Universität
Münster

Hierarchical Approximate SVD

Christian Himpe[1], Tobias Leibner[1], Stephan Rave[1]
[1]University of Münster, Germany

Oberseminar Numerik

Münster, November 22, 2023

RB for Nonlinear Evolution Equations

Full order model

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

$$\partial_t u_\mu(t) + \mathcal{L}_\mu(u_\mu(t)) = 0, \quad u_\mu(0) = u_0,$$

where $\mathcal{L}_\mu\colon \mathcal{P} \times V_h \to V_h$ is a nonlinear finite volume operator.

RB for Nonlinear Evolution Equations

Full order model

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

$$\partial_t u_\mu(t) + \mathcal{L}_\mu(u_\mu(t)) = 0, \quad u_\mu(0) = u_0,$$

where $\mathcal{L}_\mu \colon \mathcal{P} \times V_h \to V_h$ is a nonlinear finite volume operator.

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.

$$\partial_t u_{\mu,N}(t) + P_{V_N}(\mathcal{L}_\mu(u_{\mu,N}(t))) = 0, \quad u_{\mu,N}(0) = P_{V_N}(u_0),$$

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

RB for Nonlinear Evolution Equations

Full order model

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

$$\partial_t u_\mu(t) + \mathcal{L}_\mu(u_\mu(t)) = 0, \quad u_\mu(0) = u_0,$$

where $\mathcal{L}_\mu \colon \mathcal{P} \times V_h \to V_h$ is a nonlinear finite volume operator.

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.

$$\partial_t u_{\mu,N}(t) + P_{V_N}(\mathcal{L}_\mu(u_{\mu,N}(t))) = 0, \quad u_{\mu,N}(0) = P_{V_N}(u_0),$$

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

- Still expensive to evaluate projected operator $P_{V_N} \circ \mathcal{L}_\mu \colon V_N \longrightarrow V_h \longrightarrow V_N$
  $\Longrightarrow$ use hyper-reduction (e.g. empirical interpolation).

Basis Generation

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)

Basis Generation

Offline phase

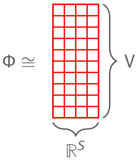
Basis for V_N is computed from **solution snapshots** $u_{\mu_s}(t)$ of full order problem via:

- 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

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



Optimality of POD

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

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

Example: RB Approximation of Li-Ion Battery Models

ulm university universität ulm

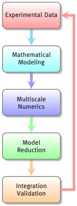
DLR Institute of Technical Thermodynamics

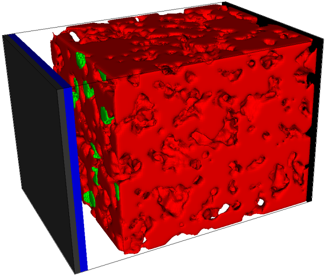
MULTIBAT

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 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

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:

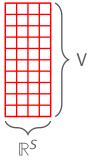
- Proper Orthogonal Decomposition (POD)
- POD-Greedy (= greedy search in μ + POD in t)

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

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

$\Phi \cong$

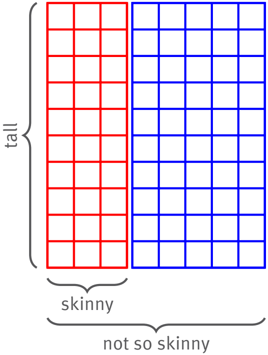


Optimality of POD

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

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

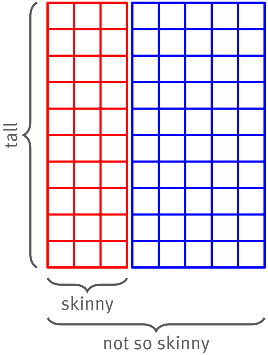
Are your tall and skinny matrices not so skinny anymore?



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.

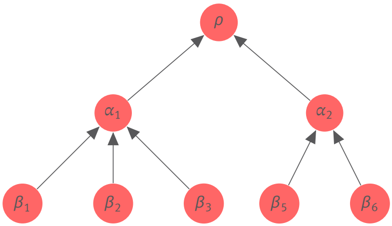
Disclaimer

- 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.

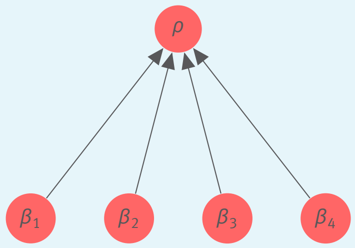
HAPOD – Hierarchical Approximate POD



- 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.

