Proceedings of Machine Learning Research vol 242:130–141, 2024

The Behavioral Toolbox

Ivan Markovsky IMARKOVSKY@CIMNE.UPC.EDU
Catalan Institution for Research and Advanced Studies (ICREA), Pg. Lluis Companys 23, 08010 Barcelona, and Int. Centre for Numerical Methods in Engineering (CIMNE), Gran Capità, 08034 Barcelona, Spain

Abstract

The Behavioral Toolbox is a collection of Matlab functions for modeling, analysis, and design of dynamical systems using the behavioral approach to systems theory and control. It implements newly emerged direct data-driven methods as well as classical parametric representations of linear time-invariant systems. At the core of the toolbox is a nonparameteric representation of the finite-horizon behavior by an orthonormal basis. The current version has education and research goals and isn't intended for handling "big data". The paper presents five problems—checking systems equality, interconnection of systems, errors-in-variables least-squares smoothing, missing input estimation, and data-driven forecasting—and describes their solution by the methods in the toolbox.

Keywords: Behavioral approach, Data-driven methods, System identification, Matlab, Software.

1. Introduction

At the core of the behavioral approach is the notion of a dynamical system as a set of trajectories—the behavior Willems (1986, 1987, 2007). For numerical computations, however, a representation of the behavior is needed. The representation used in the Behavioral Toolbox is a basis for the behavior restricted to a finite-horizon. It is nonparameteric and has close connection to newly emerged data-driven methods, inspired by Willems et al. (2005) fundamental lemma. Unlike classical nonparameteric representations, such as the convolution, a basis for the restricted behavior doesn't assume an input/output partitioning of the variables and is a direct application of the behavioral approach.

For an in-depth introduction to the philosophy of the behavioral approach, its differences from the classical approach, and its relation to data-driven methods for systems and control, refer to the overview papers Markovsky and Dörfler (2021); Markovsky et al. (2023a). The notation in this paper is also the same as in the overviews (see Table 1). The methodology used in the Behavioral Toolbox is published in Markovsky and Rapisarda (2008); Markovsky (2013, 2017); Markovsky and Dörfler (2023, 2022). The contribution of this paper is the first software package for modeling, analysis, and design of dynamical systems in the behavioral setting.

$w \in (\mathbb{R}^q)^{\mathbb{N}}$, $w : \mathbb{N} \to \mathbb{R}^q$	q-variate real discrete-time signal with time axis $\mathbb{N}$
$w\vert_T := \big(w(1), \ldots, w(T)\big)$	restriction of w to the interval $[1, T]$
$w = w_{\text{ini}} \wedge w_{\text{f}}$	concatenation of trajectories w_{ini} and w_{f}
$\mathscr{B} \subset (\mathbb{R}^q)^{\mathbb{N}}$	discrete-time dynamical system with q variables
$\mathscr{B}\vert_T := \{ w\vert_T \mid w \in \mathscr{B} \}$	restriction of $\mathscr{B}$ to the interval $[1, T]$
$\boldsymbol{m}(\mathscr{B})$ / $\boldsymbol{\ell}(\mathscr{B})$ / $\boldsymbol{n}(\mathscr{B})$	number of inputs / lag / order of $\mathscr{B}$
$\mathscr{H}_T(w)$	Hankel matrix with T block rows constructed from w

Table 1: Summary of notation.

We showcase the Behavioral Toolbox in five examples. They admit solutions in the classical setting as well as the behavioral setting. The latter are based on standard linear algebraic operations. They are conceptually simple and easy to implement. They are also applicable when the underlying system is unknown but data are available from the system. In this case, the methods in the toolbox achieve a direct map from the observed data to the desired solution. Classical solutions require a preliminary identification step. The current implementation of the methods in the Behavioral Toolbox, however, makes them less efficient computationally than alternative classical methods.

First, we consider the basic question when two systems are equal. In the classical setting, equality of linear time-invariant systems represented in state-space can be verified by finding a state transformation that makes the corresponding systems' parameters equal. In the behavioral setting, the question is equivalent to verifying equality of subspaces—a classical linear algebra problem. The second example is interconnection of systems. As a specific example, we show that series connection in the input/output setting can be obtained by intersection of behaviors. The third example is optimal signal-from-noise separation. The classical solution, which uses a state-space representation of the data-generating system, is the celebrated Kalman smoother. The interpretation of the problem in the behavioral setting is projection on the behavior, *i.e.*, finding the nearest trajectory of a system to a given signal. The fourth example is the observer problem in the behavoral setting, *i.e.*, inferring one set of variables of a dynamical system from another set of variables. The observer is applied for estimation of a missing input. The fifth example illustrates the direct data-driven approach on a forecasting problem, where instead of a system a trajectory of the system is given.

The toolbox, the examples in this paper, and additional examples are available from https://imarkovs.github.io/bt.tar

2. When are two systems equal?

The question "How to check if two systems are equal?" immediately leads to the questions "How are the systems specified?" and "What does it mean that they are equal?". Intuitively we consider two systems equal when they have the same "external behaviors". The behavioral approach turns this intuition into a definition: the system *is* the external behavior. Transfer functions and state-space equations can be used to *represent* systems, but are not systems. Thus, from the behavioral perspective, statements such as "transfer function system" and "the system $x(t+1) = Ax(t) + Bu(t)$" are nonsense. This is not *just* a terminology convention. As discussed next, the distinction between a system (set of trajectories) and representation (equation that represents the system) is essential.

