Reduced Basis methods with a DUNE–Matlab–Communication–Interface
Outline

Reduced basis method

RBmatlab–Dune-RB-Interface

dune-rb Module

RBMatlab
Reduced Basis Method Overview

RB Scenario:
- Parametrized partial differential equations for (non-stationary) problems
- Applications relying on time-critical or many repeated simulations, e.g. design, control, optimization applications.

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

References: [Patera&Rozza, 2006], [Haasdonk et al., 2008]
Reduced Basis Method Overview

RB Scenario:
- Parametrized partial differential equations for (non-stationary) problems
- Applications relying on time-critical or many repeated simulations, e.g. design, control, optimization 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)  http://morepas.org
Reduced Basis Method Overview

RB Scenario:
- Parametrized partial differential equations for (non-stationary) problems
- Applications relying on time-critical or many repeated simulations, e.g. design, control, optimization applications.

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

References: [Patera&Rozza, 2006], [Haasdonk et al., 2008]
Reduction Basis Method Overview

RB Scenario:
- Parametrized partial differential equations for (non-stationary) problems
- Applications relying on time-critical or many repeated simulations, e.g. design, control, optimization applications.

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

References: [Patera&Rozza, 2006], [Haasdonk et al., 2008]
Reduced Basis Method Overview

RB Scenario:
- Parametrized partial differential equations for (non-stationary) problems
- Applications relying on time-critical or many repeated simulations, e.g. design, control, optimization 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)  http://morepas.org
Derivation of Reduced Numerical Scheme

1. Analytical formulation

For $\mu \in \mathcal{P} \subset \mathbb{R}^p$, find $u : [0, T_{\text{max}}] \rightarrow \mathcal{W} \subset L^2(\Omega)$, s.t.

$$\partial_t u(t) - \mathcal{L}(\mu)[u(t)] = 0,$$

$$u(0) = u_0(\mu)$$

plus (parameter dependent) boundary conditions.
Derivation of Reduced Numerical Scheme

1. Analytical formulation

For \( \mu \in \mathcal{P} \subset \mathbb{R}^p \), find \( u : [0, T_{\text{max}}] \rightarrow \mathcal{W} \subset L^2(\Omega) \), s.t.

\[
\partial_t u(t) - \mathcal{L}(\mu)[u(t)] = 0, \quad u(0) = u_0(\mu)
\]

plus (parameter dependent) boundary conditions.

2. Discretization (e.g. FV, FE, DG)

For \( \mu \in \mathcal{P} \) compute \( \{u_h^k(\mu)\}_{k=0}^K \subset \mathcal{W}_h \subset L^2(\Omega) \) by

\[
\begin{align*}
  u_h^0(\mu) &:= P_h[u_0(\mu)] \\
  u_h^k(\mu) &:= u_h^{k-1}(\mu) + \Delta t \mathcal{L}_h(\mu)[u_h^{k-1}(\mu)].
\end{align*}
\]
Derivation of Reduced Numerical Scheme (cont.)

- Generate reduced basis space $\mathcal{V}_{\text{red}} := \text{span} \{ \varphi_i \}_{i=1}^N \subset \mathcal{V}_h$ with POD-Greedy algorithm.

- Assume separable initial data and operator

$$u_0 = \sum_{q=1}^{Q_{u_0}} \sigma_{u_0}^q(\mu) u_0^q, \quad \mathcal{L} = \sum_{q=1}^{Q_L} \sigma_L^q(\mu) \mathcal{L}^q$$

with parameter dependent and independent parts.

- Compute offline–vectors and –matrices from parameter independent contributions for numerical scheme

$$(P_0^q, q)_n = \int_{\Omega} u_0^q \varphi_n, \quad q = 1, \ldots, Q_{u_0}, n = 1, \ldots, N$$

$$(L^q)_{n,m} = \int_{\Omega} \mathcal{L}^q [\varphi_m] \varphi_n, \quad q = 1, \ldots, Q_L, m, n = 1, \ldots, N$$
Derivation of Reduced Numerical Scheme (cont.)

Generate reduced basis space $\mathcal{W}_{\text{red}} := \text{span} \{ \phi_i \}_{i=1}^N \subset \mathcal{W}_h$ with POD-Greedy algorithm.

Assume separable initial data and operator

$$ u_0 = \sum_{q=1}^{Q_{u0}} \sigma_{u_0}^q(\mu) u_0^q \quad \mathcal{L} = \sum_{q=1}^{Q_{L}} \sigma_{L}^q(\mu) \mathcal{L}^q $$

with parameter dependent and independent parts.

