Westfälische
Wilhelms-Universität
Münster

Dune-RB: An abstract reduced basis model order reduction interface

living.knowledge
WWU Münster

Felix Albrecht,
Martin Drohmann,
Bernard Haasdonk, Sven
Kaulmann, Mario Ohlberger

20/06/2012

Outline

Reduced basis method

Abstract software concept

Proof of concept: Poisson equation

Outlook

Table of Contents

Reduced Basis Method

RB Scenario:

- Parametrized PDEs
- time-critical applications
- many-query applications

Goals:

- Offline-/Online decomposition
- Efficient reduced simulations
- A posteriori error control

References: [Patera&Rozza, 2006], [Haasdonk et al., 2008]

Martin Drohmann (mdrohmann@wwu.de)

Reduced Basis Method

RB Scenario:

- Parametrized PDEs
- time-critical applications
- many-query applications

Goals:

- Offline-/Online decomposition
- Efficient reduced simulations
- A posteriori error control

References: [Patera&Rozza, 2006], [Haasdonk et al., 2008]

Reduced Basis Method

RB Scenario:

- Parametrized PDEs
- time-critical applications
- many-query applications

Goals:

- Offline-/Online decomposition
- Efficient reduced simulations
- A posteriori error control

References: [Patera&Rozza, 2006], [Haasdonk et al., 2008]

Linear stationary problems

Find solutions $u \in \mathcal{W}$, such that

$$\mathcal{L}[u] = b \qquad \text{in } \Omega. \tag{1}$$

- Linear operator $\mathcal{L}$

Linear stationary problems

For all $\boldsymbol{\mu} \in \mathcal{P} \subset \mathbb{R}^p$, find solutions $u(\boldsymbol{\mu}) \in \mathcal{W}$, such that

$$\mathcal{L}(\boldsymbol{\mu})\,[u(\boldsymbol{\mu})] = b(\boldsymbol{\mu}) \qquad \text{in } \Omega(\boldsymbol{\mu}). \tag{1}$$

- Linear operator $\mathcal{L}$
- Parametrization

Linear stationary problems

For all $\boldsymbol{\mu} \in \mathcal{P} \subset \mathbb{R}^p$, find solutions $u(\boldsymbol{\mu}) \in \mathcal{W}$, such that

$$\mathcal{L}(\boldsymbol{\mu})\,[u(\boldsymbol{\mu})] = b(\boldsymbol{\mu}) \qquad \text{in } \Omega(\boldsymbol{\mu}). \tag{1}$$

- Linear operator $\mathcal{L}$
- Parametrization
- Separability of parameter and space variable

$$\mathcal{L}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_L} \sigma_L^q(\boldsymbol{\mu})\mathcal{L}^q \tag{2}$$

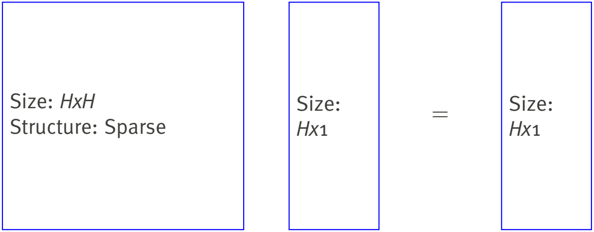
$$b(\boldsymbol{\mu}) = \sum_{q=1}^{Q_b} \sigma_b^q(\boldsymbol{\mu}) b^q \tag{3}$$

Linear stationary problems

Discretization (linear with affine parameter dependence)

For $\boldsymbol{\mu} \in \mathcal{P}$ find $u_h(\boldsymbol{\mu}) \in \mathcal{W}_h \subset \mathcal{W}$, such that

$$\mathcal{L}_h(\boldsymbol{\mu})\,[u_h(\boldsymbol{\mu})] = b_h(\boldsymbol{\mu}).$$



Reduced basis scheme

Assume we have reduced basis space: $\mathcal{W}_{\text{red}} := \text{span}\,\{u_h(\boldsymbol{\mu}^n)\}_{n=1}^N$

Reduced basis scheme

Assume we have reduced basis space: $\mathcal{W}_{\text{red}} := \text{span}\,\{u_h(\boldsymbol{\mu}^n)\}_{n=1}^N$

