MATLAB code reproducing the simulation results in "Identifiability in the Behavioral Setting"

Identification methods

The model parameters are obtained from the approximate left kernel `R` of a data matrix `D` via rank-`r` approximation. When the structure of `D` is ignored the unstructured low-rank approximation is computed with the function `lra`.

```
function [R, P, Dh] = lra(D, r)
[u, s, v] = svd(D); R = u(:, (r + 1):end)'; P = u(:, 1:r);
if nargout > 2, Dh = u(:, 1:r) * s(1:r, 1:r) * v(:, 1:r)'; end
```

The data matrix `D` may be:

- mosaic Hankel matrix, constructed from multiple trajectories (most general case):

```
function H = moshank(w, i)
if iscell(w)
  N = length(w); H = []; for k = 1:N, H = [H blkhank(w{k}, i)]; end
else
  H = blkhank(w, i);
end
```

- Hankel matrix, constructed from a single trajectory:

```
function H = blkhank(w, i, j)
[q, T] = size(w); if T < q, w = w'; [q, T] = size(w); end
if nargin < 3 | isempty(j), j = T - i + 1; end
if j <= 0, error('Not enough data.'), end
H = zeros(i * q, j);
for ii = 1:i
  H(((ii - 1) * q + 1):(ii * q), :) = w(:, ii:(ii + j - 1));
end
```

- Page matrix, constructed from a single trajectory:

```
function P = page(w, L)
[q, T] = size(w); if T < q, w = w'; [q, T] = size(w); end % assuming T > q
Tp = floor(T / L); P = reshape(w(:, 1:(Tp*L)), q * L, Tp);
```

- trajectory matrix, composed of multiple trajectories of length $\ell+1$.

We consider the SISO case, so that the required rank reduction is by one. The function `w2r` combines four different methods.

```
function R = w2r(w, n, method)
if exist('method') && strcmp(method, 'slra'), R = ss2r(ident(w, 1, n)); return, end
if exist('method')
  W = method(w, n + 1); r = 2 * n + 1;  % data matrix and required rank
```

```
elseif exist('n')
  W = blkhank(w, n + 1); r = 2 * n + 1; % the default method is 'Hankel'
else
  W = w; r = size(W, 1) - 1;
end
R = lra(W, r); R = R / R(end); % approximate left kernel + normlization
```

A conversion from a state space to kernel representation $\mathscr{B} = \ker R(\sigma)$ is done by the function `ss2r`.

```
function R = ss2r(sys)
n = order(sys); [q, p] = tfdata(tf(sys), 'vec');
m = length(q) - 1; R = zeros(1, 2 * (n + 1));
R(1:2:end) = -flip([zeros(n - m) q]); R(2:2:end) = flip(p);
```

The method is selected with the parameter `method`:

- `method = 'slra'` (requires the IDENT package) evokes a structured low-rank approximation method on the (mosaic) Hankel matrix,
- `method = '@blkhank'` (default) evokes `lra` on the Hankel matrix,
- `method = '@page'` evokes `lra` on the Page matrix (default), and
- not specifying both `method` and `n` evokes `lra` on `w`.

The last option is used when `w` consists of stacked next to each other trajectories of length `n+1`.

Projection on the behavior

In order to find the best approximation $\widehat{w}$ of given time series w_{d} within a given model $\mathscr{B} = \ker R(\sigma)$, we solve the problem

$$\text{minimize} \quad \text{over } \widehat{w} \quad \|w_{\mathrm{d}} - \widehat{w}\| \quad \text{subject to} \quad \widehat{w} \in \mathscr{B}|_T,$$

i.e., project w_{d} on $\mathscr{B}|_T$. A fast method for computing the projection is implemented in the function `misfit` of the IDENT package. If the ident package is not installed, a slow method using the function `multmatrix` is given below:

```
function wh = w2wh(w, R)
try
  [M, wh] = misfit(w, r2ss(R)); % fast implementation
catch
  T = size(w, 1); B = null(multmat(R, T, 2));
  wh = B * pinv(B) * vec(w'); wh = reshape(wh, 2, T)';
end

function S = multmat(R, T, q)
[g, nc] = size(R); n = T - nc / q + 1;
S = zeros(n * g, nc + (n - 1) * q);
for i = 1:n
  S((1:g) + (i - 1) * g, (1:nc) + (i - 1) * q) = R;
end

function sysh = r2ss(R)
q = - flip(R(1:2:end)); p = flip(R(2:2:end)); sysh = ss(tf(q, p, 1));
```

