System identification by structured low-rank approximation

Ivan Markovsky and Konstantin Usevich

School of Electronics and Computer Science
University of Southampton
{im,ku}@ecs.soton.ac.uk

Abstract

Identification problem with no a priori separation of the variables into inputs and outputs and representation invariant approximation criterion is considered. The model class consists of linear time-invariant systems of bounded complexity (number of inputs and order) and the approximation criterion is the minimum of the ℓ_2-norm distance between the given time series and a time series that is consistent with the approximate model. The problem is equivalent to and is solved as a block-Hankel structured low-rank approximation problem. Software implementing the approach in practice is developed and tested on benchmark problems from the DAISY dataset. Additional features of the software are specification of exact variables, fixed initial conditions, and identification from multiple time series with different length.

Keywords: system identification, errors-in-variables, output error, model reduction, low-rank approximation, reproducible research, DAISY.

1 Introduction

This document presents software for approximate linear time-invariant system identification. A discrete-time dynamical system $\mathscr{B} \subset (\mathbb{R}^q)^{\mathbb{Z}}$ is a collection of trajectories (q-variables time-series $w : \mathbb{Z} \to \mathbb{R}^q$). No a priori separation of the variables into inputs and output is made and the system is not a priori bound to a particular representation. However, the variables w can always be partitioned into inputs u (free variables) and outputs y (dependent variables) and the system can be represented in various equivalent forms, *e.g.*, the ubiquitous input/state/output representation

$$x(t+1) = Ax(t) + Bu(t), \qquad y(t) = Cx(t) + Du(t). \tag{I/S/O}$$

The number of inputs $\mathtt{m}$, the number of outputs $\mathtt{p}$, and the minimal state dimension $\mathtt{n}$ of an input/state/output representation are invariant of the representation and in particular of the input/output partitioning.

The class of finite dimensional linear time-invariant systems with q variables and at most $\mathtt{m}$ inputs is denoted by $\mathscr{L}^q_{\mathtt{m}}$. The number of inputs and the minimal state dimension specify the complexity of the system in the sense that the dimension of the restriction of $\mathscr{B}$ to the interval $[1, T]$, where $T \geq \mathtt{n}$, is a $T\mathtt{m} + \mathtt{n}$ dimensional subspace. Equivalently, the complexity of the system can be specified by the input dimension and the *lag* of the system. The lag of $\mathscr{B}$ is the minimal natural number ℓ, for which there exists an ℓth order difference equation

$$R_0 w(t) + R_1 w(t+1) + \cdots + R_\ell w(t+\ell) = 0 \tag{DE}$$

representation of the system, *i.e.*, $\mathscr{B} = \{ w \mid \text{(DE) holds} \}$. The subset of $\mathscr{L}^q_{\mathtt{m}}$ with lag at most ℓ is denoted by $\mathscr{L}^q_{\mathtt{m},\ell}$.

The considered identification problem is the global total least squares problem [17, 13]:

> Given a time series $w_{\mathrm{d}} \in (\mathbb{R}^q)^T$ and a complexity specification $(\mathtt{m}, \ell)$, find the system
>
> $$\widehat{\mathscr{B}} := \arg\min_{\mathscr{B} \in \mathscr{L}^q_{\mathtt{m},\ell}} M(w_{\mathrm{d}}, \mathscr{B}), \quad \text{where} \quad M(w_{\mathrm{d}}, \mathscr{B}) := \min_{\widehat{w} \in \mathscr{B}} \|w_{\mathrm{d}} - \widehat{w}\|^2_{\ell_2}. \tag{SYSID}$$

The number $M(w_{\mathrm{d}}, \mathscr{B})$ is called the *misfit* between w_{d} and $\mathscr{B}$. It shows how much the model $\mathscr{B}$ fails to "explain" the data w_{d}. The optimal approximate modeling problem (SYSID) aims to find the system $\widehat{\mathscr{B}}$ in the model class $\mathscr{L}^{q}_{\mathtt{m},\ell}$ that best fits the data according to the misfit criterion.

Special cases of (SYSID) are static data modeling ($\ell = 0$) and output-only or autonomous system identification ($\mathtt{m} = 0$). The solution approach, described next, leads to an algorithm that covers these special cases. The following are additional features of the approach.

1. Elements of the given time series w_{d} can be specified as "exact", in which case they appear unmodified in the approximation $\widehat{w}$. For example, in output error identification problems the variables are a priori partitioned into inputs and outputs and the input variables are exact while the output variables are perturbed by measurement noise.

2. The approximation $\widehat{w}$ can be constrained to be a trajectory of the model $\mathscr{B}$, generated under a priori fixed initial conditions w_{ini}, *i.e.*,
$$\begin{bmatrix} w_{\mathrm{ini}} \\ \widehat{w} \end{bmatrix} \in \mathscr{B}.$$
(Note that problem (SYSID) identifies the model without prior knowledge about the initial conditions, under which the data w_{d} is generated, *i.e.*, w_{ini} is a free variable.) In identification from impulse or step response data, however, the initial conditions are know exactly. When available this prior information should be taken into account by constraining w_{ini}.

3. Multiple time series
$$w^k_{\mathrm{d}} = \big(w^k_{\mathrm{d}}(1), \ldots, w^k_{\mathrm{d}}(T_k)\big), \qquad k = 1, \ldots, N,$$
with possibly different number of samples can be used simultaneously in solving (SYSID). In this case, the misfit between the data w_{d} and the model $\mathscr{B}$ is defined as
$$M(w_{\mathrm{d}}, \mathscr{B}) := \min_{\widehat{w}^1, \ldots, \widehat{w}^k \in \mathscr{B}} \sum_{k=1}^{N} \|w^k_{\mathrm{d}} - \widehat{w}^k\|^2_{\ell_2}.$$
This feature is also useful when the data is generated from a single experiment with blocks of ℓ or more consecutive missing samples [16].

