Universität Münster

pyMOR: From Model Order Reduction to Numerics in Abstract Spaces

Linus Balicki[1], Hendrik Kleikamp[2], Petar Mlinarić[1], Stephan Rave[2], Jens Saak[3], Felix Schindler[4]

[1]Virginita Tech
[2]University of Münster, Germany
[3]MPI DCTS
[4]Arup

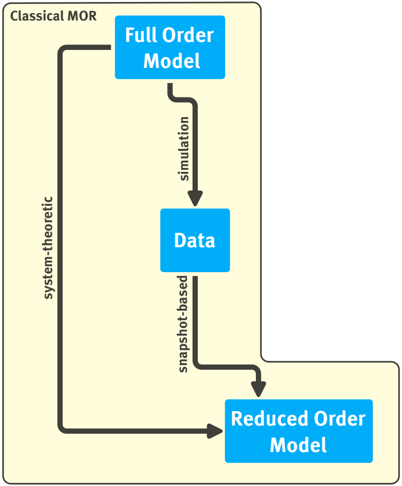
PDESoft 2024

Cambridge, July 2, 2024

living.knowledge

Mathematics Münster
Cluster of Excellence

What is Model Order Reduction?

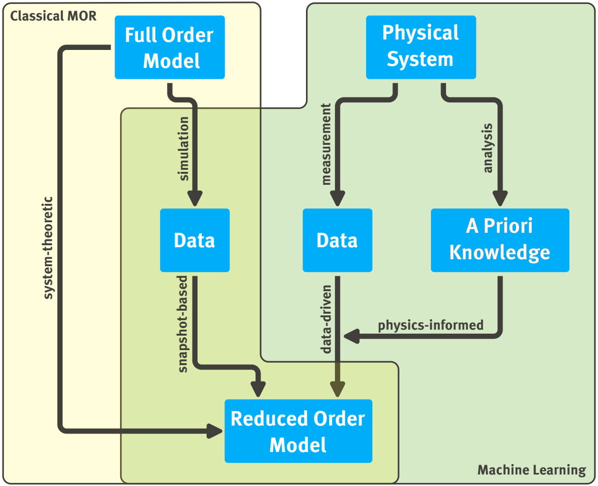


Compute computationally efficient surrogate models for

- optimization,
- control,
- real-time predictions,
- ...

Certification: rigorously control approximation error a priori or a posteriori.

What is Model Order Reduction?



Compute computationally efficient surrogate models for

- optimization,
- control,
- real-time predictions,
- ...

Certification: rigorously control approximation error a priori or a posteriori.

Reduced Basis Methods for Elliptic Problems

Parametric linear elliptic problem (full order model)

For given parameter $\mu \in \mathcal{P}$, find $u_h(\mu) \in V_h$ s.t.

$$
\begin{aligned}
a(u_h(\mu), v_h; \mu) &= f(v_h) \qquad\qquad \forall v_h \in V_h \\
y_h(\mu) &= g(u_h(\mu))
\end{aligned}
$$

Reduced Basis Methods for Elliptic Problems

Parametric linear elliptic problem (full order model)

For given parameter $\mu \in \mathcal{P}$, find $u_h(\mu) \in V_h$ s.t.

$$\begin{aligned} a(u_h(\mu), v_h; \mu) &= f(v_h) \qquad\qquad \forall v_h \in V_h \\ y_h(\mu) &= g(u_h(\mu)) \end{aligned}$$

Parametric linear elliptic problem (reduced order model)

For given $V_N \subset V_h$, let $u_N(\mu) \in V_N$ be given by Galerkin proj. onto V_N, i.e.

$$\begin{aligned} a(u_N(\mu), v_N; \mu) &= f(v_N) \qquad\qquad \forall v_N \in V_N \\ y_N(\mu) &= g(u_N(\mu)) \end{aligned}$$

RB Methods – Computing V_N

Weak greedy basis generation

function $\textsc{Weak-Greedy}(\mathcal{S}_{train} \subset \mathcal{P}, \varepsilon)$
  $V_N \leftarrow \{0\}$
  while $\max_{\mu \in \mathcal{S}_{train}} \textsc{Err-Est}(\textsc{ROM-Solve}(\mu), \mu) > \varepsilon$ **do**
    $\mu^* \leftarrow \operatorname{arg-max}_{\mu \in \mathcal{S}_{train}} \textsc{Err-Est}(\textsc{ROM-Solve}(\mu), \mu)$
    $V_N \leftarrow \operatorname{span}(V_N \cup \{\textsc{FOM-Solve}(\mu^*)\})$
  end while
  return V_N
end function

Err-Est

Use residual-based error estimate w.r.t. FOM (finite dimensional $\rightsquigarrow$ can compute dual norms).