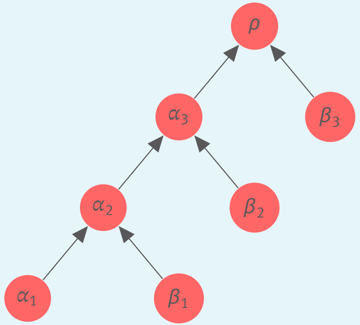
HAPOD – Special Cases

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
$D\colon \mathcal{S} \to \mathcal{L}_{\mathcal{T}}$	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
$\tilde{\mathcal{S}}_\alpha$	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)

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

[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)

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

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:

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

Then:

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

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

Moreover:

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

HAPOD – Theoretical Analysis

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

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

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

Then:

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

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

Moreover:

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

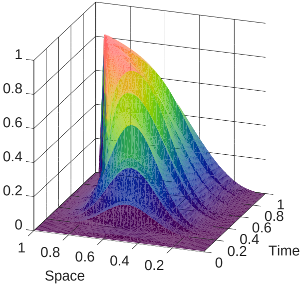
Incremental HAPOD Example

Compress state trajectory of forced inviscid Burgers equation:

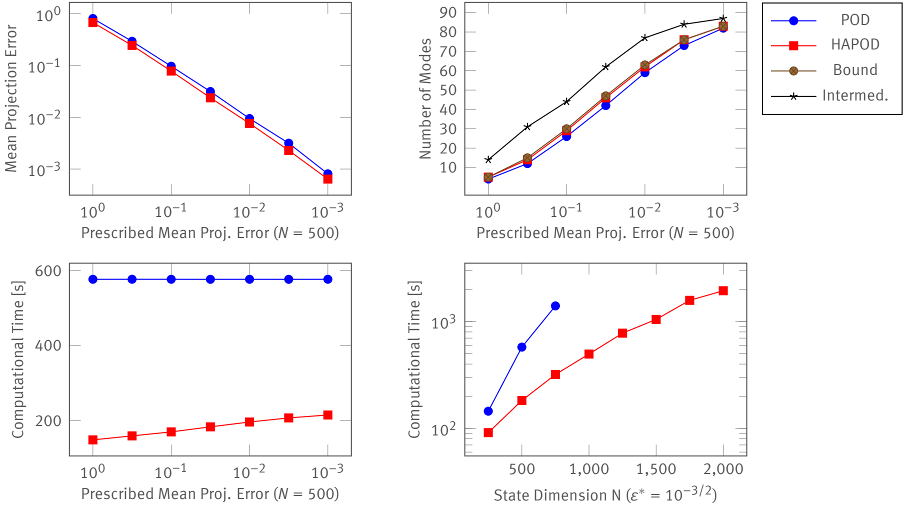
$$\begin{aligned} \partial_t z(x,t) + z(x,t) \cdot \partial_x z(x,t) &= u(t)\exp\left(-\frac{1}{20}\left(x-\frac{1}{2}\right)^2\right), && (x,t) \in (0,1) \times (0,1), \\ z(x,0) &= 0, && x \in [0,1], \\ z(0,t) &= 0, && t \in [0,1], \end{aligned}$$

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

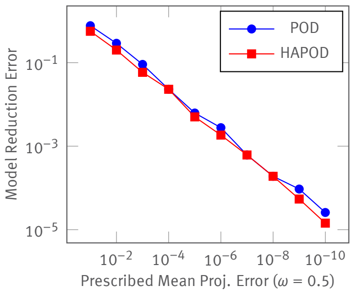


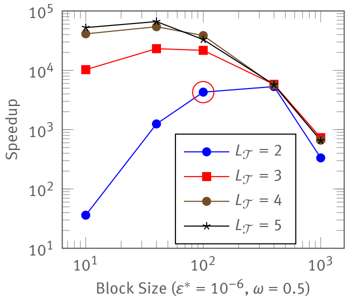
Distributed HAPOD Example

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.



