WWU MÜNSTER

Model Order Reduction of Large-Scale Systems

Linus Balicki, Andreas Buhr, René Fritze, Martin Gander, Petar Mlinaric, Tim Keil, Mario Ohlberger, Stephan Rave, Felix Schindler

IANS Seminar

Stuttgart, June 23, 2021

living.knowledge

MM Mathematics Münster Cluster of Excellence

Outline

1. Introduction to Reduced Basis Methods
2. HAPOD – Hierarchical Approximate POD
3. Localized Reduced Basis Additive Schwarz Methods
4. Two-Scale Reduced Basis Localized Orthogonal Decomposition
5. Model Order Reduction with pyMOR

Not featured:

- Model order reduction of problems with moving shocks/boundaries via nonlinear approximation.

Introduction to Reduced Basis Methods

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.

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

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.

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

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**
        $\mu^* \leftarrow \text{arg-max}_{\mu\in\mathcal{S}_{train}} \textsc{Err-Est}(\textsc{ROM-Solve}(\mu), \mu)$
        $V_N \leftarrow \text{span}(V_N \cup \{\textsc{FOM-Solve}(\mu^*)\})$
    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**
    $\mu^* \leftarrow \text{arg-max}_{\mu \in \mathcal{S}_{train}} \textsc{Err-Est}(\textsc{ROM-Solve}(\mu), \mu)$
    $V_N \leftarrow \text{span}(V_N \cup \{\textsc{FOM-Solve}(\mu^*)\})$
  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 dual weighted residual approach for improved convergence w.r.t to output $y_N(\mu)$.

RB Methods – Online Efficiency

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.

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

Affine decomposition

Assume that a_μ can be written as

$$a(u, v; \mu) = \sum_{q=1}^{Q} \theta_q(\mu) a_q(u, v).$$

RB Methods – Online Efficiency

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.

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

Affine decomposition

Assume that a_μ can be written as

$$a(u, v; \mu) = \sum_{q=1}^{Q} \theta_q(\mu) a_q(u, v).$$

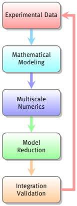
Offline/Online splitting

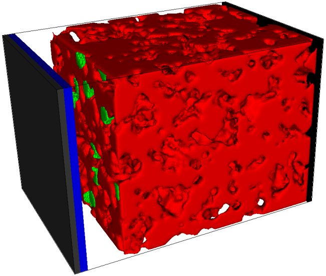
By pre-computing

$$a_q(\varphi_i, \varphi_j),\ f(\varphi_i),\ g(\varphi_i)$$

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

Example: RB Approximation of Li-Ion Battery Models





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

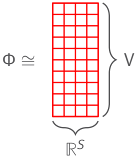
Basis for V_N is computed from **solution snapshots** $u_{\mu_s}(t)$ of full order problem via:

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

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

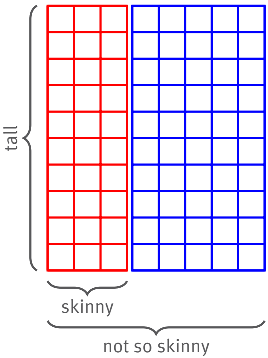


Optimality of POD

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

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

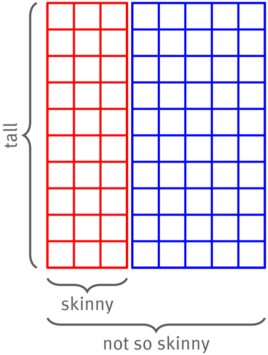
Are your tall and skinny matrices not so skinny anymore?



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.

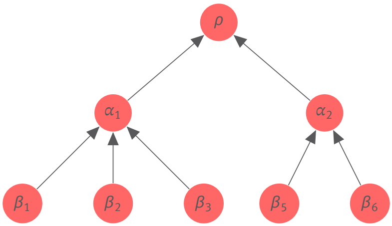
Disclaimer

- 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.

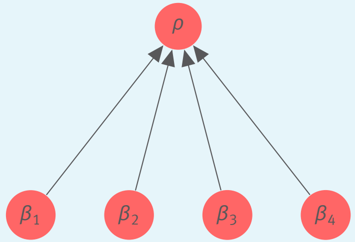
HAPOD – Hierarchical Approximate POD



- 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.

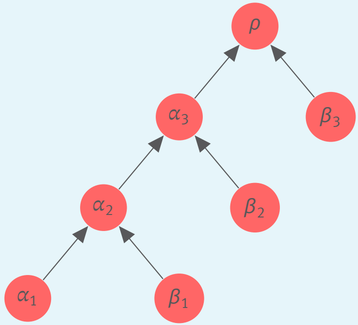
HAPOD – Special Cases

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

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

[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:*

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

Then:

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

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

Moreover:

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

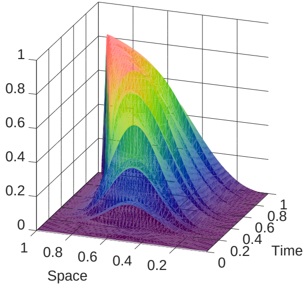
Incremental HAPOD Example

Compress state trajectory of forced inviscid Burgers equation:

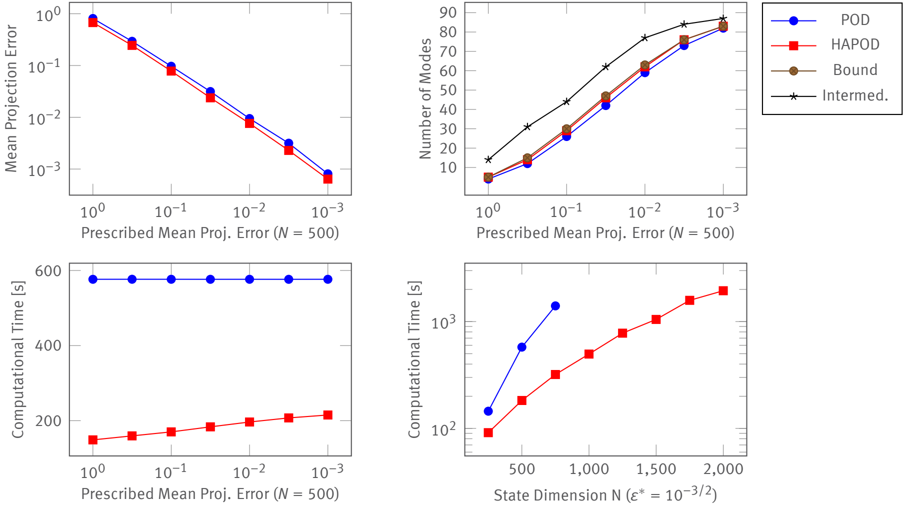
$$\begin{aligned} \partial_t z(x,t) + z(x,t)\cdot\partial_x z(x,t) &= u(t)\exp(-\frac{1}{20}(x-\frac{1}{2})^2), & (x,t) &\in (0,1)\times(0,1),\\ z(x,0) &= 0, & x &\in [0,1],\\ z(0,t) &= 0, & t &\in [0,1], \end{aligned}$$

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

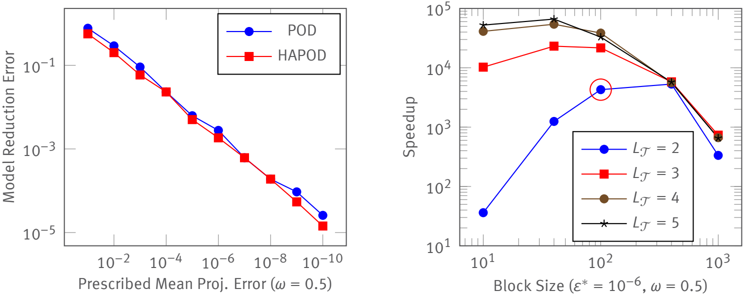


Distributed HAPOD Example

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.



