A missing data approach to data-driven filtering and control: Simulation examples addendum

Ivan Markovsky

Contents

1 EIV Kalman smoothing **1**

2 Step response simulation **3**

3 Linear quadratic step tracking **5**

4 Realization **7**

5 Noisy realization **8**

This document shows simulation examples with the matrix completion approach for data-driven signal processing, presented in "A missing data approach to data-driven filtering and control". The EIV Kalman smoothing example is included in Section VI of the paper. Due to space limitations, the other examples are presented only in this document. The full code used to generate the results is listed, making the results easily reproducible.

1 EIV Kalman smoothing

- Problem (the references correspond to the equations and bibliography in the paper)
  - classical: (4) + (8)
  - data-driven: (7)
- Data generating system: $\overline{\mathscr{B}} = \{ (u,y) \mid u - \sigma u + \sigma^2 u = 0.81y - 1.456\sigma y + \sigma^2 y \}$.

```
% <system-1>
clear all, n = 2; sys0 = ss(tf([1 -1 1], [1 -1.456 0.81], 1));
```

- Identification data: $w_{\mathrm{d}} = \overline{w}_{\mathrm{d}} + \widetilde{w}$, where $\overline{w}_{\mathrm{d}} \in \overline{\mathscr{B}}$ and $\widetilde{w}$ is zero mean, white Gaussian noise.

```
% <ident-data>
Td = 100; rng('default')
ud0 = rand(Td, 1); yd0 = lsim(sys0, ud0); wd0 = [ud0 yd0];
wt = randn(Td, 2); wd = wd0 + 0.1 * wt / norm(wt) * norm(wd0);
```

- Trajectory $w = w_\mathrm{p} \wedge w_\mathrm{f}$: w_p missing, $w_\mathrm{f} = \overline{w}_\mathrm{f} + \widetilde{w}_\mathrm{f}$, where $\overline{w}_\mathrm{f}$ is the step response of $\overline{\mathscr{B}}$ and $\widetilde{w}_\mathrm{f}$ is zero mean, white Gaussian noise.

```
% <w-est>
wp = NaN * ones(n, 2); Tf = 20;
uf0 = ones(Tf, 1); yf0 = step(sys0, Tf - 1); wf0 = [uf0 yf0];
wft = randn(Tf, 2); wf = wf0 + 0.05 * wft / norm(wft) * norm(wf0);
```

- Validation criterion: $e := ||\overline{w}_\mathrm{f} - \widehat{w}_\mathrm{f}||/||\overline{w}_\mathrm{f}||$

```
% <est-error>
e = @(wfh) norm(wf0 - wfh, 'fro') / norm(wf0, 'fro');
```

- Model based Kalman smoother: (19)

```
function [wh, x0h] = eiv_ks(w, sys)
T = size(w, 1); n = size(sys, 'order'); % assume SISO
h = impulse(sys, T - 1);
TT = toeplitz(h, [h(1) zeros(1, T - 1)]);
I = eye(n);
for i = 1:n,
  O(:, i) = initial(sys, I(:, i), T - 1);
end
A = [zeros(T, n) eye(T); O TT];
x0uh = A \ w(:);
yh = [O TT] * x0uh;
wh = [x0uh(n + 1:end), yh];
x0h = x0uh(1:n);
```

- Subspace data-driven method: [11]

```
function wh = eiv_ks_dd(wd, w, n, m)
if ~exist('m') || isempty(m), m = 1; end
Tf = size(w, 1); H = blkhank(wd, Tf);
[R, P] = lra(H, Tf * m + n);
wh = reshape(P * P' * vec(w'), 2, Tf)';
```

- We verify that with knowledge of the true model $\overline{\mathscr{B}}$, the result of estimating the missing values in w, using the `misfit` function, gives the correct result, *i.e.*, it coincides with the result of the `eiv_ks`.

```
% <est-check>
[M, wh] = misfit([wp; wf], sys0);
wfh = wh(n + 1:end, :);
wfh_ks  = eiv_ks(wf, sys0);
check_misfit = norm(wfh - wfh_ks, 'fro') / norm(wfh_ks, 'fro')
```

$$\Rightarrow \quad 3.7316\,(\text{-}09)$$

("⇒" indicates a result of the evaluation of the previous block of code.)

