Westfälische
Wilhelms-Universität
Münster

POD-DEIM Model Order Reduction for Large Nonlinear Problems

Christian Himpe, Tobias Leibner and Stephan Rave

Séminaire d'analyse numérique
Université de Genève

Genève
April 30, 2019

Outline

Reduced Basis Methods and POD

Reduced Basis Methods

Parametric linear parabolic problem (full order model)

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

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

Reduced Basis Methods

Parametric linear parabolic problem (full order model)

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

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

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.

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

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

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**
    $\mu^* \leftarrow \operatorname{arg-max}_{\mu \in \mathcal{S}_{train}} \textsc{Err-Est}(\textsc{ROM-Solve}(\mu), \mu)$
    $V_N \leftarrow \textsc{Basis-Ext}(V_N, \textsc{FOM-Solve}(\mu^*))$
  end while
  return V_N
end function

BASIS-EXT

1. Compute $u_{\mu^*}^{\perp}(t) = (I - P_{V_N})u_{\mu^*}(t)$.
2. Add $\text{POD}(u_{\mu^*}^{\perp}(t))$ 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).

Stephan Rave (stephan.rave@wwu.de)

RB Methods – Online Efficiency

Parametric linear parabolic problem (reduced order model)

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

Affine decomposition

Assume that b_μ can be written as

$$
b_\mu(v, u) = \sum_{q=1}^{Q} \theta_q(\mu) b_q(v, u).
$$

RB Methods – Online Efficiency

Parametric linear parabolic problem (reduced order model)

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

Affine decomposition

Assume that b_μ can be written as

$$b_\mu(v, u) = \sum_{q=1}^{Q} \theta_q(\mu) b_q(v, u).$$

Offline/Online splitting

By pre-computing

$$\langle \varphi_i, \varphi_j \rangle, \; b_q(\varphi_i, \varphi_j), \; f(\varphi_i), \; g(\varphi_i)$$

for a reduced basis $\varphi_1, \ldots, \varphi_N$ of V_N, solving ROM becomes independent of dim V_h.

RB Methods – Nonlinear Problems

Parametric nonlinear parabolic problem (full order model)

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

where $\mathcal{A}_\mu : V_h \to V_h^*$ is a nonlinear operator.

RB Methods – Nonlinear Problems

Parametric nonlinear parabolic problem (full order model)

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

where $\mathcal{A}_\mu : V_h \to V_h^*$ is a nonlinear operator.

Parametric nonlinear parabolic problem (reduced order model)

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

RB Methods – Nonlinear Problems

Parametric nonlinear parabolic problem (full order model)

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

where $\mathcal{A}_\mu : V_h \to V_h^*$ is a nonlinear operator.

Parametric nonlinear parabolic problem (reduced order model)

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

Problem: No offline/online splitting of nonlinear

$$\mathcal{A}_\mu : V_N \longrightarrow V_h \longrightarrow {V_N}^*.$$

Same problem for non-affinely decomposed b_μ.

RB Methods – Empirical Interpolation

EI – abstract version

Let normed Space V, functionals $\Psi \subseteq V^*$ and training set $\mathcal{M} \subset V$ be given.
Construct via EI-GREEDY algorithm:

1. Interpolation basis $b_1, \ldots b_M \in \operatorname{span} \mathcal{M}$,
2. Interpolation functionals $\psi_1, \ldots, \psi_M \in \Psi$.

The empirical interpolant $\mathcal{I}_M(v)$ of an arbitrary $v \in V$ is then determined by

$$\mathcal{I}_M(v) \in \operatorname{span}\{b_1, \ldots, b_M\} \quad \text{and} \quad \psi_m(\mathcal{I}_M(v)) = \psi_m(v) \quad 1 \leq m \leq M.$$

EI Cheat Sheet

	V	Ψ	online
function EI	function space	point evaluations	evaluation at 'magic points'
operator EI	range of (discrete) operator	DOFs	local evaluation at selected DOFs
matrix DEIM	matrices of given shape	matrix entries	assembly of selected entries

RB Methods – Hyper-Reduction

Reduced Order Model (with EI)

