Validation of an algorithm for kernel representation of the sum of two behaviors

Antonio Fazzi, Ivan Markovsky

The goal of the next experiment is to validate the following algorithm for the computation of a kernel representation of the sum of two linear time-invariant behaviors. If R_a, R_b are the kernel representations of the original systems, then

1. Compute the number of outputs of the sum $\mathbf{p}(\mathscr{B}_+) = q - \mathbf{m}(\mathscr{B}_+)$, where q is the number of variables and $\mathbf{m}$ is the number of inputs
2. Compute the value of L such that the number of rows of the Sylvester matrix
$$\mathscr{S} = \begin{bmatrix} \mathscr{S}_L(R_a) \\ \mathscr{S}_L(R_b) \end{bmatrix} \tag{1}$$
exceeds the number of columns by (at least) $\mathbf{p}(\mathscr{B}_+)$
3. Compute the left kernel basis $\begin{bmatrix} Z_a & -Z_b \end{bmatrix}$ of the Sylvester matrix $\mathscr{S}$
4. Define R as
$$\begin{bmatrix} R_0 & R_1 & \cdots & R_\ell \end{bmatrix} := Z_a \mathscr{S}_L(R_a) = Z_b \mathscr{S}_L(R_b). \tag{2}$$

This is presented in **Addition and intersection of linear time invariant behaviors**. The literate programming style [1, 5] is used in the following to make the experiment reproducible; the reader can simply copy and paste the Matlab code in a command line.

The first step is the generation of the data. Two nontrivial systems with 3 variables (1 input and 2 outputs) and order 2 are built via the function `drss`:

```
%Parameters
q=3;                % nr. of variables
m1=1; m2=1;            % nr. of inputs
l1=1; l2=1;            % systems lag l
```

```
p1=q-m1; p2=q-m2;  % nr. of outputs
n1=l1*p1; n2=l2*p2;% systems order

%Generate systems
sys1=drss(n1,p1,m1);  % first system
sys2=drss(n2,p2,m2);  % second system
```

To generate the trajectories of the two systems, we first simulate the outputs by providing a random input to the Matlab function `lsim`, and then we stack both the input and the outputs in a matrix.

```
T=100;             % time span;
u1=randn(T,m1);    % random input
y1=lsim(sys1,u1);  % output 1st system
w1=[u1 y1];        % trajectory 1st system
u2=randn(T,m2);    % random input
y2=lsim(sys2,u2);  % output 2nd system
w2=[u2 y2];        % trajectory 2nd system
```

A trajectory of the sum is simply given by the addition $w_1 + w_2$. We compute then the number of inputs (and outputs) and the order of the sum by choosing two different values for L in the equation rank $\mathscr{H}_L(w) = \mathbf{n}(\mathscr{B}) + L\mathbf{m}(\mathscr{B})$.

```
w = w1+w2;
[T, q]=size(w);
L=floor((T+1) / (q+1));
r1=rank(blkhank(w,L));
r2=rank(blkhank(w,L-1));
c = round([L 1; L-1 1] \ [r1; r2]);
m=c(1);  n=c(2);
p=q-m;
```

Observe that $p = 1 > 0$, as a consequence of the number of inputs of the original systems and the fact that such inputs are random (their intersection is empty).

What we need is an algorithm to compute the kernel representations of the two systems directly from a given trajectory. this is done via the function w2r [3]. We can then compute the kernel representations R_1 from w_1, R_2 from w_2. and $Rsum$ from $w_1 + w_2$ (the last is possible once we know the lag of the sum, which is simply given by n/p).

```
R1=w2r(w1,l1,m1,n1); % ker rep. sys1
R2=w2r(w2,l2,m2,n2); % ker rep. sys2
Rsum=w2r(w1+w2,n/p,m1+m2,n1+n2); % ker rep. sum.
```

where the other inputs to the function w2r are the system lag ℓ, the number of inputs and the system order, respectively. The representation $Rsum$ will be used as term of comparison with the proposed algorithm.

Remark 1. *The function* `w2r` *calls a function for building block-Hankel matrices. This is available from the slra toolbox [2].*

The idea to validate (2) is to compare it with the kernel representation Rsum. They are both kernel representations of the same system but computed via different algorithms.

First of all we need a function for building a (block) Toeplitz matrix

```
function TP = blktoep2(P,q, T)
%P matrix collecting the coefficients of the matrix polynomial
%q number of variables
%T+1 number of rows

m = size(P, 1);
l1 = size(P, 2) / q;

TP = zeros((T+1) * m, l1 * q);
for i = 1:T+1
    TP((i - 1) * m + (1:m), (i - 1) * q + (1:size(P,2))) = P;
end
```

For the computation of R in (2) we first need to compute the correct size of each multiplication block. This is done by setting the difference between the number of rows and the number of columns equal to p. Each block needs to have $L = \frac{p+l1p1+l2p2}{p1+p2-q}$ columns.

```
L=(p+l1*p1+l2*p2)/(p1+p2-q);
l=max(l1,l2);
rowS=L-(l+1)
```

The number `rowS` defines the number of rows in each multiplication block. The implementation of the algorithm follows.

```
S1=blktoep2(R1, q, rowS + l - l1);
S2=blktoep2(R2, q, rowS + l - l2);
S=[S1; S2];
Z=null(S')'  %kernel sylvester matrix
R=Z(:,1:size(S1,1))*S1; % ker. rep. sum
```

We should expect that the two representations Rsum and R are numerically different, i.e., they differ by the product with a unimodular matrix polynomial [4]. To see that they actually represent the same system, we can check that the time series $w_1 + w_2$ is in the kernel of both R and Rsum (assuming they are minimal).

```
norm(Rsum*blkhank(w1+w2,n/p+1));
norm(R*blkhank(w1+w2,n/p+1));
```

where the function `blkhank` comes from the *slra* toolbox [2] and generates a block Hankel matrix with a number of block rows equal to the second input.

References

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

[2] I. Markovsky and K. Usevich. Software for weighted structured low-rank approximation. *J. Comput. Appl. Math.*, 256:278–292, 2014.

[3] I. Markovsky, J. C. Willems, S. Van Huffel, and B. De Moor. *Exact and Approximate Modeling of Linear Systems: A Behavioral Approach.* SIAM, 2006.

[4] J. W. Polderman and J. C. Willems. *Introduction to Mathematical Systems Theory*, volume 26 of *Texts in Applied Mathematics.* Springer New York, New York, NY, 1998.

[5] N. Ramsey. Literate programming simplified. *IEEE Software*, 11:97–105, 1994.