Compute offline–vectors and –matrices from parameter independent contributions for numerical scheme

$$ (P^0,q)_n = \int_\Omega u_0^q \phi_n, \quad q = 1, \ldots, Q_{u0}, n = 1, \ldots, N $$

$$ (L^q)_{n,m} = \int_\Omega \mathcal{L}^q [\phi_m] \phi_n, \quad q = 1, \ldots, Q_{L}, m, n = 1, \ldots, N $$

and a posteriori error estimation (not discussed here).
3. Reduced numerical scheme

For $\mu \in P$ compute $\lbrace u_{\text{red}}^k(\mu) \rbrace_{k=0}^K \subset \mathcal{W}_{\text{red}} \subset \mathcal{W}_h$ by

\[
\begin{align*}
(u_{\text{red}}^0(\mu) - P_h[u_0(\mu)], \varphi) &= 0 \quad \forall \varphi \in \mathcal{W}_{\text{red}}, \\
(u_{\text{red}}^k(\mu) - u_{\text{red}}^{k-1}(\mu) + \Delta t \mathcal{L}_h(\mu) [u_{h}^{k-1}(\mu)], \varphi) &= 0 \quad \forall \varphi \in \mathcal{W}_{\text{red}},
\end{align*}
\]
3. Reduced numerical scheme

For $\mu \in \mathcal{P}$ compute $\{u_{\text{red}}^k(\mu)\}_{k=0}^K \subset \mathcal{W}_{\text{red}} \subset \mathcal{W}_h$ by

$$(u_{\text{red}}^0(\mu) - P_h[u_0(\mu)], \varphi) = 0 \quad \forall \varphi \in \mathcal{W}_{\text{red}},$$

$$(u_{\text{red}}^k(\mu) - u_{\text{red}}^{k-1}(\mu) + \Delta t L_h(\mu) [u_h^{k-1}(\mu)], \varphi) = 0 \quad \forall \varphi \in \mathcal{W}_{\text{red}},$$

or equivalently compute vectors $a^k(\mu) \subset \mathbb{R}^N$ for $k = 0, \ldots, K$ such that

$$u_{\text{red}}^k(\mu) = \sum_{n=1}^N a_n^k \varphi_n$$

by

$$a^0(\mu) := \sum_{q=1}^{Q_{\mu_0}} \sigma_{\mu_0}^q(\mu) P_{\mu_0}^{q},$$

$$a^k(\mu) := a^{k-1}(\mu) + \Delta t \sum_{q=1}^{Q_L} \sigma_L^q(\mu) L^q a^{k-1}.$$
Reduced basis generation

Use error estimator $\eta(\mu)$, s.t. $\max_{k=0,\ldots,K} \| u_h^k(\mu) - u_{\text{red}}^k(\mu) \|_{W_h} \leq \eta(\mu)$. 

POD-greedy algorithm

**INPUT:** $M_{\text{train}} \subset P$, $\varepsilon_{\text{tol}}$, $N_{\text{max}}$

**OUTPUT:** $W_{\text{red}}$

1. Initialize reduced basis:
   $\Phi_{N_0} \leftarrow \{ \phi_n \}_{n=1}^{N_0}$

2. Repeat
   1. Find worst approximated trajectory:
      $\mu_{\text{max}} \leftarrow \arg \max_{\mu \in M_{\text{train}}} \eta(\mu)$
   2. Compute trajectory
      $\bar{\mathcal{u}}_k^h(\mu_{\text{max}})_{k=0}^K$
   3. Compute new basis function:
      $\phi_{N+1} \leftarrow \text{POD} \left( \bar{\mathcal{u}}_k^h(\mu_{\text{max}}) - P_{\text{red}} \hat{\mathcal{u}}_k^h(\mu_{\text{max}})_{k=0}^K \right)$
      $N \leftarrow N + 1$
   Until $\eta(\mu_{\text{max}})$ falls beneath given tolerance $\varepsilon_{\text{tol}}$ or $N = N_{\text{max}}$.

$W_{\text{red}} \leftarrow \text{span} \{ \phi_n \}_{n=1}^N$
Reduced basis generation

Use error estimator $\eta(\mu)$, s.t. $\max_{k=0,\ldots,K} \| u_h^k(\mu) - u_{\text{red}}^k(\mu) \|_{\mathcal{V}_h} \leq \eta(\mu)$.

POD-greedy algorithm

**INPUT:** $M_{\text{train}} \subset \mathcal{P}$, $\varepsilon_{\text{tol}}$, $N_{\text{max}}$

**OUTPUT:** $\mathcal{V}_{\text{red}}$

*Initialize reduced basis:*

\[
\Phi_{N_0} \leftarrow \{ \varphi_n \}_{n=1}^{N_0}
\]

$N \leftarrow N_0$
Reduced basis generation

Use error estimator $\eta(\mu)$, s.t. $\max_{k=0,\ldots,K} \| u_h^k(\mu) - u_{\text{red}}^k(\mu) \|_{\mathcal{W}_h} \leq \eta(\mu)$.

