Received: 31 May 2019 | Accepted: 28 June 2019

DOI: 10.1002/pamm.201900459

System-theoretic model order reduction with pyMOR

Linus Balicki[1,*], **Petar Mlinarić**[2,**], **Stephan Rave**[3,***], and **Jens Saak**[2,†]

[1] Otto von Guericke University Magdeburg, Universitätsplatz 2, 39106 Magdeburg, Germany

[2] Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstraße 1, 39106 Magdeburg, Germany

[3] University of Münster, Orleans-Ring 10, 48149 Münster, Germany

This paper shows recent developments in pyMOR, in particular the addition of system-theoretic methods. All methods are implemented using pyMOR's abstract interfaces, which allows the application to partial differential equation (PDE) models implemented with third-party libraries. We demonstrate this by applying balanced truncation to a PDE model discretized in FEniCS.

1 Introduction

pyMOR is a free software library for building model order reduction (MOR) applications in the Python programming language. All of its algorithms are based on abstract interfaces which allow seamless integration with external partial differential equation (PDE) solver packages (see [1]). Its initial focus was on reduced basis methods for linear and nonlinear parameterized PDEs. We present here recent developments in system-theoretic MOR methods and related algorithms.

In Section 2, we give a brief overview of linear systems and balanced truncation (BT). Next, we describe pyMOR's abstract interfaces and the implementation of the methods in Section 3. We demonstrate BT applied to a PDE model discretized using FEniCS in Section 4.

2 System-theoretic model order reduction

We consider continuous-time, linear time-invariant systems

$$E\dot{x}(t) = Ax(t) + Bu(t), \quad y(t) = Cx(t), \quad x(0) = 0, \tag{1}$$

with $E, A \in \mathbb{R}^{n\times n}$ (E invertible), $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$, input $u(t) \in \mathbb{R}^m$, state $x(t) \in \mathbb{R}^n$, and output $y(t) \in \mathbb{R}^p$. The goal of MOR is to find a reduced-order model (ROM)

$$\hat{E}\dot{\hat{x}}(t) = \hat{A}\hat{x}(t) + \hat{B}u(t), \quad \hat{y}(t) = \hat{C}\hat{x}(t), \quad \hat{x}(0) = 0, \tag{2}$$

with $\hat{E}, \hat{A} \in \mathbb{R}^{r\times r}$, $\hat{B} \in \mathbb{R}^{r\times m}$, $\hat{C} \in \mathbb{R}^{p\times r}$, for some $r \ll n$ such that $\|y - \hat{y}\|$ is small for every input u in some function space. BT is a projection-based method where $\hat{E}, \hat{A}, \hat{B}, \hat{C}$ are obtained by Petrov-Galerkin projection of the original system matrices on appropriate ansatz and test spaces. The main computational cost in finding these spaces lies in solving two large-scale Lyapunov equations: $APE^{\mathrm{T}} + EPA^{\mathrm{T}} + BB^{\mathrm{T}} = 0$ and $A^{\mathrm{T}}QE + E^{\mathrm{T}}QA + C^{\mathrm{T}}C = 0$. Low-rank approximations for P and Q are computed using the low-rank alternating directions implicit (LR-ADI) method. The method consists of solving a sequence of linear systems $(p_iE + A)x_i = b_i$ with shifts p_i.

3 Generic interfaces for model order reduction

pyMOR's PDE solver integration is based on three abstract interfaces: `VectorArrays`, `Operators`, and `Models`. Each `VectorArray` is an element of a `VectorSpace`, `Operator` is a mapping (possibly nonlinear or parametric) between two `VectorSpaces`, and a `Model` is a structured collection of `Operators`. The interfaces are completely agnostic of the solver's internal data representation. All reduction algorithms in pyMOR are implemented in terms of these interfaces.

In particular, in pyMOR 0.5, `LTISystem` is a model representing (1), where E, A, B, C are `Operators`. In case of the example in Section 4, these operators come from a discretization in FEniCS. Via linear combination and the `Operator's apply_inverse` method, the LR-ADI algorithm instructs FEniCS to solve $(p_iE + A)x_i = b_i$.