RB Methods – Computing V_N

Weak greedy basis generation

function $\textsc{Weak-Greedy}(\mathcal{S}_{train} \subset \mathcal{P}, \varepsilon)$
    $V_N \leftarrow \{0\}$
    while $\max_{\mu \in \mathcal{S}_{train}} \textsc{Err-Est}(\textsc{ROM-Solve}(\mu), \mu) > \varepsilon$ **do**
        $\mu^* \leftarrow \operatorname{arg-max}_{\mu \in \mathcal{S}_{train}} \textsc{Err-Est}(\textsc{ROM-Solve}(\mu), \mu)$
        $V_N \leftarrow \operatorname{span}(V_N \cup \{\textsc{FOM-Solve}(\mu^*)\})$
    end while
    return V_N
end function

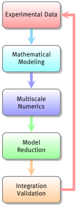
Err-Est

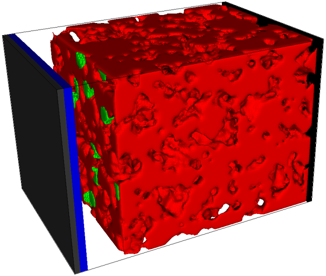
Use residual-based error estimate w.r.t. FOM (finite dimensional $\rightsquigarrow$ can compute dual norms).

- Use parameter separability / hyperreduction to gain online efficiency.
- Use dual-weighted residual approach for improved convergence w.r.t to output $y_N(\mu)$.

Example: RB Approximation of Li-Ion Battery Models

ulm university universität ulm
Institute of Technical Thermodynamics
MULTIBAT
SPONSORED BY THE Federal Ministry of Education and Research
Fraunhofer ITWM
Westfälische Wilhelms-Universität Münster





MULTIBAT: Gain understanding of degradation processes in rechargeable Li-Ion Batteries through mathematical modeling and simulation at the pore scale.

FOM:

- 2.920.000 DOFs
- Simulation time: ≈ 15.5h

ROM:

- Snapshots: 3
- dim V_N = 245
- Rel. err.: $< 4.5 \cdot 10^{-3}$
- Reduction time: ≈ 14h
- Simulation time: ≈ 8m
- Speedup: 120

pyMOR – Model Order Reduction with Python

Goal 1

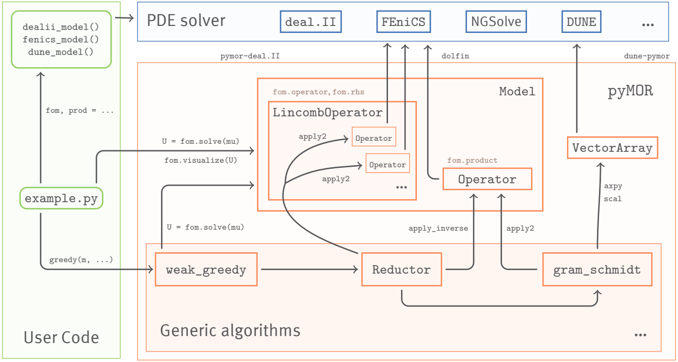
One library for algorithm development *and* real-world applications.

Goal 2

Large collection of MOR algorithms in a unified language.

- Started late 2012, 23k lines of Python code, 10k single commits.
- BSD-licensed, open development model, 41 contributors.
- Quick prototyping with `NumPy`/`SciPy`-based discretization toolkit.
- Seamless integration with high-performance PDE solvers.
  Official bindings for: `deal.II`, `DUNE`, `FEniCS`, `NGSolve`

Generic Interfaces for MOR



- VectorArray, Operator, Model classes represent objects in solver's memory.
- No communication of high-dimensional data.
- Tight, low-level integration with external solver.
- No MOR-specific code in solver.

