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 paper presents five problems—checking systems equality, interconnection of systems, errors-in-variables least-squares smoothing, missing input estimation, and data-driven forecasting—and their solution by the methods in the toolbox as well as implementation details for some of the methods that led to new results of broader interest.

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 based on the fundamental lemma Willems et al. (2005). 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. (2023). The notation used in this document is also the same as the one in the overview papers. A summary is given in Table 1.

$w \in (\mathbb{R}^q)^{\mathbb{N}}$, $w : \mathbb{N} \to \mathbb{R}^q$	q-variate real discrete-time signal with time axis $\mathbb{N}$
$w\vert_T := \big(w(1), \dots, w(T)\big)$	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 examples. The examples admit solutions in the classical as well as in the behavioral settings. 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 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. We show that series connection in the input/output setting is a special case of 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* its external behavior.

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 Toolbox of Matlab:

```
sys1 == sys2 % -> not implemented
```

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 for linear time-invariant systems is due to the perception that linear time-invariant systems (`ss` objects in particular) are not comparable. In fact, linear time-invariant systems are comparable but the parameters of their state-space representations are not. The shift in perception brought by the behavioral approach is that *parameters of a representation are not the system.* 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. 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, high-level problems related to systems (*e.g.*, check if $\mathscr{B}^1 = \mathscr{B}^2$) are stated without involving systems' representations. The high-level 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 transfer functions, it may be beneficial to switch to another one, say state-space. The one solving the problem should not be obliged to use a particular representation. Lower-level problem formulation, however, may ask specifically about solving a high-level problem using a particular representation or method.

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 for it. Provided that T is long enough (often this 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 $\mathscr{B}^1|_T = \mathscr{B}^2|_T$—a basic linear algebra problem, for which there are existing methods. As usual in science and engineering, a new problem is solved by reducing it to an already solved problem. An extra benefit of reducing the problem to basic linear algebra operations is existence of high-quality software that is easily accessible.

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 for the finite-horizon behaviors $\mathscr{B}^1|_T$, $\mathscr{B}^2|_T$. 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. Then, $\mathscr{B}^1|_T = \mathscr{B}^2|_T$ is checked numerically by 1) verifying that the distance (defined as the principal angle) between $\mathscr{B}^1|_T$, $\mathscr{B}^2|_T$ is smaller than a tolerance 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

$$w^1 = \begin{bmatrix} w_1^1 \\ w_2^1 \end{bmatrix} = \begin{bmatrix} u^1 \\ y^1 \end{bmatrix} \quad \text{and} \quad w^2 = \begin{bmatrix} w_1^2 \\ w_2^2 \end{bmatrix} = \begin{bmatrix} u^2 \\ y^2 \end{bmatrix}.$$

```
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 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 dynamics of the interconnected system $\mathscr{B}_{\text{int}}$ is fully specified by its finite horizon behavior with horizon $T = \boldsymbol{\ell}(\mathscr{B}^1) + \boldsymbol{\ell}(\mathscr{B}^2)$. Since in the example the systems are single output the lags are equal to the orders of the systems.

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

Thus, for the computation of $\mathscr{B}_{\text{int}}$, we restrict to a finite horizon $[1, \boldsymbol{\ell}(\mathscr{B}^1) + \boldsymbol{\ell}(\mathscr{B}^2)]$. The functions `B2BT` and `R2BT` of the toolbox construct the finite horizon behavior $\mathscr{B}|_T$ of an 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 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 `BT_` are equal:

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

We've found the series connection of $\mathscr{B}_1$ and $\mathscr{B}_2$ using directly bases for their finite-horizon behaviors without computing and using model parameters.

4. Signal from noise separation

The problem considered next is: 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 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). 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 simple 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 $\mathscr{B}_T$, so `wh = BT * BT′ * vec(w′)` is the signal $\widehat{w}$ solving (KS).

The model-based solution using Riccati equations is computationally more efficiently, however, it is harder to derive and implement. Indeed, `wh = BT * BT′ * vec(w′)` is hard to beat in length and simplicity of implementation.

5. Missing input estimation

Consider an 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 problem considered is to find u_{missing}, given u_{given}, y, and $\mathscr{B}$, *i.e.*, estimate a missing input u_{missing} of the system. The problem is a special case of the observer problem in the behavioral setting Bisiacco et al. (2006). Using the nonparameteric representation of the restricted behavior, the solution of the missing input estimation problem leads to the solution:

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

where $B_T|_{I_{\text{missing}}}$ 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)';
```

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 problem has a unique solution, so that the missing input is recovered exactly. Indeed,

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

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 given data $\left[\begin{smallmatrix} u_{\text{given}} \\ y \end{smallmatrix}\right]$, we have found the missing input u_{missing}.

6. Direct data-driven forecasting

The behavioral toolbox is based on the nonparametric representation of the finite-horizon behavior $\mathscr{B}|_T$. Such a representation can be obtained from another representation or from data, *i.e.*, a trajectory $w_{\text{d}} \in (\mathbb{R}^q)^{T_{\text{d}}}$ of the system $\mathscr{B}$. Leveraging the linearity and time-invariance properties of the system and assuming sufficiently long ($T_{\text{d}} > T$) trajectory w_{d} of the system, the image of the Hankel matrix $\mathscr{H}_T(w_{\text{d}})$ constructed from the data is included in the finite-horizon behavior $\mathscr{B}|_T$. Moreover, assuming that $\operatorname{rank} \mathscr{H}_T(w_{\text{d}})$ is sufficiently high, more specifically (Markovsky and Dörfler, 2023, Corollary 21)

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

we have that

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

i.e., 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 point that a representation of the finite-horizon behavior $\mathscr{B}|_T$ can be constructed directly from data using the Hankel matrix makes the methods in the Behavioral Toolbox applicable in situations where a model is not available but data from the system can be collected. In such situations, classical methods require a parametric system identification step prior to applying the methods. The methods in the toolbox, in contrast, implement the direct map from the given data to the desired solution. This direct map is theoretically interesting. As shown next it is also practically useful.