Solution approach

For given $w = \big(w(1), \ldots, w(T)\big)$ and $\ell \in \mathbb{N}$, we define the block Hankel matrix
$$\mathscr{H}_{\ell+1}(w) := \begin{bmatrix} w(1) & w(2) & \cdots & w(T-\ell) \\ w(2) & w(3) & \cdots & w(T-\ell+1) \\ \vdots & \vdots & & \vdots \\ w(\ell+1) & w(\ell+2) & \cdots & w(T) \end{bmatrix}.$$

The identification problem (SYSID) is solved in the following equivalent formulation:
$$\widehat{R} = \arg \min_{R, RR^\top = I_p} \left(\min_{\widehat{w}} \ \|w_{\mathrm{d}} - \widehat{w}\|^2_{\ell_2} \quad \text{subject to} \quad R\mathscr{H}_{\ell+1}(\widehat{w}) = 0 \right). \tag{SLRA}$$

The parameter R of (SLRA) is related to the parameters $R_0, R_1, \ldots, R_\ell$ of the difference equation representation (DE) of the approximating system $\mathscr{B}$ as follows:
$$R = \begin{bmatrix} R_0 & R_1 & \cdots & R_\ell \end{bmatrix}, \qquad \text{where} \quad R_i \in \mathbb{R}^{\mathtt{p} \times q}.$$

Algorithms and software for structured low-rank approximation are developed in [12, 10]. We use the C++ solver of [11] as the core computational tool for solving the system identification problem (SYSID). In fact, the software presented in this paper can be viewed as an interface to the structured low-rank approximation solver for the purpose

of linear time-invariant system identification. On its own, the solver computes a parameter $\widehat{R}$ of the difference equation representation of the optimal approximating system $\widehat{\mathscr{B}}$. The system identification function converts the parameter $\widehat{R}$ to the parameters $(\widehat{A},\widehat{B},\widehat{C},\widehat{D})$ of an input/state/output representation of $\widehat{\mathscr{B}}$ in order to facilitate the usage of the model by other analysis and synthesis tools. In addition, a system $\mathscr{B}_{\rm ini} \in \mathscr{L}^{q}_{\mathtt{m},\ell}$, specified by parameters $(A_{\rm ini},B_{\rm ini},C_{\rm ini},D_{\rm ini})$ can be used as an initial approximation for the optimization algorithm. The parameters $(A_{\rm ini},B_{\rm ini},C_{\rm ini},D_{\rm ini})$ are converted to the parameter $R_{\rm ini}$, used by the structured low-rank approximation solver. Although the identification function has as external interface the (I/S/O) representation of the model, the internal computations are done via the parameter R of the difference equation representation.

⟨*structure specification*⟩≡
```
s.m = ell1 * ones(q, 1); s.n = T - ell; r = ell1 * m + n;
L = q * ell1; phi = []; for i = 1:ell1, phi = [phi, i:ell1:L]; end
s.phi = eye(L); s.phi = s.phi(phi, :);
```

2 Usage

The function `ident` solves the approximate identification problem (SYSID), and `misfit` computes the misfit $M(w_{\rm d},\mathscr{B})$. Both functions use the input/state/output representation (I/S/O), so that they implement the following mappings:

`ident`: $(w_{\rm d},\mathtt{m},\ell) \mapsto (\widehat{A},\widehat{B},\widehat{C},\widehat{D})$, where $(\widehat{A},\widehat{B},\widehat{C},\widehat{D})$ define a (locally) optimal solution of (SYSID), and

⟨`ident` *function definition*⟩≡
```
function [sysh, info, wh, xini] = ident(w, m, ell, varargin)
```

`misfit`: $\big(w_{\rm d},(A,B,C,D)\big) \mapsto (M,\widehat{w})$, where M is the misfit between the model, defined by (A,B,C,D), and $w_{\rm d}$.

⟨`misfit` *function definition*⟩≡
```
function [M, wh, xini] = misfit(w, sysh, varargin)
```

- `w` is the given time series $w_{\rm d}$ — a real Matlab array of dimension $T \times q \times N$, where T is the number of samples, q is the number of variables, and N is the number of time series. In case of multiple time series of different dimensions, `w` should be specified as a cell array with N cells, each one of which is a $T_i \times q$ matrix containing the ith time series, *i.e.*,

$$\mathtt{w(t,:,k)} = \big(w^k(t)\big)^\top \qquad \text{or} \qquad \mathtt{w\{k\}(t,:)} = \big(w^k(t)\big)^\top.$$

⟨*define* `T, q, N`⟩≡
```
if ~iscell(w)
  [T, q, N] = size(w); T = ones(N, 1) * T;
else
  N = length(w); for k = 1:N, [T(k), q] = size(w{k}); end, T = T(:);
end
```

- (`m`, `ell`) is the complexity specification (input dimension and lag).
- `varargin` is/are optional arguments(s) specifying exact variables, exact initial conditions, and options for the optimization solver, used by the `ident` function. The options can be passed to the functions `ident` and `misfit` as fields of a variable or as pairs (`'name', 'value'`) of input arguments.
  - `'exct'` (default value `[]`) — vector of indices for exact variables.

⟨`ident` *named arguments*⟩≡
```
ip.addParamValue('exct', [], @(exct) isempty(exct) || ...
                                     (all(exct >= 1 & exct <= q)))
```

⟨*structure specification*⟩+≡
```
if ~isempty(opt.exct), s.w = ones(q, 1); s.w(opt.exct) = inf; end
```

– `'wini'` (default value `[]`) — specifies exact initial conditions. If `wini = []`, initial conditions are not specified. If `wini = 0`, exact zero initial conditions are specified, *i.e.*, $\left[\begin{smallmatrix} 0 \\ \widehat{w}^k \end{smallmatrix}\right] \in \widehat{\mathscr{B}}$, More generally, `wini` $= w_{\text{ini}}$ is an ℓ samples long trajectory (specified by an $\ell \times q \times N$ array or a N-dimensional cell array of $\ell \times q$ matrices), defining initial conditions for the time series $\widehat{w}$, *i.e.*, $\left[\begin{smallmatrix} w^k_{\text{ini}} \\ \widehat{w}^k \end{smallmatrix}\right] \in \widehat{\mathscr{B}}$.

⟨`ident` *named arguments*⟩+≡
```
ip.addParamValue('wini', [])
```

⟨*initial conditions*⟩≡
```
if iscell(w)
  if isempty(opt.wini)
    opt.wini = cell(1, N);
  elseif ~iscell(opt.wini) & (opt.wini == 0)
    for k = 1:N, wini{k} = zeros(ell, q); end, opt.wini = wini;
  else
    for k = 1:N
      if opt.wini{k} == 0, opt.wini{k} = zeros(ell, q); end
    end
  end
elseif opt.wini == 0
  opt.wini = zeros(ell, q, N);
end
```

⟨*structure specification*⟩+≡
```
if ~iscell(opt.wini) && ~isempty(opt.wini)
  W = ones(T, q, N); W(:, opt.exct, :) = inf;
  s.n = s.n + ell; T = T + ell;
  s.w = vec([inf * ones(ell, q, N); W]);
elseif iscell(opt.wini) && ~isempty(cell2mat(opt.wini))
  s.w = [];
  for k = 1:N
    W = ones(T(k), q); W(:, opt.exct) = inf;
    if ~isempty(opt.wini{k})
      s.n(k) = s.n(k) + ell; T(k) = T(k) + ell;
      s.w = [s.w; vec([inf * ones(ell, q); W])];
    else
      s.w = [s.w; W(:)];
    end
  end
end
```

– `'sys0'` — initial approximation: an input/state/output representation of a system, given as a state space (`ss`) object, with `m` inputs, `p` $:= q -$ `m` outputs, and order `n` $:= \ell$`p`. (Default value is computed by unstructured low-rank approximation.)