Find $u_{\mu,N} \in V_N$ s.t.

$$\langle v, \partial_t u_{\mu,N}(t)\rangle + \langle v, \{I_M \circ \mathcal{A}_{M,\mu} \circ R_{M'}\}(u_{\mu,N}(t))\rangle = f(v) \quad \forall v \in V_N,$$

where

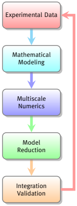
$R_{M'}\colon V_h \to \mathbb{R}^{M'}$	restriction to M' DOFs needed for local evaluation
$\mathcal{A}_{M,\mu}\colon \mathbb{R}^{M'} \to \mathbb{R}^{M}$	local evaluation of $\mathcal{A}_\mu$ at M interpolation DOFs
$I_M\colon \mathbb{R}^{M} \to V_h^*$	linear interpolation operator

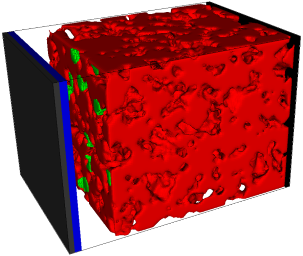
Offline/Online splitting

- Pre-compute the linear operators $\langle \cdot, I_M(\cdot)\rangle$ and $R_{M'}$ w.r.t. basis of V_N.
- Effort to evaluate $\langle \cdot, I_M \circ \mathcal{A}_{M,\mu} \circ R_{M'}(\cdot)\rangle$ w.r.t. this basis:

$$\mathcal{O}(MN) + \mathcal{O}(M) + \mathcal{O}(MN).$$

MULTIBAT: RB Approximation of Li-Ion Battery Models





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: Numerical Results

Model:

- Half-cell with plated Li
- μ = 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: $\approx$ 15.5h
- Projection: $\approx$ 14h
- Red. model: $\approx$ 8m
- Speedup: **120**

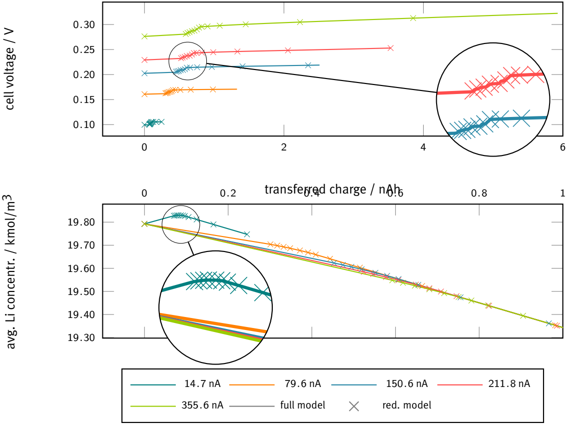
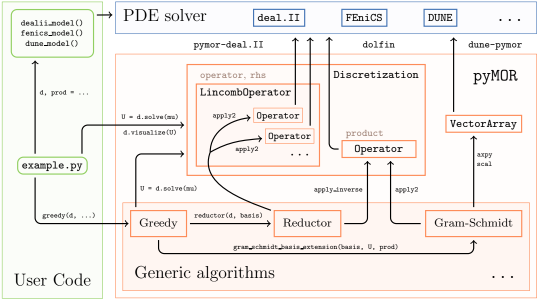


Figure: Validation of reduced order model output for random discharge currents; solid lines: full order model, markers: reduced order model.

pyMOR – Model Order Reduction with Python



- Quick prototyping with Python.
- Seamless integration with high-performance PDE solvers.
- Out of box MPI support for reduction algs. and PDE solvers.
- BSD-licensed, fork us on Github!

pyMOR School

October 7-11, 2019

MPI Magdeburg

https://school.pymor.org

HAPOD – Hierarchical Approximate POD

Computing V_N with POD

Offline phase

Basis for V_N is computed from **solution snapshots** $u_{\mu_s}(t)$ of full order problem via:

- Proper Orthogonal Decomposition (POD)
- POD-Greedy (= greedy search in μ + POD in t)

