pyMOR – Reduced Order Modeling with Python [⋆]

Linus Balicki [*] **René Fritze** [**] **Petar Mlinarić** [*] **Stephan Rave** [**]
Jens Saak [***] **Felix Schindler** [**]

[*] *Department of Mathematics, Virginia Tech, Blacksburg, USA (e-mail: {balicki,mlinaric}@vt.edu).*
[**] *Applied Mathematics, University of Münster, Germany (e-mail: {rene.fritze,stephan.rave,felix.schindler}@uni-muenster.de)*
[***] *Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg, Germany (e-mail: saak@mpi-magdeburg.mpg.de)*

Abstract: pyMOR is a free and open source software library for writing model order reduction applications with the Python programming language. Implemented algorithms include both Reduced Basis and system-theoretic reduction methods, as well as non-intrusive approaches such as approximation with artificial neural networks. All these algorithms can be easily integrated with external high-performance PDE solver packages. In this poster contribution we give a brief overview on the design of pyMOR. Further, we will present in more details two main features of the upcoming 2022.1 release: 1. a new and unified model hierarchy, 2. new discretization routines to create these models from common analytical problem definitions using different PDE solver backends.

Keywords: model reduction, software, reduced basis method, balanced truncation, IRKA, POD, empirical interpolation, artificial neural networks

1. INTRODUCTION

Over the last decade, Model Order Reduction (MOR) has become an essential tool in mathematical modeling and simulation workflows, significantly speeding up computation times, especially in multi-query contexts such as optimization, optimal control or interactive design applications (see, e.g., Benner et al. (2020)).

Since MOR methods work on top of existing ODE/PDE solvers, their implementation is often non-trivial and requires knowledge of both the given solver and the MOR method to be implemented. pyMOR (Milk et al. (2016); Balicki et al. (2019); Mlinarić et al. (2021), `https://www.pymor.org`) is a free and open source MOR library for the Python programming language which facilitates the integration of MOR methods with high-performance solvers by expressing MOR algorithms via operations on simple solver interface classes.

In this poster contribution we give a short overview on pyMOR's design (Section 2). In Section 3 we discuss two new features of the upcoming 2022.1 release, which significantly facilitate the construction and reduction of full-order models (FOMs) in pyMOR.

[⋆] Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy EXC 2044 –390685587, Mathematics Münster: Dynamics–Geometry–Structure and under RA 3055/1-1, SA 3477/1-1: pyMOR – Nachhaltige Software zur Modell-Ordnungs-Reduktion.

2. SOFTWARE DESIGN

pyMOR's design is based on the idea of expressing all MOR operations through low-level, MOR agnostic interface classes for interacting with the solver that implements the FOM. In particular this allows to easily prototype new MOR algorithms using a lightweight discretization library, such as the toolkit shipped with pyMOR, and later use the same implementation for a large application problem implemented in a specialized high-performance code.

2.1 Interfaces

pyMOR interacts with external solvers through `VectorArrays`, `Operators` and `Models`. A `VectorArray` represents an ordered collection of vectors of the same dimension and allows the usual linear algebra operations such as inner products or linear combinations. `Operators` encapsulate matrices or nonlinear operators, which can be `applied` to `VectorArrays`. A `Model` encodes the mathematical structure of the given FOM and exposes the solvers' simulation routines via the `solve` method.

We emphasize that all required interface operations can be expected to be already implemented in the external solver. In particular, no MOR-specific code has to be added to integrate a new solver with pyMOR. The integration usually is technically realized by compiling the external solver as a Python extension module, but also network or disk-based communication is possible.

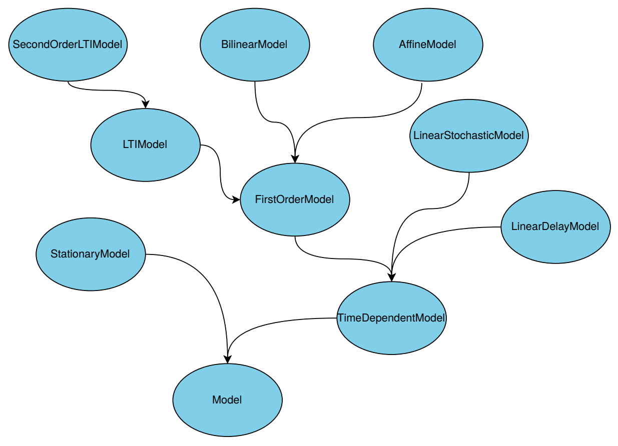

Fig. 1. Hierarchy of `Models` in pyMOR 2022.1 (subject to change). `Reductors` for a base class can be applied to each subclass in the hierarchy.