The subspace data-driven method `eiv_ks_dd` also gives the correct result when the data w_d is exact, i.e., $w_\mathrm{d} = \overline{w_\mathrm{d}}$

```
wfh_ss = eiv_ks_dd(wd0, wf, n);
check_ss = norm(wfh_ks - wfh_ss, 'fro') / norm(wfh_ks, 'fro')
```

$$\Rightarrow \quad 1.2531\,(\text{-}15)$$

- Next, we compare the results of
  1. `ident` i.e., estimation of the missing values in w without using the true model,
  2. `ident` + `eiv_ks` i.e., identification of a model $\widehat{\mathscr{B}}$ for $\overline{\mathscr{B}}$ using the data w_d and applying the model based Kalman smoother `eiv_ks` on w_f with the identified model $\widehat{\mathscr{B}}$, and
  3. `eiv_ks_dd`, i.e., the alternative subspace data-driven EIV Kalman filtering method.

```
% <est-id>
[sysh, info, wh] = ident({wd, [wp; wf]}, 1, n);

[sysh_id, info_id] = ident(wd, 1, n);
wfh_id = eiv_ks(wf,sysh_id);

wfh_ss = eiv_ks_dd(wd, wf, n);

[e(wh{2}(n + 1:end, :)) e(wfh_id) e(wfh_ss) e(wf)]
```

$$\Rightarrow \quad 0.0310 \quad 0.0315 \quad 0.0412 \quad 0.0581$$

For explanation of the result see Section VI of the paper.

- Code from this section: `test_est.m`

2 Step response simulation

- Problem
  - classical: (4) + (9)
  - data-driven problem:

$$\text{minimize} \quad \text{over } \widehat{w}_\mathrm{d} \text{ and } \widehat{y}_\mathrm{f} \quad \|w_\mathrm{d} - \widehat{w}_\mathrm{d}\| \quad \text{subject to} \quad \widehat{w} = w_\mathrm{p} \wedge (u_\mathrm{f}, \widehat{y}_\mathrm{f}) \in \mathscr{B}_{\mathrm{mpum}}(\widehat{w}_\mathrm{d}) \in \mathscr{L}_{1,\ell}$$

- Data generating system $\overline{\mathscr{B}}$ and the identification data w_d are the same as in the EIV Kalman smoothing example.
- Trajectory $w = w_\mathrm{p} \wedge w_\mathrm{f}$: $w_\mathrm{p} = 0$ exact, $u_\mathrm{f} = 1$ exact, and y_f missing.

```
% <w-simulation>
Tf = 20; s0 = step(sys0, Tf - 1);
uf = ones(Tf, 1); yf = NaN * ones(Tf, 1); wf = [uf yf];
```

- Validation criterion: $e := ||\bar{s} - \widehat{y}_{\mathrm{f}}||/||\bar{s}||$, where $\bar{s}$ is the step response of the system $\overline{\mathscr{B}}$.

```
% <sim-error>
e = @(sh) norm(s0 - sh) / norm(s0);
```

- We verify that with knowledge of the true model $\overline{\mathscr{B}}$, the result of estimating the missing values in w, using the `misfit` function is correct, *i.e.*, $\widehat{y}_{\mathrm{f}} = \bar{s}$
  - `opt.exct = 1` specifies that the u component of the trajectory $w = (u, y)$ is exact, and
  - `opt.wini = 0` specifies zero initial conditions.

```
% <sim-check>
opt.exct = 1; opt.wini = 0;
[M, wfh] = misfit(wf, sys0, opt); sh = wfh(:, 2); e(sh)
```

$$\Rightarrow \quad 7.5067\text{-}16$$

- Next, we compare the results of
  1. `ident` i.e., estimation of the missing values in w without using the true model, and
  2. `ident + step` i.e., identification of a model $\widehat{\mathscr{B}}$ for $\overline{\mathscr{B}}$ using the data w_{d} and simulating the impulse response, using the identified model $\widehat{\mathscr{B}}$, and
  3. subspace data-driven simulation method [11], implemented in the function `uy2h`.

```
% <sim-result>
opt.exct = {[], 1}; opt.wini = {[], 0};
[sysh, info, wh] = ident({wd wf}, 1, n, opt);
sh = wh{2}(:, 2);

[sysh_id, info_id, wd_id] = ident(wd, 1, n);
sh_id = step(sysh_id, Tf - 1);

addpath ~/mfiles/detss
sh_ss = cumsum(uy2h(wd(:, 1), wd(:, 2), 2, 2, Tf));

[e(sh) e(sh_id) e(sh_ss)]
```

$$\Rightarrow \quad 0.0655 \quad 0.0655 \quad 0.1356$$

The results of the data-driven and classical approaches are equal up to numerical errors. The reason is that the objective function of the data-driven problem coincides with the objective function of the identification problem. Indeed, $||w - \widehat{w}||_v = 0$, because w contains only exact and missing data. Another justification of the equivalence of the classical and the data-driven methods for data-driven simulation is that in this case the trajectory w_{f} does not carry information about the data generating system. Thus, the system identification and data-driven simulation methods use the same data.