Computing V_N with POD

Offline phase

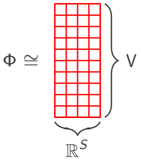
Basis for V_N is computed from **solution snapshots** $u_{\mu_s}(t)$ of full order problem via:

- Proper Orthogonal Decomposition (POD)
- POD-Greedy (= greedy search in μ + POD in t)

POD (a.k.a. PCA, Karhunen–Loève decomposition)

Given Hilbert space V, $\mathcal{S} := \{v_1, \ldots, v_S\} \subset V$, the k-th POD mode of $\mathcal{S}$ is the k-th left-singular vector of the mapping

$$\Phi : \mathbb{R}^S \to V, \quad e_s \to \Phi(e_s) := v_s$$

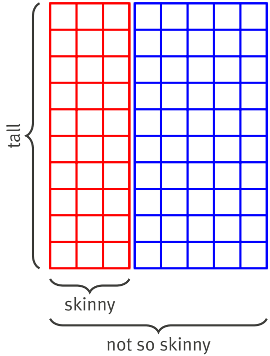


Optimality of POD

Let V_N be the linear span of first N POD modes, then:

$$\sum_{s\in\mathcal{S}} \|s - P_{V_N}(s)\|^2 = \sum_{m=N+1}^{|\mathcal{S}|} \sigma_m^2 = \min_{\substack{X\subset V \\ \dim X \leq N}} \sum_{s\in\mathcal{S}} \|s - P_X(s)\|^2$$

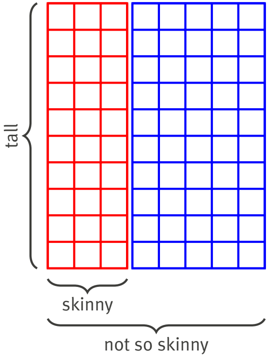
Are your tall and skinny matrices not so skinny anymore?



POD of large snapshot sets:

- large computational effort
- parallelization?
- data $>$ RAM $\Longrightarrow$ disaster

Are your tall and skinny matrices not so skinny anymore?



POD of large snapshot sets:

- large computational effort
- parallelization?
- data $>$ RAM $\Longrightarrow$ disaster

Solution: PODs of PODs!

Disclaimer

- You might have done this before.

Disclaimer

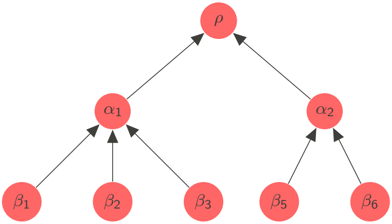
- You might have done this before.

- Others have done it before – often well-hidden in a paper on entirely different topic.
  We are aware of:
  [Qu, Ostrouchov, Samatova, Geist, 2002], [Paul-Dubois-Taine, Amsallem, 2015],
  [Brands, Mergheim, Steinmann, 2016], [Iwen, Ong, 2017].

Disclaimer

- You might have done this before.
- Others have done it before – often well-hidden in a paper on entirely different topic.
  We are aware of:
  [Qu, Ostrouchov, Samatova, Geist, 2002], [Paul-Dubois-Taine, Amsallem, 2015],
  [Brands, Mergheim, Steinmann, 2016], [Iwen, Ong, 2017].
- Our contributions:
  1. Formalization for arbitrary trees of worker nodes.
  2. Extensive theoretical error and performance analysis.
  3. A recipe for selecting local truncation thresholds.
  4. Extensive numerical experiments for different application scenarios.
- Can be trivially extended to low-rank approximation of snapshot matrix by keeping track of right-singular vectors.

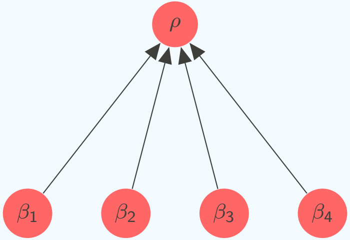
HAPOD – Hierarchical Approximate POD



- Input: Assign snapshot vectors to leaf nodes β_i as input.
- At each node α:
    1. Perform POD of input vectors with given local ℓ^2-error tolerance $\varepsilon(\alpha)$.
    2. Scale POD modes by singular values.
    3. Send scaled modes to parent node as input.
- Output: POD modes at root node ρ.