* linus.balicki@ovgu.de

** Corresponding author: mlinaric@mpi-magdeburg.mpg.de

*** stephan.rave@uni-muenster.de

† saak@mpi-magdeburg.mpg.de

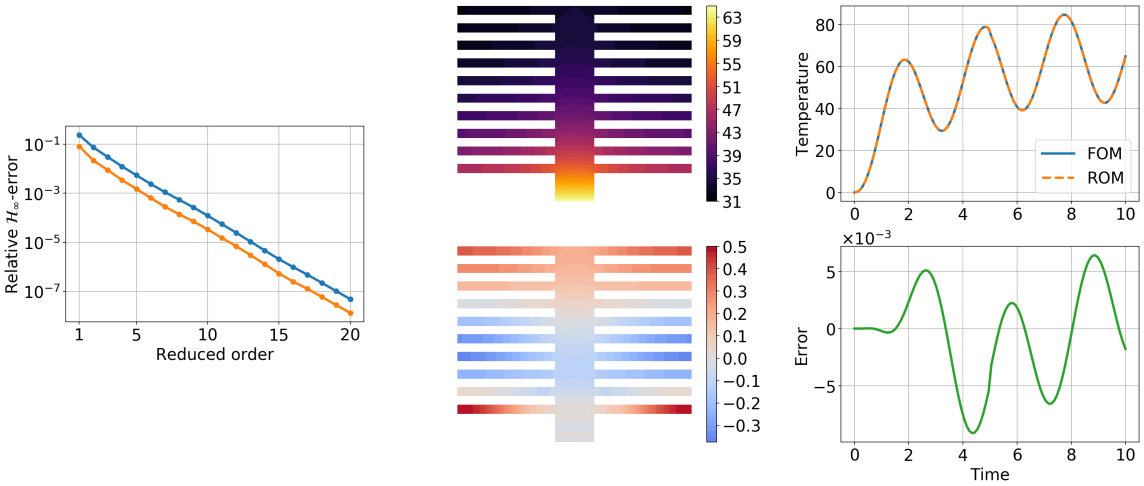

Fig. 1: Left column: Upper and lower bounds for the relative $\mathcal{H}_\infty$ reduction error when using balanced truncation. Middle column: Full-order model snapshot at time $t = 10$ and its reconstruction error using a reduced-order model of order 10. Right column: The full-order and reduced-order models' outputs and the error.

4 Numerical example

As an example, we consider the heat equation over a cross section of a heat sink (see [2] for the problem description). Finite element discretization, using FEniCS, leads to a system of the form (1) with $n = 12\,296$, $m = 1$, and $p = 1$. The top-middle plot in Figure 1 shows the state of the full-order model at $t = 10$ when the input is $u(t) = \sin(\frac{\pi}{3}t)^2$. The average run time is about 8 seconds on a laptop (model: Vaio VJS132C11L; processor: Intel© Core™ i7-8550U CPU @ 1.80 GHz, 4 cores, 8 logical processors, hyper-threading activated; RAM: 8 GB; cache: 8 MB).

We applied BT (using pyMOR 0.5) to get a ROM of order 10, with a run time of about 7 seconds. Based on Hankel singular values, we find the relative $\mathcal{H}_\infty$-error lies in the interval $[7.27 \times 10^{-5}, 1.23 \times 10^{-4}]$, shown also in the left plot of Figure 1. The relative $\mathcal{H}_2$-error can be computed and its value is 7.37×10^{-3}. In the bottom row of Figure 1, we see the state and output errors of the ROM. As expected, we see that the output is approximated better than the state. The average run time of the simulation is about 15 milliseconds, which is roughly 500 times faster then for the full-order model.

The entire script to generate these results can be found in [2].

5 Conclusion and outlook

We demonstrated BT applied to PDE models discretized in the PDE solver package FEniCS. This is possible due to the LR-ADI method's implementation using pyMOR's abstract interfaces.

The example used FEniCS, but other solver packages are also possible. Currently also deal.II, DUNE, and NGSolve are supported. Custom (domain specific) solvers can be easily integrated with pyMOR.

In pyMOR 0.5, further variants of BT for first-order systems are available, together with a few methods for second-order systems. Furthermore, interpolatory methods for first-order and second-order systems are also included.

Some PDE solver packages do not support complex numbers (e.g., FEniCS). In the example, we considered a system with symmetric E and A, where real shifts p_i were sufficient. For general systems, we plan to add support to pyMOR for solving linear systems with complex shifts based on real linear algebra to further enable applicability of system-theoretic methods to models defined with such PDE solver packages.

Acknowledgements This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2044 - 390685587, Mathematics Münster:Dynamics - Geometry - Structure, the German Federal Ministry of Education and Research (BMBF) under contract 05M18PMA, and DFG "pyMOR - Sustainable Software for Model Order Reduction" project (RA 3055/1-1, SA 3477/1-1).

References

[1] R. Milk, S. Rave, and F. Schindler, SIAM J. Sci. Comput. **38(5)**, pp. S194–S216 (2016).
[2] P. Mlinarić and S. Rave, pyMOR demo for balanced truncation applied to a heat sink model in FEniCS, Zenodo (2019), http://doi.org/10.5281/zenodo.3232900.