The result of the subspace method has higher estimation error. Indeed, the subspace method does not use nonlinear optimization and as a result yields a suboptimal estimator, while the other methods are statistically optimal (maximum likelihood estimators in the EIV setting).

- Code from this section: `test_sim.m`

3 Linear quadratic step tracking

- Problem
  - classical: (4) + (10)
  - data-driven:

$$\text{minimize} \quad \text{over } \widehat{w}_\mathrm{d} \text{ and } \widehat{w}_\mathrm{f} \quad \gamma \underbrace{\|w_\mathrm{d} - \widehat{w}_\mathrm{d}\|_2^2}_{\text{identification error}} + \underbrace{\|y_\mathrm{f} - \widehat{y}_\mathrm{f}\|_2^2}_{\text{tracking error}} \quad \text{subject to} \quad \widehat{w} = w_\mathrm{p} \wedge \widehat{w}_\mathrm{f} \in \mathscr{B}_{\mathrm{mpum}}(\widehat{w}_\mathrm{d}) \in \mathscr{L}_{1,\ell}.$$

  The parameter γ is user defined and allow for a trade-off between identification and tracking errors.

- Data generating system: the system used above is invertible, so that any reference output can be tracked perfectly by a suitable input. In order to have tracking error, we modify the system so that it is not invertible.

```
% <system-2>
n = 2; sys0 = ss(tf([1 -1],[1 -1.456 0.81],1));
```

- Identification data w_d: generated in the same way as above (EIV setup)
- Initial conditions for the tracking problem: set to zero
- Trajectory $w = w_\mathrm{p} \wedge w_\mathrm{f}$: $w_\mathrm{p} = 0$ exact, u_f missing, $y_\mathrm{f} = 1$

```
% <w-ctr>
Tf = 30; uf = NaN * ones(Tf, 1); yf = ones(Tf, 1); wf = [uf yf];
```

- Validation criterion: $e := ||y_\mathrm{f} - \widehat{y}_\mathrm{f}||/||y_\mathrm{f}||$

```
% <ctr-error>
e = @(yfh) norm(yf - yfh) / norm(yf);
```

- Model based least-squares output tracking: minimize over u_f $||y_\mathrm{f} - \mathscr{O}_{T_\mathrm{f}} x_\mathrm{ini} - \mathscr{T}_{T_\mathrm{f}} u_\mathrm{f}||$

```
function [uh, yh] = ls_track(sys, y, x0)
[T, p] = size(y); n = size(sys, 'order'); % asume single input
if nargin < 3, x0 = zeros(n, 1); end
h = impulse(sys, T - 1); h = vec(h');

TT = zeros(p * T, T);
for i = 1:T,
  TT(:, i) = [zeros(p * (i - 1), 1); h(1:(p * (T - i + 1)))];
end
```

```
I = eye(n);
for i = 1:n,
  O(:, i) = vec(initial(sys, I(:, i), T - 1)');
end

uh = pinv(TT) * (vec(y') - O * x0);
yh = reshape(O * x0 + TT * uh, p, T)';
```

- We verify that with knowledge of the true model $\overline{\mathscr{B}}$, the result of estimating the missing values in w, using the `misfit` function is correct, *i.e.*, it coincides with the result of the `ls_track`.

```
% <ctr-check>
opt.wini = 0; % zero initial conditions
[M, wfh] = misfit(wf, sys0, opt); ufh = wfh(:, 1);
[ufh_ls, yfh_ls] = ls_track(sys0, yf); ufh_ls_opt = ufh_ls;
norm(ufh_ls - ufh) / norm(ufh_ls)
```

$$\Rightarrow \quad 4.2057\,(\text{-}05)$$

- Next, we compare the results of

  1. `ident` i.e., estimation of the missing values in w without using the true model, and
  2. `ident` + `ls_track` i.e., identification of a model $\widehat{\overline{\mathscr{B}}}$ for $\overline{\mathscr{B}}$ using the data w_{d} and least-squares tracking of y_{f} using the identified model $\widehat{\overline{\mathscr{B}}}$.

```
% <ctr-results>
opt.wini = 0; opt.method = 'q';
[sysh, info, wfh] = ident({100 * wd, wf}, 1, n, opt); ufh = wfh{2}(:, 1);
[sysh_id, info_id] = ident(wd, 1, n, opt);
[ufh_ls, yfh_ls] = ls_track(sysh_id, yf);
```