Let's create a random discrete-time linear time-invariant system, defined by a state-space representation:

```
m = 2; p = 2; n = 3; sys1 = drss(n, p, m);
```

and another system that is obtained from the first one by a random change of the state-space basis:

```
sys2 = ss2ss(sys1, rand(n));
```

The external behaviors of the two systems are the same and in this sense the systems are *equal*. This fact, however, can not be inferred by comparing directly their state-space parameters and the equal to operation == is not supported in the Control System Toolbox of Matlab:

```
sys1 == sys2 % -> error
```

gives error "`Operator '==' is not supported for operands of type 'ss'.`"

The problem of finding when two systems are equal admits many solutions. The reason == is not supported is due to the perception that linear time-invariant systems (`ss` objects in particular) are not comparable. In fact, systems when viewed as behaviors are comparable. The shift of perception brought by the behavioral approach that *system are not parameters of a particular representation* has far reaching consequences. In the behavioral setting, the answer to the question when a system $\mathscr{B}^1$ is equal to a system $\mathscr{B}^2$ follows directly from the definition of a system as a set of trajectories: "when $\mathscr{B}^1 = \mathscr{B}^2$". The answer doesn't suggest methods for doing the job, however. For this, one has to choose particular representations of the systems; hence the many possible solution methods.

Another shift of perception brought by the behavioral approach is that definitions (*e.g.*, "$\mathscr{B}^1$ equals $\mathscr{B}^2$" means "$\mathscr{B}^1 = \mathscr{B}^2$") are stated in terms of the behavior, *i.e.*, systems' definitions do not involve systems' representations. Also, problems related to systems (*e.g.*, check if $\mathscr{B}^1 = \mathscr{B}^2$) are stated without involving systems' representations. The problem statement shows "what we are after", irrespective of how we may go about achieving it. Indeed, when the systems are initially given by some representations, say state-space, it may be advantageous to switch to another one, say transfer functions. *Choosing the representation is part of the solution, not the problem formulation.*

The Behavioral Toolbox makes the abstract set theoretic framework of the behavioral approach actionable by restricting the infinite-horizon behavior $\mathscr{B}$ to a finite time-horizon $[1,T]$, resulting in a finite-dimensional subspace $\mathscr{B}|_T \subset (\mathbb{R}^q)^T$, and constructing a basis $\{b^1,\ldots,b^r\}$ for $\mathscr{B}|_T$, *i.e.*,

$$\mathscr{B}|_T = \text{image} \underbrace{\begin{bmatrix} b^1 & \cdots & b^r \end{bmatrix}}_{B_T}, \qquad \text{where } r := \dim \mathscr{B}|_T. \tag{B_T}$$

Provided that the horizon T is "long enough" (this often means T larger than the lag of the system), a problem about $\mathscr{B}$ can be reduced to an equivalent problem about $\mathscr{B}|_T$. In particular, one can check $\mathscr{B}^1 = \mathscr{B}^2$ by checking equality of finite-dimensional subspace $\mathscr{B}^1|_T = \mathscr{B}^2|_T$, with $T = \max\{\ell(\mathscr{B}^1), \ell(\mathscr{B}^2)\}$ — a basic linear algebra problem, for which there are existing methods.

The function `equal` of the toolbox implements the check for equality of systems:

```
equal(sys1, sys2) % -> true
```

First, it converts the `ss` objects `sys1` and `sys2` to bases B_T^1 and B_T^2 for the finite-horizon behaviors $\mathscr{B}^1|_T$ and $\mathscr{B}^2|_T$. This is done by a "data-driven approach" that simulates a random trajectory of the system, constructs a Hankel matrix of the trajectory, and computes a basis for the image of the Hankel matrix (see Section 7.1). Then, $\mathscr{B}^1|_T = \mathscr{B}^2|_T$ is checked numerically by 1) verifying that the distance, defined as the principal angle (Golub and Van Loan, 1996, Section 12.4), between $\mathscr{B}^1|_T$ and $\mathscr{B}^2|_T$ is smaller than a tolerance related to the numerical errors and 2) $\dim \mathscr{B}^1|_T = \dim \mathscr{B}^2|_T$.

3. Interconnection of systems

In the classical setting, interconnection of systems is done by *input-to-output connection*: the output of one system is fed to an input of another system. In the behavioral setting, interconnection is viewed as *variables sharing*: some variables of one system are set equal to some variables of another system. Input-to-output connection is a special case of variables sharing. Indeed, the former restricts the latter by selecting input/output partitionings of the variables of the systems and equating input to output variables only. Willems (2007) argues that interconnection of *physical systems* is always variables sharing and only incidentally (typically in man-made systems) input-to-output assignment.