Reduced simulation (linear with affine parameter dependence)

For $\boldsymbol{\mu} \in \mathcal{P}$ find $u_h(\boldsymbol{\mu}) \in \mathcal{W}_h$, such that

$$\mathcal{L}_h(\boldsymbol{\mu})\,[u_h(\boldsymbol{\mu})] = b_h(\boldsymbol{\mu}).$$

Reduced basis scheme

Assume we have reduced basis space: $\mathcal{W}_{\text{red}} := \text{span}\,\{u_h(\boldsymbol{\mu}^n)\}_{n=1}^N$

Reduced simulation (linear with affine parameter dependence)

For $\boldsymbol{\mu} \in \mathcal{P}$ find $u_{\text{red}}(\boldsymbol{\mu}) \in \mathcal{W}_{\text{red}} \subset \mathcal{W}_h$, such that

$$\mathcal{L}_{\text{red}}(\boldsymbol{\mu})\,[u_{\text{red}}(\boldsymbol{\mu})] = b_{\text{red}}(\boldsymbol{\mu}).$$

Martin Drohmann (mdrohmann@wwu.de)

Reduced basis scheme

Assume we have reduced basis space: $\mathcal{W}_{\text{red}} := \text{span}\{u_h(\boldsymbol{\mu}^n)\}_{n=1}^N$

Reduced simulation (linear with affine parameter dependence)

For $\boldsymbol{\mu} \in \mathcal{P}$ find $u_{\text{red}}(\boldsymbol{\mu}) \in \mathcal{W}_{\text{red}} \subset \mathcal{W}_h$, such that

$$\mathcal{L}_{\text{red}}(\boldsymbol{\mu})[u_{\text{red}}(\boldsymbol{\mu})] = b_{\text{red}}(\boldsymbol{\mu}).$$

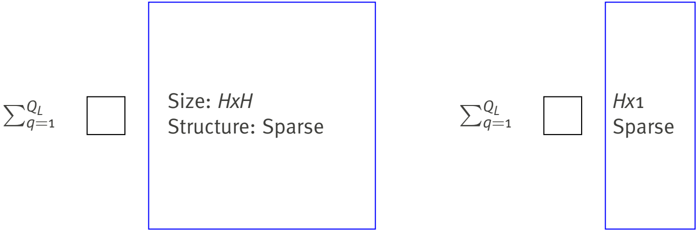
Size: *NxN* Structure: Dense	Size: *Nx*1	=	Size: *NxN* Structure: Sparse	Size: *Nx*1

Offline-/Online decomposition

Parameter separation

$\mathcal{L}_h$ and b_h can be written as

$$\mathcal{L}_h(\boldsymbol{\mu})[u_h] = \sum_{q=1}^{Q_L} \sigma_L^q(\boldsymbol{\mu})\mathcal{L}_h^q[u_h], \qquad b_h(\boldsymbol{\mu}) = \sum_{q=1}^{Q_b} \sigma_b^q(\boldsymbol{\mu}) b_h^q$$



Offline–matrices

Assume RB space spanned by N (orthonormal) basis functions $\Phi := \{\varphi_i\}_{i=1}^N$.
Reduction of parameter independent components during offline phase: $\mathcal{L}_{\text{red}}^q := \mathcal{P}_{\text{red}}\left[\mathcal{L}_h^q\right]$ und $b_{\text{red}}^q := \mathcal{P}_{\text{red}}\left[b_h^q\right]$

$\underline{\mathcal{L}_{\text{red}}^q}$ =
Size: NxN
Structure: Dense

$\underline{\Phi}^t$
Size: NxH
Structure: Dense

$\underline{\mathcal{L}_h}^q$
Size: HxH
Structure: Sparse

$\underline{\Phi}$
Size: HxN
Structure: Dense

Online assembling

$$\mathcal{L}_{\text{red}}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_L} \sigma_L^q(\boldsymbol{\mu}) \ \mathcal{P}_{\text{red}}[\mathcal{L}_h^q]$$

$$\mathcal{L}\text{red}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_L} \square$$

Size: *NxN*
Structure: Dense

Efficient error bound

Residual

The norm of the residual

$$R(\boldsymbol{\mu}) := \mathcal{L}_h(\boldsymbol{\mu})\,[u_{\mathrm{red}}(\boldsymbol{\mu})] - b_h. \tag{4}$$

is efficiently computable by offline-/online decomposition!