⟨`ident` *named arguments*⟩+≡
```
ip.addParamValue('sys0', [], @(sys0) isempty(sys0) || isa(sys0, 'ss'));
ip.parse(varargin{:}); opt = ip.Results;
```

⟨*initial approximation*⟩≡
```
if ~isempty(opt.sys0), ⟨sys0 ↦ R⟩, opt.Rini = R'; end
```

– Arguments allowing the user to specify different optimization algorithms (`'method'`), control the displayed information (`'disp'`), and change the convergence criteria (`'tolX'`, `'tolGrad'`, and `'maxiter'`) are described in the structured low-rank approximation user's manual [11].

⟨`ident` *named arguments*⟩+≡

```
ip.addParamValue('disp', 'off');
ip.addParamValue('method', 'll');
ip.addParamValue('maxiter', 100);
ip.addParamValue('tol', 1e-5);
```

- `sysh` is an input/state/output representation of the identified or validated system $\widehat{\mathscr{B}}$.
- `info` is a structure, containing exit information from the structured low-rank approximation solver: `info.M` is the misfit $M(w_{\mathrm{d}}, \widehat{\mathscr{B}})$, `info.time` is the execution time, and `info.iter` is the number of iterations.
- `M` is the misfit $M(w_{\mathrm{d}}, \widehat{\mathscr{B}})$.
- `wh` is the optimal approximating time series.
- `xini` is a matrix whose columns are the initial condition, under which $\widehat{w}^k$, $k = 1, \ldots, N$ are obtained.

3 Model reduction example

As a test example, consider a mechanical system consisting of n point masses connected in a chain by ideal springs and ideal dampers. The first and the last masses are also connected to walls. Friction force, proportional to the speed, acts on all masses. The parameters of the system are the masses $m_1, \ldots, m_N$, the spring and damper coefficients $k_0, k_1, \ldots, k_N$ and $d_0, d_1, \ldots, d_N$, the length δ of the springs, and the friction coefficient f, which are all nonnegative numbers. Setting the coefficients of the most left and/or most right springs and dampers to zero has the effect of detaching the chain of masses from the left and/or right walls.

The system dynamics is described by the system of differential equations

$$
\begin{aligned}
m_1 \ddot{p}_1 &= -(k_0 + k_1) p_1 + k_1 p_2 - (d_0 + d_1 + f) \dot{p}_1 + d_1 \dot{p}_2 + (k_0 - k_1) \delta \\
m_2 \ddot{p}_2 &= k_1 p_1 - (k_1 + k_2) p_2 + k_2 p_3 + d_1 \dot{p}_1 - (d_1 + d_2 + f) \dot{p}_2 + d_2 \dot{p}_3 + (k_1 - k_2) \delta \\
&\vdots \\
m_i \ddot{p}_i &= k_{i-1} p_{i-1} - (k_{i-1} + k_i) p_i + k_i p_{i+1} + d_{i-1} \dot{p}_{i-1} - (d_{i-1} + d_i + f) \dot{p}_i + d_i \dot{p}_{i+1} + (k_{i-1} - k_i) \delta \\
&\vdots \\
m_N \ddot{p}_N &= k_{N-1} p_{N-1} - (k_{N-1} + k_N) p_N + k_{N-1} \dot{p}_{N-1} - (d_{N-1} + d_N + f) \dot{p}_N + (k_{N-1} + k_R N) \delta .
\end{aligned}
$$

The positions y of the masses with indexes in the set $\mathscr{Y}$ are observed and external forces u are applied to the masses with indexes in the set $\mathscr{U}$. The external forces u and the positions y are viewed as, respectively, control inputs and observed outputs of the system. The dynamics of the system can be represented by input/state/output representation (I/S/O) with parameters

$$
A = \begin{bmatrix} 0 & I_N & 0 \\ A_{21} & A_{22} & A_{23} \\ 0 & 0 & 0 \end{bmatrix} \in \mathbb{R}^{(2N+1)\times(2N+1)}, \qquad B = \begin{bmatrix} 0 \\ E_{\mathscr{Y}} \\ 0 \end{bmatrix}, \qquad C = \begin{bmatrix} E_{\mathscr{Y}}^{\top} & 0 & 0 \end{bmatrix}, \qquad D = 0, \qquad x_{2N+1}(0) = 1,
$$

where

$$
A_{21} = \begin{bmatrix}
-\frac{k_0+k_1}{m_1} & \frac{k_2}{m_1} & & & \\
\frac{k_1}{m_2} & -\frac{k_1+k_2}{m_2} & \frac{k_3}{m_2} & & \\
 & \ddots & \ddots & \ddots & \\
 & & \frac{k_{N-2}}{m_{N-2}} & -\frac{k_{N-2}+k_{N-1}}{m_{N-2}} & \frac{k_N}{m_{N-2}} \\
 & & & \frac{k_{N-1}}{m_N} & -\frac{k_{N-1}+k_N}{m_N}
\end{bmatrix}
$$

and

$$A_{22} = \begin{bmatrix} -\frac{d_0+d_1+f}{m_1} & \frac{d_2}{m_1} & & & \\ \frac{d_1}{m_2} & -\frac{d_1+d_2+f}{m_2} & \frac{d_3}{m_2} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{d_{N-2}}{m_{N-2}} & -\frac{d_{N-2}+d_{N-1}+f}{m_{N-2}} & \frac{d_N}{m_{N-2}} \\ & & & \frac{d_{N-1}}{m_N} & -\frac{d_{N-1}+d_N+f}{m_N} \end{bmatrix}, \qquad A_{23} = \begin{bmatrix} \frac{k_0-k_1}{m_1} \\ \frac{k_1-k_2}{m_2} \\ \vdots \\ \frac{k_{N-2}-k_{N-1}}{m_{N-1}} \\ \frac{k_{N-1}+Nk_N}{m_N} \end{bmatrix} \delta.$$

The model is constructed so that x_{2N+1} is a constant equal to $x_{2N+1}(0) = 1$.