Next, we show how series connection fits into the general setting of interconnection by variables sharing. Consider two single-input single-output linear time-invariant systems $\mathscr{B}^1$ and $\mathscr{B}^2$ with input/output partitionings $w^1 = \begin{bmatrix} w_1^1 \\ w_2^1 \end{bmatrix} = \begin{bmatrix} u^1 \\ y^1 \end{bmatrix}$ and $w^2 = \begin{bmatrix} w_1^2 \\ w_2^2 \end{bmatrix} = \begin{bmatrix} u^2 \\ y^2 \end{bmatrix}$.

```
n1 = 2; B1 = drss(n1);
n2 = 2; B2 = drss(n2);
```

The variables sharing corresponding to the series connection of $\mathscr{B}^1$ and $\mathscr{B}^2$ is $w_2^1 = w_1^2$, or written in a "kernel form":

$$\begin{bmatrix} 0 & 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} w^1 \\ w^2 \end{bmatrix} = 0. \qquad (w_2^1 = w_1^2)$$

```
R3 = [0 1 -1 0];
```

Equation ($w_2^1 = w_1^2$) defines the static system $\mathscr{B}^3 := \{ w \mid w_2^1 = w_1^2 \}$.

In order to construct the interconnection system $\mathscr{B}_{\text{int}}$, first, we define the joint behavior

$$\text{append}(\mathscr{B}^1, \mathscr{B}^2) := \left\{ \begin{bmatrix} w^1 \\ w^2 \end{bmatrix} \;\middle|\; w^1 \in \mathscr{B}^1,\ w^2 \in \mathscr{B}^2 \right\} \qquad \text{(append)}$$

of $\mathscr{B}^1$ and $\mathscr{B}^2$ without the interconnection law. The interconnection system $\mathscr{B}_{\text{int}}$ is obtained then as the intersection of the joint behavior $\text{append}(\mathscr{B}^1, \mathscr{B}^2)$ with $\mathscr{B}^3$,

$$\mathscr{B}_{\text{int}} = \text{append}(\mathscr{B}^1, \mathscr{B}^2) \cap \mathscr{B}^3.$$

It can be shown that the interconnected system $\mathscr{B}_{\text{int}}$ is fully specified by its finite horizon behavior with horizon $T = \boldsymbol{\ell}(\mathscr{B}^1) + \boldsymbol{\ell}(\mathscr{B}^2)$. Thus, for the computation of $\mathscr{B}_{\text{int}}$, we restrict to $[1, T]$. In the example, the systems are single output, so that the lags of the systems are equal to their orders.

```
T = n1 + n2;
```

The functions `B2BT` and `R2BT` construct orthonormal basis for the finite horizon behavior $\mathscr{B}|_T$ of a linear time-invariant system $\mathscr{B}$, specified by a state-space and a kernel representation, respectively:

```
BT1 = B2BT(B1, T); BT2 = B2BT(B2, T); BT3 = R2BT(R3, 4, T);
```

The function `BTappend` constructs a basis for the finite-horizon behavior of $\text{append}(\mathscr{B}^1, \mathscr{B}^2)$ and `BTintersect` computes the intersection of of two subspaces:

```
BT12    = BTappend(BT1, 2, BT2, 2);
BT12int = BTintersect(BT12, BT3);
```

Next, we verify that the resulting interconnected system $\mathscr{B}_{\text{int}}$ corresponds to the series connection of $\mathscr{B}^1$ and $\mathscr{B}^2$:

```
B12 = B2 * B1;
```

For this purpose, we project the behavior of $\mathscr{B}_{\text{int}}$ on the $\begin{bmatrix} w_1^1 \\ w_2^2 \end{bmatrix}$ variables (thus eliminating the variables w_2^1 and w_1^2):

```
BT12_ = BTproject(BT12int, q, [1 4]);
```

and, using the `equal` function, verify that the systems defined by `B12` and `BT12_` are equal:

```
equal(B12, BT12_) % -> 1
```

We found the behavior of the series connection of $\mathscr{B}_1$ and $\mathscr{B}_2$ directly from the behaviors of $\mathscr{B}_1$ and $\mathscr{B}_2$ without computing parametric representations of $\mathscr{B}_1$ and $\mathscr{B}_2$ and using their parameters.

4. Signal from noise separation

The signal from noise separation problem considered in this section is defined as follows.

> Given a system $\mathscr{B}$ and a "noisy" signal $w = \overline{w} + \widetilde{w}$, where $\overline{w} \in \mathscr{B}|_T$ is the to-be-estimated "true" signal and $\widetilde{w}$ is a zero-mean white Gaussian noise, find the maximum-likelihood estimate $\widehat{w}$ of $\overline{w}$.

The setup where the data w is a true value $\overline{w}$ plus noise $\widetilde{w}$ is called *errors-in-variables* Söderström (2018). Like the behavioral approach, it treats all variables on an equal footing without separating them into inputs and outputs. By definition, the errors-in-variables setup is statistical and leads to stochastic estimation problems. Thus, it complements the behavioral approach, which is by definition deterministic. (Currently there is no fully developed extension of the behavioral approach to stochastic systems, see Willems (2013) for behavioral approach of static stochastic systems.)

In order to simulate data in the errors-in-variables setup, we use the function `B2w` which returns a random trajectory $w \in \mathscr{B}|_T$.

```
m = 1; p = 1; n = 3; q = m + p; B = drss(n, p, m);
T = 10; w0 = B2w(B, T); wn = randn(size(w0));
w = w0 + 0.1 * norm(w0) * wn / norm(wn);
```

The maximum-likelihood estimate $\widehat{w}$ of $\overline{w}$ in the errors-in-variables setup is the solution of the optimization problem

$$\text{minimize} \quad \text{over } \widehat{w} \quad \|w - \widehat{w}\| \quad \text{subject to} \quad \widehat{w} \in \mathscr{B}|_T, \tag{KS}$$

i.e., the statistically optimal estimate $\widehat{w}$ has a geometric interpretation as the projection of the noisy signal w on the finite-horizon behavior $\mathscr{B}|_T$ of the data-generating system $\mathscr{B}$.