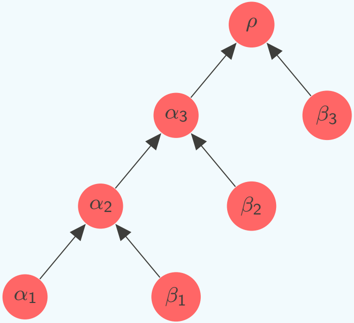
HAPOD – Special Cases

Distributed HAPOD



- Distributed, communication avoiding POD computation.

Incremental HAPOD



- On-the-fly compression of large trajectories.

HAPOD – Some Notation

Trees

$\mathcal{T}$	the tree
$\rho_{\mathcal{T}}$	root node
$\mathcal{N}_{\mathcal{T}}(\alpha)$	nodes of $\mathcal{T}$ below or equal node α
$\mathcal{L}_{\mathcal{T}}$	leafs of $\mathcal{T}$
$L_{\mathcal{T}}$	depth of $\mathcal{T}$

HAPOD

$\mathcal{S}$	snapshot set
$D : \mathcal{S} \to \mathcal{L}_{\mathcal{T}}$	snapshot to leaf assignment
$\varepsilon(\alpha)$	error tolerance at α
$\lvert \mathrm{HAPOD}[\mathcal{S}, \mathcal{T}, D, \varepsilon](\alpha) \rvert$	number of HAPOD modes at α
$\lvert \mathrm{POD}(\mathcal{S}, \varepsilon) \rvert$	number of POD modes for error tolerance ε
P_α	orth. proj. onto HAPOD modes at α
$\widetilde{\mathcal{S}}_\alpha$	snapshots at leafs below α

HAPOD – Theoretical Analysis

Theorem (Error bound[1])

$$\sum_{s \in \widetilde{\mathcal{S}}_\alpha} \|s - P_\alpha(s)\|^2 \leq \sum_{\gamma \in \mathcal{N}_\mathcal{T}(\alpha)} \varepsilon(\gamma)^2.$$

[1] For special cases in appendix of [Paul-Dubois-Taine, Amsallem, 2015].

HAPOD – Theoretical Analysis

Theorem (Error bound[1])

$$\sum_{s \in \widetilde{\mathcal{S}}_\alpha} \|s - P_\alpha(s)\|^2 \leq \sum_{\gamma \in \mathcal{N}_\mathcal{T}(\alpha)} \varepsilon(\gamma)^2.$$

Theorem (Mode bound)

$$\Big|\mathrm{HAPOD}[\mathcal{S}, \mathcal{T}, D, \varepsilon](\alpha)\Big| \leq \Big|\mathrm{POD}\Big(\widetilde{\mathcal{S}}_\alpha, \varepsilon(\alpha)\Big)\Big|.$$

[1] For special cases in appendix of [Paul-Dubois-Taine, Amsallem, 2015].

HAPOD – Theoretical Analysis

Theorem (Error bound[1])

$$\sum_{s \in \widetilde{\mathcal{S}}_\alpha} \|s - P_\alpha(s)\|^2 \leq \sum_{\gamma \in \mathcal{N}_\mathcal{T}(\alpha)} \varepsilon(\gamma)^2.$$

Theorem (Mode bound)

$$\left|\mathrm{HAPOD}[\mathcal{S}, \mathcal{T}, D, \varepsilon](\alpha)\right| \leq \left|\mathrm{POD}\left(\widetilde{\mathcal{S}}_\alpha, \varepsilon(\alpha)\right)\right|.$$

But how to choose ε in practice?

- Prescribe error tolerance ε^* for final HAPOD modes.
- Balance quality of HAPOD space (number of additional modes) and computational efficiency ($\omega \in [0, 1]$).
- Number of input snapshots should be irrelevant for error measure (might be even unknown a priori). Hence, control ℓ^2-*mean* error $\frac{1}{|\mathcal{S}|}\sum_{s \in \mathcal{S}} \|s - P_{\rho_\mathcal{T}}(s)\|^2$.

