots & \tilde{w}(T-L+1) & \tilde{w}(T-L+2) \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
\tilde{w}(L+1) & \tilde{w}(L+2) & \cdots & \tilde{w}(T-1) & \tilde{w}(T)
\end{bmatrix}
\]

Suffices to compute a set of generators of the sub-module of annihilators of the MPUM. Also, we would like to do this computation

recursively and approximately.
Idea derived from the case $T = \infty$.

Assume time-series data $\mathbb{D} = \{d_1, d_2, \cdots, d_N\}$, $d_k \in (\mathbb{R}^w)^N$.

! Compute the MPUM in $\mathcal{L}^w \sim$ polynomial matrix $R_\mathbb{D}$.

1. $R_0 = I$

2. from $R_k \mapsto R_{k+1}$:
   - Compute $e_{k+1} := R_k(\sigma)d_{k+1}$.
   - Compute $E_{k+1}$ corresponding to the MPUM of $e_{k+1}$
   - $R_{k+1} = E_{k+1}R_k$

3. $R_\mathbb{D} = R_N$

Reduces pbm to the computation of the MPUM for one time series.
Recursive in $T$

MPUM with one time-series, $d$, time-axis $\mathbb{N}$

$$d = (\cdots, d(t), \cdots, d(-1), d(0))$$

Use the previous algorithm with the time-series data

$$d_{-k} = (\cdots, d(-k-1), d(-k))$$, $-k \in \mathbb{N}$

1. $R_{k_0}$ given, say $I$

2. from $R_{-k} \mapsto R_{-k+1}$:
   - $e_{-k+1} := R_{-k}(\sigma^{-1})d_{-k+1}$. Looks as $(\cdots, 0, \cdots, 0, *)$
   - Compute $E_{-k+1}$ the MPUM of $e_{-k+1}$. Very simple!
   - $R_{-k+1} = E_{-k+1}R_{-k}$

3. $R\{d\} = R_0$
Recursive in $T$

In order to apply this to

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

we miss an initial condition. This may be circumvented by considering instead the extended time-series

$$\cdots, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} \tilde{w}(1) \\ 0 \end{bmatrix}, \begin{bmatrix} \tilde{w}(2) \\ 0 \end{bmatrix}, \cdots, \begin{bmatrix} \tilde{w}(T) \\ 0 \end{bmatrix}$$

and discarding certain of the relations obtained.

Can be implemented using approximate linear algebra computations.
We need to compute a ‘module basis’ of the left kernel of

\[
\begin{bmatrix}
\tilde{w}(1) & \tilde{w}(2) & \cdots & \tilde{w}(T - L - 1) & \tilde{w}(T - L) \\
\tilde{w}(2) & \tilde{w}(3) & \cdots & \tilde{w}(T - L) & \tilde{w}(T - L + 1) \\
\tilde{w}(3) & \tilde{w}(4) & \cdots & \tilde{w}(T - L + 1) & \tilde{w}(T - L + 2) \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
\tilde{w}(L + 1) & \tilde{w}(L + 2) & \cdots & \tilde{w}(T - 1) & \tilde{w}(T)
\end{bmatrix}
\]
Consider the Hankel matrices

\[
\begin{bmatrix}
\tilde{w}(1) & \tilde{w}(2) & \cdots & \tilde{w}(T - \Delta - 2) & \tilde{w}(T - \Delta - 1) \\
\tilde{w}(2) & \tilde{w}(3) & \cdots & \tilde{w}(T - \Delta - 1) & \tilde{w}(T - \Delta) \\
\tilde{w}(3) & \tilde{w}(4) & \cdots & \tilde{w}(T - \Delta) & \tilde{w}(T - \Delta + 1) \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
\tilde{w}(\Delta) & \tilde{w}(\Delta + 1) & \cdots & \tilde{w}(T - 1) & \tilde{w}(T)
\end{bmatrix}
\]

and let $\Delta$ vary from 1 to $L + 1$. 
Recursive in annihilators

Basic idea.

Step 1: Compute (SVD) basis $R_0$ for left kernel of

$$[\bar{w}(1) \quad \bar{w}(2) \quad \ldots \quad \bar{w}(T-1) \quad \bar{w}(T)]$$

and its orthogonal complement $S_0$.