Problem (KS) does not involve a particular representation of the system $\mathscr{B}$. It states what we are after without suggesting a method for solving the problem. The solution can be obtained using a state-space representation, which leads to a Riccati equation Markovsky and De Moor (2005). An alternative solution using the nonparameteric representation of the restricted behavior is:

```
BT = B2BT(B, T); wh = BT * BT' * vec(w'); wh = reshape(wh, q, T)';
```

Since the basis `BT` of $\mathscr{B}_T$ is orthonormal, `BT * BT'` is the orthogonal projector on $\mathscr{B}_T$ and `wh` is the solution of (KS).

The solution using the Riccati equation is a computationally efficient method for solving (KS), *i.e.*, for finding the projection $\widehat{w}$ of w on $\mathscr{B}|_T$. It produces the same result as `wh = BT * BT' * vec(w')`, however, it is more difficult to derive and implement. Also, as shown in Section 6, the solution based on the nonparametric representation of the finite-horizon behavior, is applicable when the data-generating system $\mathscr{B}$ is unknown but data $w_{\mathrm{d}} \in \mathscr{B}|_{T_{\mathrm{d}}}$ is available. In this case, the solution presented above becomes a direct data-driven method for errors-in-variables smoothing. The transition from model-based to data-driven becomes trivial using the behavioral approach and the nonparametric representation (B_T) of the finite-horizon behavior.

The solution of the smoothing problem presented in this section is easily generalizable to situation where some variables are exact (noise free), other variables are inexact (corrupted by noise), and a third set of variables are missing (not specified). This more general setup is considered in Markovsky and Dörfler (2022), where the problem is interpolation and approximation of trajectories of dynamical systems. The next section presents a special case of missing data estimation.

5. Missing input estimation

Consider a linear time-invariant system $\mathscr{B}$, which variables w are separated as follows: $w = \begin{bmatrix} u_{\text{missing}} \\ u_{\text{given}} \\ y \end{bmatrix}$. As the notation suggests, u_{missing} is an unknown / to-be-estimated input, u_{given} is an known / observed input, and y is known / observed outputs.

```
m_missing = 1; m_given = 1; m = m_missing + m_given;
p = 2; n = 3; q = m + p; B = drss(n, p, m); T = 10; w = B2w(B, T);
w_missing = w(:, 1:m_missing); w_given = w(:, m_missing+1:end);
```

The missing input estimation problem is defined as follows.

> Given the data u_{given}, y, and the system $\mathscr{B}$, find the missing input u_{missing} of $\mathscr{B}$.

It is a special case of the observer problem in the behavioral setting Bisiacco et al. (2006). Using the nonparameteric representation of the restricted behavior (B_T), the solution is given as an explicit map from the given data $\begin{bmatrix} u_{\text{given}} \\ y \end{bmatrix}$ to the estimate $\widehat{u}_{\text{missing}}$ of the missing input u_{missing}:

$$\widehat{u}_{\text{missing}} = B_T|_{I_{\text{missing}}} \left(B_T|_{I_{\text{given}}} \right)^+ \begin{bmatrix} u_{\text{given}} \\ y \end{bmatrix}.$$

Here $B_T|_{I_{\text{missing}}}$ and $B_T|_{I_{\text{given}}}$ are the submatrices of B_T corresponding to the missing and the given data, respectively, and $(\cdot)^+$ denotes the pseudo-inverse. In Matlab code, the solution is:

```
BT = B2BT(B, T);
[BT_missing, BT_given] = BT2UYT(BT, m_missing, m_given + p);
wh_missing = BT_missing * BT_given' * vec(w_given');
wh_missing = reshape(wh_missing, m_missing, T)';
```

As before, the first step of the solution is the construction of the nonparameteric representation (B_T) of $\mathscr{B}$. The function `BT2UYT` selects the submatrices $B_T|_{I_{\text{missing}}}$ and $B_T|_{I_{\text{given}}}$ of B_T. In `BT_missing * BT_given' * vec(w_given')`, we exploited the fact that B_T is orthonormal, replacing the pseudo-inverse $\left(B_T|_{I_{\text{given}}} \right)^+$ with the transposed $\left(B_T|_{I_{\text{given}}} \right)^\top$.

Under the verifiable from the data condition

$$\operatorname{rank} B_T|_{I_{\text{given}}} = \boldsymbol{m}(\mathscr{B})T + \boldsymbol{n}(\mathscr{B}),$$

```
rank(BT_given) == T * m + n % -> 1
```

the missing input estimation problem has a unique solution, *i.e.*, $\widehat{u}_{\text{missing}} = u_{\text{missing}}$ and therefore u_{missing} is recovered exactly. Indeed,

```
e = norm(w_missing - w_missing) / norm(w_missing) % -> 0
```

In summary, applying standard linear algebra operations (pseudo-inverse and matrix multiplication) on the basis B_T of the finite-horizon behavior $\mathscr{B}|_T$ and the data $\begin{bmatrix} u_{\text{given}} \\ y \end{bmatrix}$, we recovered u_{missing}. Moreover, under verifiable from the data assumptions, the recovery is possible and is exact.