With

$$\left\| (\mathcal{L}_h(\boldsymbol{\mu}))^{-1} \right\|_{\mathcal{W}_h} \leq C(\boldsymbol{\mu}), \tag{5}$$

the approximation error can be bounded by

$$\| u_h(\boldsymbol{\mu}) - u_{\mathrm{red}}(\boldsymbol{\mu}) \|_{\mathcal{W}_h} \leq \eta(\boldsymbol{\mu}) := C(\boldsymbol{\mu})\, \| R(\boldsymbol{\mu}) \| \,. \tag{6}$$

All Offline–Matrices

$$\underline{\mathcal{L}^q_{\text{red}}} := \left(\underline{\mathcal{L}^q_h}\,\underline{\Phi}\right)^t W\underline{\Phi} \qquad \text{for } q = 1,\dots,Q_L, \tag{7}$$

$$\underline{b^q_{\text{red}}} := (\underline{b^q_h})^t W\underline{\Phi} \qquad \text{for } q = 1,\dots,Q_b, \tag{8}$$

$$\underline{\mathcal{K}^{q,q'}_{\text{red}}} := \left(\underline{\mathcal{L}^q_h}\,\underline{\Phi}\right)^t W\underline{\mathcal{L}^{q'}_h}\,\underline{\Phi} \qquad \text{for } q,q' = 1,\dots,Q_L, \tag{9}$$

$$\underline{m^{q,q'}_{\text{red}}} := \left(\underline{\mathcal{L}^q_h}\,\underline{\Phi}\right)^t W\underline{b^{q'}_h} \qquad \text{for } q = 1,\dots,Q_L,\ q' = 1,\dots,Q_b \text{ and} \tag{10}$$

$$\underline{n^{q,q'}_{\text{red}}} := (\underline{b^q_h})^t W\underline{b^{q'}_h} \qquad \text{for } q,q' = 1,\dots,Q_b. \tag{11}$$

What about Φ_N?

BASIC-greedy algorithm

BASIC-GREEDY($M_{\text{train}}, \varepsilon_{\text{tol}}, N_{\max}$)

– *Initialize reduced basis of dimension* Υ_0:

$\Phi_{N_0} \leftarrow$ INITBASIS()

$N \leftarrow N_0$

repeat

– *Find worst approximated parameter:*

$(\boldsymbol{\mu}_{\max}) \leftarrow \arg\max_{\boldsymbol{\mu} \in M_{\text{train}}} \eta_N(\boldsymbol{\mu})$

– *Extend reduced basis by snapshot:*

$\Phi_{N+1} \leftarrow \Phi_N \oplus u_h(\boldsymbol{\mu})$

$N \leftarrow N + 1$

until $\max_{\boldsymbol{\mu} \in M_{\text{train}}} \eta_N(\boldsymbol{\mu}) \leq \varepsilon_{\text{tol}}$ *or* $N > N_{\max}$

return *reduced data:* Φ_N

BASIC-greedy algorithm

BASIC-GREEDY$(M_{\text{train}}, \varepsilon_{\text{tol}}, N_{\max})$

– *Initialize reduced basis of dimension* Υ_0*:*

$\Phi_{N_0} \leftarrow$ INITBASIS()

$N \leftarrow N_0$

repeat

– *Find worst approximated parameter:*

$(\boldsymbol{\mu}_{\max}) \leftarrow \arg\max_{\boldsymbol{\mu} \in M_{\text{train}}} \eta_N(\boldsymbol{\mu})$

– *Extend reduced basis by snapshot:*

$\Phi_{N+1} \leftarrow \Phi_N \oplus u_h(\boldsymbol{\mu})$

$N \leftarrow N + 1$

until $\max_{\boldsymbol{\mu} \in M_{\text{train}}} \eta_N(\boldsymbol{\mu}) \leq \varepsilon_{\text{tol}}$ *or* $N > N_{\max}$

return *reduced data:* Φ_N

Convergence of greedy algorithm

Convergence depends on parametrization, i.e. manifold $\mathcal{S}(\mathcal{P}) := \{u_h(\boldsymbol{\mu}) | \boldsymbol{\mu} \in \mathcal{P}\}$.