⟨*form A*⟩≡
```
if N > 1
    a21 =   diag(K(2:end - 1) ./ M(2:end), -1) ...
          - diag((K(1:end - 1) + K(2:end)) ./ M) ...
          + diag(K(2:end - 1) ./ M(1:end - 1), 1);
    a22 =   diag(D(2:end - 1) ./ M(2:end), -1) ...
          - diag((D(1:end - 1) + D(2:end) + f) ./ M) ...
          + diag(D(2:end - 1) ./ M(1:end - 1), 1);
else
    a21 = - sum(K) / M; a22 = - (sum(D) + f) / M;
end
a = [zeros(N) eye(N); a21 a22];
```

⟨*form A*⟩+≡
```
a_ext = zeros(2 * N, 1);
if N > 1
    a_ext(N + 1:end, 1) = [(K(1:end - 2) - K(2:end - 1)) ./ M(1:end - 1);
                           (K(end - 1) + N * K(end)) / M(end)] * delta;
else
    a_ext(2) = sum(K) / M;
end
a = [a, a_ext; zeros(1, 2 * N + 1)];
```

⟨*form B, C, D*⟩≡
```
n = 2 * N + 1; m = length(IN); p = length(OUT);
b = zeros(n, m); b(N + IN, 1:end) = 1;
c = zeros(p, n); c(1:p, OUT) = 1; d = zeros(p, m);
```

⟨*define* sys⟩≡
```
function s = sys(N, varargin)
⟨parse model parameters⟩
⟨form A⟩
⟨form B, C, D⟩
s = ss(a, b, c, d);
```

⟨*parse model parameters*⟩≡
```
ip = inputParser; ip.KeepUnmatched = true;
ip.addParamValue('delta', 1)
ip.addParamValue('f', 0)
ip.addParamValue('M', ones(N, 1))
ip.addParamValue('K', ones(N + 1, 1))
ip.addParamValue('D', ones(N + 1, 1))
ip.addParamValue('IN', 1)
ip.addParamValue('OUT', round(N / 2))
ip.parse(varargin{:}); opt = ip.Results;
names = fieldnames(opt);
for i = 1:length(names)
    eval(sprintf('%s = opt.%s;', names{i}, names{i}));
end
```

⟨*test* sys⟩≡
```
N = 7; opt.f = 0.05;
opt.M = (N:-1:1)';
opt.K = [0; ones(N - 1, 1); 0];
opt.D = [0; ones(N - 1, 1); 0];
s = sys(N, opt); n = size(s, 'order');
rank(ctrb(s))

% y = impulse(s); T = length(y);
T = 300; u = randn(T, 1); % [1; zeros(T - 1, 1)]; %
x0 = [1:N, zeros(1, N), 1]';
[y, t, x] = lsim(s, u, 0:(T - 1), x0);

%figure(1), plot(t, y), figure(2), plot(t, x(:, 1:N))
%figure(3), for i = 1:N, plot(t, x(:, i)), pause(.5), end
%rk = rank(blkhank(y, n + 1))

sh = ident([u y], 1, n - 1);
sort(eig(sh)), sort(eig(c2d(s, 1)))
```

Multiple trajectories of different lengths

The following script simulates multiple trajectories of a linear time-invariant system, adds noise to the true data, and applies the `ident` function to find a model from the noisy data. The simulation parameters are as follows:

⟨test⟩≡
```
clear all, warning off
ell = 3; p = 1; m = 1; T = [30 50];
nl = 0.2; eiv = 1; opt.wini = 0;
```

The true data is generated as a set of random trajectories of the true system.

⟨test⟩+≡
```
sys0 = drss(ell * p, p, m); N = length(T);
if opt.wini == 0;
  xini = zeros(p * ell, 1);
else,
  xini = rand(p * ell, 1);
end
for k = 1:N,
  u0{k} = randn(T(k), 1); [eye(m); zeros(T(k) - m, m)]; % randn, ones
  y0{k} = lsim(sys0, u0{k}, [], xini);
end
```

The data used for identification is a noise curropted version of the true data.

⟨test⟩+≡
```
if eiv,
  for k = 1:N, w0{k} = [u0{k} y0{k}]; end, opt.exct = [];
else
  w0 = y0; opt.exct = 1:m;
end
for k = 1:N
  wn = randn(size(w0{k}));
  w{k} = w0{k} + nl * norm(w0{k}) * wn / norm(wn);
  if ~eiv, w0{k} = [u0{k} y0{k}]; w{k} = [u0{k} w{k}]; end
end
if N == 1, w0 = w0{1}; w = w{1}; end
```

A model in the model class $\mathscr{L}_{\mathtt{m},\ell}$ (which contrains the true system) is idenitifed by the `ident` function:

⟨test⟩+≡
```
[sysh, info, wh, xini] = ident(w, m, ell, opt);
[M, wh, xini] = misfit(w, sysh, opt);
```

The true, noisy, and approximating trajectores are plotted and shown in Figure **??**.

⟨test⟩+≡
```
for k = 1:N,
  figure(k)
  plot(w0{k}(:, 2), 'k'), hold on,
  plot(w{k}(:, 2), 'k:'),
  plot(wh{k}(:, 2), 'b-')
end
```

The optimal misfit is generically independent of the input/output partitioning

The optimal misfit value of the (SYSID) problem is invariant to permutation of the elements of w_{d}, *i.e.*, for an arbitrary permutation matrix $\Pi \in \mathbb{R}^{q\times q}$,

$$\min_{\mathscr{B}\in\mathscr{L}^q_{\mathtt{m},\ell}} M(w_{\mathrm{d}},\mathscr{B}) = \min_{\mathscr{B}\in\mathscr{L}^q_{\mathtt{m},\ell}} M(\Pi w_{\mathrm{d}},\mathscr{B}).$$

This property of the problem allows to make no distinction of the ordering of the variables. In the next simulation example, we verify the invariance property numerically.

⟨*input/output partition*⟩≡
```
l = 1;
[sysh1, info1, wh1, xini1] = ident(w, m, ell); info1
perm = randperm(m + p), w2 = w(:,perm);
[sysh2, info2, wh2, xini2] = ident(w2, m, ell); info2
norm(wh1(:, perm) - wh2)
```

Although (SYSID) is permutation invariant, the optimization problem may behave differently for different permutation matrices Π, because of existence of multiple local minima of the misfit function.

