WWU MÜNSTER

pyMOR

# Generic Algorithms and Interfaces for Model Order Reduction

Linus Balicki[1], René Fritze[2], Hendrik Kleikamp[2], Petar Mlinarić[1], Stephan Rave[2], Felix Schindler[2]
[1]Virginia Tech, Blacksburg, USA, [2]Uni Münster, Germany

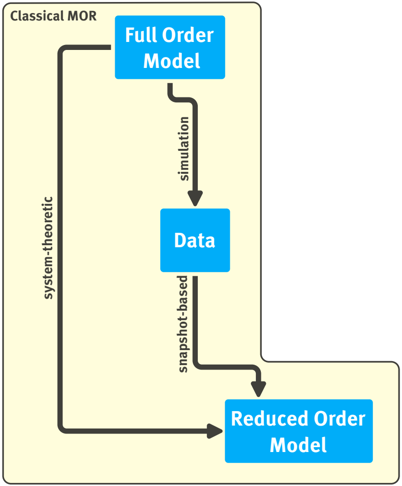
Workshop on Open Source Software and Granular Matter

Enschede, June 1, 2023

living.knowledge

Mathematics Münster
Cluster of Excellence

# What is Model Order Reduction?

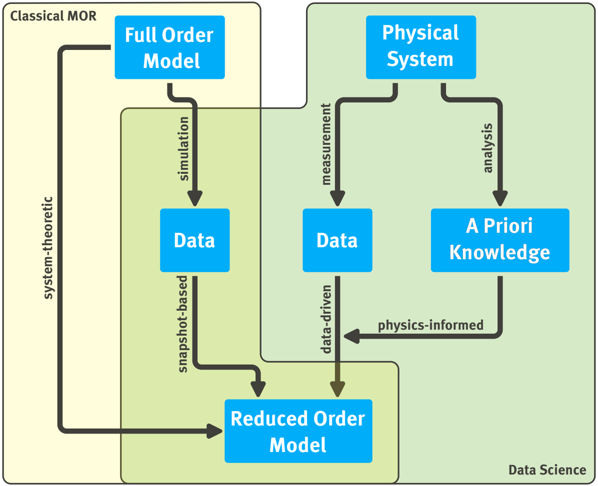



**Compute** computationally efficient surrogate models for

- 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.

# System-Theoretic Model Order Reduction

## Linear time invariant system (full order model)

Input $u(t) \in \mathbb{R}^m$ to state $x(t) \in \mathbb{R}^n$ to output $y(t) \in \mathbb{R}^p$ mapping is given by

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t). \end{aligned}$$

# System-Theoretic Model Order Reduction

## Linear time invariant system (full order model)

Input $u(t) \in \mathbb{R}^m$ to state $x(t) \in \mathbb{R}^n$ to output $y(t) \in \mathbb{R}^p$ mapping is given by

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t). \end{aligned}$$

## Model Reduction Magic

Compute 'good' $W, V \in \mathbb{R}^{n\times r}$, $r \ll n$, s.t. $W^T V = I$

## Linear time invariant system (reduced order model)

Input $u(t) \in \mathbb{R}^m$ to state $x(t) \in \mathbb{R}^r$ to output $y(t) \in \mathbb{R}^p$ mapping is given by

$$\begin{aligned} \dot{x}(t) &= (W^T A V)\,x(t) + (W^T B)\,u(t) \\ y(t) &= (CV)\,x(t). \end{aligned}$$

# System-Theoretic MOR – Computing $V$ and $W$

## Balanced Truncation

- Compute reachability Gramian $\mathcal{P}$ and observability Gramian $\mathcal{Q}$ subject to

$$A\mathcal{P} + \mathcal{P}A^T + BB^T = 0$$
$$A^T\mathcal{Q} + \mathcal{Q}A + C^TC = 0.$$

- Simultaneously diagonalize $\mathcal{P}$ and $\mathcal{Q}$.
- Select $V$, $W$ by truncating states which are both hard to reach and hard to observe.