Projecting an elliptic Model

Mathematical formulation

$$A(\mu)u(\mu) = F$$

$$A_r(\mu) := V^T A(\mu) V$$

$$F_r(\mu) := V^T F$$

$$A_r(\mu)u_r(\mu) = F_r$$

Projecting an elliptic Model

Mathematical formulation

$$A(\mu)u(\mu) = F$$

$$A_r(\mu) := V^T A(\mu) V$$

$$F_r(\mu) := V^T F$$

$$A_r(\mu)u_r(\mu) = F_r$$

Low-level interface

```
u = A_op.apply_inverse(F, mu)
A_r = V.inner(A_op.apply(V, mu))
F_r = V.inner(F)
u_r = np.linalg.solve(A_r, F_r)
```

Projecting an elliptic Model

Mathematical formulation

$$
\begin{aligned}
A(\mu)u(\mu) &= F \\
A_r(\mu) &:= V^T A(\mu) V \\
F_r(\mu) &:= V^T F \\
A_r(\mu)u_r(\mu) &= F_r
\end{aligned}
$$

Low-level interface

```
u = A_op.apply_inverse(F, mu)
A_r = V.inner(A_op.apply(V, mu))
F_r = V.inner(F)
u_r = np.linalg.solve(A_r, F_r)
```

Automatic projection (parameter-aware)

```
A_r_op = project(A, V, V)
```

Projecting an elliptic Model

Mathematical formulation

$$A(\mu)u(\mu) = F$$

$$A_r(\mu) := V^T A(\mu) V$$

$$F_r(\mu) := V^T F$$

$$A_r(\mu)u_r(\mu) = F_r$$

Low-level interface

```
u = A_op.apply_inverse(F, mu)
A_r = V.inner(A_op.apply(V, mu))
F_r = V.inner(F)
u_r = np.linalg.solve(A_r, F_r)
```

Automatic projection (parameter-aware)

```
A_r_op = project(A, V, V)
```

High-level code

```
u = fom.solve(mu)
red = StationaryRBReductor(fom, V)
rom = red.reduce()
u_r = rom.solve()
```

Building a model

Building a model

Using NumPy/SciPy matrices

```
fom = LTIModel.from_matrices(A, B, C, D, E)
```

Building a model

Using NumPy/SciPy matrices

```
fom = LTIModel.from_matrices(A, B, C, D, E)
```

Using builtin discretization toolkit

```
p = thermal_block_problem((2,3))  # or define your own problem
fom, _ = discretize_stationary_cg(p, diameter=1/100)
```

Building a model

Using NumPy/SciPy matrices

```
fom = LTIModel.from_matrices(A, B, C, D, E)
```

Using builtin discretization toolkit

```
p = thermal_block_problem((2,3))  # or define your own problem
fom, _ = discretize_stationary_cg(p, diameter=1/100)
```

Using an external solver

```
from pymor.discretizers.fenics import discretize_stationary_cg
fom, _ = discretize_stationary_cg(p, diameter=1/100)
```

Building a model

Using NumPy/SciPy matrices

```
fom = LTIModel.from_matrices(A, B, C, D, E)
```

Using builtin discretization toolkit

```
p = thermal_block_problem((2,3))  # or define your own problem
fom, _ = discretize_stationary_cg(p, diameter=1/100)
```

Using an external solver

```
from pymor.discretizers.fenics import discretize_stationary_cg
fom, _ = discretize_stationary_cg(p, diameter=1/100)
```

Manual integration of external solver

```
from pymor.bindings.fenics import FenicsMatrixOperator, FenicsVectorSpace
space = FenicsVectorSpace(V)
A = FenicsMatrixOperator(A, V, V)
```

Generic Algorithms

Models

StationaryModel	PHLTIModel	BilinearModel	QuadraticHamiltonianModel
InstationaryModel	SecondOrderModel	TransferFunction	LinearStochasticModel
LTIModel	LinearDelayModel		

Algorithms

POD	certified RB	parametric PG projection	balanced truncation
P-AAA new!	HAPOD	adaptive greedy basis generation	empirical interpolation
DEIM	TF-IRKA	non-intrusive MOR with ANNs	Arnoldi eigensolver
DMD new!	rational Arnoldi	low-rank ADI Lyapunov solver	randomized GSVD new!
IRKA	PSD cotangent lift new!	low-rank ADI Riccati solver	randomized eigensolver new!
SAMDP	PSD complex SVD new!	bitangential Hermite interpolation	biorthogonal Gram-Schmidt
LGMRES	modal truncation	Gram-Schmidt with reiteration	tangential rational Krylov
LSMR	time steppers	symplectic Gram-Schmidt new!	Newton algorithm
LSQR	SLYCOT support	PSD SVD-like decomposition new!	second-order BT/IRKA

MOR or More?