4 Performance on real-life data

In this section, the performance of the method, described in the paper, is tested on benchmark problems from the data base for system identification DAISY [2]. The data come from a number of applications: process industry, electrical, mechanical, and environmental. We choose a validation criterion that measures the predictive power of the model: how accurate the model can fit a part of the data that is not used for identification. Values for the identification methods' parameters that correspond to this validation criterion are chosen and fixed for all data sets. Although often better results can be obtained by (preprocessing of the data and tuning of the identification method parameters), our tests reflect the view that the identification process should be done with as little human interaction as possible. We apply the methods choosing only the model class (specified by a bound on the model complexity) and the identification/validation criterion (specified by desired notion of approximation or disturbance/noise properties).

4.1 Database for system identification DAISY

The considered data sets are listed in Table 1. References and details about the nature and origin of the data is given in Appendix B. The data is preprocessed only by centering it. The model's lag ℓ is chosen manually for each data set from the complexity–accuracy trade-off curve (misfit as a function of the lag).

The data $w_{\mathrm{d}} = (u_{\mathrm{d}}, y_{\mathrm{d}})$ in all examples is split into identification and validation parts. For a chosen $x \in [0,100]$, the first or last x% of the data, denoted w_{idt}, are used for identification, and the remaining $(100-x)$% of the data, denoted w_{val}, are used for validation. A model $\widehat{\mathscr{B}}$ is identified from w_{idt} by an identification method and is validated on w_{val} by the validation criterion defined next. The model class is linear time-invariant systems with a bound ℓ on the lag (degree of a difference equation representation or equivalently observability index). Bounding the lag by ℓ corresponds to bounding the order by $\ell\mathtt{p}$, where $\mathtt{p}$ is the number of outputs.

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

Table 1: Examples from DAISY. T—number of data points, m—number of inputs, p—number of outputs, ℓ—lag of the identified model.

4.2 Validation criterion and results

The validation criterion is the "simulation fit" computed by the function `compare` of the System Identification Toolbox. Given a time series $w_{\mathrm{d}} = (u_{\mathrm{d}}, y_{\mathrm{d}})$ and a model $\mathscr{B}$, define the approximation $\widehat{y}$ of y in $\mathscr{B}$ as follows:

$$\widehat{y}\big((u_{\mathrm{d}}, y_{\mathrm{d}}), \mathscr{B}\big) := \min_{\widehat{y}} \|y_{\mathrm{d}} - \widehat{y}\| \quad \text{subject to} \quad \mathrm{col}(u_{\mathrm{d}}, \widehat{y}) \in \mathscr{B}.$$

(The optimization is carried over the initial conditions that generate $\widehat{y}$ from the given input u_{d}.) Let $\bar{y}$ be the mean of y_{d}, *i.e.*, $\bar{y} := \sum_{t=1}^{T} y_{\mathrm{d}}(t)/T$. With this notation, the fit of w_{d} by $\mathscr{B}$ is defined as

$$F(w_{\mathrm{d}}, \mathscr{B}) := 100 \, \frac{\max\big(0, 1 - \|y_{\mathrm{d}} - \widehat{y}(w_{\mathrm{d}}, \mathscr{B})\|\big)}{\|y_{\mathrm{d}} - \bar{y}\|}.$$

We compare the fitting criterion $F(w_{\mathrm{val}}, \widehat{\mathscr{B}})$ for the models produced by the compared identification methods.

4.3 Identification from step response

⟨*step response data*⟩≡

```
clear all
d = load('distill2.dat'); T = 30; u = zeros(T, 3, 3);
for i = 1:3,
  u(:, i, i) = 1;
  y(:, :, i) = [d(1:T, 2 * i), d(1:T, 2 * i + 1)];
end
for i = 1:3, for j = 1:2,
  y(:, j, i) = y(:, j, i) / norm(y(:, j, i));
end, end
figure
for i = 1:3, for j = 1:2,
  subplot(2, 3, (j - 1) * 3 + i), plot(y(:, j, i))
  ax = axis; axis([1, T, ax(3:4)])
end, end
```