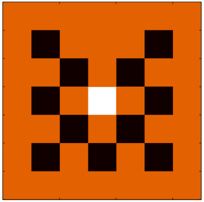
[2] See: http://modelreduction.org/index.php/Synthetic_parametric_model

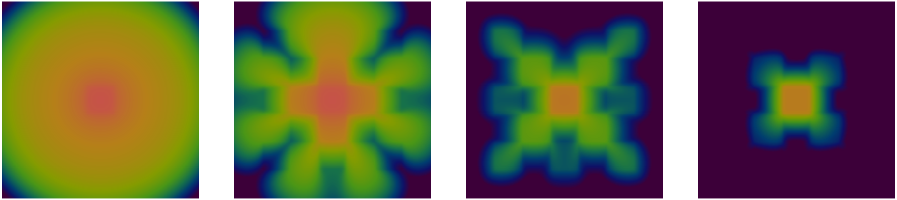
HAPOD – HPC Example

Neutron transport equation

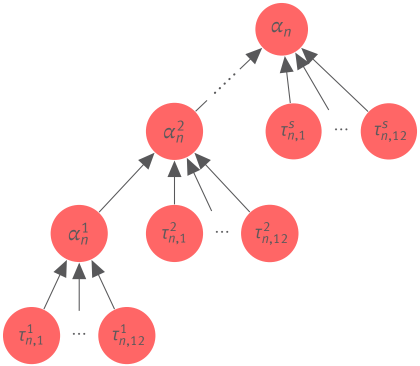
$$\partial_t \psi(t,\mathbf{x},\mathbf{v}) + \mathbf{v}\cdot\nabla_{\mathbf{x}}\psi(t,\mathbf{x},\mathbf{v}) + \sigma_t(\mathbf{x})\psi(t,\mathbf{x},\mathbf{v}) = \frac{1}{|V|}\sigma_s(\mathbf{x})\int_V \psi(t,\mathbf{x},\mathbf{w})\,\mathrm{d}\mathbf{w} + Q(\mathbf{x})$$

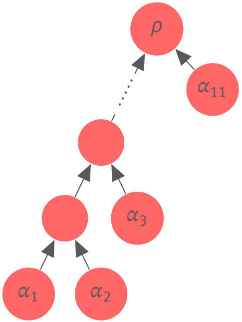
- Moment closure/FV approximation.
- Varying absorbtion and scattering coefficients.
- Distributed snapshot and HAPOD computation on PALMA cluster (125 cores).





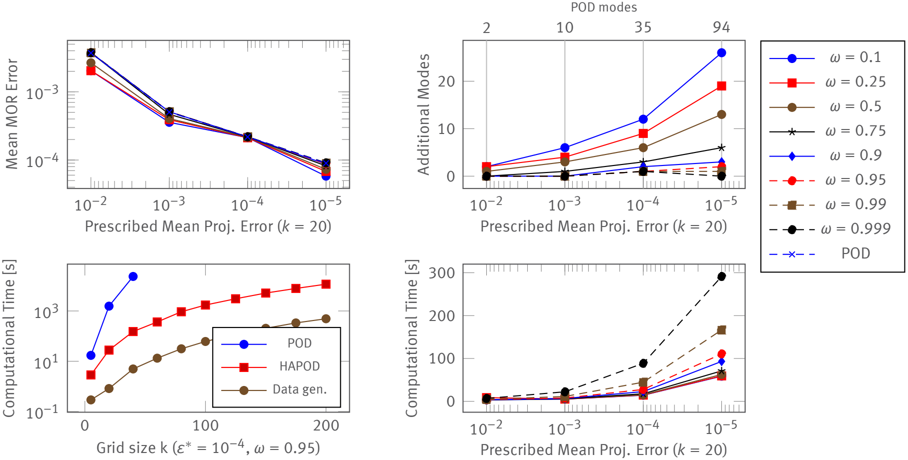
HAPOD – HPC Example





- 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

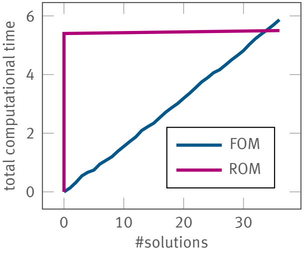


- $\approx 39.000 \cdot k^3$ doubles of snapshot data (≈ 2.5 terabyte for $k = 200$).

Localized Reduced Basis Additive Schwarz Methods

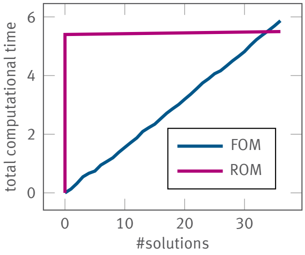
RB Method – Caveats

- Offline time too large in not-so-many-query scenarios?
- $\mathcal{P}$ too large?



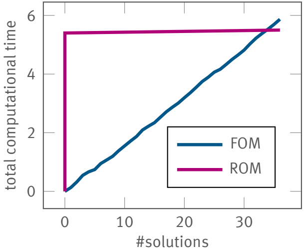
RB Method – Caveats

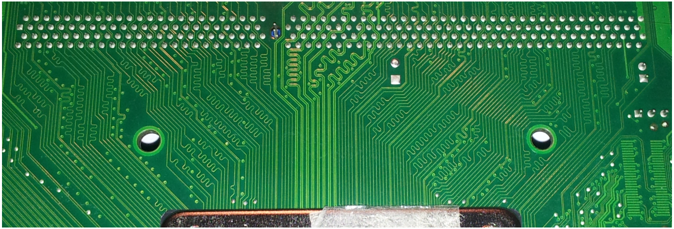
- Offline time too large in not-so-many-query scenarios?
- $\mathcal{P}$ too large?
- Only local influence of μ?



RB Method – Caveats

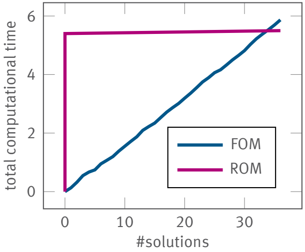
- Offline time too large in not-so-many-query scenarios?
- $\mathcal{P}$ too large?
- Only local influence of μ?
- Local geometry changes?

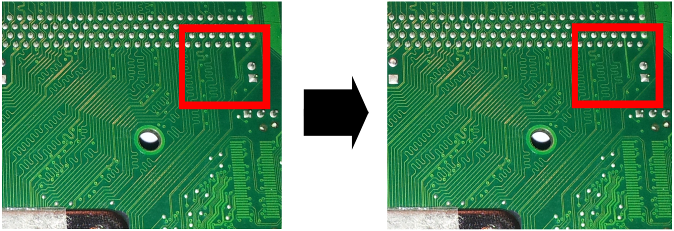




RB Method – Caveats

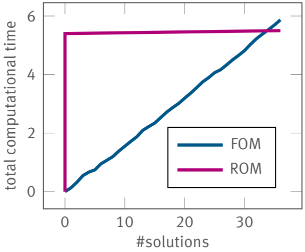
- Offline time too large in not-so-many-query scenarios?
- $\mathcal{P}$ too large?
- Only local influence of μ?
- Local geometry changes?

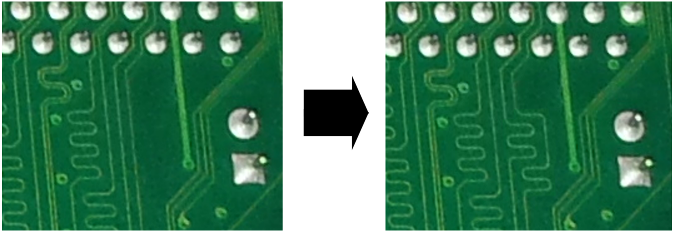




RB Method – Caveats

- Offline time too large in not-so-many-query scenarios?
- $\mathcal{P}$ too large?
- Only local influence of μ?
- Local geometry changes?





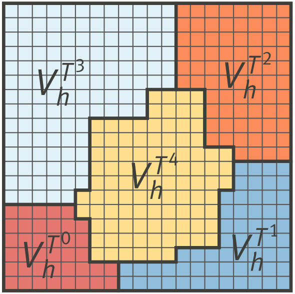
Localized RB Methods for Elliptic Problems

Idea of the **LRBMS**: given a finely-resolved grid τ_h [ALBRECHT ET AL., 2012]

- decompose approximation space into *local* spaces $V_h = \bigoplus_{T \in \mathcal{T}_H} V_h^T$
- associated with subdomains $T \in \mathcal{T}_H$

  independent local discretizations and approximation spaces (CG or DG)

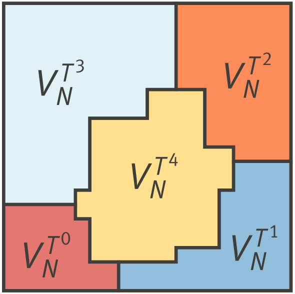
  and global SWIPDG coupling [ERN, STEPHANSEN, ZUNINO, 2009]



Localized RB Methods for Elliptic Problems

Idea of the **LRBMS**: given a finely-resolved grid τ_h [ALBRECHT ET AL., 2012]

- decompose approximation space into *local* spaces $V_h = \bigoplus_{T \in \mathcal{T}_H} V_h^T$
- associated with subdomains $T \in \mathcal{T}_H$
  independent local discretizations and approximation spaces (CG or DG)
  and global SWIPDG coupling [ERN, STEPHANSEN, ZUNINO, 2009]
- build local reduced spaces $V_N^T \subset V_h^T$
- reduced *broken* space $V_N = \bigoplus_{T \in \mathcal{T}_H} V_N^T$