# System-Theoretic MOR – Computing $V$ and $W$

## Balanced Truncation

- Compute reachability Gramian $\mathcal{P}$ and observability Gramian $\mathcal{Q}$ subject to

$$A\mathcal{P} + \mathcal{P}A^T + BB^T = 0$$
$$A^T\mathcal{Q} + \mathcal{Q}A + C^TC = 0.$$

- Simultaneously diagonalize $\mathcal{P}$ and $\mathcal{Q}$.
- Select $V$, $W$ by truncating states which are both hard to reach and hard to observe.

## Rational interpolation

- Construct $V$, $W$, s.t. the transfer function (Laplace transform of impulse response)

$$H(s) = C(sI - A)^{-1}B$$

  is interpolated (including higher moments) at points $s_1, \dots, s_k$ by a rational function.
- Methods: Moment matching, Padé approximation, IRKA, ...
- Computed using rational Krylov methods.

# Reduced Basis Methods (easiest case)

## Parametric linear parabolic problem (full order model)

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

$$\begin{aligned} u_\mu(x, 0) &= u_0(x), \\ \partial_t u_\mu(x,t) - \nabla \cdot (\sigma_\mu(x) \nabla u_\mu(x,t)) &= f(x), \\ y_\mu(t) &= g(u_\mu(\cdot, t)). \end{aligned}$$

# Reduced Basis Methods (easiest case)

## Parametric linear parabolic problem (full order model)

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

$$\begin{aligned} u_\mu(0) &= u_0, \\ \langle v, \partial_t u_\mu(t)\rangle + b_\mu(v, u_\mu(t)) &= \ell(v) \qquad\qquad \forall v \in V_h, \\ y_\mu(t) &= g(u_\mu(t)). \end{aligned}$$

# Reduced Basis Methods (easiest case)

## Parametric linear parabolic problem (full order model)

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

$$\begin{aligned} u_\mu(0) &= u_0, \\ \langle v, \partial_t u_\mu(t) \rangle + b_\mu(v, u_\mu(t)) &= \ell(v) \qquad \forall v \in V_h, \\ y_\mu(t) &= g(u_\mu(t)). \end{aligned}$$

## Parametric linear parabolic problem (reduced order model)

For given $V_N \subset V_h$, let $u_{\mu,N}(t) \in V_N$ be given by Galerkin proj. onto $V_N$, i.e.

$$\begin{aligned} u_{\mu,N}(0) &= P_{V_N}(u_0), \\ \langle v, \partial_t u_{\mu(t),N} \rangle + b_\mu(v, u_{\mu(t),N}) &= \ell(v) \qquad \forall v \in V_N, \\ y_{\mu,N}(t) &= g(u_{\mu,N}(t)), \end{aligned}$$

where $P_{V_N}\colon V_h \to V_N$ is orthogonal proj. onto $V_N$.

# RB Methods – Computing $V_N$

## Weak greedy basis generation

1: **function** $\textsc{Weak-Greedy}(\mathcal{S}_{train} \subset \mathcal{P}, \varepsilon)$
2: $\quad V_N \leftarrow \{0\}$
3: $\quad$ **while** $\max_{\mu \in \mathcal{S}_{train}} \textsc{Err-Est}(\textsc{ROM-Solve}(\mu), \mu) > \varepsilon$ **do**
4: $\quad\quad \mu^* \leftarrow \operatorname{arg-max}_{\mu \in \mathcal{S}_{train}} \textsc{Err-Est}(\textsc{ROM-Solve}(\mu), \mu)$
5: $\quad\quad V_N \leftarrow \textsc{Basis-Ext}(V_N, \textsc{FOM-Solve}(\mu^*))$
6: $\quad$ **end while**
7: $\quad$ **return** $V_N$
8: **end function**

## Basis-Ext

1. Compute $u_{\mu^*}^{\perp}(t) = (I - P_{V_N})u_{\mu^*}(t)$.
2. Add $\mathrm{POD}(u_{\mu^*}^{\perp}(t))$ to $V_N$ (leading left-singular vectors of snapshot matrix).

## Err-Est

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

# Fancy RB example: MULTIBAT

ulm university universität uulm

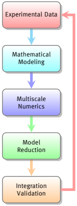
DLR Institute of Technical Thermodynamics

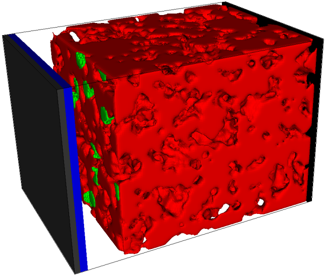
MULTIBAT

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.