In practice the data is almost never exact. In case of noisy data, satisfying certain stochastic assumptions (true value plus zero mean white Gaussian noise), the maximum-likelihood solution, which is theoretically optimal, leads to a *Hankel structured low-rank approximation* problem Markovsky (2019). The Hankel structured low-rank approximation problem is a nonconvex optimization problem. All currently existing methods are heuristics for solving this nonconvex problem.

Next, we show the performance of the methods in the toolbox. For comparison, we apply the classical approach of model-identification (using state-of-the-art parametric system identification methods) followed by a model-based solution. The problem considered is prediction, *i.e.*, simulation of a "future" trajectory of unknown system. The data is the "Robot arm" benchmark from the database for system identification DAISY De Moor et al. (1997).

```
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 $T_{\text{ini}} = 20$ 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`. Note that once the past horizon T_{ini} is chosen, it has no tunable 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 takes as an input a model. The model is obtained by a subspace identification method, implemented in the `n4sid` function from the System Identification Toolbox. The model-based approach requires a hyper-parameter—the model order `n`. It is chosen as 8 by trail-and-error, aiming for optimal performance. In this hyper-parameter tuning the validation data is used which gives unfair advantage of the model-based method.

```
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, it is faster in the example.

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 output of the validation part of the data and $\widehat{y}_v$ is the forecast obtained by the method.

```
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 has no hyper-parameters.

7. Implementation details

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 a basis $\{b^1,\dots,b^r\}$ for $\mathscr{B}|_T$ as a representation of $\mathscr{B}$:

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

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 $g_1,\dots,g_r \in \mathbb{R}$ 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 $\mathscr{H}_T(w_\text{d})$, and computes an orthonormal basis for $\mathscr{B}|_T$:

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

The computation of a basis B_T from a trajectory $w_\text{d} \in \mathscr{B}|_{T_\text{d}}$ requires finding the dimension of $\mathscr{B}|_T$. Under (GPE) (which is almost certainly satisfied for a sufficiently long random trajectory $w_\text{d} \in \mathscr{B}|_T$), $\dim \mathscr{B}|_T = \text{rank}\, \mathscr{H}_T(w_\text{d})$. Thus, $\dim \mathscr{B}|_T$ can be found by rank computation. A robust way of computing $\text{rank}\, \mathscr{H}_T(w_\text{d})$ is thresholding the singular values of $\mathscr{H}_T(w_\text{d})$. 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 dimension is computed from (Markovsky and Dörfler, 2023, Corollary 5):

$$\dim \mathscr{B}|_T = mT + n, \quad \text{for } T \geq \ell. \tag{$\dim \mathscr{B}|_T$}$$

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 permutation matrix Π is specified in the functions of the toolbox by a vector `io`, such that `w(io)` $\mapsto$ `[u; y]`. The restricted behavior with permuted variables is created with the function `BT2BT`:

```
io = flip(1:q); BTp = BT2BT(BT, q, io);
```

The function `BT2UYT` extracts from the basis of the restricted behavior $\mathscr{B}|_T$ 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 describe a data-driven method for finding an 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_{ini}. Therefore, for any $w_{\text{ini}} \in (\mathbb{R}^q)^{T_{\text{ini}}}$ with $T_{\text{ini}} \geq \ell$ and $u_{\text{f}} \in (\mathbb{R}^m)^{T_{\text{f}}}$ with $T_{\text{f}} \geq \ell$, there exist a unique $y_{\text{f}} \in (\mathbb{R}^p)^{T_{\text{f}}}$, such that

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

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

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

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

7.3. Analysis

We show how properties of the system can be found directly from its restricted behavior (rather than from parametric representations as done by classical analysis methods). The properties considered are: system's complexity, controllability, H_∞-norm, and distance between systems. The following subbehaviors of $\mathscr{B}$ are of special interest: $\mathscr{Y}_0$ — *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, $\mathscr{B}_{\text{c}}$ — *controllable subbehavior*, *i.e.*, the set of trajectories that are patchable with zero past trajectory, and $\mathscr{B}_{\text{p}}$ — *periodic subbehavior*, *i.e.*, the set of periodic trajectories.

Using the zero initial conditions subbehavior $\mathscr{B}_0 \subset \mathscr{B}|_T$, we can find the zero initial conditions input-to-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 method is implemented in the function `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).

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 controllable subbehavior $\mathscr{B}_c \subset \mathscr{B}|_T$ for $T \geq \boldsymbol{\ell}(\mathscr{B})$ is $mT+n$-dimensional if and only if $\mathscr{B}$ is controllable. This leads to a data-driven controllability test that is implemented in the function `isunctr`. 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 in the function `distunctr`. Using the zero initial conditions input-to-output map H_T of $\mathscr{B}$ for a given input/output partitioning allows us to compute the finite-horizon H_∞-norm of $\mathscr{B}$. The method is implemented in the function `BT2Hinf`. The function `Bdist` computes distance between systems, defined as the principal angle between the finite-horizon behaviors of the systems.

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.

I. Markovsky. *Low-Rank Approximation: Algorithms, Implementation, Applications*. Springer, 2019.

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.

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

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, 2023.

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

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. *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.