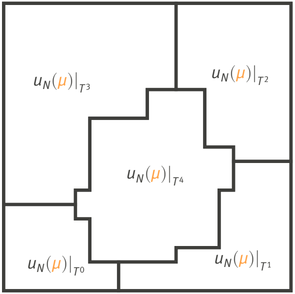


Localized RB Methods for Elliptic Problems

Idea of the **LRBMS**: given a finely-resolved grid τ_h [Albrecht et al., 2012]

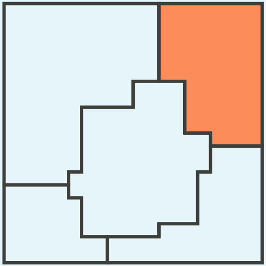
- decompose approximation space into *local* spaces $V_h = \oplus_{T \in \mathcal{T}_H} V_h^T$
- associated with subdomains $T \in \mathcal{T}_H$
  independent local discretizations and approximation spaces (CG or DG)
  and global SWIPDG coupling [Ern, Stephansen, Zunino, 2009]
- build local reduced spaces $V_N^T \subset V_h^T$
- reduced *broken* space $V_N = \oplus_{T \in \mathcal{T}_H} V_N^T$

- larger V_N, but sparse ROM system matrices
- initialization of V_N^T:
  - empty
  - global solution snapshots
  - **local training**



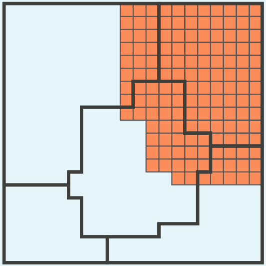
Offline Initialization of V_N

Training algorithm (adapted from [Buhr, Engwer, Ohlberger, R, 2017]) for all $T \in \mathcal{T}_H$



Offline Initialization of V_N

Training algorithm (adapted from [Buhr, Engwer, Ohlberger, R, 2017]) for all $T \in \mathcal{T}_H$



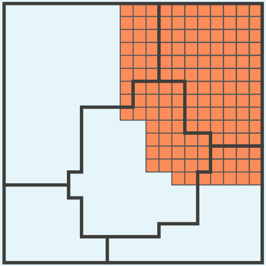
- For every $\mu \in \mathcal{S}_{train} \subset \mathcal{P}$:
  - Solve training problem on oversampling subdomain $T^\delta \supset T$:

$$a(\varphi_{h,0}(\mu), v_h; \mu) = f(v_h) \quad \text{in } T^\delta$$

$$\varphi_{h,0}(\mu) = 0 \quad \text{on } \partial T^\delta$$

Offline Initialization of V_N

Training algorithm (adapted from [Buhr, Engwer, Ohlberger, R, 2017]) for all $T \in \mathcal{T}_H$

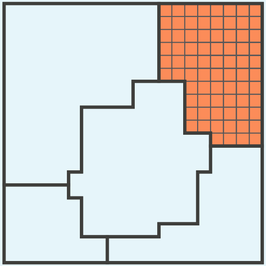


- For every $\mu \in \mathcal{S}_{train} \subset \mathcal{P}$:
  - Solve training problem on oversampling subdomain $T^\delta \supset T$:
    $$a(\varphi_{h,0}(\mu), v_h; \mu) = f(v_h) \qquad \text{in } T^\delta$$
    $$\varphi_{h,0}(\mu) = 0 \qquad \text{on } \partial T^\delta$$
  - For $1 \leq k \leq K$, solve training problem:
    $$a(\varphi_{h,k}(\mu), v_h; \mu) = 0 \qquad \text{in } T^\delta$$
    $$\varphi_{h,k}(\mu) = g_k \qquad \text{on } \partial T^\delta$$

    for K random Dirichlet data functions g_k on ∂T^δ.

Offline Initialization of V_N

Training algorithm (adapted from [Buhr, Engwer, Ohlberger, R, 2017]) — for all $T \in \mathcal{T}_H$



- For every $\mu \in \mathcal{S}_{train} \subset \mathcal{P}$:
  - Solve training problem on oversampling subdomain $T^\delta \supset T$:

$$a(\varphi_{h,0}(\mu), v_h; \mu) = f(v_h) \quad \text{in } T^\delta$$
$$\varphi_{h,0}(\mu) = 0 \quad \text{on } \partial T^\delta$$

  - For $1 \leq k \leq K$, solve training problem:

$$a(\varphi_{h,k}(\mu), v_h; \mu) = 0 \quad \text{in } T^\delta$$
$$\varphi_{h,k}(\mu) = g_k \quad \text{on } \partial T^\delta$$

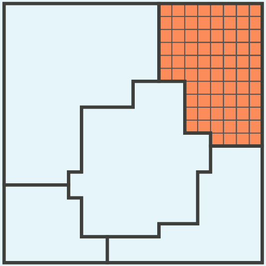
    for K random Dirichlet data functions g_k on ∂T^δ.

- Initialize local RB space on T as

$$V_N^T := \operatorname{span} \bigcup_{\mu \in \mathcal{S}_{train}} \left\{ \varphi_{h,0}(\mu)\big|_T, \ldots, \varphi_{h,K}(\mu)\big|_T \right\}.$$

Offline Initialization of V_N

Training algorithm (adapted from [Buhr, Engwer, Ohlberger, R, 2017]) for all $T \in \mathcal{T}_H$



- For every $\mu \in \mathcal{S}_{train} \subset \mathcal{P}$:
  - Solve training problem on oversampling subdomain $T^\delta \supset T$:

$$a(\varphi_{h,0}(\mu), v_h; \mu) = f(v_h) \quad \text{in } T^\delta$$
$$\varphi_{h,0}(\mu) = 0 \quad \text{on } \partial T^\delta$$

  - For $1 \leq k \leq K$, solve training problem:

$$a(\varphi_{h,k}(\mu), v_h; \mu) = 0 \quad \text{in } T^\delta$$
$$\varphi_{h,k}(\mu) = g_k \quad \text{on } \partial T^\delta$$

    for K random Dirichlet data functions g_k on ∂T^δ.

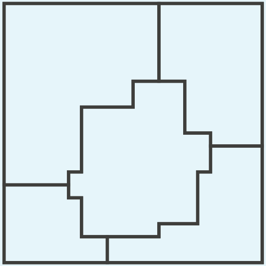
- Initialize local RB space on T as

$$V_N^T := \operatorname{span} \bigcup_{\mu \in \mathcal{S}_{train}} \{ \varphi_{h,0}(\mu)|_T, \dots, \varphi_{h,K}(\mu)|_T \}.$$

- Use greedy algorithm for large $\mathcal{S}_{train}$.

Online-Adaptive Enrichment of V_N

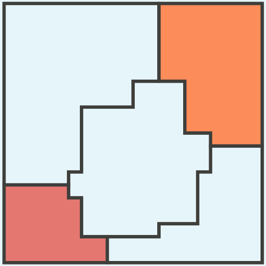
Enrichment algorithm for some $\mu \in \mathcal{P}$



- compute reduced solution $u_N(\mu)$
- estimate error $\eta_{h,N}(\mu)$
- if $\eta_{h,N}(\mu) > \Delta$, start intermediate local enrichment phase:
  - compute local error indicators

Online-Adaptive Enrichment of V_N

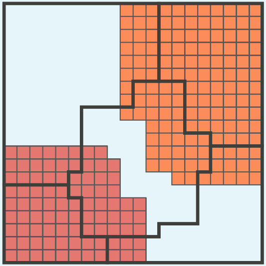
Enrichment algorithm — for some $\mu \in \mathcal{P}$



- compute reduced solution $u_N(\mu)$
- estimate error $\eta_{h,N}(\mu)$
- if $\eta_{h,N}(\mu) > \Delta$, start intermediate local enrichment phase:
  - compute local error indicators
  - mark subdomains for enrichment: $\mathcal{X} = \mathtt{mark}(\mathcal{T}_H)$