6. Direct data-driven forecasting

The behavioral toolbox is based on the nonparametric representation (B_T) of the finite-horizon behavior $\mathscr{B}|_T$. Such a representation can be obtained from another representation or from data, *i.e.*, a trajectory $w_{\mathrm{d}} \in (\mathbb{R}^q)^{T_{\mathrm{d}}}$ of the system $\mathscr{B}$. Leveraging the linearity and time-invariance properties of the system, for $w_{\mathrm{d}} \in \mathscr{B}|_{T_{\mathrm{d}}}$ with $T_{\mathrm{d}} > T$, the image of the Hankel matrix $\mathscr{H}_T(w_{\mathrm{d}})$ constructed from w_{d} is included in the finite-horizon behavior $\mathscr{B}|_T$. Moreover, assuming that $T \geq \boldsymbol{\ell}(\mathscr{B})$ and

$$\operatorname{rank} \mathscr{H}_T(w_{\mathrm{d}}) = \boldsymbol{m}(\mathscr{B})T + \boldsymbol{n}(\mathscr{B}), \tag{GPE}$$

we have that (Markovsky and Dörfler, 2023, Corollary 21)

$$\operatorname{image} \mathscr{H}_T(w_{\mathrm{d}}) = \mathscr{B}|_T. \tag{DD-REPR}$$

The raw data w_{d} in the form of the Hankel matrix $\mathscr{H}_T(w)$ serves as a nonparameteric representation of $\mathscr{B}|_T$. The key assumption (GPE) is called *generalized persistency of excitation*. It is verifiable from the data and the prior knowledge of the system's complexity: number of inputs $\boldsymbol{m}(\mathscr{B})$, lag $\boldsymbol{\ell}(\mathscr{B})$, and order $\boldsymbol{n}(\mathscr{B})$. (Markovsky et al. (2023b) compare and contrast (Markovsky and Dörfler, 2023, Corollary 21) with the fundamental lemma of Willems et al. (2005).)

In case of noisy data obtained in the errors-in-variables setup, the maximum-likelihood solution leads to a *Hankel structured low-rank approximation* problem Markovsky (2019), which is a nonconvex optimization problem. Currently existing methods are heuristics for solving this nonconvex problem. Next, we show the performance of the methods in the toolbox on real-data from data-base for system identification DAISY, De Moor et al. (1997). The problem considered is /forecasting/, *i.e.*, simulation of a "future" trajectory of unknown system Markovsky and Dörfler (2022). The data set is the "Robot arm" benchmark. For comparison, as a reference method we use the classical model-based approach, where a model is first identified from the data and then the model is used for finding the predicted signal.

```
robot_arm_; m = 1; p = 1; n = 8;
wd = [u y]; [Td, q] = size(wd);
```

The data is split into an identification part followed by a validation part. The last $T_{\mathrm{ini}} = 20$ samples of the identification part are also used for estimation of the initial conditions for the validation part.

```
Tv = 50; Ti = Td - Tv;
wi = wd(1:Ti, :); wv = wd(Ti+1:end, :);
Tini = n; w_ini = wi(end - Tini + 1:end, :);
```

The prediction method based on the nonparameteric finite-horizon representation of the behavior is implemented in the function `u2y`. With T_{ini} chosen, `u2y` has no hyper-parameters.

```
tic, yh_dd = u2y(hank(wi, Tv + Tini), q, wv(:, 1:m), w_ini);
t_dd = toc % -> 0.0186
```

The corresponding model-based method is the function `forecast` from the System Identification Toolbox of Matlab, which requires a model. The model is obtained by a subspace identification method, implemented in the `n4sid` function, which has a hyper-parameter—the model order n. It is chosen as $n = 8$ by trial-and-error using the validation data.

```
tic, Bh = n4sid(iddata(wi(:, m+1:end), wi(:, 1:m)), n);
f = forecast(Bh), iddata(w_ini(:,m+1:end), w_ini(:,1:m)), Tv, wv(:,1:m));
yh_mb = f.OutputData; t_mb = toc % -> 1.4176
```

Note that although the direct data-driven approach is computationally less efficient than the model-based approach, in the example, it is faster.

The results of the two methods are compared in terms of the relative prediction error: $e := \|y_v - \widehat{y}_v\|/\|y_v\|$, where y_v is the to-be-forecast validation output and $\widehat{y}_v$ is the forecast.

```
e = @(yh) 100 * norm(wv(:, m+1:end) - yh) / norm(wv(:, m+1:end));
[e(yh_dd), e(yh_mb)] % -> 3.5531    3.8398
```

In the example, the results of the two methods are similar. The model-based method, however, required tuning of a hyper-parameter (for which we've selected optimal value using the validation data), while the direct data-driven method is hyper-parameters free.

7. Implementation details

Section 7.1 presents details about the implementation of the "data-driven" approach for constructing (B_T). Sections 7.2 and 7.2 show how input/output partitioning and analysis problems can be found directly from (B_T). Section 7.4 presents methods for computing parametric representations.