Kolmogorov N-width

$$d_N(\mathcal{S}(\mathcal{P}), \mathcal{W}_h) := \inf_{\substack{\hat{\mathcal{W}} \subset \mathcal{W}_h \\ \dim(\hat{\mathcal{W}})=N}} \sup_{v_h \in \mathcal{S}(\mathcal{P})} \min_{u_h \in \hat{\mathcal{W}}} \|v_h - u_h\|_{\mathcal{W}_h} \tag{12}$$

Binev et al., 2011 show: Exponential or polynomial convergence of $d_N(\mathcal{S}(\mathcal{P}), \mathcal{W}_h)$ with growing N is inherited by Greedy algorithm.

Convergence of greedy algorithm

Convergence depends on parametrization, i.e. manifold $\mathcal{S}(\mathcal{P}) := \{u_h(\boldsymbol{\mu}) | \boldsymbol{\mu} \in \mathcal{P}\}$.

Kolmogorov N-width

$$d_N(\mathcal{S}(\mathcal{P}), \mathcal{W}_h) := \inf_{\substack{\hat{\mathcal{W}} \subset \mathcal{W}_h \\ \dim(\hat{\mathcal{W}})=N}} \sup_{v_h \in \mathcal{S}(\mathcal{P})} \min_{u_h \in \hat{\mathcal{W}}} \|v_h - u_h\|_{\mathcal{W}_h} \tag{12}$$

Binev et al., 2011 show: Exponential or polynomial convergence of $d_N(\mathcal{S}(\mathcal{P}), \mathcal{W}_h)$ with growing N is inherited by Greedy algorithm.

But: It is infeasible to obtain convergence constants a priori.

$\Rightarrow$

1. smart basis generation algorithms
2. rapid protoyping of reduced basis methods

Convergence of greedy algorithm

Convergence depends on parametrization, i.e. manifold $\mathcal{S}(\mathcal{P}) := \{u_h(\boldsymbol{\mu}) | \boldsymbol{\mu} \in \mathcal{P}\}$.

Kolmogorov N-width

$$d_N(\mathcal{S}(\mathcal{P}), \mathcal{W}_h) := \inf_{\substack{\hat{\mathcal{W}} \subset \mathcal{W}_h \\ \dim(\hat{\mathcal{W}})=N}} \sup_{v_h \in \mathcal{S}(\mathcal{P})} \min_{u_h \in \hat{\mathcal{W}}} \|v_h - u_h\|_{\mathcal{W}_h} \tag{12}$$

Binev et al., 2011 show: Exponential or polynomial convergence of $d_N(\mathcal{S}(\mathcal{P}), \mathcal{W}_h)$ with growing N is inherited by Greedy algorithm.

But: It is infeasible to obtain convergence constants a priori.

$\Rightarrow$

1. smart basis generation algorithms
2. rapid protoyping of reduced basis methods

Table of Contents

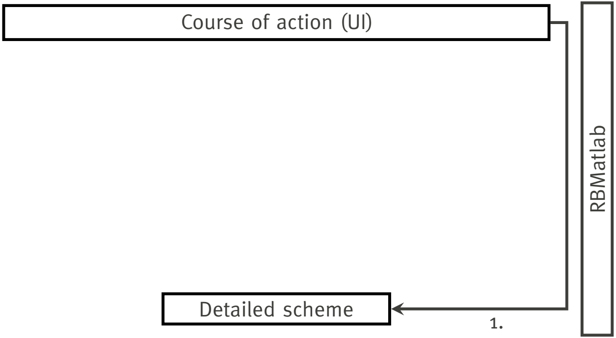
Goals specification

- Abstract reduced basis framework
- Re-use of (existing) implementations of PDE discretizations
- Rapid prototyping of reduced basis methods