Online-Adaptive Enrichment of V_N

Enrichment algorithm — for some $\mu \in \mathcal{P}$



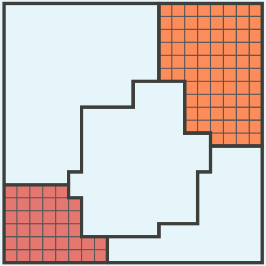
- compute reduced solution $u_N(\mu)$
- estimate error $\eta_{h,N}(\mu)$
- if $\eta_{h,N}(\mu) > \Delta$, start intermediate local enrichment phase:
  - compute local error indicators
  - mark subdomains for enrichment: $\mathcal{X} = \mathtt{mark}(\mathcal{T}_H)$
  - solve corrector problem on oversampling subdomain $T^\delta \supset T$ for all $T \in \mathcal{X}$:

$$a(\varphi_h(\mu), v_h; \mu) = f(v_h) \qquad \text{in } T^\delta$$

$$\varphi_h(\mu) = u_N(\mu) \qquad \text{on } \partial T^\delta$$

Online-Adaptive Enrichment of V_N

Enrichment algorithm — for some $\mu \in \mathcal{P}$



- compute reduced solution $u_N(\mu)$
- estimate error $\eta_{h,N}(\mu)$
- if $\eta_{h,N}(\mu) > \Delta$, start intermediate local enrichment phase:
  - compute local error indicators
  - mark subdomains for enrichment: $\mathcal{X} = \texttt{mark}(\mathcal{T}_H)$
  - solve corrector problem on oversampling subdomain $T^\delta \supset T$ for all $T \in \mathcal{X}$:

$$a(\varphi_h(\mu), v_h; \mu) = f(v_h) \qquad \text{in } T^\delta$$

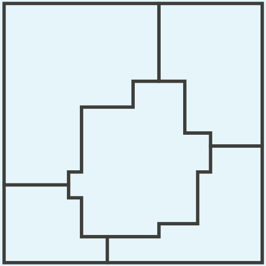
$$\varphi_h(\mu) = u_N(\mu) \qquad \text{on } \partial T^\delta$$

  - extend local reduced basis for all $T \in \mathcal{X}$:

$$V_N^T := \operatorname{span} V_N^T \cup \{\varphi_h(\mu)|_T\}$$

Online-Adaptive Enrichment of V_N

Enrichment algorithm — for some $\mu \in \mathcal{P}$



- compute reduced solution $u_N(\mu)$
- estimate error $\eta_{h,N}(\mu)$
- if $\eta_{h,N}(\mu) > \Delta$, start intermediate local enrichment phase:
  - compute local error indicators
  - mark subdomains for enrichment: $\mathcal{X} = \mathtt{mark}(\mathcal{T}_H)$
  - solve corrector problem on oversampling subdomain $T^\delta \supset T$ for all $T \in \mathcal{X}$:
    $$a(\varphi_h(\mu), v_h; \mu) = f(v_h) \qquad \text{in } T^\delta$$
    $$\varphi_h(\mu) = u_N(\mu) \qquad \text{on } \partial T^\delta$$
  - extend local reduced basis for all $T \in \mathcal{X}$:
    $$V_N^T := \operatorname{span} V_N^T \cup \{\varphi_h(\mu)|_T\}$$
  - update reduced quantities
  - compute updated reduced solution $u_N(\mu)$ and $\eta_{h,N}(\mu)$

Online-Adaptive Enrichment of V_N

Enrichment algorithm — for some $\mu \in \mathcal{P}$

- compute reduced solution $u_N(\mu)$
- estimate error $\eta_{h,N}(\mu)$
- if $\eta_{h,N}(\mu) > \Delta$, start intermediate local enrichment phase:
  - compute local error indicators
  - mark subdomains for enrichment: $\mathcal{X} = \mathtt{mark}(\mathcal{T}_H)$
  - solve corrector problem on oversampling subdomain $T^\delta \supset T$ for all $T \in \mathcal{X}$:

$$a(\varphi_h(\mu), v_h; \mu) = f(v_h) \qquad \text{in } T^\delta$$

$$\varphi_h(\mu) = u_N(\mu) \qquad \text{on } \partial T^\delta$$

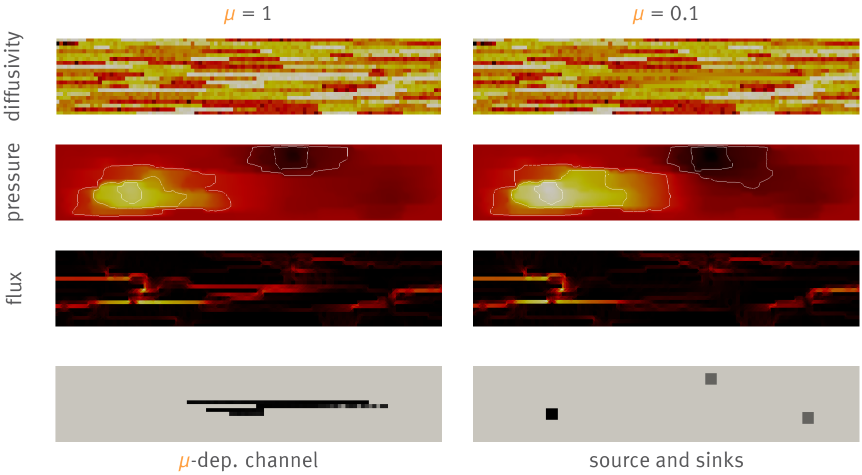
  - extend local reduced basis for all $T \in \mathcal{X}$:

$$V_N^T := \operatorname{span} V_N^T \cup \{ \varphi_h(\mu)|_T \}$$

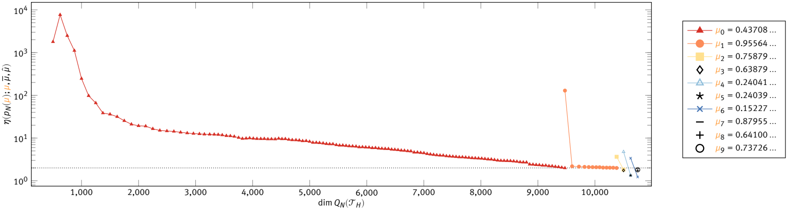
  - update reduced quantities
  - compute updated reduced solution $u_N(\mu)$ and $\eta_{h,N}(\mu)$
- iterate until $\eta_{h,N}(u_{\mu,N}) \leq \Delta$, return $u_N(\mu)$

$\dim V_h(T^\delta)$ |

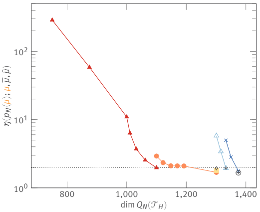
LRBMS with online enrichment: Example SPE10



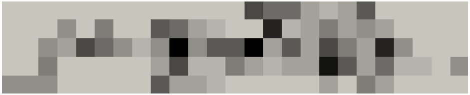
LRBMS with online enrichment: Example SPE10



Convergence history of LRBMS with initially empty V_N



LRBMS initialized with 2 solution snapshots



Distribution of local basis size after online enrichment.

Related Approaches (incomplete)

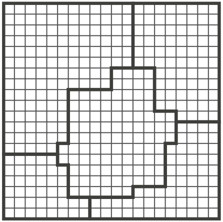
- Reduced Basis Element Method
  [Maday, Ronquist, 2002]

- Port-Reduced Static Condensation Reduced Basis Element Method
  [Eftang, Patera, 2013]

- Generalized Multiscale Finite Element Methods
  [Efendiev, Galvis, Hou 2013]

- Reduced Basis Hybrid Method
  [Iapichino, Quarteroni, Rozza, Volkwein, 2014]

- ArbiLoMod, a Simulation Technique Designed for Arbitrary Local Modifications
  [Buhr, Engwer, Ohlberger, R, 2017]



Questions

- Where should be enriched?
- How fast will enrichment converge?
- Which training method to combine with enrichment?
- How to balance training and enrichment?

Goal: Minimize total number of local V_h-dependent computations/communication events.

Connections with Domain Decomposition Methods

- Local enrichment function $\varphi_h(\mu)|_T$

$$
\begin{aligned}
a(\varphi_h(\mu), v_h; \mu) &= f(v_h) && \text{in } T^\delta \\
\varphi_h(\mu) &= u_N(\mu) && \text{on } \partial T^\delta
\end{aligned}
$$

corresponds to subdomain solution in Restricted Additive Schwarz (RAS) method.

Connections with Domain Decomposition Methods

- Local enrichment function $\varphi_h(\mu)|_T$

$$a(\varphi_h(\mu), v_h; \mu) = f(v_h) \qquad \text{in } T^\delta$$
$$\varphi_h(\mu) = u_N(\mu) \qquad \text{on } \partial T^\delta$$

corresponds to subdomain solution in Restricted Additive Schwarz (RAS) method.

- In particular (for minimal overlap):

enrichment + Galerkin projection onto V_N

$\cong$

locally(!) adaptive [Spillane, 2016] **RAS multi-preconditioned CG** [Bridson, Greif, 2006]

Connections with Domain Decomposition Methods

- Local enrichment function $\varphi_h(\mu)|_T$

$$a(\varphi_h(\mu), v_h; \mu) = f(v_h) \qquad \text{in } T^\delta$$
$$\varphi_h(\mu) = u_N(\mu) \qquad \text{on } \partial T^\delta$$

  corresponds to subdomain solution in Restricted Additive Schwarz (RAS) method.

