Lehrstuhl für Regelungstechnik
Fakultät für Maschinenwesen
Technische Universität München

sss & sssMOR

Analysis and Reduction of Large-Scale Dynamic Systems in MATLAB

A. Castagnotto, M. Cruz Varona, B. Lohmann

Abstract

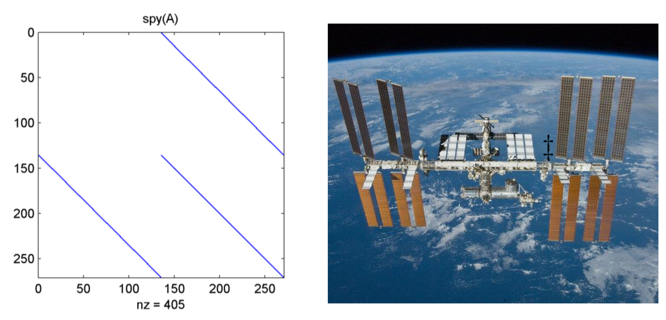
The accurate modeling of dynamical systems often results in a large number of differential equations. In this case, the system matrices then easily become too large to define state-space models (**ss** objects) in MATLAB. In this contribution we present two new toolboxes that allow the definition and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

sss

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 ($N \gg 10^3$), then the system matrices are generally sparse, meaning that the number of nonzero entries is much smaller than N^2.

$$\left.\begin{aligned} \boxed{E}\,\boxed{\dot{x}} &= \boxed{A}\,\boxed{x} + \boxed{B}\,\boxed{u} \\ \boxed{y} &= \boxed{C}\,\boxed{x} + \boxed{D}\,\boxed{u} \end{aligned}\right\} x \in \mathbb{R}^N$$



Sparsity pattern and fotograph of the ISS model.[2]

The Control System Toolbox in MATLAB[1] 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

```
sys = ss(A,B,C,D)  or  sys = dss(A,B,C,D,E)
```

is only possible up until an order of $\mathcal{O}(10^4)$ on a standard computer. Indeed, the definition of a full identity matrix of size 10^5 requires 80 GB of storage, while its sparse counterpart requires only 2.4 MB to be stored!

Functionality

sss exploits 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

```
sys = sss(A,B,C,D,E)
```

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

Functions

Manipulation:

```
>> truncate(sys,p,m); connect(sys1,sys2);…
>> sys1-sys2; sys1*sys2; c2d(sysC,Ts);…
```

Frequency domain analysis:

```
>> freqresp(sys,w); bode(sys); sigma(sys);…
```

Time domain analysis:

```
>> impulse(sys); step(sys); lsim(sys,u,Ts);…
```

Additional properties:

```
>> sys.isDae; sys.isSym; sys.isSimo;…
>> norm(sys,2); norm(sys,inf); isstable(sys);…
>> eigs(sys); spy(sys); diag(sys);…
```

Compatibility

Old Code:

```
>> sys = ss(A,B,C,D)
>> myCode(sys)
```

New Code:

```
>> sys = sss(A,B,C,D)
>> myCode(sys)
```

sssMOR

High fidelity reduced order modeling

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

$$\left.\begin{aligned} \overbrace{\boxed{W^\top}\,\boxed{E}\,\boxed{V}}^{E_r}\,\boxed{\dot{x}_r} &= \overbrace{\boxed{W^\top}\,\boxed{A}\,\boxed{V}}^{A_r}\,\boxed{x_r} + \overbrace{\boxed{W^\top}\,\boxed{B}}^{B_r}\,\boxed{u} \\ \boxed{y_r} &= \underbrace{\boxed{C}\,\boxed{V}}_{C_r}\,\boxed{x_r} + \underbrace{\boxed{D}}_{D_r}\,\boxed{u} \end{aligned}\right\} x_r \in \mathbb{R}^n$$

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.

Functionality

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

Function	Description
`modalMor(sys,n)`	Modal truncation with preservation of dominant modes
`tbr(sys,n)`	Balanced truncation with preservation of dominant Hankel Singular Values
`rk(sys,s0)`	Krylov-Subspace-Method with matching of transfer function Taylor series coefficients
`irka(sys,s0)`	Iterative Rational Krylov Algorithm for $\mathcal{H}_2$-optimal reduction
`cirka(sys,s0)`	Confined IRKA algorithm for fast $\mathcal{H}_2$-optimal reduction
`spark(sys,s0)`	Stability Preserving, Adaptive Rational Krylov Algorithm
`porkV(…)`	$\mathcal{H}_2$-Pseudo-Optimal Rational Krylov Algorithm
`cure(sys)`	CUmulative REduction with adaptive choice of reduced order

sss and sssMOR – extensions of the Control System Toolbox

In the following, we show the advantage of using **sss** and **sssMOR** by running the <u>same</u> 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`)[3].