7.1. Nonparameteric representation of the restricted behavior

The restricted behavior $\mathscr{B}|_T$ of a linear time-invariant system is a shift-invariant subspace of $(\mathbb{R}^q)^T$. The methods developed in the toolbox use (B_T) as a representation of $\mathscr{B}$. With some abuse of notation, we refer to the matrix $B_T := \begin{bmatrix} b^1 & \cdots & b^r \end{bmatrix} \in \mathbb{R}^{qT\times r}$ of the basis vectors as the basis. The representation (B_T) of $\mathscr{B}|_T$ is nonparameteric because it involves $r := \dim \mathscr{B}|_T$ parameters $g_1, \ldots, g_r \in \mathbb{R}$ to specify $w \in \mathscr{B}|_T$ via $w = B_T g$. For $T \geq \boldsymbol{\ell}(\mathscr{B}) + 1$, $\mathscr{B}|_T$ defines $\mathscr{B}$ (Markovsky and Dörfler, 2023, Lemma 13), so (B_T) is a *nonparameteric representation* of $\mathscr{B}$.

The function `B2BT` implements a data-driven approach. Instead of using the parameters of the state-space representation of $\mathscr{B}$, it computes a random trajectory w_d of $\mathscr{B}$, forms the Hankel matrix $\mathscr{H}_T(w_\mathrm{d})$, and computes an orthonormal basis for $\mathscr{B}|_T$:

$$(A,B,C,D) \xmapsto{\texttt{lsim}} w_\mathrm{d} \xmapsto{\texttt{hank}} \mathscr{H}_T(w_\mathrm{d}) \xmapsto{\texttt{orth}} \mathscr{B}|_T$$

The computation of a basis B_T from a trajectory $w_\mathrm{d} \in \mathscr{B}|_{T_\mathrm{d}}$ requires finding the dimension of $\mathscr{B}|_T$. Under (GPE) (which is almost certainly satisfied for a sufficiently long random trajectory $w_\mathrm{d} \in \mathscr{B}|_T$), $\dim \mathscr{B}|_T = \operatorname{rank} \mathscr{H}_T(w_\mathrm{d})$. Thus, $\dim \mathscr{B}|_T$ can be found by rank computation. A robust way of computing $\operatorname{rank} \mathscr{H}_T(w_\mathrm{d})$ is thresholding the singular values of $\mathscr{H}_T(w_\mathrm{d})$. The functions of the toolbox that construct B_T from data therefore have an optional threshold parameter `tol`. Alternatively, the user can specify model's complexity, in which case the rank is computed from the relation (Markovsky and Dörfler, 2023, Corollary 5)

$$\operatorname{rank} B_T = \dim \mathscr{B}|_T = \boldsymbol{m}(\mathscr{B})T + \boldsymbol{n}(\mathscr{B}), \quad \text{for } T \geq \boldsymbol{\ell}(\mathscr{B}). \tag{$\dim \mathscr{B}|_T$}$$

7.2. Input/output partitionings

A partitioning of the variables $w(t) \in \mathbb{R}^q$ into inputs $u(t) \in \mathbb{R}^m$ and outputs $y(t) \in \mathbb{R}^p$ is defined by a permutation matrix $\Pi \in \mathbb{R}^{q\times q}$ as follows:

$$w \mapsto (u,y) : \begin{bmatrix} u \\ y \end{bmatrix} := \Pi w \qquad \text{and} \qquad (u,y) \mapsto w : \; w = \Pi^\top \begin{bmatrix} u \\ y \end{bmatrix}. \tag{I/O}$$

Πw reorders the variables w, so that the first m variables are the inputs and the remaining $p := q-m$ variables are the outputs. The restricted behavior with permuted variables is created with the function `BT2BT`. The function `BT2UYT` extracts from B_T its input and output components.

The number of inputs m is uniquely defined by the behavior. However, an input/output partitioning of the variables is in general not unique. The currently available methods in the literature for finding an input/output partitioning of a system $\mathscr{B}$ are based on parametric representations of the system. Next, we present a data-driven method for checking if an given partitioning is a valid input/output partitioning directly from the finite-horizon behavior $\mathscr{B}|_T$. The key observation is that (I/O) is an input/output partitioning of $\mathscr{B}$ if and only if it is possible to simulate any trajectory of $\mathscr{B}$ by choosing the input u and the initial conditions, specified by a past trajectory $w_{\rm ini}$, *i.e.*, for any $w_{\rm ini} \in (\mathbb{R}^q)^{T_{\rm ini}}$ with $T_{\rm ini} \geq \ell$ and $u_{\rm f} \in (\mathbb{R}^m)^{T_{\rm f}}$ with $T_{\rm f} \geq \ell$, there exist a unique $y_{\rm f} \in (\mathbb{R}^p)^{T_{\rm f}}$, such that

$$w_{\rm ini} \wedge \Pi^\top \begin{bmatrix} u_{\rm f} \\ y_{\rm f} \end{bmatrix} \in \mathscr{B}|_{T_{\rm ini}+T_{\rm f}}.$$

Let $B_{T_{\rm ini}+T_{\rm f}}$ be a basis for $\mathscr{B}|_{T_{\rm ini}+T_{\rm f}}$ and let $\begin{bmatrix} W_{\rm ini} \\ U_{\rm f} \end{bmatrix}$ be the the submatrix of $B_{T_{\rm ini}+T_{\rm f}}$, corresponding to the initial trajectory $w_{\rm ini}$ and input $u_{\rm f}$. Then, (I/O) is an input/output partitioning of $\mathscr{B}$ if and only if

$$\operatorname{rank} \begin{bmatrix} W_{\rm ini} \\ U_{\rm f} \end{bmatrix} = m(T_{\rm ini}+T_{\rm f}) + n.$$

The method is implemented in `is_io`, which checks if a given partitioning is a possible input/output partitioning, and `BT2IO`, which finds all possible input/output partitionings.