In case of multiple trajectories of length $\ell + 1$, the projection on $\mathscr{B}|_{\ell+1}$ is simpler and faster to compute via the function `W2Wh`.

```
function Wh = W2Wh(W, R)
P = null(R); Wh = P * (P \ W);
```

Simulation setup

The data generating system `sys0` is is used as a benchmark for data-driven control in [2] and [1]. It is a 4th order single-input single-output system:

```
%% true-system
q = [ 0 0 0 0.28261 0.50666];
p = [1 -1.41833 1.58939 -1.31608 0.88642];
sys0 = ss(tf(q, p, 0.05)); n = order(sys0); R0 = ss2r(sys0);
```

A kernel representation $\mathscr{B} = \ker R(\sigma)$ is defined by the parameter `R0`

Two data sets are used:

1. a single trajectory $w_{\mathrm{d}} \in (\mathbb{R}^2)^T$ and
2. multiple trajectories $\mathscr{W}_{\mathrm{d}} = \{w_{\mathrm{d}}^1, \ldots, w_{\mathrm{d}}^N\} \subset (\mathbb{R}^2)^{n+1}$.

They are generated in the errors-in-variables setup: $w_{\mathrm{d}} = \bar{w} + \tilde{w}$, where $\bar{w} \in \bar{\mathscr{B}}|_T$ is a true trajectory and the $\tilde{w}$ is additive noise, which is independent identically distributed, zero mean, Gaussian with standard deviation s. The multiple trajectories $w_{\mathrm{d}}^1, \ldots, w_{\mathrm{d}}^N$ are also generated in the errors-in-variables setup from $\bar{\mathscr{B}}$ with the same noise assumptions.

```
%% true-data
u0 = rand(T, 1); x0 = rand(n, 1); y0 = lsim(sys0, u0, [], x0); w0 = [u0, y0];
W0 = zeros(2 * (n+1), N);
for k = 1:N
  u0k = rand(Tshort, 1); x0k = rand(n, 1); y0k = lsim(sys0, u0k, [], x0k);
  W0(:, k) = vec([u0k y0k]');
end

%% noisy-data
w = w0 + s * randn(size(w0)); w = w(1:T, :);
W = W0 + s * randn(size(W0)); W = W(:, 1:N);
```

The validation criteria are:

1. parameter error:

$$e_r := \frac{\|\bar{R} - \widehat{R}\|}{\|\bar{R}\|},$$

```
E{1} = @(Rh) norm(R0 - Rh) / norm(R0);
E_name{1} = '\bar{R}';
```

2. error with respect to the true single trajectory $\bar{w}$:

$$e_{\bar{w}} := \frac{\|\bar{w} - \widehat{w}\|}{\|\bar{w}\|},$$

where $\widehat{w}$ is the projection of $\bar{w}$ on the model $\widehat{\mathscr{B}}$

```
E{2} = @(wh) norm(w0(1:T, :) - wh, 'fro') / norm(w0(1:T, :), 'fro');
E_name{2} = 'w_d';
```

3. error with respect to the noisy single trajectory w_{d}:

$$e_{w_{\mathrm{d}}} := \frac{\|w_{\mathrm{d}} - \widehat{w}\|}{\|w_{\mathrm{d}}\|},$$

where w_{d} is the projection of w_{d} on the model $\widehat{\mathscr{B}}$

```
E{3} = @(wh) norm(w - wh, 'fro') / norm(w, 'fro');
E_name{3} = '\bar{w}';
```

4. error with respect to the true multiple trajectories $\bar{\mathscr{W}}$:

$$e_{\bar{\mathscr{W}}} := \frac{\sqrt{\sum_{i=1}^{N} \|\bar{w}^i - \widehat{w}^i\|_2^2}}{\sqrt{\sum_{i=1}^{N} \|\bar{w}^i\|_2^2}},$$

where $\widehat{w}^i$ is the projection of $\bar{w}^i$ on the model $\widehat{\mathscr{B}}$, and

```
E{4} = @(Wh) norm(W0(:, 1:N) - Wh, 'fro') / norm(W0(:, 1:N), 'fro');
E_name{4} = '\bar{\cal W}';
```

