Abstract

The accurate modeling of dynamical systems often results in a large number of differential equations. In this case, the system matrices then easily become too large to define state-space models (as objects) in MATLAB. In this contribution we present two new toolboxes that allow the definition and analysis of large-scale models by introducing sparse-space-state objects (ssM). Through model order reduction (ssMOR) it is possible to obtain high fidelity, low order approximations of the relevant dynamics to further reduce the computational complexity.

Exploit the sparsity of system matrices

Linear time-invariant systems are often represented by state-space models for the purpose of control design. If the original order \( N \) is very high (\( N \gg 10^5 \)), then the system matrices are generally sparse, meaning that the number of nonzero entries is much smaller than \( N^2 \).

\[
E = \begin{bmatrix} A & B \end{bmatrix} \quad \text{or} \quad E = \begin{bmatrix} A, B, C, D, E \end{bmatrix}
\]

The Control System Toolbox in MATLAB is not able to exploit this characteristic and stores all matrices as “full”. For this reason, the definition of state-space models by the commands

\[
\text{sys} = \\text{ss}(A,B,C,D) \quad \text{or} \quad \text{sys} = \text{discrete}(A,B,C,D,E)
\]

is only possible up until an order of \( 10^6 \) on a standard computer. Indeed, the definition of a full identity matrix of size \( 10^6 \) requires \( 80 \) GB of storage, while its sparse counterpart requires only 2.4 MB to be stored!

**ssS**

**ssMOR**

Exploit the sparsity of the system matrices, leading to substantial advantages in terms of storage and computational requirements. Sparse-state-space models can be defined by simply calling

\[
\text{sys} = \text{ss}(A,B,C,D)
\]

In addition, ssS contains many of the analysis function available in the Control System Toolbox adapted to exploit the sparsity, whenever possible.

**Functionalities**

**Model order reduction**

Model order reduction can be achieved with ssMOR by passing an ssS object to the respective MOR function, together with appropriate reduction parameters.

**Functionality**

Even when using ssS, computations with large-scale models can still be prohibitively demanding. For this reason, we often seek reduced order models of much lower order \( n < N \) as high fidelity approximations of the full order dynamics. The process of model order reduction (MOR) can be seen as a Petrov-Galerkin projection

\[
E x = (A - \varepsilon B)x + \varepsilon C \quad \text{or} \quad x = E^{-1}(A - \varepsilon B)x + \varepsilon C
\]

Accordingly, the task of MOR can be translated to finding appropriate projection matrices \( V, W \) depending on the properties of the original model to be preserved. Classical methods include modal truncation, balanced truncation and rational Krylov methods, while IRKA and CURED SPARK are examples of state-of-the-art functions.

**ssS and ssMOR – extensions of the Control System Toolbox**

In the following, we show the advantage of using ssS and ssMOR by running the same analysis code on different model classes and comparing the computational effort for analysis. The simulations were run on a benchmark model of order \( N = 1357 \) representing the cooling of a steel profile (rail_1357).

```
function analyzeModel(sys)
    % Analyze LTI model
    % Model order
    n = size(sys.A,1);
    % Memory requirement to store sys
    info = whos('sys');
    memory = info.bytes;
    % Check stability
    stabCheck = isstable(sys);
    % Modal order
    figure; bodemag(sys); figure; bodemag(sys);
    % H2 and Hinf norms
    h2norm = norm(sys);
    hinfnorm = norm(sys,inf);
end
```

```
function ssS
    % ssS and ssMOR are open-source toolboxes released under BSD license to foster the education and exchange in the field of MOR. More information available under www.rt.mw.tum.de/sssS or www.rt.mw.tum.de/sssMOR.
```

**ssS and ssMOR**

ssS and ssMOR are open-source toolboxes released under BSD license to foster the education and exchange in the field of MOR. More information available under www.rt.mw.tum.de/sssS or www.rt.mw.tum.de/sssMOR.

```
function ssM
    % ssMOR - a comparison of reduction methods
    In the following we give a small comparison of some reduction methods, contained in ssMOR, with respect to reduction time and approximation quality. Note that these may vary depending on the model and the reduction parameters selected. The rail model of order \( N = 1357 \) is reduced to an order \( n = 18 \). Krylov-algorithms were initialized at \( s_0 = 0 \).
```

Functionality

```
function ssM = 
    % ssMOR - a comparison of reduction methods
    In the following we give a small comparison of some reduction methods, contained in ssMOR, with respect to reduction time and approximation quality. Note that these may vary depending on the model and the reduction parameters selected. The rail model of order \( N = 1357 \) is reduced to an order \( n = 18 \). Krylov-algorithms were initialized at \( s_0 = 0 \).
```

**ssS** and **ssMOR** are open-source toolboxes released under BSD license to foster the education and exchange in the field of MOR. More information available under www.rt.mw.tum.de/sssS or www.rt.mw.tum.de/sssMOR.

```
function ssS
    % ssS and ssMOR are open-source toolboxes released under BSD license to foster the education and exchange in the field of MOR. More information available under www.rt.mw.tum.de/sssS or www.rt.mw.tum.de/sssMOR.
```