[1] For special cases in appendix of [Paul-Dubois-Taine, Amsallem, 2015].

HAPOD – Theoretical Analysis

Theorem (ℓ^2-mean error and mode bounds)

Choose local POD error tolerances $\varepsilon(\alpha)$ *for* ℓ^2*-approximation error as:*

$$\varepsilon(\rho_{\mathcal{T}}) := \sqrt{|S|} \cdot \omega \cdot \varepsilon^*, \qquad \varepsilon(\alpha) := \sqrt{\widetilde{\mathcal{S}}_\alpha} \cdot (L_{\mathcal{T}} - 1)^{-1/2} \cdot \sqrt{1 - \omega^2} \cdot \varepsilon^*.$$

Then:

$$\frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \|s - P_{\rho_{\mathcal{T}}}(s)\|^2 \leq {\varepsilon^*}^2 \quad \textit{and} \quad |\operatorname{HAPOD}[\mathcal{S}, \mathcal{T}, D, \varepsilon]| \leq |\overline{\operatorname{POD}}(\mathcal{S}, \omega \cdot \varepsilon^*)|,$$

where $\overline{\operatorname{POD}}(\mathcal{S}, \varepsilon) := \operatorname{POD}(\mathcal{S}, |\mathcal{S}| \cdot \varepsilon)$.

Moreover:

$$|\operatorname{HAPOD}[\mathcal{S}, \mathcal{T}, D, \varepsilon](\alpha)| \leq |\overline{\operatorname{POD}}(\widetilde{\mathcal{S}}_\alpha, (L_{\mathcal{T}} - 1)^{-1/2} \cdot \sqrt{1 - \omega^2} \cdot \epsilon^*)|$$

HAPOD – Theoretical Analysis

Theorem (ℓ^2-mean error and mode bounds)

Choose local POD error tolerances $\varepsilon(\alpha)$ *for* ℓ^2*-approximation error as:*

$$\varepsilon(\rho_{\mathcal{T}}) := \sqrt{|S|} \cdot \omega \cdot \varepsilon^*, \qquad \varepsilon(\alpha) := \sqrt{\widetilde{\mathcal{S}}_\alpha} \cdot (L_{\mathcal{T}} - 1)^{-1/2} \cdot \sqrt{1 - \omega^2} \cdot \varepsilon^*.$$

Then:

$$\frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \|s - P_{\rho_{\mathcal{T}}}(s)\|^2 \leq \varepsilon^{*2} \quad \textit{and} \quad |\operatorname{HAPOD}[\mathcal{S}, \mathcal{T}, D, \varepsilon]| \leq |\overline{\operatorname{POD}}(\mathcal{S}, \omega \cdot \varepsilon^*)|,$$

where $\overline{\operatorname{POD}}(\mathcal{S}, \varepsilon) := \operatorname{POD}(\mathcal{S}, |\mathcal{S}| \cdot \varepsilon)$.

Moreover:

$$\begin{aligned} |\operatorname{HAPOD}[\mathcal{S}, \mathcal{T}, D, \varepsilon](\alpha)| &\leq |\overline{\operatorname{POD}}(\widetilde{\mathcal{S}}_\alpha, (L_{\mathcal{T}} - 1)^{-1/2} \cdot \sqrt{1 - \omega^2} \cdot \epsilon^*)| \\ &\leq \min_{N \in \mathbb{N}}(d_N(\mathcal{S}) \leq (L_{\mathcal{T}} - 1)^{-1/2} \cdot \sqrt{1 - \omega^2} \cdot \epsilon^*). \end{aligned}$$

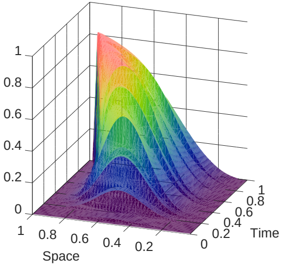
Incremental HAPOD Example

Compress state trajectory of forced inviscid Burgers equation:

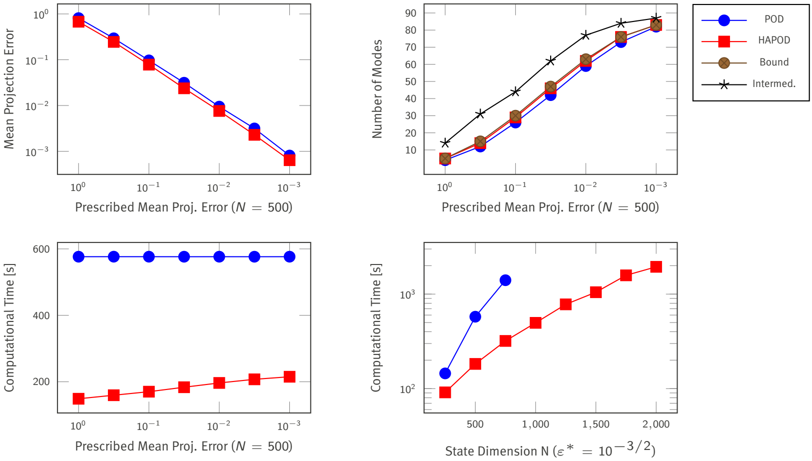
$$\begin{aligned} \partial_t z(x,t) + z(x,t)\cdot\partial_x z(x,t) &= u(t)\exp\left(-\frac{1}{20}(x-\frac{1}{2})^2\right), & (x,t) &\in (0,1)\times(0,1), \\ z(x,0) &= 0, & x &\in [0,1], \\ z(0,t) &= 0, & t &\in [0,1], \end{aligned}$$

where $u(t) \in [0, 1/5]$ iid. for 0.1% random timesteps, otherwise 0.

- Upwind finite difference scheme on uniform mesh with $N = 500$ nodes.
- 10^4 explicit Euler steps.
- 100 sub-PODs, $\omega = 0.75$.
- All computations on Raspberry Pi 1B single board computer (512MB RAM).



Incremental HAPOD Example

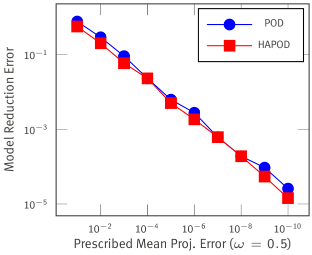


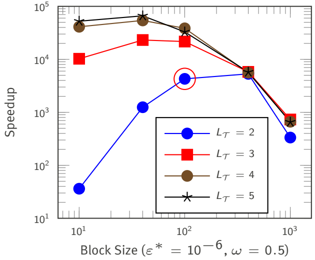
Distributed HAPOD Example

Distributed computation and POD of empirical cross Gramian:

$$\widehat{W}_{X,ij} := \sum_{m=1}^{M} \int_0^\infty \langle x_i^m(t), y_m^j(t) \rangle \, dt \in \mathbb{R}^{N\times N}$$

- ‘Synthetic’ benchmark model[2] from MORWiki with parameter $\theta = \frac{1}{10}$.
- Partition $\widehat{W}_X$ into 100 slices of size 10.000×100.





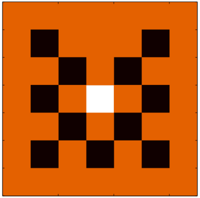
[2] See: http://modelreduction.org/index.php/Synthetic_parametric_model

