# Simulations in the paper **Addition and intersection of linear time-invariant behaviors**

Antonio Fazzi[a], Ivan Markovsky[b]

[a] *University of Padova*
*Department of Information Engineering*
*Via Gradenigo 6/b, 35131 Padova, Italy*

[b] *International Centre for Numerical Methods in Engineering (CIMNE)*
*and Catalan Institution for Research and Advanced Studies (ICREA)*
*Gran Capitn, 08034 Barcelona, Spain*

---

In this short document we write down some numerical simulations related to the examples in [1]. The literate programming style [2, 3] is used in the following. This makes the experiment reproducible (the reader can copy and paste the code in the Matlab command line, but the numerical values will be different due to the randomness of the data).

## 1. Scalar autonomous systems with simple poles

We copy below [1, Lemma 6], whose numerical simulation follows.

**Lemma 1.** *Let $\mathcal{A}$ and $\mathcal{B}$ be two scalar autonomous linear time-invariant behaviors with minimal kernel representations $R_a(\sigma), R_b(\sigma)$, defined by the scalar polynomials $R_a(z), R_b(z)$. The polynomials for the minimal kernel representations $R_+(z)$ of $\mathcal{A}+\mathcal{B}$ and $R_\cap(z)$ of $\mathcal{A}\cap\mathcal{B}$ are given by, respectively, the least common multiple and the greatest common divisor of $R_a(z)$ and $R_b(z)$.*

The following simulation example reproduces the results of Lemma 1. In the following code chunks we use the function `blkhank(p, l, T)` to build block-Hankel matrices with `l` rows and `T` columns from a time series `p`. It can be downloaded from the *slra* toolbox [4].

The first step is to generate two scalar autonomous systems with a given set of poles and the corresponding trajectories. For convenience, the starting systems are generated in state-space form.

```
sys1 = zpk([],[-1.1, 0.1, 1], 1, 1);
sys1 = ss(sys1);                        %first system
sys2 = zpk([],[-0.5, -0.2, 1], 1, 1);
sys2 = ss(sys2);                        %second system
```

*Email addresses:* antonio.fazzi@unipd.it (Antonio Fazzi), imarkovsky@cimne.upc.edu (Ivan Markovsky)

```
y1 = initial(sys1, randn(3, 1), 20);  %trajectory first system
y2 = initial(sys2, randn(3, 1), 20);  %trajectory second system
```

A trajectory of the sum is given simply by adding the two trajectories of the original systems. We compute, then, the kernel representations of `sys1, sys2` and `sys1+sys2` by computing the left null spaces of the Hankel matrices $H_4$(`y1`), $H_4$(`y_2`) and $H_6$(`y1+y2`), respectively, using the function `blkhank`. The lag $\ell$ for each system is its number of poles (or possibly it can be computed as $\ell = n/p$, where $n, p$ are the outputs of Algorithm **??**.

```
R1 = null(blkhank(y1, 4, length(y1) - 3)')';
R1 = fliplr(R1)
-0.6675   0.0000   0.7410   -0.0734

R2 = null(blkhank(y2, 4, length(y2) - 3)')';
R2 = fliplr(R2)
0.8276   -0.2483   -0.4966   -0.0828

R = null(blkhank(y1 + y2, 6, length(y1+y2) - 5)')'
R = fliplr(R)
-0.5816   -0.4071   0.5874   0.3879   0.0198   -0.0064
```

We can finally check that the roots of the three polynomials are exactly the expected poles. Observe that the presence of the common pole in the starting systems was reflected in the polynomials degrees. The roots of the computed polynomials also reveal the non-trivial common factor between `R1` and `R2`.

## 2. A single-input single-output system and an autonomous system

In [1, Section 5.2] we showed that given a single-input single-output system

$$\mathcal{A} = \ker \begin{bmatrix} q_a(\sigma) & p_a(\sigma) \end{bmatrix}.$$

and an autonomous systems

$$\mathcal{B} = \ker \begin{bmatrix} 1 & 0 \\ 0 & p_b(\sigma) \end{bmatrix}.$$

, a kernel representation of the sum $\mathcal{A} + \mathcal{B}$ is given by

$$R_+(z) = p_b(z) \begin{bmatrix} q_a(z) & -p_a(z) \end{bmatrix}. \tag{1}$$

We check it numerically.

The first system is a Single-Input Single Output controllable system. The second system is autonomous (that is, uncontrollable). Observe that now we have two system variables, so to match the sizes of the system trajectories we have to add a zero input to the trajectory of the autonomous system.

```
sys1 = drss(1, 1, 1);  %first system
sys2 = drss(1, 1, 0);  %second system

u1 = randn(20, 1);     %random input
y1 = lsim(sys1, u1);   %output first system
w1 = [u1 y1];          %trajectory first system
y2 = initial(sys2, randn(1, 1), 19);  %output second system
w2 = [zeros(20, 1) y2];   %trajectory second system
```

Then, we can compute the kernel representations starting from the system trajectories. We use the function `blkhank` to build the Hankel matrices $H_2$(`w1`), $H_2$(`w2`) and $H_3$(`w1+w2`). About the autonomous system, we want a square matrix polynomial (as in (**??**)) but we will find more annihilators because of the presence of the zero rows corresponding to the input. Hence only a subset of the computed annihilators is selected.

```
R1 = null(blkhank(w1, 2, 19)')';
p1 = [R1(4), R1(2)];
q1 = [R1(3), R1(1)];

R2 = null(blkhank(w2, 2, 19)')';
R2 = R2([1, 3],:)   %check the  subset of rows
p2 = [R2(1, 4), R2(1, 2)];

R = null(blkhank(w1 + w2, 3, 18)')';
p = [R(6), R(4), R(2)];
q = [R(5), R(3), R(1)];
```

It holds true that `p = conv(p2, p1)` and `q = conv(p2, q1)` (up to a normalization), so we checked the expression (1) numerically.

## References


[1] A. Fazzi, I. Markovsky, Addition and intersection of linear time-invariant behaviors, `https://arxiv.org/abs/2107.01588`.

[2] D. Knuth, Literate programming, Comput. J. 27 (2) (1984) 97–111.

[3] N. Ramsey, Literate programming simplified, IEEE Software 11 (1994) 97–105.

[4] I. Markovsky, K. Usevich, Software for weighted structured low-rank approximation, J. Comput. Appl. Math. 256 (2014) 278–292.