- In particular (for minimal overlap):

  enrichment + Galerkin projection onto V_N

  $\cong$

  locally(!) adaptive [Spillane, 2016] RAS multi-preconditioned CG [Bridson, Greif, 2006]

- Moreover:

  offline training of V_N

  $\cong$

  construction of coarse space

  e.g. DtN [Nataf et al., 2011], GenEO [Spillane et al., 2014], SHEM [Gander, Loneland, Rahman, 2015]

A Localized RB Additive Schwarz Method

A Localized RB Additive Schwarz Method

1. Choose overlapping DD $T \in \mathcal{T}_H$ and define local FEM spaces $V_h^T \subset V_h$ as usual.

A Localized RB Additive Schwarz Method

1. Choose overlapping DD $T \in \mathcal{T}_H$ and define local FEM spaces $V_h^T \subset V_h$ as usual.

2. Use RB methods to construct coarse space V_N^0 for which abstract Schwarz framework guarantees robustness of AS+CG iterations for every μ.

A Localized RB Additive Schwarz Method

1. Choose overlapping DD $T \in \mathcal{T}_H$ and define local FEM spaces $V_h^T \subset V_h$ as usual.

2. Use RB methods to construct coarse space V_N^0 for which abstract Schwarz framework guarantees robustness of AS+CG iterations for every μ.

3. In each iteration compute solution $u_N(\mu)$ via Galerkin projection onto $V_N^0 \oplus V_N^T$.

A Localized RB Additive Schwarz Method

1. Choose overlapping DD $T \in \mathcal{T}_H$ and define local FEM spaces $V_h^T \subset V_h$ as usual.
2. Use RB methods to construct coarse space V_N^0 for which abstract Schwarz framework guarantees robustness of AS+CG iterations for every μ.
3. In each iteration compute solution $u_N(\mu)$ via Galerkin projection onto $V_N^0 \oplus V_N^T$.
4. Use RB estimator $\eta_{h,N}(u_N(\mu);\mu)$ to locally enrich V_N^T whith AS corrections where needed:

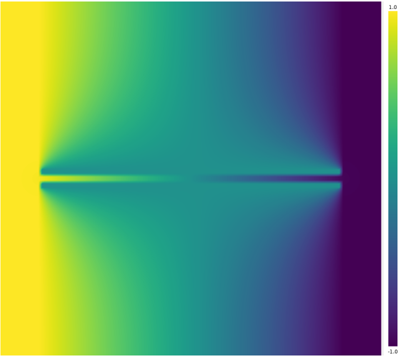
$$\eta_{h,N}(u_N(\mu);\mu)^2 := C(\mu)^2 \sum_{T \in \mathcal{T}_H} \left(\sup_{v_h \in V_h^T} \frac{f(v_h) - a(u_N(\mu), v_h; \mu)}{\|v_h\|} \right)^2$$

where, with C_{stab} the stability constant of decomposition $V_h = V_N^0 + \sum_{T \in \mathcal{T}_H} V_h^T$:

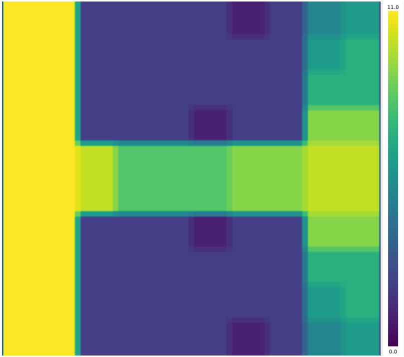
$$C(\mu) \leq C_{inf\text{-}sup}(\mu) \cdot C_{stab}$$

Simple Experiment (without μ, local non-parametric changes)

Solution (contrast: 10^5)



Number of local solutions (max=11)

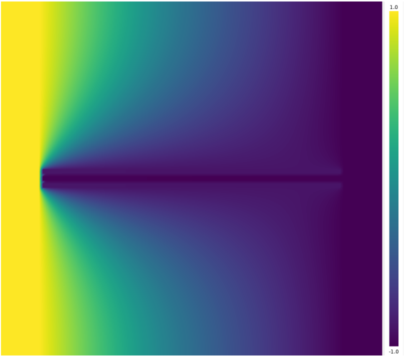


- 10 × 10 subdomains
- 4 elements overlap
- 6 GenEO basis functions per domain

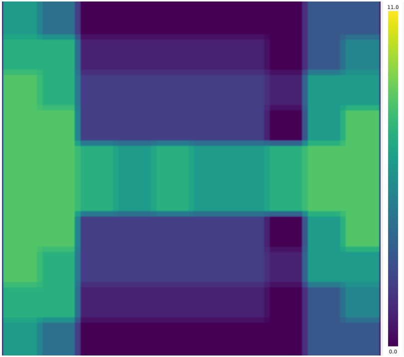
- enrich where $\|\mathcal{R}|_T\| \geq 0.5/|\mathcal{T}_H| \cdot \|\mathcal{R}\|$
- update V_N^0, keep localized soution in V_N^T for next problem

Simple Experiment (without μ, local non-parametric changes)

Solution (contrast: 10^5)



Number of local solutions (max=11)

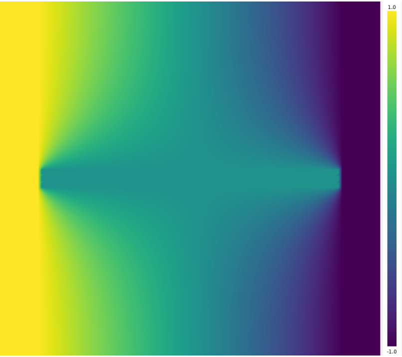


- 10 × 10 subdomains
- 4 elements overlap
- 6 GenEO basis functions per domain

- enrich where $\|\mathcal{R}|_T\| \geq 0.5/|\mathcal{T}_H| \cdot \|\mathcal{R}\|$
- update V_N^0, keep localized soution in V_N^T for next problem

Simple Experiment (without μ, local non-parametric changes)

Solution (contrast: 10^5)



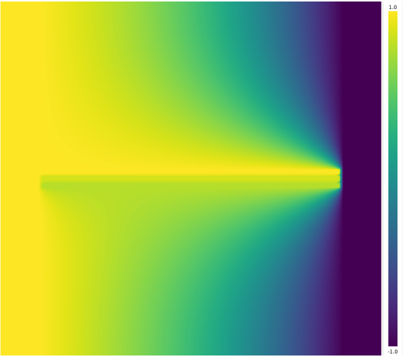
Number of local solutions (max=11)

- 10×10 subdomains
- 4 elements overlap
- 6 GenEO basis functions per domain

- enrich where $\|\mathcal{R}|_T\| \geq 0.5/|\mathcal{T}_H| \cdot \|\mathcal{R}\|$
- update V_N^0, keep localized soution in V_N^T for next problem

Simple Experiment (without μ, local non-parametric changes)

Solution (contrast: 10^5)



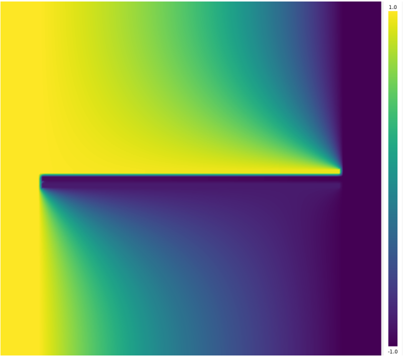
Number of local solutions (max=11)

- 10 × 10 subdomains
- 4 elements overlap
- 6 GenEO basis functions per domain

- enrich where $\| \mathcal{R}|_T \| \geq 0.5 / |\mathcal{T}_H| \cdot \|\mathcal{R}\|$
- update V_N^0, keep localized soution in V_N^T for next problem

Simple Experiment (without μ, local non-parametric changes)

Solution (contrast: 10^5)



- 10 × 10 subdomains
- 4 elements overlap
- 6 GenEO basis functions per domain

Number of local solutions (max=11)

- enrich where $\|\mathcal{R}|_T\| \geq 0.5/|\mathcal{T}_H| \cdot \|\mathcal{R}\|$
- update V_N^0, keep localized soution in V_N^T for next problem

Some Remarks

- Communication of V_h-dependet data only with neighbors of enriched subdomains.
- localized enrichment $\cong$ flexible multi-preconditioned projected CG with full orthogonalization.
- More iterations but less work.

	iterations	local solutions
PCG	118	11800
PCG + RB solution as initial value	84	8400
enrich localized (keep solutions in V_N^T)	**38**	**1803**
enrich everywhere (keep solutions in V_N^T)	36	3600
enrich localized (keep updates in V_N^T)	33	1718
enrich everywhere (keep updates in V_N^T)	29	2900

Two-Scale Reduced Basis Localized Orthogonal Decomposition

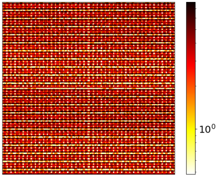
Multiscale Model Problem

Parameterized diffusion equation

For a fixed parameter $\mu \in \mathcal{P}$ find u_μ s.t.