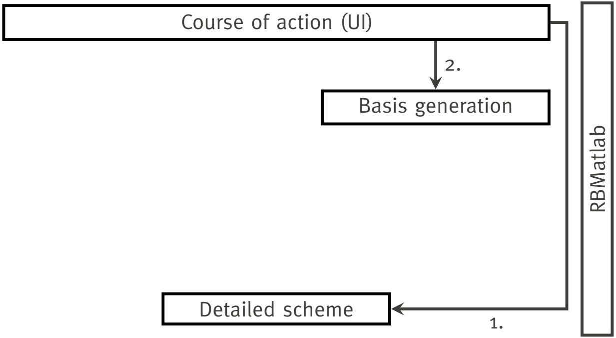
Course of action in RBmatlab

1.	`sim=detailed_simulation(model, model_data)`	Computes parameter dependent high dimensional solutions $u_h(\boldsymbol{\mu})$.
2.	`detailed_data=gen_detailed_data(model, model_data)`	Generates reduced basis space (Greedy algorithm).
3.	`reduced_data=gen_reduced_data(model, detailed_data)`	Generates reduced vectors and matrices using the method `rb_operators`.
4.	`rb_sim=rb_simulation(model, reduced_data)`	Computes a reduced solution Dofs $\underline{u_{\text{red}}}(\boldsymbol{\mu})$ and the a posteriori error estimate $\eta(\boldsymbol{\mu})$.
5.	`sim=rb_reconstruction(model, detailed_data, rb_sim)`	Computes a reduced solution $u_{\text{red}}(\boldsymbol{\mu})$.
6.	`p=plot_sim_data(model, detailed_data, sim)`	Visualizes a vector given by `sim`.

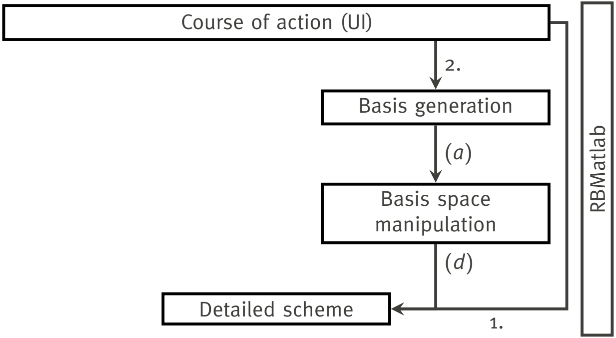
RBmatlab method call graph



RBmatlab method call graph

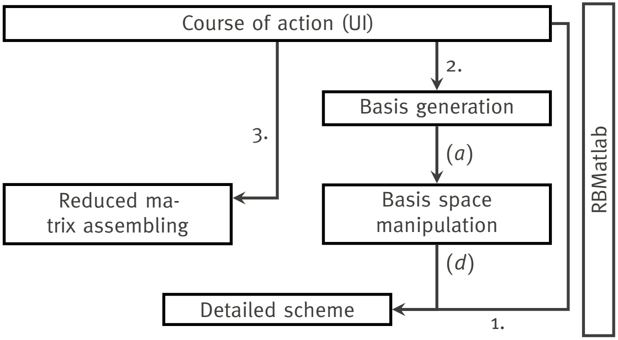


RBmatlab method call graph

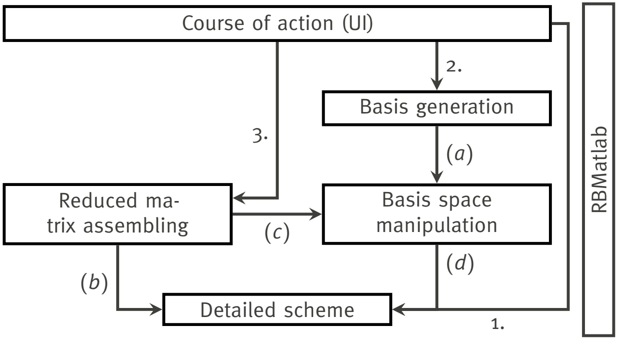


Martin Drohmann (mdrohmann@wwu.de)