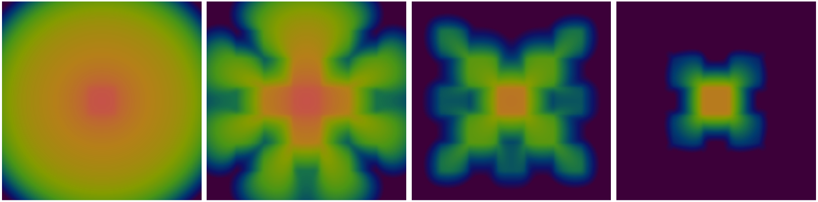
HAPOD – HPC Example

Neutron transport equation

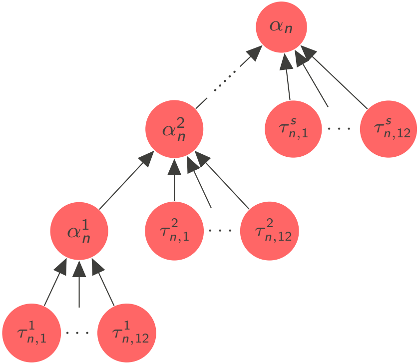
$$\partial_t \psi(t,\mathbf{x},\mathbf{v}) + \mathbf{v}\cdot\nabla_{\mathbf{x}}\psi(t,\mathbf{x},\mathbf{v}) + \sigma_t(\mathbf{x})\psi(t,\mathbf{x},\mathbf{v}) = \frac{1}{|V|}\sigma_s(\mathbf{x})\int_V \psi(t,\mathbf{x},\mathbf{w})\,d\mathbf{w} + Q(\mathbf{x})$$

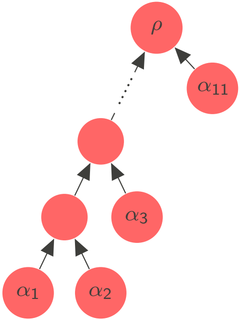
- Moment closure/FV approximation.
- Varying absorbtion and scattering coefficients.
- Distributed snapshot and HAPOD computation on PALMA cluster (125 cores).





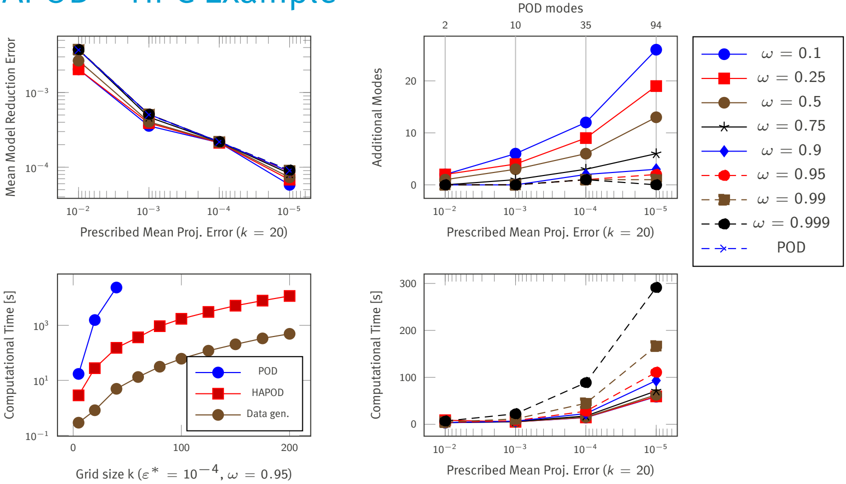
HAPOD – HPC Example





- HAPOD on compute node n. Time steps are split into s slices. Each processor core computes one slice at a time, performs POD and sends resulting modes to main MPI rank on the node.
- Incremental HAPOD is performed on MPI rank 0 with modes collected on each node.

HAPOD – HPC Example



- $\approx 39.000 \cdot k^3$ doubles of snapshot data (≈ 2.5 terabyte for $k = 200$).

What About Nonlinear Problems?

For nonlinear problems, we also need to generate a basis for EI.

What About Nonlinear Problems?

For nonlinear problems, we also need to generate a basis for EI.

- In case of DEIM, EI basis is computed as POD of operator evaluations.

What About Nonlinear Problems?

For nonlinear problems, we also need to generate a basis for EI.

- In case of DEIM, EI basis is computed as POD of operator evaluations.
- $\rightsquigarrow$ Use HAPOD to simultaneously compute RB and DEIM bases.
- Interpolation DOFs are chosen afterwards only using DEIM basis as data (EI-GREEDY).