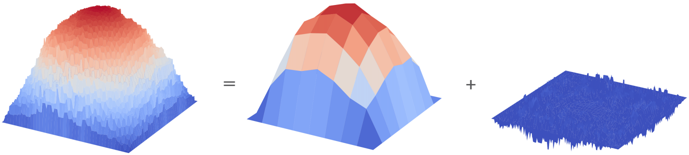
$$\begin{aligned} -\nabla \cdot A_\mu \nabla u_\mu &= f, && \text{in } \Omega, \\ u_\mu &= 0, && \text{on } \partial\Omega, \end{aligned} \qquad \text{or in weak form} \qquad a_\mu(u_\mu, v) = F(v), \quad \forall v \in V$$

- Parameter space $\mathcal{P} \subset \mathbb{R}^m$, $m \in \mathbb{N}$
- Bounded Lipschitz domain $\Omega \subset \mathbb{R}^d$, Hilbert space V.
- $f \in L^2(\Omega)$, bilinear form a_μ and functional $F \in V'$.
- Homogeneous Dirichlet boundary conditions.
- $A_\mu \in L^\infty(\Omega, \mathbb{R}^{d \times d})$ symmetric and uniformly elliptic: $0 < \alpha \leq A_\mu \leq \beta < \infty$.
- Possibly high variations in A_μ (e.g. due to soil composition).



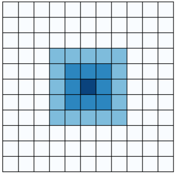
Multiscale Orthogonal Decomposition

- Fine mesh $\mathcal{T}_h$ and coarse mesh $\mathcal{T}_H$ with maximal element diameter $H \gg h$, FE spaces V_h and $V_H := V_h \cap \mathcal{P}_1(\mathcal{T}_H)$.
- Interpolation operator $\mathcal{I}_H: V_h \to V_H$ (e.g. L^2-projection).
- Finescale space $V^\mathrm{f} := \ker(\mathcal{I}_H) = \{v \in V_h \,|\, \mathcal{I}_H(v) = 0\}$, decomposition $V = V_H + V^\mathrm{f}$.



- Finescale correction $\mathcal{Q}_\mu: V_H \to V^\mathrm{f}$ defined by $$a_\mu(\mathcal{Q}_\mu v_H, v^\mathrm{f}) = a_\mu(v_H, v^\mathrm{f}), \qquad \forall\, v^\mathrm{f} \in V^\mathrm{f}.$$
- Multiscale space $V_\mu^\mathrm{ms} := (I - \mathcal{Q}_\mu)V_H$.
- a-orthogonal decomposition $V_h = V_\mu^\mathrm{ms} \oplus_a V^\mathrm{f}$.

Localization



- Truncated finescale space $V^{\mathrm{f}}(U_k(T)) := \left\{ v \in V^{\mathrm{f}} \,\middle|\, v|_{\Omega \setminus U_k(T)} = 0 \right\}$.
- For each $T \in \mathcal{T}_H$, define localized correctors $\mathcal{Q}_k^T v_H \in V^{\mathrm{f}}(U_k(T))$

$$a_{U_k(T)}(\mathcal{Q}_k^T v_H, v^{\mathrm{f}}) = a_T(v, v^{\mathrm{f}}), \qquad \forall\, v^{\mathrm{f}} \in V^{\mathrm{f}}(U_k(T)),$$

- Localized corrector operator $\mathcal{Q}_k = \sum_{T \in \mathcal{T}_H} \mathcal{Q}_k^T$.
- LOD space $V_k^{\mathrm{ms}} := \{\lambda_x - \mathcal{Q}_k \lambda_x \,|\, x \in \mathcal{N}_H\}$

Lemma [Målqvist/Peterseim ’14]

The correctors $\mathcal{Q}$ decay exponentially, e.g.

$$\|\mathcal{Q} - \mathcal{Q}_k\| \leq C_{\mathcal{Q}} k^{d/2} \theta^k \|\mathcal{Q}\|,$$

where $0 < \theta < 1$ and $C_{\mathcal{Q}}$ depends on α/β but not on the variations of A_μ.

Petrov Galerkin Formulation [Elfverson/Ginting/Henning '15]

Petrov–Galerkin LOD method

Find $u_H^{\text{ms}} \in V_k^{\text{ms}}$ such that

$$a(u_k^{\text{ms}}, v) = F(v), \qquad \forall\, v \in V_H.$$

- No interaction between correctors required.
- Reduced memory consumption.
- Still similar convergence results.

Convergence theorem

$$\left\|u_{h,\mu} - u_{H,k,\mu}\right\|_{L^2} + \left\|u_{h,\mu} - u_{H,k,\mu}^{\text{ms}}\right\|_1 \lesssim (H + \theta^k k^{d/2})\|f\|_{L^2(\Omega)}$$

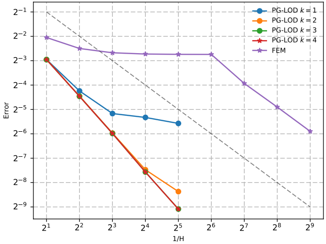


Figure: Energy error $|||u_\varepsilon - u_{H,k}^{\text{ms}}|||$ for the PG–LOD and $|||u_\varepsilon - u_h|||$ for the FEM for 1d model problem from **[Peterseim'16]**.

Two-Scale Formulation of the LOD

Two-Scale space

$$\mathfrak{V} := V_H \oplus V^{\mathrm{f}}_{h,k,T_1} \oplus \cdots \oplus V^{\mathrm{f}}_{h,k,T_{|\mathcal{T}_H|}}$$

$$|||\mathfrak{u}|||_1^2 := \|u_H\|_1^2 + \sum_{T \in \mathcal{T}_H} \left\|u^{\mathrm{f}}_T\right\|_1^2$$

Two-Scale Formulation of the LOD

Two-Scale space

$$\mathfrak{V} := V_H \oplus V^{\mathrm{f}}_{h,k,T_1} \oplus \cdots \oplus V^{\mathrm{f}}_{h,k,T_{|\mathcal{T}_H|}}$$

$$|||\mathfrak{u}|||_1^2 := \|u_H\|_1^2 + \sum_{T\in\mathcal{T}_H} \left\|u^{\mathrm{f}}_T\right\|_1^2$$

Two-scale bilinear form

$$\mathfrak{B}_\mu(\mathfrak{u}, \mathfrak{v}) := a_\mu(u_H - \sum_{T\in\mathcal{T}_H} u^{\mathrm{f}}_T, v_H) + \rho^{1/2} \sum_{T\in\mathcal{T}_H} a_\mu(u^{\mathrm{f}}_T, v^{\mathrm{f}}_T) - a^T_\mu(u_H, v^{\mathrm{f}}_T),$$

Two-Scale Formulation of the LOD

Two-Scale space

$$\mathfrak{V} := V_H \oplus V^{\mathrm{f}}_{h,k,T_1} \oplus \cdots \oplus V^{\mathrm{f}}_{h,k,T_{|\mathcal{T}_H|}}$$

$$|||\mathfrak{u}|||_1^2 := \|u_H\|_1^2 + \sum_{T \in \mathcal{T}_H} \left\|u^{\mathrm{f}}_T\right\|_1^2$$

Two-scale bilinear form

$$\mathfrak{B}_\mu(\mathfrak{u}, \mathfrak{v}) := a_\mu(u_H - \sum_{T \in \mathcal{T}_H} u^{\mathrm{f}}_T, v_H) + \rho^{1/2} \sum_{T \in \mathcal{T}_H} a_\mu(u^{\mathrm{f}}_T, v^{\mathrm{f}}_T) - a^T_\mu(u_H, v^{\mathrm{f}}_T),$$

Proposition

The two-scale solution $\mathfrak{u}_\mu \in \mathfrak{V}$ of

$$\mathfrak{B}_\mu(\mathfrak{u}_\mu, \mathfrak{v}) = F(v_H) \qquad \forall \mathfrak{v} \in \mathcal{V}.$$

is uniquely determined and given by $\mathfrak{u}_\mu = \left[u_{H,k,\mu},\ \mathcal{Q}^{T_1}_{k,\mu}(u_{H,k,\mu}),\ \ldots,\ \mathcal{Q}^{T_{|\mathcal{T}_H|}}_{k,\mu}(u_{H,k,\mu})\right]$.