7.3. Analysis

Properties of the system, such as system's complexity, controllability, and H_∞-norm can be found directly from its restricted behavior rather than from parametric representations as done by classical analysis methods. The complexity $\boldsymbol{c}(\mathscr{B})$ of a linear time-invariant system $\mathscr{B}$ is defined as the triple: number of inputs $\boldsymbol{m}(\mathscr{B})$, lag $\boldsymbol{\ell}(\mathscr{B})$, and order $\boldsymbol{n}(\mathscr{B})$. Although in the classical setting the order $\boldsymbol{n}(\mathscr{B})$ is defined via a minimal state-space representation, it is a property of the system $\mathscr{B}$ and can be computed directly from the restricted behavior $\mathscr{B}|_T$ (provided $T \geq \boldsymbol{\ell}(\mathscr{B})+1$). Also, the number of inputs $\boldsymbol{m}(\mathscr{B})$ can be computed from $\mathscr{B}|_T$ without reference to a particular input/output representation. The method for finding $\boldsymbol{m}(\mathscr{B})$ and $\boldsymbol{n}(\mathscr{B})$ from $\mathscr{B}|_T$ is based on (dim $\mathscr{B}|_T$). Evaluating dim $\mathscr{B}|_{t_i}$ for $t_1 \neq t_2 \geq \boldsymbol{\ell}(\mathscr{B})$, *e.g.*, $t_1 = T$ and $t_2 = T-1$, we obtain the system of equations

$$\begin{bmatrix} T & 1 \\ T-1 & 1 \end{bmatrix} \begin{bmatrix} m \\ n \end{bmatrix} = \begin{bmatrix} \dim \mathscr{B}|_T \\ \dim \mathscr{B}|_{T-1} \end{bmatrix},$$

from which $m = \boldsymbol{m}(\mathscr{B})$ and $n = \boldsymbol{n}(\mathscr{B})$ can be found. The resulting method is implemented in the function `BT2c`. The system's complexity connects nonparameteric and parametric representations and is a critical first step in finding a parametric representation of the system.

The following subbehaviors of $\mathscr{B}$ are used by methods in the toolbox: $\mathscr{Y}_0$ — *zero-input subbehavior*, *i.e.*, the set of transient responses, $\mathscr{U}_0$ — *zero-output subbehavior*, *i.e.*, the set of inputs blocked by the system, $\mathscr{B}_0$ — *zero initial conditions subbehavior*, *i.e.*, the set of zero initial conditions trajectories, $\mathscr{B}_{\rm c}$ — *controllable subbehavior*, *i.e.*, the set of trajectories that are patchable with zero past trajectory, and $\mathscr{B}_{\rm p}$ — *periodic subbehavior*, *i.e.*, the set of periodic trajectories. Finite-horizon representations of these subbehaviors are computed from (B_T) by the following functions: `BT2Y0`, `BT2U0`, `BT2B0`, `BT2BC`, and `BT2BP`. The subbehaviors $\mathscr{Y}_0$, $\mathscr{U}_0$, $\mathscr{B}_0$, $\mathscr{B}_{\rm c}$, and $\mathscr{B}_{\rm p}$ have also independent interest and some of them are used in identification, signal processing, and control methods. For example, finding $\mathscr{Y}_0$ is a key step in the MOESP-type subspace identification methods Verhaegen and Dewilde (1992) and $\mathscr{B}_{\rm p}$ allows frequency-domain analysis of the system.

Using the zero initial conditions subbehavior $\mathscr{B}_0 \subset \mathscr{B}|_T$, we can find the input-output map $H_T : u|_T \mapsto y|_T$. Let B_0 be the matrix of the basis vectors for $\mathscr{B}_0$ and let U_0, Y_0 be the submatrices of B_0 corresponding to the inputs and the outputs. Then, $H_T = Y_0 U_0^{-1}$. The corresponding function is `BT2HT`. The $pT \times mT$ matrix H_T is the finite-horizon representation of the transfer function $H(z)$ of the system $\mathscr{B}$ corresponding to the input/output partitioning (I/O). Using H_T, we can find also the finite-horizon H_∞-norm of the system. The method is implemented in the function `BT2Hinf`.

The controllable subbehavior $\mathscr{B}_c \subset \mathscr{B}|_T$ for $T \geq \ell(\mathscr{B})$ is $mT+n$-dimensional if and only if $\mathscr{B}$ is controllable. This leads to a data-driven controllability test. Moreover, $mT+n-\dim \mathscr{B}_c|_T$ is the number of uncontrollable modes of $\mathscr{B}$. Based on $\mathscr{B}_c$, a quantitative test for controllability—a distance to uncontrollability—is implemented.