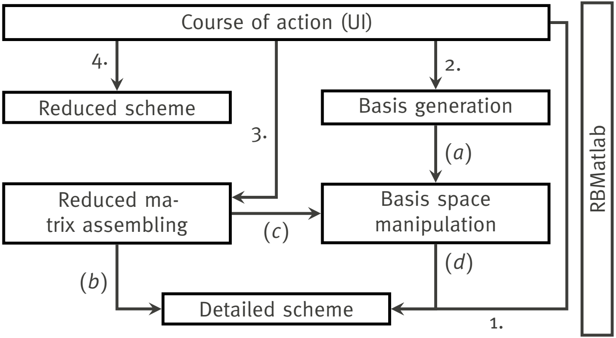
RBmatlab method call graph



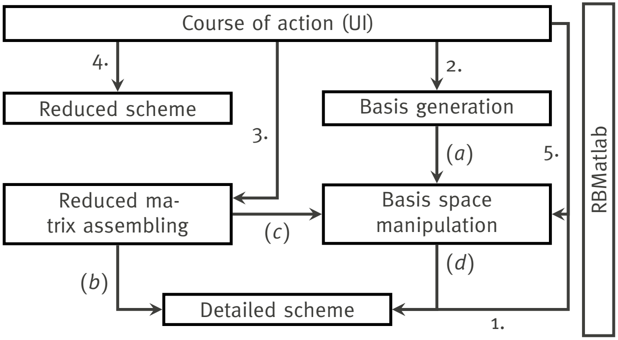
RBmatlab method call graph



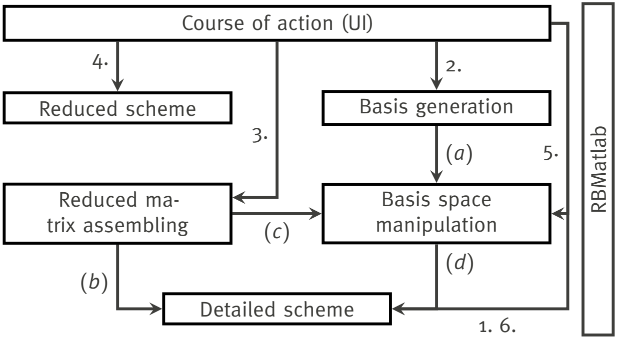
RBmatlab method call graph



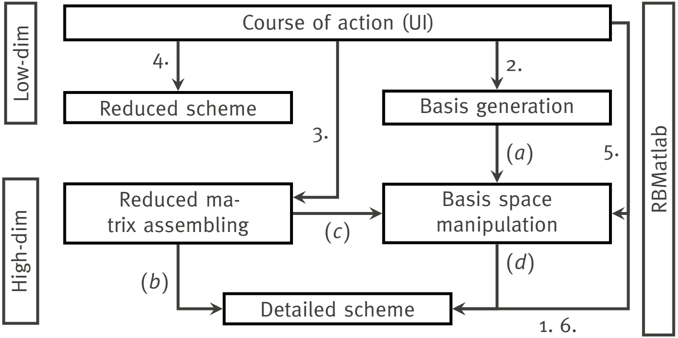
RBmatlab method call graph



RBmatlab method call graph



RBmatlab method call graph



Martin Drohmann (mdrohmann@wwu.de)

Observations: High–dimensional computations

Tier I

The detailed scheme needs to fulfill an interface:

- `solve()` method
- export of (separable) system matrices and vectors (sparse)
- `visualize()` method

Observations: High–dimensional computations

Tier I

The detailed scheme needs to fulfill an interface:

- `solve()` method
- export of (separable) system matrices and vectors (sparse)
- `visualize()` method

Tier II

- mostly problem independent
- works on dense matrices

Observations: High–dimensional computations

Tier I

The detailed scheme needs to fulfill an interface:

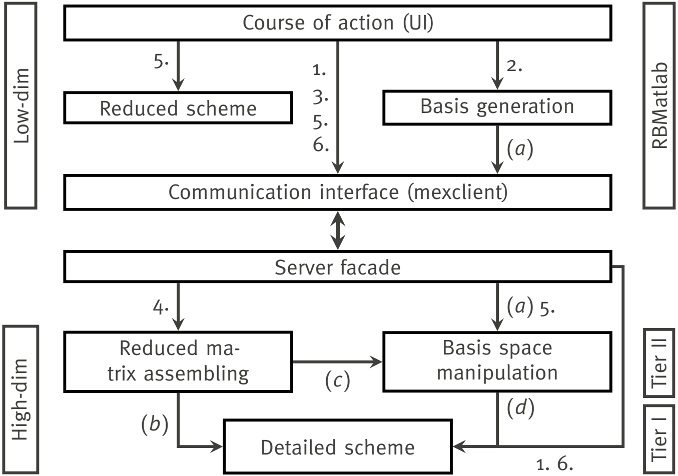
- `solve()` method
- export of (separable) system matrices and vectors (sparse)
- `visualize()` method

Tier II

- mostly problem independent
- works on dense matrices

Idea: Separate from low–dimensional computations
$\Rightarrow$ Specialized hardware, and software framework

Abstract software concept



Martin Drohmann (mdrohmann@wwu.de)

High–dimensional computations (DUNE-RB)

- Storage / manipulation of reduced spaces
- Efficient high dimensional linear algebra algorithms
  - POD, orthonormalization, Gram–Matrix computations
- Parametrization
- (Implemented in C++)
  - based on DUNE core modules (http://dune-project.org)

The glue

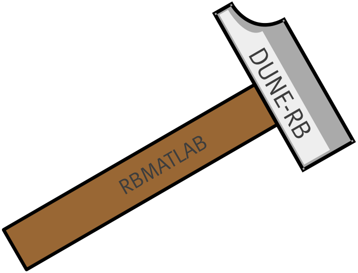
- Communication between Dune-RB and RBmatlab can be realized by
  1. compiling Dune-RB example as (mex-) library for matlab, or
  2. TCP/IP communication between two stand–alone applications.

Serialization of Matlab data containers:
dense matrices, structs, cell arrays, strings

Martin Drohmann (mdrohmann@wwu.de)

Summary

We can build reduced basis hammers for discrete problems...



Summary

And only need to make our problems look like nails!

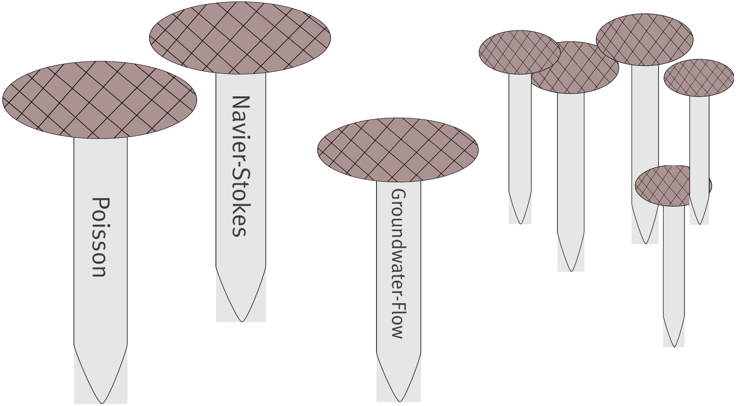


Table of Contents

Implementation details

- Finite volume discretization
- Based on Dune-PdeLab operators
- Linear algebra on both tiers with Eigen (http://eigen.tuxfamily.org)
- Visualization: vtk file output

Poisson problem

Domain: $\Omega \subset \mathbb{R}^d, d = 2, 3$.

$$
\begin{aligned}
-k\Delta u - mu &= 1 && \text{in } \Omega \\
u &= 0 && \text{on } \partial\Omega
\end{aligned}
$$

$\boldsymbol{\mu} := (k, m) \in \mathcal{P} := [1, 10] \times [0, 0.2]$.