Two-scale Stability Estimate

Proposition

Let $\rho := C_{\text{ovl}} \cdot \kappa$, then $\mathfrak{B}_\mu$ is $|||\cdot|||_{a,\mu}$-$|||\cdot|||_1$-continuous and inf-sup stable with the following bounds on the respective constants:

$$\sup_{0\neq\mathfrak{u}\in\mathfrak{V}} \sup_{0\neq\mathfrak{v}\in\mathfrak{V}} \frac{\mathfrak{B}_\mu(\mathfrak{u},\mathfrak{v})}{|||\mathfrak{u}|||_{a,\mu}\cdot|||\mathfrak{v}|||_1} \leq \beta^{1/2} \quad \text{and} \quad \inf_{0\neq\mathfrak{u}\in\mathfrak{V}} \sup_{0\neq\mathfrak{v}\in\mathfrak{V}} \frac{\mathfrak{B}_\mu(\mathfrak{u},\mathfrak{v})}{|||\mathfrak{u}|||_{a,\mu}\cdot|||\mathfrak{v}|||_1} \geq \gamma_k/\sqrt{5}.$$

where

$$|||\mathfrak{u}|||^2_{a,\mu} := \|u_H - \sum_{T\in\mathcal{T}_H} u_T^{\mathrm{f}}\|^2_{a,\mu} + \rho \sum_{T\in\mathcal{T}_H} \|\mathcal{Q}^T_{k,\mu}(u_H) - u_T^{\mathrm{f}}\|^2_{a,\mu}$$

Error Bound

$$\left\{\|u_{H,k,\mu} - u_H\|_1^2 + \rho \sum_{T\in\mathcal{T}_H} \|\mathcal{Q}^T_{k,\mu}(u_H) - u_T^{\mathrm{f}}\|_1^2\right\}^{1/2} \leq \sqrt{5} C_{\mathcal{I}_H} \alpha^{-1/2} \gamma_k^{-1} \sup_{\mathfrak{v}\in\mathfrak{V}} \frac{\mathfrak{F}(\mathfrak{v}) - \mathfrak{B}_\mu(\mathfrak{u},\mathfrak{v})}{|||\mathfrak{v}|||_1}$$

$$\leq \sqrt{15} C_{\mathcal{I}_H} (C_{\text{ovl}} + 1)^{1/2} \kappa^{1/2} \gamma_k^{-1} \beta^{1/2} \left\{\|u_{H,k,\mu} - u_H\|_1^2 + \rho \sum_{T\in\mathcal{T}_H} \|\mathcal{Q}^T_{k,\mu}(u_H) - u_T^{\mathrm{f}}\|_1^2\right\}^{1/2}.$$

Two-Scale Reduced Basis Approach

Stage 1 (for each $T \in \mathcal{T}_H$)

ROM:

$$a_\mu(\mathcal{Q}^{T,rb}_{k,\mu}(v_H), v^{\mathrm{f}}_T) = a^T_\mu(v_H, v^{\mathrm{f}}_T), \qquad \forall v^{\mathrm{f}}_T \in V^{\mathrm{f},rb}_{k,T}.$$

Output:

$$\mathbb{K}^{rb}_\mu := \sum_{T \in \mathcal{T}_H} \mathbb{K}^{rb}_{T,\mu}, \qquad \left(\mathbb{K}^{rb}_{T,\mu}\right)_{ji} := \left(A_\mu(x_T\nabla - \nabla \mathcal{Q}^{T,rb}_{k,\mu})\phi_i\,,\, \nabla\phi_j\right)_{U_k(T)}$$

Error bound:

$$\|\mathcal{Q}^T_{k,\mu}(v_H) - \mathcal{Q}^{T,rb}_{k,\mu}(v_H)\|_{a,\mu} \leq \alpha^{-1/2} \sup_{v^{\mathrm{f}}_T \in V^{\mathrm{f}}_{h,k,T}} \frac{a^T_\mu(v_H, v^{\mathrm{f}}_T) - a_\mu(\mathcal{Q}^{T,rb}_{k,\mu}(v_H), v^{\mathrm{f}}_T)}{\|v^{\mathrm{f}}_T\|_1}.$$

Basis generation: weak greedy algorithm

Two-Scale Reduced Basis Approach

Stage 2

ROM:

$$\mathfrak{u}_\mu^{rb} := \operatorname*{arg\,min}_{\mathfrak{u}\in\mathfrak{V}^{rb}} \sup_{\mathfrak{v}\in\mathfrak{V}} \frac{\mathfrak{F}(\mathfrak{v}) - \mathfrak{B}_\mu(\mathfrak{u},\mathfrak{v})}{|||\mathfrak{v}|||_1}.$$

Error bound:

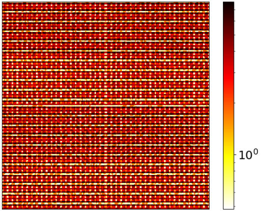
$$\left\{\|u_{H,k,\mu} - u_H\|_1^2 + \rho \sum_{T\in\mathcal{T}_H} \|\mathcal{Q}_{k,\mu}^T(u_H) - u_T^{\mathrm{f}}\|_1^2\right\}^{1/2} \leq \sqrt{5}C_{\mathcal{I}_H}\alpha^{-1/2}\gamma_k^{-1} \sup_{\mathfrak{v}\in\mathfrak{V}} \frac{\mathfrak{F}(\mathfrak{v}) - \mathfrak{B}_\mu(\mathfrak{u},\mathfrak{v})}{|||\mathfrak{v}|||_1}$$

Basis generation: weak greedy algorithm; snapshots computed with:

$$\mathbb{K}_\mu^{rb} \cdot \underline{u}_{H,k,\mu} = \mathbb{F}$$

$$\mathfrak{u}_\mu := \left[u_{H,k,\mu}, \mathcal{Q}_{k,\mu^*}^{T_1,rb}(u_{H,k,\mu}), \dots, \mathcal{Q}_{k,\mu^*}^{T,rb}(u_{H,k,\mu})\right]$$

Numerical Experiment



- $\mathcal{P} := [1, 5]^3$
- $|\mathcal{T}_h| = 67,108,864$
- $|\mathcal{T}_H| = 4,096$
- $1,024$ processes
- $\kappa \approx 16$
- Stage 2 greedy until Stage 1 error dominates.

tolerance ε_1	10^{-1}		10^{-2}	
method	RBLOD	TSRBLOD	RBLOD	TSRBLOD
$t_1^{\text{offline}}(T)$	4994	4052	10393	11241
t_1^{offline}	26008	20382	48379	53279
t_2^{offline}	-	5754	-	10385
t^{offline}	26008	26136	83403	63665
cum. size St.1	147473	94417	278528	193289
av. size St.1	9.00	23.05	17.00	47.19
size St.2	-	10	-	18
t^{LOD}	484.58		493.06	
t^{online}	3.93	0.0006	4.62	0.001
speed-up w.r.t LOD	123.15	8.32e5	106.79	4.93e5
$e_{\text{LOD}}^{H^1,\text{rel}}$	6.40e-4	1.99e-3	2.56e-5	2.04e-5
$e_{\text{LOD}}^{L^2,\text{rel}}$	1.74e-4	1.95e-3	1.86e-6	8.86e-6

Model Order Reduction with pyMOR

pyMOR main developers

Linus Balicki

René Fritze

Petar Mlinarić

Stephan Rave

Felix Schindler

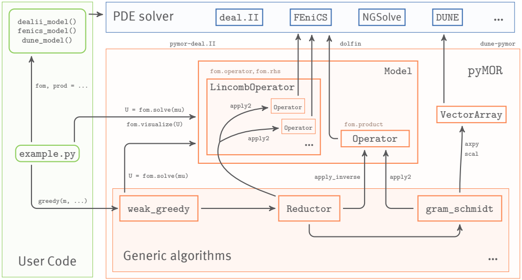
pyMOR – Model Order Reduction with Python

Goal

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

- Started late 2012, 20k lines of Python code, 6k 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 Algorithms and 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.

Implemented Algorithms

- Gram-Schmidt, POD, HAPOD.
- Greedy basis generation with different extension algorithms.
- Automatic (Petrov-)Galerkin projection of arbitrarily nested affine combinations of operators.
- Interpolation of arbitrary (nonlinear) operators, EI-Greedy, DEIM.
- A posteriori error estimation.
- System theory methods: balanced truncation, IRKA, ...
- Iterative linear solvers, eigenvalue computation, Newton algorithm, time-stepping algorithms.
- New! Non-intrusive MOR using artificial neural networks.

Feature Tour: FEniCS Support

- Directly interfaces FEniCS LA backend, no copies needed.
- Use same MOR code as with builtin discretization toolkit!
- Builtin support for empirical interpolation.
- Thermal block demo:
  30 SLOC FEniCS +
  15 SLOC wrapping for pyMOR.
- Easily increase FEM order, etc.