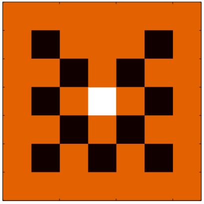


[2]See: http://modelreduction.org/index.php/Synthetic_parametric_model

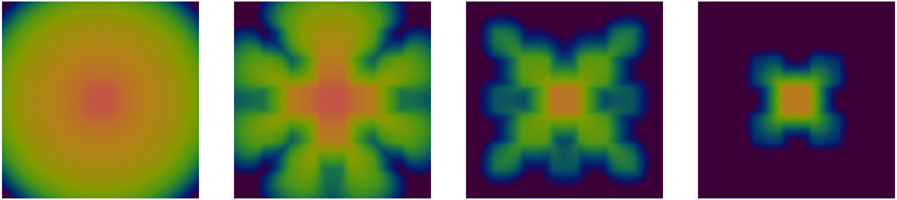
HAPOD – HPC Example

Neutron transport equation

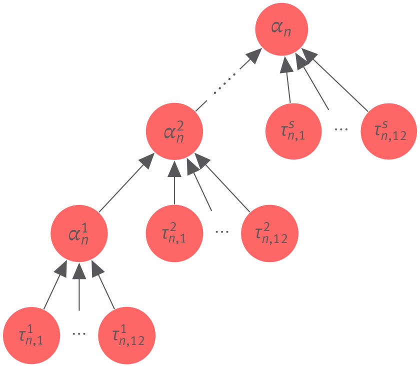
$$\partial_t \psi(t, \mathbf{x}, \mathbf{v}) + \mathbf{v} \cdot \nabla_{\mathbf{x}} \psi(t, \mathbf{x}, \mathbf{v}) + \sigma_t(\mathbf{x}) \psi(t, \mathbf{x}, \mathbf{v}) = \frac{1}{|V|} \sigma_s(\mathbf{x}) \int_V \psi(t, \mathbf{x}, \mathbf{w}) \, \mathrm{d}\mathbf{w} + Q(\mathbf{x})$$

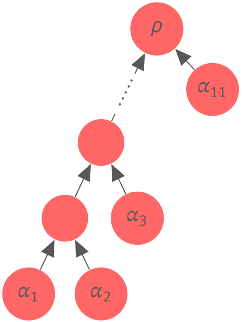


- Moment closure/FV approximation.
- Varying absorbtion and scattering coefficients.
- Distributed snapshot and HAPOD computation on PALMA cluster (125 cores).



HAPOD – HPC Example

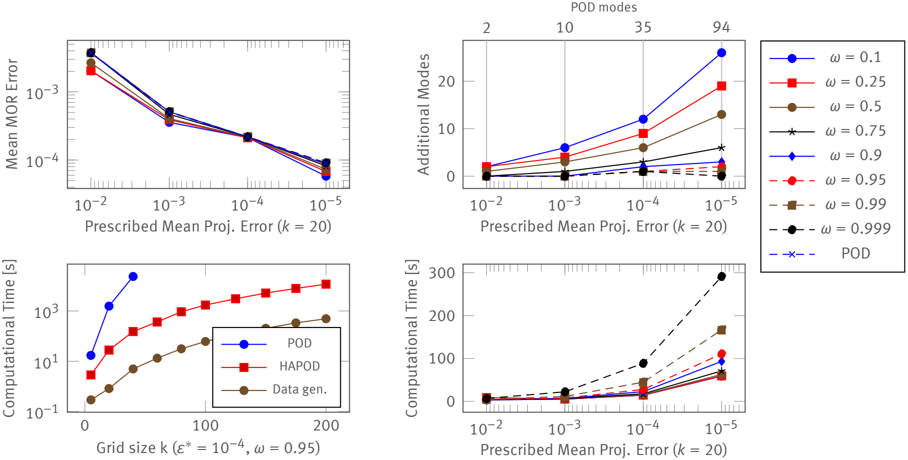




- HAPOD on compute node n. Time steps are split into s slices. Each processor core computes one slice at a time, performs POD and sends resulting modes to main MPI rank on the node.

- 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.

- 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).

Where are my right-singular vectors?!

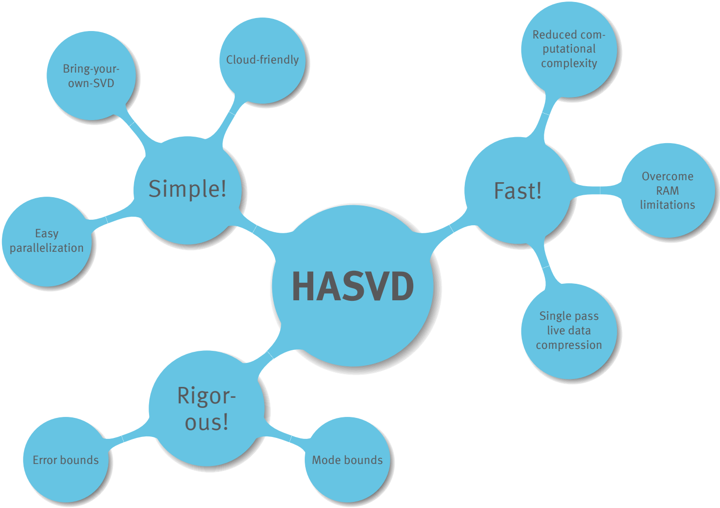
Where are my right-singular vectors?!

At the blackboard!

HASVD vs. Stoachstic SVD

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

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


HASVD
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
`pip install pymor`

Matlab HAPOD implementation:
`git clone https://github.com/gramian/hapod`

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