Keep $R_0$ as valid zero-th order laws, and replace $\bar{w}$ by

$$\bar{w}' = S_0 \bar{w} = (\bar{w}'(1), \bar{w}'(2), \ldots, \bar{w}'(T)), \quad \bar{w}'(t) \in \mathbb{R}^{w'}$$

This has no more zero-th order laws.
Step 2: (SVD) \( \mathbf{R}_1 = \begin{bmatrix} n_0 & n_1 \end{bmatrix} \), \( n_0, n_1 \in \mathbb{R}^{1 \times w'} \) in left kernel

\[
\begin{bmatrix}
\tilde{w}'(1) & \tilde{w}'(2) & \cdots & \tilde{w}'(T - 2) & \tilde{w}'(T - 1) \\
\tilde{w}'(2) & \tilde{w}'(3) & \cdots & \tilde{w}'(T - 1) & \tilde{w}'(T)
\end{bmatrix}
\]

Organize \( \mathbf{R}_1 \) as the polynomial row vector

\[
n(\xi) = n_0 + n_1 \xi = \begin{bmatrix} r_1(\xi) & r_2(\xi) & \cdots & r_w(\xi) \end{bmatrix}
\]

Compute (Bézout) \( C \in \mathbb{R}^{(w' - 1) \times w'}[\xi] \) such that \( \begin{bmatrix} n[\xi] \\ C[\xi] \end{bmatrix} \) is unimodular.

Keep \( n \) as a valid first order law, and replace \( \tilde{w}' \) by

\[
\tilde{w}'' = C(\sigma)\tilde{w}' = (\tilde{w}''(1), \tilde{w}''(2), \ldots, \tilde{w}''(T - 1)) \quad \tilde{w}''(t) \in \mathbb{R}^{w' - 1}
\]

etc.
Both recursions can be combined, leading to very efficient ways of finding an MPUM.

This is effective for exact data (or in finite field case).
Behavior of the algorithm for $T$ large
Typical way of evaluating SYSID algorithms:

Assume that

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

is generated by an element of the model class.

Does the algorithm return the model that generated the data for large \( T \), or in the limit as \( T \to \infty \) (consistency)?
The MPUM in $\mathfrak{L}_L^w$ for

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

returns $\mathcal{B}$ if

1. $\tilde{w} \in \mathcal{B}_{[1,T]}$
2. $L$ is sufficiently large
3. $\mathcal{B}$ is controllable
4. the input component in $\tilde{w}$ is persistently exciting of sufficiently high order

The left kernel of the Hankel matrix is then module-like.
Assume \( \tilde{w} = (\tilde{u}, \tilde{y}) \) generated by behavior \( \mathcal{B} \). Then

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

has ‘correct’ kernel & image if

1. \( \Delta > \text{lag}(\mathcal{B}) \)

2. \( \mathcal{B} \) controllable

3. \( \tilde{u} \) is persistently exciting of order \( \Delta + n(\mathcal{B}) \)
Identifiability

\[
\begin{bmatrix}
\tilde{u}(1) & \tilde{u}(2) & \tilde{u}(3) & \cdots & \tilde{u}(T - L(\mathcal{B})) \\
\tilde{y}(1) & \tilde{y}(2) & \tilde{y}(3) & \cdots & \tilde{y}(T - L(\mathcal{B})) \\
\tilde{u}(2) & \tilde{u}(3) & \tilde{u}(4) & \cdots & \tilde{u}(T - L(\mathcal{B}) + 1) \\
\tilde{y}(2) & \tilde{y}(3) & \tilde{y}(4) & \cdots & \tilde{y}(T - L(\mathcal{B}) + 1) \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\tilde{u}(L(\mathcal{B}) + 1) & \tilde{u}(L(\mathcal{B}) + 2) & \tilde{u}(L(\mathcal{B}) + 3) & \cdots & \tilde{u}(T) \\
\tilde{y}(L(\mathcal{B}) + 1) & \tilde{y}(L(\mathcal{B}) + 2) & \tilde{y}(L(\mathcal{B}) + 3) & \cdots & \tilde{y}(T)
\end{bmatrix}
\]

kernel det. laws of the system (has rank \( m(\mathcal{B})(L(\mathcal{B}) + 1) + n(\mathcal{B}) \) if

\[
\begin{bmatrix}
\tilde{u}(1) & \tilde{u}(2) & \cdots & \tilde{u}(T - L(\mathcal{B}) - n(\mathcal{B}) - 1) \\
\tilde{u}(2) & \tilde{u}(3) & \cdots & \tilde{u}(T - L(\mathcal{B}) - n(\mathcal{B})) \\
\vdots & \vdots & \ddots & \vdots \\
\tilde{u}(L(\mathcal{B}) + n(\mathcal{B}) + 1) & \tilde{u}(L(\mathcal{B}) + n(\mathcal{B}) + 2) & \cdots & \tilde{u}(T)
\end{bmatrix}
\]

has rank \( m(\mathcal{B})(L(\mathcal{B}) + n(\mathcal{B}) + 1) \).
From the data to the state trajectory
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