2.2 Algorithms

Based on the aforementioned interface classes, pyMOR implements various MOR algorithms such as Reduced Basis (RB) methods, Proper Orthogonal Decomposition (POD), (discrete) empirical interpolation, Balanced Truncation, the Iterative Rational Krylov Algorithm (IRKA), as well as non-intrusive data-driven methods, such as approximation with artificial neural networks. As an example, the Petrov-Galerkin projection $W^T \cdot A \cdot V$ of the full-order matrix A onto bases spanned by the columns of V and W can be written as

```
W.inner(A.apply(V)),
```

where A is given as an `Operator` and V, W are `VectorArrays`. All MOR algorithms are realized as `Reductor` objects, which transform a given full-order `Model` to a corresponding reduced-order `Model` (ROM) of similar type, where the FOM's `Operators` have been replaced by new `Operators` storing their reduced counterparts. Due to the general nature of pyMOR's interfaces, pyMOR also implements further algorithms, such as Gram-Schmidt orthonormalization, a Newton algorithm or different time steppers.

3. NEW FEATURES IN PYMOR 2022.1

In this section we highlight two main new features in pyMOR related to building and reducing `Models` in pyMOR.

3.1 A new Model hierarchy

To our knowledge, pyMOR is the only available software package which includes a large variety of both RB methods, which are geared at parameterized PDEs, as well as system-theoretic methods such as Balanced Truncation or IRKA, which originally were mainly developed for LTI systems. For years, both branches of MOR have been developed mostly independently from one another, which, so far, has also been reflected in pyMOR, where system-theoretic `reductors` operate on `LTIModels` and related classes, which are incompatible with the classes required by RB `reductors`, which operate on `StationaryModels` and `InstationaryModels`.

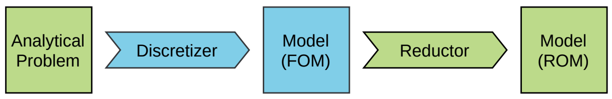

Fig. 2. Reduced-order modeling pipeline in pyMOR (blue background: solver specific code, green background: generic code).

In pyMOR 2022.1, the existing `Model` classes are refactored into a new class hierarchy that provides a unified view onto `Models` for both RB and system-theoretic methods (see Fig. 1). In particular, this allows to seamlessly apply RB `reductors` for general `FirstOrderModels` to `Models` with additional structure, such as `LTIModel`, in addition to the more specialized methods, e.g., Balanced Truncation.

3.2 Discretizers for external PDE solvers

pyMOR's builtin discretization toolkit uses `analytical-problems` as data structures to define the (parameterized) PDE problem to be solved. `Analyticalproblems` combine a definition of the computational domain with coefficient functions for the respective PDE that is given by the respective problem class. Such an `analyticalproblem` is then given as an input to a `discretizer`, which builds a corresponding `Model` using pyMOR's builtin FEM/FV `Operators`. So far, building a `Model` with an external PDE solver, even for a standard benchmark problem, required manually building the `Model` using appropriate PDE solver code and pyMOR's wrapper classes for the given solver.

Based on pyMOR's recently introduced symbolic expression library, pyMOR 2022.1 includes new `discretizers` which allow to use the same `analyticalproblems` to automatically build `Models` using external solvers, such as FEniCS, and, thus, to make use of advanced features of these solvers, such as higher-order methods or MPI parallelization. This enables a powerful reduced order modeling workflow, where the user can easily build efficient ROMs, even for complex problems, without having knowledge of the used PDE solver library (see Fig. 2).

REFERENCES

Balicki, L., Mlinarić, P., Rave, S., and Saak, J. (2019). System-theoretic model order reduction with pyMOR. *PAMM*, 19(1). doi:10.1002/pamm.201900459.

Benner, P., Schilders, W., Grivet-Talocia, S., Quarteroni, A., Rozza, G., and Miguel Silveira, L. (eds.) (2020). *Model Order Reduction: Volume 1–3*. De Gruyter.

Milk, R., Rave, S., and Schindler, F. (2016). pyMOR – Generic Algorithms and Interfaces for Model Order Reduction. *SIAM Journal of Scientific Computing*, 38(5), S194–S216. doi:10.1137/15M1026614.

Mlinarić, P., Rave, S., and Saak, J. (2021). Parametric Model Order Reduction Using pyMOR. In P. Benner, T. Breiten, H. Faßbender, M. Hinze, T. Stykel, and R. Zimmermann (eds.), *Model Reduction of Complex Dynamical Systems*, International Series of Numerical Mathematics, 357–367. Springer International Publishing, Cham. doi:10.1007/978-3-030-72983-7_17.