We actually implement a lot of fundamental non-MOR algorithms.

MOR or More?

We actually implement a lot of fundamental non-MOR algorithms.

Algorithms

POD	certified RB	parametric PG projection	balanced truncation
P-AAA	HAPOD	adaptive greedy basis generation	empirical interpolation
DEIM	TF-IRKA	non-intrusive MOR with ANNs	**Arnoldi eigensolver**
DMD	rational Arnoldi	**low-rank ADI Lyapunov solver**	**randomized GSVD**
IRKA	PSD cotangent lift	**low-rank ADI Riccati solver**	**randomized eigensolver**
SAMDP	PSD complex SVD	bitangential Hermite interpolation	**biorthogonal Gram-Schmidt**
LGMRES	modal truncation	**Gram-Schmidt with reiteration**	tangential rational Krylov
LSMR	**time steppers**	**symplectic Gram-Schmidt**	**Newton algorithm**
LSQR	SLYCOT support	**PSD SVD-like decomposition**	second-order BT/IRKA

MOR or More?

We actually implement a lot of fundamental non-MOR algorithms.

$\Longrightarrow$ Move into new base library and build ecosystem around it.

Algorithms

POD	certified RB	parametric PG projection	balanced truncation
P-AAA	HAPOD	adaptive greedy basis generation	empirical interpolation
DEIM	TF-IRKA	non-intrusive MOR with ANNs	**Arnoldi eigensolver**
DMD	rational Arnoldi	**low-rank ADI Lyapunov solver**	**randomized GSVD**
IRKA	PSD cotangent lift	**low-rank ADI Riccati solver**	**randomized eigensolver**
SAMDP	PSD complex SVD	bitangential Hermite interpolation	**biorthogonal Gram-Schmidt**
LGMRES	modal truncation	**Gram-Schmidt with reiteration**	tangential rational Krylov
LSMR	**time steppers**	**symplectic Gram-Schmidt**	**Newton algorithm**
LSQR	SLYCOT support	**PSD SVD-like decomposition**	second-order BT/IRKA

NiAS - Numerics in Abstract Spaces

NiAS - Numerics in Abstract Spaces

Goal 1: Language/library agnostic interfaces and algorithms

- Support various backends (PetSC, Eigen, NumPy, ...)
- Seamless support of C/C++, Python, Julia, Rust, ...
- Improve interoperability, comparability $\Rightarrow$ FAIR/Open Science $\Rightarrow$ MaRDI/Dmitry's talk

NiAS - Numerics in Abstract Spaces

Goal 1: Language/library agnostic interfaces and algorithms

- Support various backends (PetSC, Eigen, NumPy, ...)
- Seamless support of C/C++, Python, Julia, Rust, ...
- Improve interoperability, comparability $\Rightarrow$ FAIR/Open Science $\Rightarrow$ MaRDI/Dmitry's talk

Goal 2: Algorithms in abstract spaces

- Constrained finite-element spaces
- Subspaces defined via arbitrary projections
- Truly infinite-dimensional spaces (adaptive FEM, Chebfun, ...)
- Clarify mathematical structure (bilinear forms vs. operators, ...)

NiAS - Numerics in Abstract Spaces

Goal 1: Language/library agnostic interfaces and algorithms

- Support various backends (PetSC, Eigen, NumPy, ...)
- Seamless support of C/C++, Python, Julia, Rust, ...
- Improve interoperability, comparability $\Rightarrow$ FAIR/Open Science $\Rightarrow$ MaRDI/Dmitry's talk

Goal 2: Algorithms in abstract spaces

- Constrained finite-element spaces
- Subspaces defined via arbitrary projections
- Truly infinite-dimensional spaces (adaptive FEM, Chebfun, ...)
- Clarify mathematical structure (bilinear forms vs. operators, ...)

Status: Early planning/development (Python, C++).
Comments/contributions very welcome!

Thank you for your attention!

NiAS – Numerics in Abstract Spaces
`https://github.com/nias-project/nias`

pyMOR – Generic Algorithms and Interfaces for Model Order Reduction
SIAM J. Sci. Comput., 38(5), 2016.
`http://www.pymor.org/`

MULTIBAT: Unified Workflow for fast electrochemical 3D simulations of lithium-ion cells combining virtual stochastic microstructures, electrochemical degradation models and model order reduction
J. Comp. Sci., 2018.


pyMOR School 24
and User Meeting
August 26-30
University of Münster
https://school.pymor.org