Results

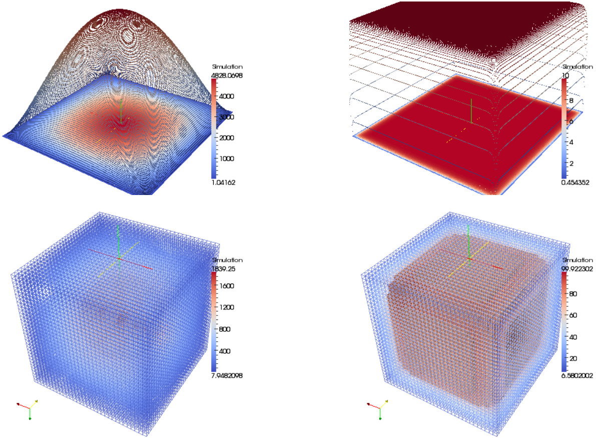


Figure: $\boldsymbol{\mu} = (1, 0)$ and $\boldsymbol{\mu} = (1, 0.1)$, Second row: 3D problem $\boldsymbol{\mu} = (1, 0)$ and $\boldsymbol{\mu} = (1, 0.01)$

Results

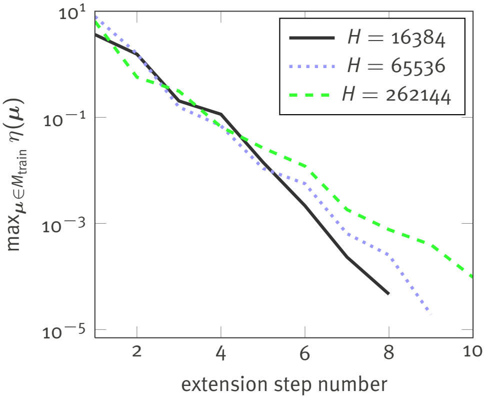


Figure: Error convergence of greedy algorithm on 200 random parameters, $\varepsilon_{\mathrm{tol}} = 10^{-4}$.

Results

H	N	max. error	detailed	ø-time[s] reduced	reconstr.	offline-time[s]
4,096	8	$8.81 \cdot 10^{-4}$	0.33	$1.53 \cdot 10^{-6}$	0.092	4.32
16,384	9	$8.92 \cdot 10^{-4}$	2.9	$1.02 \cdot 10^{-5}$	0.259	27.82
65,536	10	$5.75 \cdot 10^{-5}$	39.84	$9.37 \cdot 10^{-4}$	0.987	399.93
262,144	11	$2.15 \cdot 10^{-4}$	621.62	$9.33 \cdot 10^{-4}$	5.679	6,793.21
32,768	9	$3.61 \cdot 10^{-5}$	13.75	$8.32 \cdot 10^{-4}$	1.025	113.85

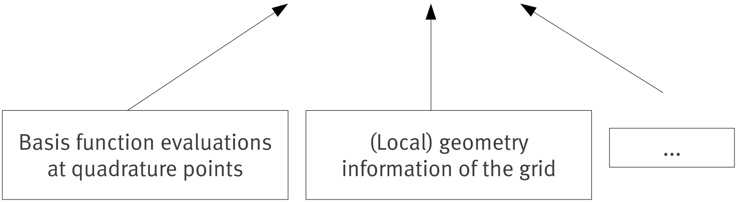
Table: Validation over 100 random test parameters. Last row $d = 3$.

Table of Contents

Interface to (non–linear) PDE discretizations

Export of local operator evaluations

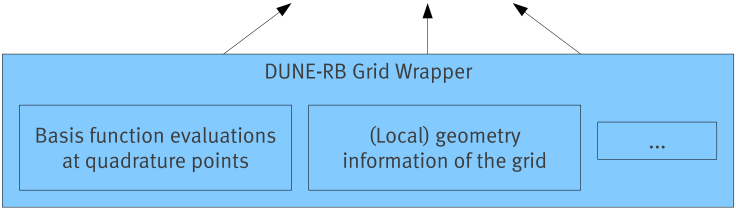
$$\mathcal{L}_{h,}(\boldsymbol{\mu})[\cdot](x_m)$$



Interface to (non–linear) PDE discretizations

Export of local operator evaluations

$$\mathcal{L}_{h,}(\boldsymbol{\mu})[\cdot](x_m)$$



Dune-RB grid wrapper

- During detailed simulation
  - Delegate calls directly to the grid
- During offline phase
  - Store all grid and function space information on the subgrid in low–dimensional data structures
- During online phase
  - Delegate calls low–dimensional data structures generated in offline phase.

More information

http://morepas.org/software