5. error with respect to the noisy multiple trajectories $\mathscr{W}_\mathrm{d}$:

$$e_{\mathscr{W}_\mathrm{d}} := \frac{\sqrt{\sum_{i=1}^{N} \|w_\mathrm{d}^i - \widehat{w}^i\|_2^2}}{\sqrt{\sum_{i=1}^{N} \|w_\mathrm{d}^i\|_2^2}},$$

where $\widehat{w}^i$ is the projection of w_d^i on the model $\widehat{\mathscr{B}}$.

```
E{5} = @(Wh) norm(W - Wh, 'fro') / norm(W, 'fro');
E_name{5} = '{\cal W}_d';
```

```
%% validate
wh = w2wh(w, Rh); Wh = W2Wh(W, Rh);
errors = [E{1}(Rh) E{2}(wh) E{3}(wh) E{4}(Wh) E{5}(Wh)];
```

Simulation example

The key question we want to explore empirically is:

> Q: For LTI system identification, should we use data from one "long" or multiple "short" experiments?

More specifically, "short" is $(\ell+1)$-samples long trajectories, where ℓ is the lag of the system. In this case, identification of the kernel parameter becomes an unstructured low-rank approximation of the data matrix. Therefore, in the errors-invariables setting assuming independent identically distributed zero mean white Gaussian noise, the `lra` method (truncation of the singular value decomposition) is statistically optimal (*i.e.*, it is maximum-likelihood).

```
%% identification-methods
M{1} = @() w2r(w, n, @blkhank); M_name{1} = 'Hankel'; M_line{1} = 'b--';
M{2} = @() w2r(w, n, @page);    M_name{2} = 'Page';   M_line{2} = 'r.-';
M{3} = @() w2r(W);              M_name{3} = 'Traj.';  M_line{3} = 'r-.';
M{4} = @() w2r(w, n, 'slra');   M_name{4} = 'ML';     M_line{4} = 'b:';
```

If the same total number of noisy samples are used in both—single and multiple experiments cases—empirical evidence suggests that the answer to question Q is: one long experiment. Next, we compare the performance of the methods when the corresponding data matrices have the same number of columns. Note that in this case the comparison is unfair, giving advantage to the method using multiple short trajectories. Indeed, a factor of ℓ more samples are used in the data matrix constructed from the multiple trajectories.

The data matrix are in the case of a

- single trajectory is $\mathscr{H}_{n+1}(w_\mathrm{d}) \in \mathbb{R}^{2(n+1)\times(T-n)}$ and
- multiple trajectories $\mathscr{H}_{n+1}(\mathscr{W}_\mathrm{d}) \in \mathbb{R}^{2(n+1)\times N}$.

We set

- $T = N(\ell+1)$, so that the total number of samples in w_d and $\mathscr{W}_\mathrm{d}$ is equal, or
- $T = N+n$, so that the number of columns in $\mathscr{H}_{n+1}(w_\mathrm{d})$ and $\mathscr{H}_{n+1}(\mathscr{W}_\mathrm{d})$ is equal.

```
%% define-T
T = N * Tshort; % same total number of samples in the Hankel and trajectory matrices
% T = N + 4; % same number of columns in the Hankel and trajectory matrices
```

```
clear all, close all
s = 0.1; Tshort = 5; N = 200; nmc = 500;
<<define-T>>
<<setup>>

%% Monte-Carlo simulation
for i = 1:nmc
  <<noisy-data>>
  <<identification-methods>>
  <<validation-criteria>>
  for k = 1:length(M)
    Rh = M{k}();
    <<validate>>
    e(i, k, :) = errors;
  end
end

%% show results
res = squeeze(mean(e));
[[' '; M_name'] [E_name; num2cell(res)]]
```