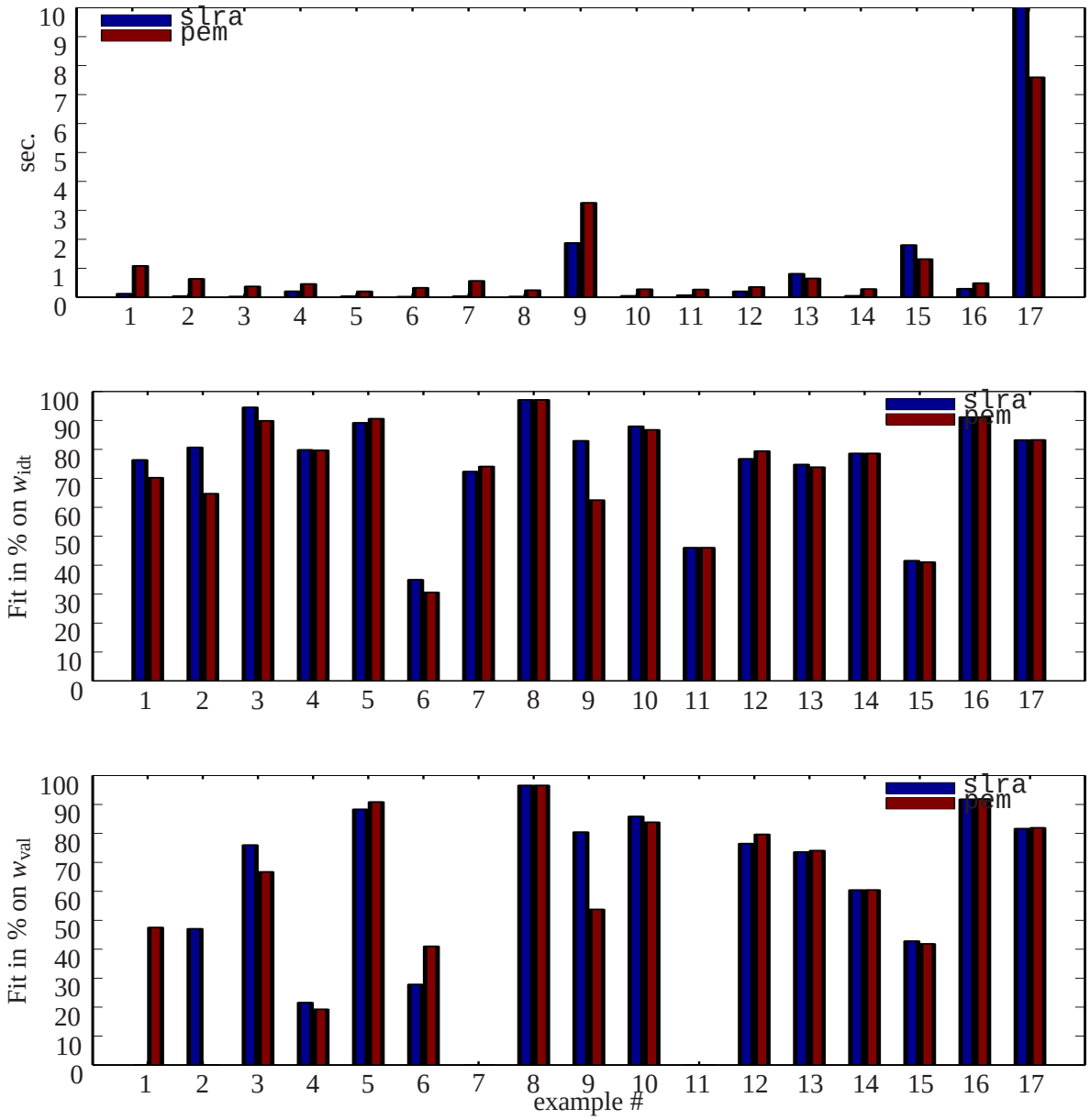

Figure 1: Results on all data sets by splitting of the data into first 70% for identification and remaining 30% for validation.

```
opt.exct = 1:3; opt.wini = 0; ell = 3;
[sysh, info, wh, xini] = ident(cat(2, u, y), 3, ell, opt);

%i = 1; j = 1:2; opt.exct = 1:length(i); ell = 2;
%[sysh, info, wh, xini] = ident(cat(2, u(:, i, i), y(:, j, i)), length(i), ell, opt);

for i = 1:3
  figure, plot(y(:, 1, i), 'k'), hold on, plot(wh(:, 4, i), 'b-')
  figure, plot(y(:, 2, i), 'k'), hold on, plot(wh(:, 5, i), 'b-')
end
```

5 Implementation

⟨*required arguments* w, m, ell⟩≡

```
ipr = inputParser;
```

```
  ipr.addRequired('w', @(w) (isnumeric(w) && ~isempty(w)) || iscell(w));
  ipr.addRequired('m', @(m) m >= 0);
  ipr.addRequired('ell', @(ell) ell >= 0);
  ipr.parse(w, m, ell);
```

$\langle (w_{\mathrm{ini}}, w_{\mathrm{d}}) \mapsto p \rangle \equiv$

```
  if ~iscell(w)
    par = vec([opt.wini; w]);
  else
    par = []; for k = 1:N, par = [par; vec([opt.wini{k}; w{k}])]; end
  end
```

$\langle \widehat{p} \mapsto \widehat{w} \rangle \equiv$

```
  if ~iscell(w)
    wh = reshape(ph, T(1), q, N);
    if ~isempty(opt.wini), wh = wh(ell1:end, :, :); end
  else
    for k = 1:N,
      wh{k} = reshape(ph(1:(q * T(k))), T(k), q); ph(1:(q * T(k))) = [];
      if ~isempty(opt.wini{k}), wh{k} = wh{k}(ell1:end, :); end
    end
  end
```

$\langle R \mapsto$ *I/S/O representation* $\rangle \equiv$

```
  if m > 0
```

$\quad \langle R \mapsto$ *I/S/O representation* ($m > 0$ *case*) $\rangle$

```
  else
```

$\quad \langle R \mapsto$ *I/S/O representation* ($m = 0$ *case*) $\rangle$

```
  end
```

$$A = \begin{bmatrix} 0 & I_{\mathrm{p}} & & \\ \vdots & & \ddots & \\ 0 & & & I_{\mathrm{p}} \\ -P_\ell^{-1}P_0 & -P_\ell^{-1}P_1 & \cdots & -P_\ell^{-1}P_{\ell-1} \end{bmatrix}, \qquad B = \begin{bmatrix} P_\ell^{-1}(Q_0 - P_0P_\ell^{-1}Q_\ell) \\ P_\ell^{-1}(Q_1 - P_1P_\ell^{-1}Q_\ell) \\ \vdots \\ P_\ell^{-1}(Q_{\ell-1} - P_{\ell-1}P_\ell^{-1}Q_\ell) \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & \cdots & 0 & I_{\mathrm{p}} \end{bmatrix}, \qquad D = P_\ell^{-1}Q_\ell$$

In the implementation below, it is taken into account that $P_\ell = I$.

$\langle R \mapsto$ *I/S/O representation* ($m > 0$ *case*) $\rangle \equiv$

```
  R3 = reshape(info.Rh, p, q, ell1);
  Q3 = R3(:, 1:m, :); P3 = - R3(:, m + 1:q, :);
  a = zeros(n); b = zeros(n, m); c = [];
  if n > 0
    a(p + 1:end, 1:n - p) = eye(n - p);
    c = [zeros(p, n - p) eye(p)];
  end
  d = Q3(:, :, ell1); ind_j = (n - p + 1):n;
  for i = 1:ell
    ind_i = ((i - 1) * p + 1):(i * p); P3i = P3(:, :, i);
    a(ind_i, ind_j) = - P3i;
    b(ind_i, :) = Q3(:, :, i) - P3i * d;
  end
  sysh = ss(a, b, c, d, 1);
```

$$\mathrm{image}(\mathscr{O}) = (R)$$

⟨$R \mapsto$ *I/S/O representation* ($m = 0$ *case*)⟩≡
```
O = null(info.Rh);
a = O(1:end - p, :) \ O(p + 1:end, :); c = O(1:p, :);
sysh = ss(a, [], c, [], 1);
```

Convert (A,B,C,D) to a parameter R. Construct the observability matrix and form the lower bock-triangular Toeplitz matrix whose first block column is

$$F = \mathrm{col}(D, CB, CAB, \ldots, CA^{\ell-1}B).$$

⟨sys0 $\mapsto R$⟩≡
```
[a, b, c, d] = ssdata(opt.sys0); L = ell1; ⟨(A,C,L) ↦ O_L(A,C)⟩
R = null(O')';
if (m > 0)
  F = [d; O(1:(end - p), :) * b];
  TT = zeros(ell1 * p, ell1 * m);
  for i = 1:ell1
    TT((i - 1) * p + 1:end, (i - 1) * m + 1: i * m) = F(1:(ell1 + 1 - i) * p, :);
  end
  Q = R * TT;
  R3 = [reshape(Q, p, m, ell1), -reshape(R, p, p, ell1)];
  R  = reshape(R3, p, ell1 * q)';
end
```