7.4. Parametric representations

Computing a kernel representation from $\mathscr{B}|_T$ is essentially applying the `null` function on $B_{\ell(\mathscr{B})+1}^\top$. The method is implemented in the functions `B2R` and `BT2R`. The inverse operation—finding the restricted behavior from a kernel or a state-space representation—is done by the functions `R2BT` and `ss2BT`, which implement the model-based approach, *i.e.*, they construct B_T from the model parameters R and (A,B,C,D), respectively. The basis computed by `ss2BT` is not orthonormal. It consists of observability and convolution matrices.

Contrary to `BT2R` (which is essentially Matlab's `null` function), computing a state-space representation from the restricted behavior is nontrivial. Indeed, it requires to do 1) state construction and 2) detect a possible input/output partitioning of the variables. The data-driven approach for these operations is implemented in the function `BT2ss`. Similarly, the transformation from a kernel representation to a state-space representation is nontrivial. A data-driven method for this operation is implemented in the function `R2ss`.

For multi-output systems, the function `B2R` computes a *nonminimal* kernel representation. Computing a minimal kernel representation or converting a nonminimal kernel representation to a minimal one are also nontrivial operations. Data-driven methods for them are implemented in the functions `BT2Rmin` and `R2Rmin`. The dichotomy of parametric vs nonparametric representations is misleading—there is a range of nonminimal parametric representations that cover the gap between minimal parametric and nonparametric representations. The problem of detecting when a parametric representation is minimal is equivalent to the one of finding the model's complexity. It is also essential in the case of identification from noisy data where the key issue is the one of achieving an accuracy–complexity trade-off.

Acknowledgments

The research leading to these results received funding from the Catalan Institution for Research and Advanced Studies (ICREA) and Fond for Scientific Research Vlaanderen (FWO) project G033822N.

References

M. Bisiacco, M.-E. Valcher, and J. C. Willems. A behavioral approach to estimation and dead-beat observer design with applications to state-space models. *IEEE Trans. Automat. Contr.*, 51(11): 1787–1797, 2006.

B. De Moor, P. De Gersem, B. De Schutter, and W. Favoreel. DAISY: A database for identification of systems. *Journal A*, 38(3):4–5, 1997. Available from http://homes.esat.kuleuven.be/~smc/daisy/.

G. Golub and C. Van Loan. *Matrix Computations*. Johns Hopkins University Press, third edition, 1996.

I. Markovsky. A software package for system identification in the behavioral setting. *Control Eng. Practice*, 21:1422–1436, 2013. doi: 10.1016/j.conengprac.2013.06.010.

I. Markovsky. A missing data approach to data-driven filtering and control. *IEEE Trans. Automat. Contr.*, 62:1972–1978, 2017. ISSN 1558–2523. doi: 10.1109/TAC.2016.2591178.

I. Markovsky. *Low-Rank Approximation: Algorithms, Implementation, Applications*. Springer, 2019. doi: 10.1007/978-3-319-89620-5.

I. Markovsky and B. De Moor. Linear dynamic filtering with noisy input and output. *Automatica*, 41(1):167–171, 2005.

I. Markovsky and F. Dörfler. Behavioral systems theory in data-driven analysis, signal processing, and control. *Annual Reviews in Control*, 52:42–64, 2021. doi: 10.1016/j.arcontrol.2021.09.005.

I. Markovsky and F. Dörfler. Data-driven dynamic interpolation and approximation. *Automatica*, 135:110008, 2022. doi: 10.1016/j.automatica.2021.110008.

I. Markovsky and F. Dörfler. Identifiability in the behavioral setting. *IEEE Trans. Automat. Contr.*, 68:1667–1677, 2023. doi: 10.1109/TAC.2022.3209954.

I. Markovsky and P. Rapisarda. Data-driven simulation and control. *Int. J. Control*, 81(12):1946–1959, 2008.

I. Markovsky, L. Huang, and F. Dörfler. Data-driven control based on behavioral approach: From theory to applications in power systems. *IEEE Control Systems Magazine*, 43:28–68, 2023a. doi: 10.1109/MCS.2023.3291638.

I. Markovsky, E. Prieto-Araujo, and F. Dörfler. On the persistency of excitation. *Automatica*, page 110657, 2023b. doi: 10.1016/j.automatica.2022.110657.

T. Söderström. *Errors-in-Variables Methods in System Identification*. Springer, 2018.

M. Verhaegen and P. Dewilde. Subspace model identification, Part 1: The output-error state-space model identification class of algorithms. *Int. J. Contr.*, 56:1187–1210, 1992.

J. C. Willems. From time series to linear system—Part I. Finite dimensional linear time invariant systems, Part II. Exact modelling, Part III. Approximate modelling. *Automatica*, 22, 23:561–580, 675–694, 87–115, 1986, 1987.

J. C. Willems. The behavioral approach to open and interconnected systems: Modeling by tearing, zooming, and linking. *IEEE Control Systems Magazine*, 27:46–99, 2007.

J. C. Willems, P. Rapisarda, I. Markovsky, and B. De Moor. A note on persistency of excitation. *Systems & Control Lett.*, 54(4):325–329, 2005. doi: 10.1016/j.sysconle.2004.09.003.

Jan C. Willems. Open stochastic systems. *IEEE Trans. Automat. Contr.*, 58(2):406–421, 2013. doi: 10.1109/TAC.2012.2210836.