Applying the control signals on the *models*, we have

```
yfh = lsim(sysh, ufh);
yfh_ls = lsim(sysh_id, ufh_ls);
[e(yfh) e(yfh_ls)]
```

$$\Rightarrow \quad 0.1645 \quad 0.1644$$

Applying the control signals on the *true system*, we have

```
yfh = lsim(sys0, ufh);
yfh_ls = lsim(sys0, ufh_ls);
yfh_ls_opt = lsim(sys0, ufh_ls_opt);
[e(yfh) e(yfh_ls) e(yfh_ls_opt)]
```

$$\Rightarrow \quad 0.2602 \quad 0.2604 \quad 0.1826$$

- Code from this section: `test_ctr.m`

4 Realization

- Data generating system $\overline{\mathscr{B}}$: the same as in the EIV Kalman smoothing example.
- Identification data: $w_\mathrm{d} = w_\mathrm{p} \wedge (u_{\mathrm{f},1} y_{\mathrm{f},1})$, see below
- Trajectory $w = w_\mathrm{p} \wedge w_\mathrm{f}$: $w_\mathrm{p} = 0$ exact, $u_\mathrm{f} = \delta$ (unit pulse) exact, $y_\mathrm{f} = y_{\mathrm{f},1} \wedge y_{\mathrm{f},2}$, where
  - $y_{\mathrm{f},1} = \bar{h}|_{T_{\mathrm{f},1}}$, with $\bar{h}$ the impulse response of $\overline{\mathscr{B}}$
  - $y_{\mathrm{f},2}$ missing

```
% <w-realization>
Tf = 30; Tf1 = 25;
uf = zeros(Tf, 1); uf(1) = 1;
yf0 = impulse(sys0, Tf - 1);
yf = yf0; yf(Tf1 + 1:end) = NaN;
wf = [uf yf];
```

- Validation criterion: $e := ||\bar{h}_2 - \widehat{y}_{\mathrm{f},2}||/||\bar{h}_2||$, where $\bar{h}_2 := \big(\bar{h}(T_{\mathrm{f},1}+1), \ldots, \bar{h}(T_\mathrm{f})\big)$

```
% <realization-error>
e = @(yfh) norm(yf0(Tf1+1:end) - yfh(Tf1+1:end)) / norm(yf0(Tf1+1:end));
```

- We verify that with knowledge of the true model $\overline{\mathscr{B}}$, the result of estimating the missing values in w, using the `misfit` function, gives the correct result, *i.e.*, it coincides with the result of the `impulse` function.

```
% <realization-check>
opt.wini = 0; opt.exct = 1;
[M, wh] = misfit(wf, sys0, opt); e(wh(:, 2))
```

$$\Rightarrow \quad 1.1100\,(\text{-}14)$$

- Next, we verify that without knowledge of the true model $\overline{\mathscr{B}}$, the result of estimating the missing values in w, using the `ident` function, also gives the correct result. As alternative method we use the nuclear norm minimization, implemented in the function `hmc_nn`.

```
% <realization-result>
[sysh, info, wfh] = ident(wf, 1, n, opt);
wh = hmc_nn([zeros(2); wf], n, 1); wh(1:2, :) = [];
[e(wfh(:, 2)) e(wh(:, 2))]
```

$$\Rightarrow \quad 0.1006\,(\text{-}06) \quad 0.0203\,(\text{-}06)$$

- Code from this section: `test_realization.m`

5 Noisy realization

- Data generating system $\overline{\overline{\mathscr{B}}}$, identification data w_{d}, trajectory w, and validation criterion are the same as in the realization problem, expect for
  - $y_{\mathrm{f},1} = \bar{h}|_{T_{\mathrm{f},1}} + \widetilde{y}_{\mathrm{f},1}$, where $\widetilde{y}_{\mathrm{f},1}$ is zero-mean, white, Gaussian noise.

```
% <w-noisy-realization>
yfn = randn(Tf, 1); yf = yf0 + 0.1 * yfn / norm(yfn) * norm(yf0);
yf(Tf1 + 1:end) = NaN; wf = [uf yf];
```

- We compare the estimation errors of the data-driven method and the nuclear norm minimization method:

```
% <noisy-realization-result>
opt.wini = 0; opt.exct = 1;
[sysh, info, wfh] = ident(wf, 1, n, opt);
wh = hmc_nn([zeros(2); wf], n, 1); wh(1:2, :) = [];
[e(wfh(:, 2)) e(wh(:, 2))]
```

$$\Rightarrow \quad 0.1458 \quad 0.3671$$

- Code from this section: `test_noisy_realization.m`