- Focus: Li-Plating.
- Li-plating initiated at interface between active particles and electrolyte.
- Need microscale models which resolve active particle geometry.
- Very large nonlinear discrete models.

# MULTIBAT: Some Results

Model:

- Half-cell with plated Li
- $\mu$ = discharge current
- 2.920.000 DOFs

Reduction:

- Snapshots: 3
- $N = 178 + 67$
- $M = 924 + 997$
- Rel. err.: $< 4.5 \cdot 10^{-3}$

Timings:

- Full model: ≈ 15.5h
- Projection: ≈ 14h
- Red. model: ≈ 8m
- Speedup: **120**

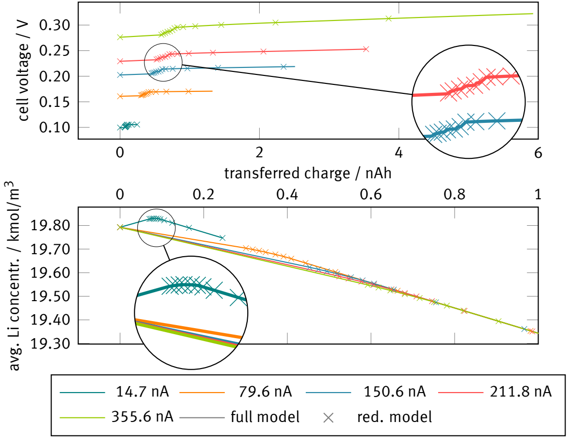



Figure: Validation of reduced order model output for random discharge currents; solid lines: full order model, markers: reduced order model.

# Software in MULTIBAT

Westfälische Wilhelms-Universität Münster

**pyMOR**
Model Order Reduction with Python

pyMOR-BEST interface

BEST

**ITWM BEST Code**
- Micro Li-Ion cell simulation

ulm university universität uulm

Geometry interface/generator

**UU Geometry modeling**

Collaborative
SEI modeling, numeric and implementation interface

**HIU/DLR cell modeling**

HIU Helmholtz Institute Ulm Electrochemical Energy Storage

**Integration Bechmark 1**
- Using all APs
- Using all Interfaces

# pyMOR – Model Order Reduction with Python

## Goal 1

One library for algorithm development *and* large-scale applications.

# pyMOR – Model Order Reduction with Python

## Goal 1

One library for algorithm development *and* large-scale applications.

## Goal 2

Unified view on MOR.

# pyMOR – Model Order Reduction with Python

## Goal 1

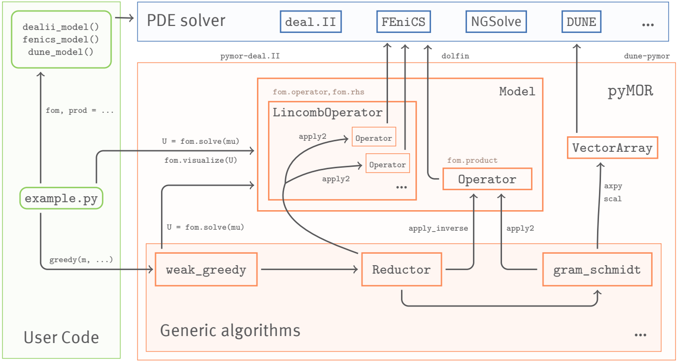
One library for algorithm development *and* large-scale applications.

## Goal 2

Unified view on MOR.

- Started late 2012, 23k lines of Python code, 8k single commits.
- BSD-licensed, fork us on GitHub!
- Quick prototyping with Python 3.
- Comes with small `NumPy`/`SciPy`-based discretization toolkit for getting started quickly.
- Seamless integration with high-performance PDE solvers.