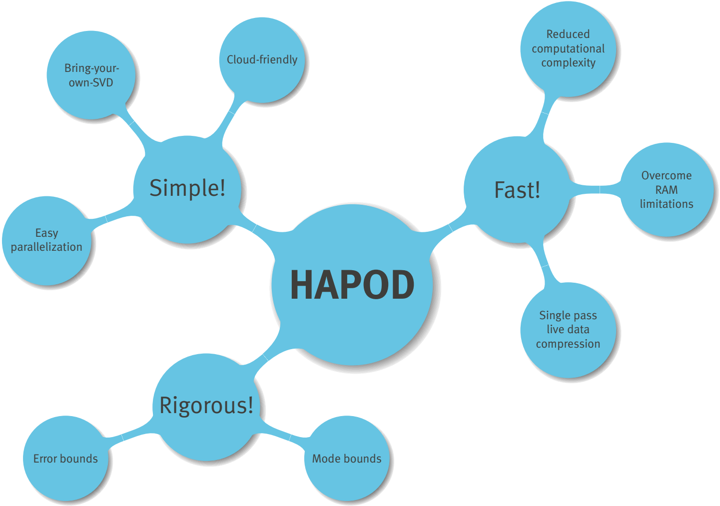
What About Nonlinear Problems?

For nonlinear problems, we also need to generate a basis for EI.

- In case of DEIM, EI basis is computed as POD of operator evaluations.
- $\rightsquigarrow$ Use HAPOD to simultaneously compute RB and DEIM bases.
- Interpolation DOFs are chosen afterwards only using DEIM basis as data (EI-Greedy).

First experiments (3D neutron transport with M_n closure):

ε_{rb}	ε_{ei}	N	M	ROM error	T_{ROM}	T_{FOM}	speedup
$1\cdot10^{-2}$	$1\cdot10^{-2}$	4	1	$4.42\cdot10^{-2}$	0.64	765.47	1,194.58
$1\cdot10^{-2}$	$5\cdot10^{-3}$	4	2	$1.48\cdot10^{-2}$	1.08	764.85	708.47
$1\cdot10^{-2}$	$1\cdot10^{-3}$	4	6	$1.05\cdot10^{-2}$	3.63	765.17	210.51
$1\cdot10^{-3}$	$1\cdot10^{-3}$	16	6	$4.46\cdot10^{-3}$	3.82	763.73	199.98
$1\cdot10^{-3}$	$5\cdot10^{-4}$	16	8	$3.65\cdot10^{-3}$	5.81	764.24	131.57
$1\cdot10^{-3}$	$1\cdot10^{-4}$	16	19	$1.37\cdot10^{-3}$	12.42	763.62	61.47
$1\cdot10^{-4}$	$1\cdot10^{-4}$	43	19	$1.71\cdot10^{-3}$	11.27	766.01	67.95
$1\cdot10^{-4}$	$5\cdot10^{-5}$	43	25	$7.20\cdot10^{-3}$	15.71	765.32	48.73
$1\cdot10^{-4}$	$1\cdot10^{-5}$	43	45	$4.49\cdot10^{-4}$	24.89	765.02	30.73


HAPOD
Simple!
Bring-your-own-SVD
Cloud-friendly
Easy parallelization
Fast!
Reduced computational complexity
Overcome RAM limitations
Single pass live data compression
Rigorous!
Error bounds
Mode bounds

Thank you for your attention!

C. Himpe, T. Leibner, S. Rave, Hierarchical Approximate Proper Orthogonal Decomposition
SIAM J. Sci. Comput., 40(5), pp. A3267-A3292

pyMOR – Generic Algorithms and Interfaces for Model Order Reduction
SIAM J. Sci. Comput., 38(5), pp. S194–S216

```
pip3 install pymor
```

Matlab HAPOD implementation:

```
git clone https://github.com/gramian/hapod
```

My homepage:
`https://stephanrave.de/`

HAPOD vs. Stoachstic SVD

	HAPOD	stoch. SVD
efficient	🙂	🙂
rigorous analysis	🙂	🙂
easy to parallelize	🙂	🙂
low-rank approximation	🙂	🙂
matrix free	🙁	🙂
single-pass	🙂	🙂
single-pass with error control	🙂	🙁
easy to implement	🙂	😐

- HAPOD is a method to efficiently obtain the POD from PODs of subsets of the data.
- HAPOD can be utilized on top of stochastic SVD methods.