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

These algorithms go to \((A, B, C, D)\) instead of to \(R\) or to \((P, Q)\). They have realization algorithms as a special case.
How does this work?

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

\[ \downarrow \]

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

Several algorithms. We give 3 of them.

Assume contr., \( \Delta > L(\mathcal{B}) \), and pers. of exc. as needed.
1. Compute ‘the’ left annihilators of $\mathcal{H}$:

\[
\begin{bmatrix}
N_1 & N_2 & N_3 & \cdots & N_\Delta \\
\tilde{w}(1) & \tilde{w}(2) & \cdots & \tilde{w}(T - \Delta + 1) \\
\tilde{w}(2) & \tilde{w}(3) & \cdots & \tilde{w}(T - \Delta + 2) \\
\tilde{w}(3) & \tilde{w}(4) & \cdots & \tilde{w}(T - \Delta + 3) \\
\vdots & \vdots & \vdots & \vdots \\
\tilde{w}(\Delta) & \tilde{w}(\Delta + 1) & \cdots & \tilde{w}(T)
\end{bmatrix}
= 0
\]
1. Compute ‘the’ left annihilators of $\mathcal{H}$:

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

Then

$$
\begin{bmatrix}
N_2 & N_3 & \cdots & N_\Delta & 0 \\
N_3 & N_4 & \cdots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
N_{\Delta - 1} & N_\Delta & \cdots & 0 & 0 \\
N_\Delta & 0 & \cdots & 0 & 0
\end{bmatrix}
\begin{bmatrix}
\tilde{x}(1) & \tilde{x}(2) & \cdots & \tilde{x}(T - \Delta + 1)
\end{bmatrix}
$$
\[
\begin{bmatrix}
\mathcal{H}_-
\end{bmatrix} =
\begin{bmatrix}
\tilde{w}(1) & \tilde{w}(2) & \cdots & \tilde{w}(T - 2\Delta + 1) \\
\tilde{w}(2) & \tilde{w}(3) & \cdots & \tilde{w}(T - 2\Delta + 2) \\
\vdots & \vdots & \ddots & \vdots \\
\tilde{w}(\Delta) & \tilde{w}(\Delta + 1) & \cdots & \tilde{w}(T - \Delta) \\
\tilde{w}(\Delta + 1) & \tilde{w}(\Delta + 2) & \cdots & \tilde{w}(T - \Delta + 1) \\
\tilde{w}(\Delta + 2) & \tilde{w}(\Delta + 3) & \cdots & \tilde{w}(T - \Delta + 2) \\
\vdots & \vdots & \ddots & \vdots \\
\tilde{w}(2\Delta) & \tilde{w}(2\Delta + 1) & \cdots & \tilde{w}(T)
\end{bmatrix}
\]

\[
\begin{bmatrix}
A & B \\
C & D
\end{bmatrix}
\]

\[\tilde{w} \mapsto \tilde{x} \mapsto \]

- PAST
- FUTURE

- ↑
- ↓

- p.24/33
2. The **intersection** of the span of the rows of $\mathcal{H}_-$

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

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

Nice num. impl. (e.g. via left kernel) $\sim$ **subspace ID**
3. Solve for $G$

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

G =

\begin{bmatrix}
\tilde{w}(1) & \cdots & \tilde{w}(T - 2\Delta + 1) \\
\vdots & \ddots & \vdots \\
\tilde{w}(\Delta) & \cdots & \tilde{w}(T - \Delta) \\
0 & \cdots & 0 \\
\vdots & \ddots & \vdots \\
0 & \cdots & 0
\end{bmatrix}
$$

$G$ computes $\tilde{x}$!