Results

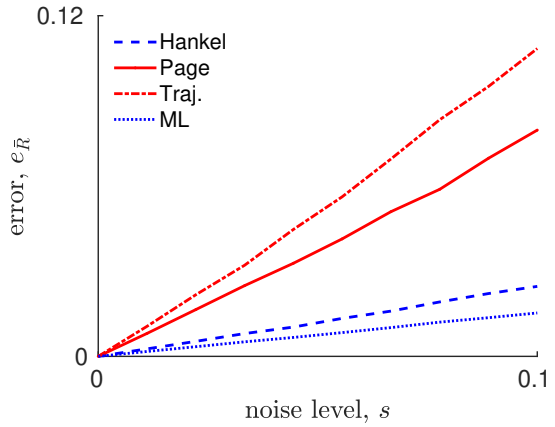
```
{' '      }    {'\bar{R}'}    {'w_d'    }    {'\bar{w}'}    {'{\cal W}_d'}    {'\bar{\cal W}'}
{'Hankel'}    {[ 0.0236]}    {[0.0914]}    {[ 0.0888]}    {[    0.1678]}    {[      0.0545]}
{'Page'  }    {[ 0.0712]}    {[0.1133]}    {[ 0.1115]}    {[    0.1681]}    {[      0.0553]}
{'Traj.' }    {[ 0.1134]}    {[0.2006]}    {[ 0.1985]}    {[    0.1684]}    {[      0.0529]}
{'ML'    }    {[ 0.0154]}    {[0.0910]}    {[ 0.0884]}    {[    0.1678]}    {[      0.0545]}
```

Error vs noise variance plot (figure 1)

```
clear all, close all
Tshort = 5; N = 200; nmc = 500;
np = 10; S = linspace(0, 0.1, np);
<<define-T>>
<<setup>>

%% Monte-Carlo simulation
for j = 1:np
  s = S(j)
  for i = 1:nmc
    <<noisy-data>>
    <<identification-methods>>
    <<validation-criteria>>
```

```
    for k = 1:length(M)
      Rh = M{k}();
      <<validate>>
      e(i, k, j, :) = errors;
    end
  end
end

X = S; f_name = 'f1'; legend_position = 'NorthWest'; x_label = 'noise level, $s$';
<<plot-results>>

%% plot-results
res = squeeze(mean(e));
for i = 1:length(E)
  figure(i), hold on
  for k = 1:length(M)
    plot(X, res(k, :, i), M_line{k})
  end
  xlabel(x_label, 'Interpreter','latex'),
  ylabel(['error, $e_{' E_name{i} '}$'], 'Interpreter','latex')
  legend(M_name, 'location', legend_position)
  legend boxoff, set(gca, 'box', 'off')
  ax = axis; axis([X(1) X(end) ax(3:4)])
  set(gca, 'xtick', [X(1) X(end)], 'ytick', ax(3:4))
  set(gca, 'fontsize', 17), eval(['print -depsc ' f_name int2str(i) '.eps'])
end
save(f_name)
```

Results

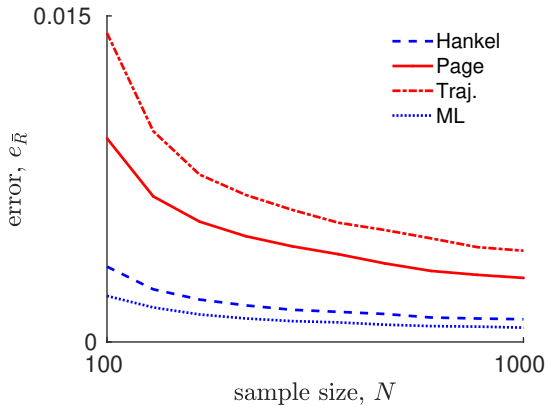


Error vs noise variance plot (figure 2)

```
clear all, close all
Tshort = 5; N = 1000; s = 0.01; nmc = 500;
np = 10; NN = linspace(100, N, np);
<<define-T>>
<<setup>>
```

```
%% Monte-Carlo simulation
for j = 1:np
  N = NN(j),
  <<define-T>>
  for i = 1:nmc
    <<noisy-data>>
    <<identification-methods>>
    <<validation-criteria>>
    for k = 1:length(M)
      Rh = M{k}();
      <<validate>>
      e(i, k, j, :) = errors;
    end
  end
end

X = NN; f_name = 'f2'; legend_position = 'NorthEast'; x_label = 'sample size, $N$';
<<plot-results>>
```

Results



References

[1] S. Formentin and I. Markovsky. "A comparison between structured low-rank approximation and correlation approach for data-driven output tracking". In: *Proc. of the IFAC Symposium on System Identification*. 2018, pp. 1068–1073.

[2] I. Landau et al. "A Flexible Transmission System as a Benchmark for Robust Digital Control". In: *European Journal of Control* 1.2 (1995), pp. 77–96.