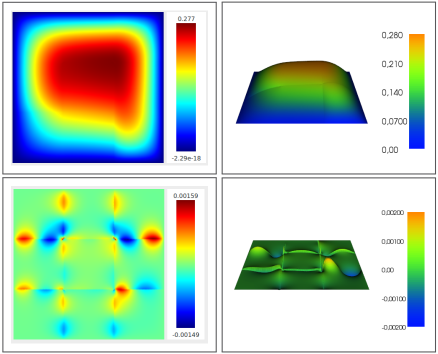


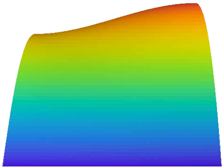
Figure: 3x3 thermal block problem
top: red. solution, bottom: red. error
left: pyMOR solver, right: FEniCS solver

Feature Tour: Empirical Interpolation with FEniCS

Nonlinear Poisson problem from FEniCS docs (for $\mu = 1$)

$$
\begin{aligned}
-\nabla \cdot \{(1 + \mu u^2(x, y)) \cdot \nabla u(x, y)\} &= x \cdot \sin(y) && \text{for } x, y \in (0, 1) \\
u(x, y) &= 1 && \text{for } x = 1 \\
\nabla u(x, y) \cdot n &= 0 && \text{otherwise}
\end{aligned}
$$

- `mesh = UnitSquareMesh(100, 100); V = FunctionSpace(mesh, "CG", 2)`.
- Time for solution: ≈ 3.4 s.
- $\mu \in [1, 1000]$, RB size: 2, EI DOFs: 5, rel. error $\approx 10^{-6}$.



- Local operator evaluation implemented using `dolfin.SubMesh`.
- Speedup: 80.
- See `fenics_nonlinear` demo.

Feature Tour: deal.II Support

- `pymor-deall.II` support module
  `https://github.com/pymor/pymor-deal.II`
- Python bindings for
  - `dealii::Vector`,
  - `dealii::SparseMatrix`.
- pyMOR wrapper classes.
- MOR demo for linear elasticity example from tutorial.

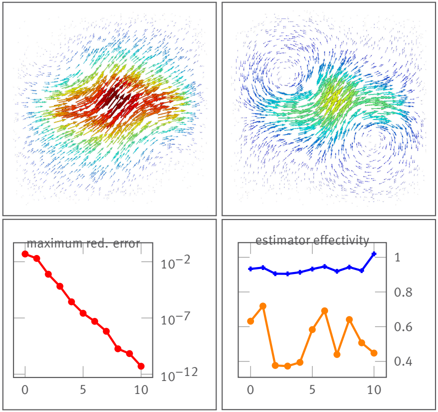


Figure: top: Solutions for $(\mu, \lambda) = (1, 1)$ and $(\mu, \lambda) = (1, 10)$, bottom: red. errs. and max./min. estimator effectivities vs. dim V_N.

Feature Tour: NGSolve Support

- Based on NGS-Py Python bindings for NGSolve.
- pyMOR wrappers for vector and matrix classes.
- 3d thermal block demo included.
- Joint work with Christoph Lehrenfeld.

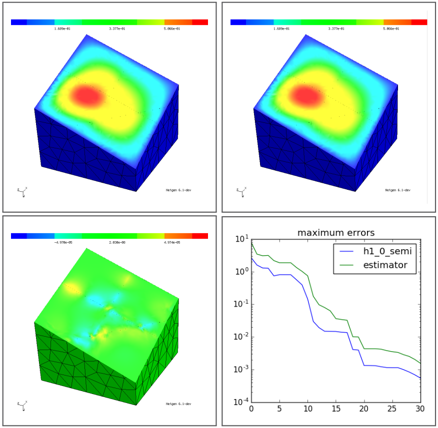


Figure: 3d thermal block problem
top: full/red. sol., bottom: err. for worst approx. μ and max. red. error vs. dim V_N.

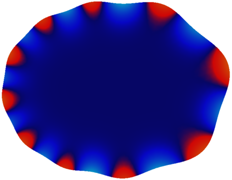
Feature Tour: MOR for an NGSolve Free Boundary Problem

[Lehrenfeld, R, 19]

Osmotic cell swelling model [Lippoth, Prokert, 2012]

Given $\Omega(0) \subset \mathbb{R}^d$, $u(0) \in H^1(\Omega(0))$ and coefficients $u_{\text{ext}}, \alpha, \beta, \gamma \in \mathbb{R}$, the **concentration** $u(t)$ and **normal velocity** w_Γ of $\Gamma(t)$ is given by:

$$
\begin{aligned}
\partial_t u - \alpha \Delta u &= 0 && \text{in } \Omega(t), \\
w_\Gamma u + \alpha \partial_{\mathbf{n}} u &= 0 && \text{on } \Gamma(t), \\
-\beta\kappa + \gamma(u - u_{\text{ext}}) &= w_\Gamma && \text{on } \Gamma(t).
\end{aligned}
$$



- ALE formulation $\rightarrow$ diffusion coeffs nonlinear in deformation field Ψ
- Empirical interpolation w.r.t. Ψ.

Feature Tour: Tools for interfacing MPI parallel solvers

- Automatically make sequential bindings MPI aware.
- Reduce HPC-Cluster models without thinking about MPI at all.
- Interactively debug MPI parallel solvers.

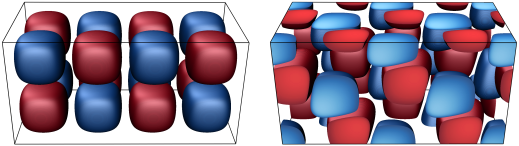


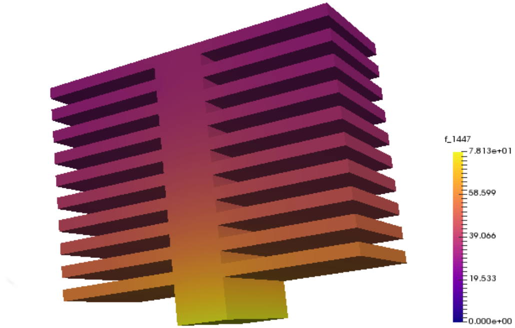
Figure: FV solution of 3D Burgers-type equation ($27.6 \cdot 10^6$ DOFs, 600 time steps) using Dune.

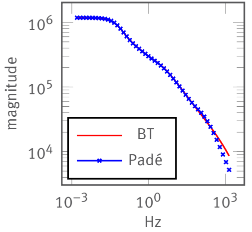
Table: Time (s) needed for solution using DUNE / DUNE with pyMOR timestepping.

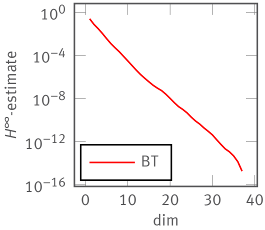
MPI ranks	1	2	3	6	12	24	48	96	192
DUNE	17076	8519	5727	2969	1525	775	395	202	107
pyMOR	17742	8904	6014	3139	1606	816	418	213	120
overhead	3.9%	4.5%	5.0%	5.7%	5.3%	5.3%	6.0%	5.4%	11.8%

Feature Tour: 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







System-Theoretic MOR with FEniCS – Implementation

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)
```

System-Theoretic MOR with FEniCS – Implementation

pyMOR wrapping

```
# def discretize (cont.)
    # monkey patch apply_inverse_adjoint, assuming symmetriy
    FenicsMatrixOperator.apply_inverse_adjoint = FenicsMatrixOperator.apply_inverse

    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
```

System-Theoretic MOR with FEniCS – Implementation

pyMOR wrapping

```
# def discretize (cont.)
    # monkey patch apply_inverse_adjoint, assuming symmetriy
    FenicsMatrixOperator.apply_inverse_adjoint = FenicsMatrixOperator.apply_inverse

    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()
```

System-Theoretic MOR with FEniCS – Implementation

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 = arnoldi(fom.A, fom.E, fom.B, [0] * r)
W = 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!

Feinauer, Hein, R, Schmidt, Westhoff, et al., *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. 31, 2019.

Himpe, Leibner, R, *Hierarchical Approximate Proper Orthogonal Decomposition*, SISC 40(5), 2018.

Buhr, Engwer, Ohlberger, R, *ArbiLoMod, a Simulation Technique Designed for Arbitrary Local Modifications*, SISC, 39(4), 2017.

Gander, R, *Localized Reduced Basis Additive Schwarz Methods*, arXiv 2103.10884, 2021.

Milk, R, Schindler, *pyMOR – Generic Algorithms and Interfaces for Model Order Reduction*, SISC 38(5), 2016.

```
pip3 install pymor
```