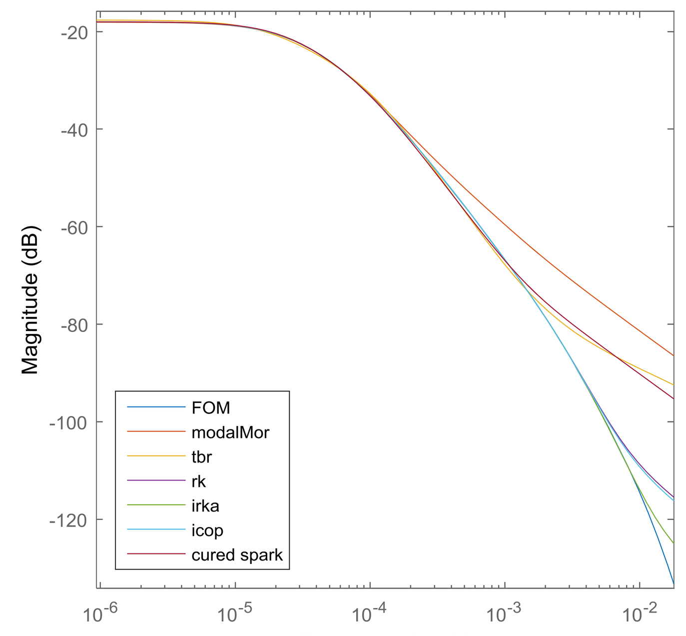
```
function analyzeModel(sys)
  % --Analyze LTI model
  %% Model order
  n = size(sys.A,1);
  %% Memory requirement to store sys
  info        = whos('sys');
  reqMem      = info.bytes
  %% Check stability
  stabCheck = isstable(sys)
  %% Bode magnitude plot
  figure; bodemag(sys)
  %% H2 and Hinf norms
  h2norm  = norm(sys)
  h8norm  = norm(sys,Inf)
```

		`ss`	`sss`	`ssRed`
order		1357	1357	50
reduction (`tbr`)	[s]	-	-	2.89
memory	[MB]	29.6	0.3	0.05
`isstable`	[s]	28.8	0.2	0.01
`bodemag`	[s]	31.6	1.1	0.63
`norm(sys)`	[s]	62.2	54.3	0.01
`norm(sys,inf)`	[s]	188	6.4	0.01

sssMOR – a comparison of reduction methods

In the following we give a small comparison of some reduction methods, contained in **sssMOR**, 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[3] of order $N = 1357$ is reduced to an order $n = 10$. Krylov-algorithms were initialized at $s0 = 0$.

	time [s]	$\frac{\lvert G-G_r\rvert_{H_2}}{\lvert G\rvert_{H_2}}$	$\frac{\lvert G-G_r\rvert_{H_\infty}}{\lvert G\rvert_{H_\infty}}$
`modalMor (sm)`	1.56	5.63e-02	7.63e-03
`tbr`	1.11	4.96e-02	4.39e-02
`rk`	0.15	4.85e-04	1.95e-05
`irka`	0.27	1.10e-04	3.62e-06
`rkIcop`	0.23	4.49e-04	1.84e-05
`cure` (with `spark`)	8.12	1.56e-02	8.85e-03



sss und **sssMOR** are open-source toolboxes released under BSD license to foster the education and exchange in the field of MOR. More information available under <u>www.rt.mw.tum.de/?sss</u> or <u>www.rt.mw.tum.de/?sssMOR</u>.

[1]MATLAB and Control System Toolbox (Release 2015b) are trademarks of The MathWorks, Inc., Natick, Massachusetts, United States.

[2]SLICOT Benchmark Models: http://slicot.org/20-site/126-benchmark-examples-for-model-reduction

[3]Available at https://simulation.uni-freiburg.de/downloads/benchmark/Steel%20Profiles%20%2838881%29

Lehrstuhl für Regelungstechnik
Prof. Dr.-Ing. habil. Boris Lohmann
www.rt.mw.tum.de
{a.castagnotto, maria.cruz, lohmann}@tum.de

Acknowledgements of financial support:
Deutsche Forschungsgemeinschaft (DFG), LO408/19-1

Acknoledgments of contribution:
Heiko Peuscher, Rudy Eid, Sylvia Cremer, Stefan Jaensch, Thomas Emmert, Jorge Silva, Rodrigo Mancilla, Siyang Hu, Niklas Kochdumper, Lisa Jeschek.

Third-Party
sss & **sssMOR** support the usage of M-M.E.S.S., the matrix equation sparse solver developed at the MPI Magdeburg by Jens Saak, Martin Köhler and Peter Benner (GPL license).