# 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.

# Generic Algorithms

## Models

| | | | |
|---|---|---|---|
| StationaryModel | PHLTIModel | BilinearModel | QuadraticHamiltonianModel |
| InstationaryModel | SecondOrderModel | TransferFunction | LinearStochasticModel |
| LTIModel | LinearDelayModel | | |

## Algorithms

| | | | |
|---|---|---|---|
| POD | certified RB | parametric PG projection | balanced truncation |
| PSD 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 |

# pyMOR Community

**Learn:**

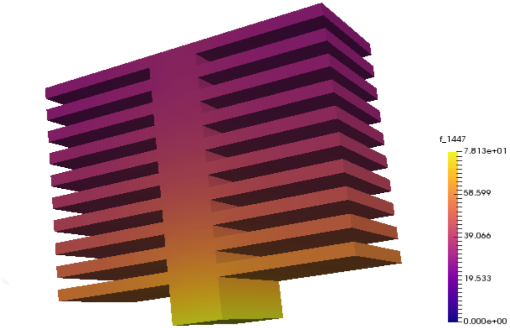
- Yearly pyMOR school (`school.pymor.org`).
- Ask questions via GitHub Discussions.
- Regular community meetings (BigBlueButton).

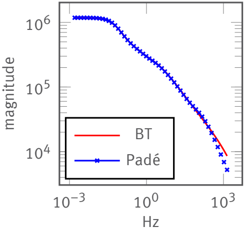
**Contribute:**

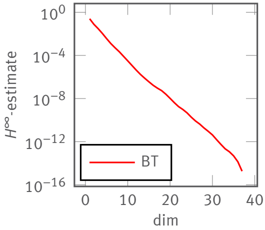
- Fix 'good first issue'.
- Get attribution via `AUTHORS.md`.
- Become contributor with push access to feature branches.
- Become main developer with full control over project.

# Example: System-Theoretic MOR with FEniCS

- MPI distributed heatsink model with FEniCS.
- Heat conduction with Robin boundary.
- Input: heat flow at base
- Output: temperature at base
- MOR: Balanced truncation and Padé approximation



# Example: System-Theoretic MOR with FEniCS

## Model assembly with FEniCS

```
def discretize():
    domain = ...
    mesh = ms.generate_mesh(domain, RESOLUTION)
    subdomain_data = ...

    V = df.FunctionSpace(mesh, 'P', 1)
    u = df.TrialFunction(V)
    v = df.TestFunction(V)
    ds = df.Measure('ds', domain=mesh, subdomain_data=boundary_markers)

    A = df.assemble(- df.Constant(100.) * df.inner(df.grad(u), df.grad(v)) * df.dx
    - df.Constant(0.1) * u * v * ds(1))
    B = df.assemble(df.Constant(1000.) * v * ds(2))
    E = df.assemble(u * v * df.dx)
```

# Example: System-Theoretic MOR with FEniCS

## pyMOR wrapping

```
# def discretize (cont.)

    space = FenicsVectorSpace(V)
    A = FenicsMatrixOperator(A, V, V)
    B = VectorOperator(space.make_array([B]))
    C = B.H
    E = FenicsMatrixOperator(E, V, V)
    fom = LTIModel(A, B, C, None, E)
    return fom
```

# Example: System-Theoretic MOR with FEniCS

## pyMOR wrapping

```
# def discretize (cont.)

    space = FenicsVectorSpace(V)
    A = FenicsMatrixOperator(A, V, V)
    B = VectorOperator(space.make_array([B]))
    C = B.H
    E = FenicsMatrixOperator(E, V, V)
    fom = LTIModel(A, B, C, None, E)
    return fom
```

## MPI wrapping

```
from pymor.tools import mpi
if mpi.parallel:
    from pymor.models.mpi import mpi_wrap_model
    fom = mpi_wrap_model(discretize, use_with=True)
else:
    fom = discretize()
```

# Example: System-Theoretic MOR with FEniCS

## Balanced Truncation

```
reductor = BTReductor(fom)
bt_rom = reductor.reduce(10)

bt_rom.mag_plot(np.logspace(-2, 4, 100), Hz=True)
```

## Padé approximation

```
k = 10
V = rational_arnoldi(fom.A, fom.E, fom.B, [0] * r)
W = rational_arnoldi(fom.A, fom.E, fom.C, [0] * r, trans=True)
pade_rom = LTIPGReductor(fom, W, V, False).reduce()

pade_rom.mag_plot(np.logspace(-2, 4, 100), Hz=True)
```

# Thank you for your attention!

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

System-theoretic model order reduction with pyMOR
PAMM 19, 2019.

Parametric model order reduction using pyMOR
Proceedings of MODRED 2019, Springer, 2020.

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.