Compute the extended observability matrix

⟨$(A,C,L) \mapsto \mathscr{O}_L(A,C)$⟩≡
```
O = c; for t = 2:L, O = [O; O(end - size(c, 1) + 1:end, :) * a]; end
```

⟨ident⟩≡
```
⟨ident inline help⟩
⟨ident function definition⟩
⟨required arguments w, m, ell⟩
⟨define T, q, N⟩
p = q - m; n = p * ell; ell1 = ell + 1;
ip = inputParser; ip.KeepUnmatched = true;
⟨ident named arguments⟩
ip.parse(varargin{:}); opt = ip.Results;
⟨initial approximation⟩
⟨initial conditions⟩
⟨(w_ini, w_d) ↦ p⟩
⟨structure specification⟩, save data
[ph, info] = slra(par, s, r, opt); info.M = info.fmin;
⟨p̂ ↦ ŵ⟩
⟨R ↦ I/S/O representation⟩
if nargout > 3, xini = inistate(wh, sysh); end
⟨define inistate⟩
```

⟨*define* inistate⟩≡
```
function xini = inistate(w, sys)
[a, b, c, d] = ssdata(sys); [p, m] = size(d); n = size(a, 1);
⟨define T, q, N⟩, L = max(T);
⟨(A,C,L) ↦ O_L(A,C)⟩, sys.Ts = -1; xini = zeros(n, N);
for k = 1:N
  if ~iscell(w)
    uk = w(:, 1:m, k); yk = w(:, (m + 1):end, k); Tk = T(1);
  else
    uk = w{k}(:, 1:m); yk = w{k}(:, (m + 1):end); Tk = T(k);
  end
  if m > 0, y0k = (yk - lsim(sys, uk, 0:(Tk - 1)))'; else, y0k = yk'; end
  xini(:, k) = O(1:(Tk * p), :) \ y0k(:);
end
```

⟨misfit⟩≡
  ⟨misfit *inline help*⟩
  ⟨misfit *function definition*⟩
  [p, m] = size(sysh); n = size(sysh, 'order'); ell = n / p;

Call ident with initial approximation sysh and zero iterations.

⟨misfit⟩+≡
  opt.sys0 = sysh; opt.maxiter = 0; opt.disp = 'off';
  [sysh, info, wh, xini] = ident(w, m, ell, opt); M = info.fmin;

Acknowledgements

The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC Grant agreement number 258581 "Structured low-rank approximation: Theory, algorithms, and applications".

References

[1] S. Bittanti and L. Piroddi. Nonlinear identification and control of a heat exchanger: a neural network approach. *Journal of the Franklin Institute*, 334(1):135–153, 1997.

[2] B. De Moor. DaISy: Database for the identification of systems. www.esat.kuleuven.be/sista/daisy/.

[3] E. Feron, M. Brenner, J. Paduano, and A. Turevskiy. Time-frequency analysis for transfer function estimation and application to flutter clearance. *AIAA J. on Guidance, Control & Dynamics*, 21(3):375–382, 1998.

[4] R. Guidorzi, M. Losito, and T. Muratori. On the last eigenvalue test in the structural identification of linear multivariable systems. In *Proc. V European meeting cybernetics systems research*, Vienna, 1980.

[5] R. Guidorzi, M. Losito, and T. Muratori. The range error test in the structural identification of linear multivariable systems. *IEEE Trans. Automat. Control*, 27:1044–1054, 1982.

[6] P. Van Den Hof and P. Schrama. Function estimation from closed loop data. *Automatica*, 29(6):1523–1527, 1993.

[7] G. Lightbody and G. Irwin. Nonlinear control structures based on embedded neural system models. *IEEE Tran. on Neural Networks*, 8(3):553–567, 1999.

[8] L. Ljung. *System Identification: Theory for the User*. Prentice-Hall, Upper Saddle River, NJ, 1999.

[9] L. Ljung. *System Identification Toolbox: User's guide*. The MathWorks, 2006.

[10] I. Markovsky. *Low Rank Approximation: Algorithms, Implementation, Applications*. Springer, 2012.

[11] I. Markovsky and K. Usevich. Software for weighted structured low-rank approximation. Technical report, ECS, Univ. of Southampton, http://eprints.ecs.soton.ac.uk/, 2012.

[12] I. Markovsky, S. Van Huffel, and R. Pintelon. Block-Toeplitz/Hankel structured total least squares. *SIAM J. Matrix Anal. Appl.*, 26(4):1083–1099, 2005.

[13] I. Markovsky, J. C. Willems, S. Van Huffel, B. De Moor, and R. Pintelon. Application of structured total least squares for system identification and model reduction. *IEEE Trans. Automat. Control*, 50(10):1490–1500, 2005.

[14] P. Van Overschee. *Subspace identification: Theory, Implementation, Application*. PhD thesis, K.U.Leuven, 1995.

[15] G. Pellegrinetti and J. Benstman. Nonlinear control oriented boiler modeling: A benchamrk problem for controller design. *IEEE Tran. Control Systems Tech.*, 40(1), 1996.

[16] R. Pintelon and J. Schoukens. Identification of continuous-time systems with missing data. *IEEE Trans. Instr. Meas.*, 48(3):736–740, 1999.

[17] B. Roorda and C. Heij. Global total least squares modeling of multivariate time series. *IEEE Trans. Automat. Control*, 40(1):50–63, 1995.

[18] P. Van Overschee and B. De Moor. *N4SID:* subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica*, 30:75–93, 1994.

[19] Y. Zhu, P. Van Overschee, B. De Moor, and L. Ljung. Comparison of three classes of identification methods. In *Proc. SYSID*, volume 1, pages 175–180, Copenhagen, Denmark, 1994.