$\equiv$ 'oblique projection
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).

\[
\begin{align*}
\text{SVD} & \quad \tilde{X} = \begin{bmatrix} \tilde{x}(1) & \tilde{x}(2) & \cdots & \tilde{x}(T) \end{bmatrix} \\
\sim & \quad \tilde{X}^{\text{red}} = \begin{bmatrix} \tilde{x}^{\text{red}}(1) & \tilde{x}^{\text{red}}(2) & \cdots & \tilde{x}^{\text{red}}(T) \end{bmatrix}
\end{align*}
\]

followed by LS solution of

\[
\begin{bmatrix} \tilde{x}^{\text{red}}(2) & \tilde{x}^{\text{red}}(3) & \cdots & \tilde{x}^{\text{red}}(T) \\ \tilde{y}(1) & \tilde{y}(2) & \cdots & \tilde{y}(T - 1) \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} \tilde{x}^{\text{red}}(1) & \tilde{x}^{\text{red}}(2) & \cdots & \tilde{x}^{\text{red}}(T - 1) \\ \tilde{u}(1) & \tilde{u}(2) & \cdots & \tilde{u}(T - 1) \end{bmatrix}
\]
<table>
<thead>
<tr>
<th>#</th>
<th>Data set name</th>
<th>$T$</th>
<th>$m$</th>
<th>$p$</th>
<th>$l$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Data of the western basin of Lake Erie</td>
<td>57</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Data of Ethane-ethylene column</td>
<td>90</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Data of a 120 MW power plant</td>
<td>200</td>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>Heating system</td>
<td>801</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>Data from an industrial dryer</td>
<td>867</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>Data of a hair dryer</td>
<td>1000</td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>Data of the ball-and-beam setup in SISTA</td>
<td>1000</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>Wing flutter data</td>
<td>1024</td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>Data from a flexible robot arm</td>
<td>1024</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>Data of a glass furnace (Philips)</td>
<td>1247</td>
<td>3</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>Heat flow density through a two layer wall</td>
<td>1680</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>Simulation of a pH neutralization process</td>
<td>2001</td>
<td>2</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>13</td>
<td>Data of a CD-player arm</td>
<td>2048</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>Data from an industrial winding process</td>
<td>2500</td>
<td>5</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>Liquid-saturated heat exchanger</td>
<td>4000</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>Data from an evaporator</td>
<td>6305</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>Continuous stirred tank reactor</td>
<td>7500</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>Model of a steam generator</td>
<td>9600</td>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>
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.
Performance

Execution time

![Execution time graph]

- stls
- pem
- subid
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?

For the $w$-behavior, this gives nothing new

(\iff \text{elimination theorem}).

So, what is the rationale for using latent variables $\varepsilon$?
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

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

\( \sim \) either \( w = \text{input, free, } \mathcal{B} = \mathbb{R}^T \)

or \( w = \text{output, } \sim \mathcal{B} \cong \text{sums of ‘exponentials’} \)

\( \sim \) 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.
Define the ‘latency’:

\[
\text{latency} (\tilde{\mathbf{w}}, \mathcal{M}) := \text{minimum } \| \tilde{\mathbf{e}} \|_{\ell^2}
\]

with the minimum taken over all \( \tilde{\mathbf{e}} \) such that

\[
R_0 \tilde{w}(t) + \cdots + R_L \tilde{w}(t + L) = M_0 \tilde{e}(t) + \cdots + M_L \tilde{e}(t + L)
\]

i.e. min. over all \( \tilde{\mathbf{e}} \) that ‘explain’ \( \tilde{w}(1), \tilde{w}(2), \ldots, \tilde{w}(T) \).

\[\leadsto \text{ system ID: search for the optimal model,} \]

\[\text{in the sense of minimal latency} \]

in a given model class.
Latency minimization

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

Latency minimization is a **deterministic** Kalman filtering pbm

The latency is actually equal to the prediction error!

$\leadsto$ deterministic interpretation, system ID toolbox, etc.
Remarks on stochastic SYSID
Why stochastic interpretation?

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

We can consider \( \varepsilon \) as a stochastic disturbance.

If we take also \( u \) as a stochastic process, then \( w \) stochastic.

SYSID pbm is then a statistical one, leading to maximum likelihood estimation (very related to PEM).

It allows evaluation of algorithms in terms of \( T \to \infty \). Nice statistical questions emerge, as consistency, asymptotic efficiency, etc.

\[ \to \text{deep theory of ARMAX systems.} \]
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.
We concentrated on **exact deterministic** SYSID.

- Nice concepts, as MPUM.
- Realization theory as special case
- Subspace algorithms very effective