A Inline help

⟨`ident` *inline help*⟩≡

```
% IDENT - Identification by structured low-rank approximation
%
% [sysh, info, wh, xini] = ident(w, m, ell, opt)
%
% Inputs:
% W   - set of trajectories, stored in an arry with dimensions
%                 #samples x #variables x #time series
%       in case of equal number of samples or a cell array with
%       #time series entries of dimension #samples x #variables
% M   - input dimension
% ELL - system lag
% OPT - options for the optimization algorithm:
%   OPT.EXCT - vector of indices for exact variables (default [])
%   OPT.SYS0 - initial approximation: an SS system with M inputs,
%              P := size(W, 2) - M outputs, and order N : = ELL * P
%   OPT.DISP - level of display [off | iter | notify | final] (default notify)
%   OPT.MAXITER - maximum number of iterations (default 100)
%   OPT.TOL  - convergence tolerance (default 1e-4)
%
% Outputs:
% SYSH - input/state/output representation of the identified system
% INFO - information from the optimization solver:
%   INFO.M - misfit ||W - WH||_F,
%   INFO.TIME - execution time for the SLRA solver,
%   INFO.ITER - number of iterations
% W    - optimal approximating time series
% XINI - initial conditions under which WH are obtained
```

⟨`misfit` *inline help*⟩≡

```
% MISFIT - Global Total Least Squares misfit ||W - Wh||_F.
%
% [M, wh, xini] = misfit(w, sys, exct)
%
% Inputs:
% W   - given a set of time series, stored in an arry with dimensions
%                 #samples x #variables x #time series           (*)
%       in case of equal number of samples or a cell array with entries
%       of the type (*) in case of different number of samples
% SYS  - state-space system with M inputs, P := size(W, 2) - M outputs,
%        and order N := L * P, where L is an integer
```

```
% EXCT - vector of indices for exact variables (default [])
%
% Outputs:
% M    - misfit ||W - WH||_F
% WH   - optimal approximating time series
% XINI - initial condition, under which WH is obtained
```

B Descriptions of the DAISY datasets

1. *Lake Erie [4]:* data of a simulation related to the identification of the western basin of Lake Erie. The inputs are the water temperature, water conductivity, water alkalinity, NO_3, and total hardness. The outputs are the dissolved oxigen and algae.

2. *Distillation column [5]:* simulated data of an ethane-ethylene distillation column. The inputs are the ratio between the reboiler duty and the feed flow, ratio between the reflux rate and the feed flow, ratio between the distillate and the feed flow, input ethane composition, and top pressure. The outputs are top ethane composition, bottom ethylene composition, and top-bottom differential pressure.

3. *Heating system:* the experiment is a simple single-input single-output heating system. The input drives a 300 Watt Halogen lamp, suspended several inches above a thin steel plate. The output is a thermocouple measurement taken from the back of the plate.

4. *Industrial dryer:* data from an industrial dryer (by Cambridge Control Ltd). The inputs are fuel flow rate, hot gas exhaust fan speed, and rate of flow of raw material. The outputs are dry bulb temperature, wet bulb temperature, and moisture content of raw material.

5. *Hair dryer [8, 9]:* laboratory setup acting like a hair dryer. Air is fanned through a tube and heated at the inlet. The air temperature is measured by a thermocouple at the output. The input is the voltage over the heating device (a mesh of resistor wires).

6. *Ball-beam [14, pages 200–206]:* data of a the ball and beam practicum at ESAT-SISTA. The input is the angle of the beam. The output is the position of the ball.

7. *Flutter [3]:* wing flutter data. Due to industrial secrecy agreements, details are not revealed. The input is highly colored.

8. *Robot arm:* data from a flexible robot arm. The arm is installed on an electrical motor. The transfer function from the measured reaction torque of the structure on the ground to the acceleration of the flexible arm is modeled. The applied input is a periodic sine sweep. The input is reaction torque of the structure. The output is acceleration of the flexible arm.

9. *Glass furnace [18]:* The inputs are the heating input and cooling input. The outputs are produced by 6 temperature sensors in a cross section of the furnace.

10. *Two layer wall:* heat flow density through a two layer wall (brick and insulation layer). The inputs are internal wall temperature and external wall temperature. The output is heat flow density through the wall.

11. *pH neutralization process:* simulation data of a pH neutralization process in a constant volume stirring tank. The inputs are the acid solution flow in liters and base solution flow in liters. The output is the pH of the solution in the tank. This process is a highly non-linear system.

12. *Data of a CD-player arm [6]:* data from the mechanical construction of a CD player arm. The inputs are the forces of the mechanical actuators while the outputs are related to the tracking accuracy of the arm. The data is measured in closed loop, and then through a two-step procedure converted to open loop equivalent data. The inputs are highly colored.

13. *Winding:* the process is a test setup of an industrial winding process. The main part of the plant is composed of a plastic web that is unwinded from first reel (unwinding reel), goes over the traction reel and is finally rewinded on the rewinding reel. Reel 1 and 3 are coupled with a DC-motor that is controlled with input set point currents I1 and I3. The angular speed of each reel (S1, S2 and S3) and the tensions in the web between reel 1 and 2 (T1) and between reel 2 and 3 (T3) are measured by dynamo tachometers and tension meters. The inputs are the angular speed of reel 1 (S1), angular speed of reel 2 (S2), angular speed of reel 3 (S3), set point current at motor 1 (I1), and set point current at motor 2 (I3). The outputs are tension in the web between reel 1 and 2 (T1) and tension in the web between reel 2 and 3 (T3).

14. *Exchanger [1]:* the process is a liquid-satured steam heat exchanger, where water is heated by pressurized saturated steam through a copper tube. The output variable is the outlet liquid temperature. The input variables are the liquid flow rate, the steam temperature, and the inlet liquid temperature. In this experiment the steam temperature and the inlet liquid temperature are kept constant to their nominal values. The heat exchanger process is a significant benchmark for nonlinear control design purposes, since it is characterized by a non minimum phase behavior. The input is the liquid flow rate. The output is the outlet liquid temperature.

15. *Industrial evaporator [19]:* a four-stage evaporator to reduce the water content of a product, for example milk. The inputs are feed flow to the first evaporator stage, vapor flow to the first evaporator stage, and cooling water flow. The outputs are the dry matter content, flow of the outcome product, and temperature of the outcome product.

16. *Tank reactor [7]:* The process is a model of a continuous stirring tank reactor, where the reaction is exothermic and the concentration is controlled by regulating the coolant flow. The input is coolant flow l/min. The outputs are concentration mol/l, and temperature Kelvin degrees.

17. *Steam generator [15]:* the data comes from a model of a steam generator at Abbott power plant in Champaign, IL. The inputs are fuel scaled 0–1, air scaled 0–1, reference level inches, and disturbance defined by the load level. The outputs are drum pressure, excess oxygen in exhaust gases %, level of water in the drum, and steam flow kg/s. To make possible the open loop identification the water level was stabilized by applying to the water flow input a feed-forward action proportional to the steam flow and a control action. The reference of this controller is the third input.