POD-greedy algorithm

\textbf{INPUT:} $M_{\text{train}} \subset \mathcal{P}$, $\varepsilon_{\text{tol}}$, $N_{\text{max}}$

\textbf{OUTPUT:} $\mathcal{W}_{\text{red}}$

\textit{Initialize reduced basis:}
\[
\Phi_{N_0} \leftarrow \{ \varphi_n \}_{n=1}^{N_0} \\
N \leftarrow N_0
\]

\textbf{repeat}
\begin{enumerate}
\item \textit{Find worst approximated trajectory:} $\mu_{\text{max}} \leftarrow \arg \max_{\mu \in M_{\text{train}}} \eta(\mu)$
\item \textit{Compute trajectory} $\{ u_h^k(\mu_{\text{max}}) \}_{k=0}^{K}$.
\item \textit{Compute new basis function:}
\[
\varphi_{N+1} \leftarrow \text{POD} \left( \{ u_h^k(\mu_{\text{max}}) - P_{\text{red}} [u_h^k(\mu_{\text{max}})] \}_{k=0}^{K} \right) \\
N \leftarrow N + 1
\]
\end{enumerate}
\textbf{until} $\eta(\mu_{\text{max}})$ falls beneath given tolerance $\varepsilon_{\text{tol}}$ \textbf{or} $N = N_{\text{max}}$. \\
$\mathcal{W}_{\text{red}} \leftarrow \text{span} \{ \varphi_n \}_{n=1}^{N}$
Reduced basis generation

Use error estimator $\eta(\mu)$, s.t. $\max_{k=0,\ldots,K} \| u_h^k(\mu) - u_{\text{red}}^k(\mu) \|_{W_h} \leq \eta(\mu)$.

**POD-greedy algorithm**

**INPUT:** $M_{\text{train}} \subset P$, $\varepsilon_{\text{tol}}$, $N_{\text{max}}$

**OUTPUT:** $W_{\text{red}}$

Initialize reduced basis:

$\Phi_{N_0} \leftarrow \{ \varphi_n \}_{n=1}^{N_0}$

$N \leftarrow N_0$

repeat

1. Find worst approximated trajectory: $\mu_{\text{max}} \leftarrow \arg \max_{\mu \in M_{\text{train}}} \eta(\mu)$

2. Compute trajectory $\{ u_h^k(\mu_{\text{max}}) \}_{k=0}^{K}$.

3. Compute new basis function:

$$\varphi_{N+1} \leftarrow \text{POD} \left( \{ u_h^k(\mu_{\text{max}}) - P_{\text{red}} [ u_h^k(\mu_{\text{max}}) ] \}_{k=0}^{K} \right)$$

$N \leftarrow N + 1$

until $\eta(\mu_{\text{max}})$ falls beneath given tolerance $\varepsilon_{\text{tol}}$ or $N = N_{\text{max}}$.

$W_{\text{red}} \leftarrow \text{span} \{ \varphi_n \}_{n=1}^{N}$
What else?

- Reduced basis methods also work for nonlinear schemes and non-separable operators. (⇒ empirical interpolation)
- A posteriori error estimators with offline/-online decomposition
- More sophisticated reduced basis generation algorithms
What else?

- Reduced basis methods also work for nonlinear schemes and non-separable operators. (⇒ empirical interpolation)
- A posteriori error estimators with offline/-online decomposition
- More sophisticated reduced basis generation algorithms

But you got the idea:

1. Usage of existing numerical schemes for basis generation
2. Low-dimensional and efficient online computations
What else?

- Reduced basis methods also work for nonlinear schemes and non-separable operators. (⇒ empirical interpolation)
- A posteriori error estimators with offline/-online decomposition
- More sophisticated reduced basis generation algorithms

But you got the idea:

1. Usage of existing numerical schemes for basis generation
2. Low-dimensional and efficient online computations

For 1, we want to use DUNE, for 2, we don’t.

Martin Drohmann (mdrohmann@wwu.de)  http://morepas.org
Motivation: Comparison of Matlab and Dune

<table>
<thead>
<tr>
<th>Matlab</th>
<th>Dune</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ Easy to learn and use</td>
<td>+ Flexible and efficient</td>
</tr>
<tr>
<td>+ Huge library of mathematical functions (statistical data, postprocessing, plots,…)</td>
<td>+ Provides complex numerical schemes</td>
</tr>
<tr>
<td>- Slow for interpreted code parts</td>
<td>- Less easy to learn</td>
</tr>
<tr>
<td>- Memory constraints</td>
<td></td>
</tr>
</tbody>
</table>
Motivation: Comparison of Matlab and Dune

Matlab

+ Easy to learn and use
+ Huge library of mathematical functions (statistical data, postprocessing, plots, ...)
- Slow for interpreted code parts
- Memory constraints

Dune

+ Flexible and efficient
+ Provides complex numerical schemes
- Less easy to learn

⇒ Separate implementation of offline-/online-phases in RBmatlab and dune-rb, respectively.

Martin Drohmann (mdrohmann@wwu.de)  http://morepas.org
high dim computation
communication of low dim data
low dim computation

Control structures
▶ parameter
▶ model

Offline data
▶ reduced basis space
▶ grid
▶ high dim operators

Visualization

Solvers

Network

Server

client

Reduced simulation
▶ "POD-greedy"

Visualization control

Error estimators

Visualization

RBmatlab

Comsol

dune

RBmatlab

others

Martin Drohmann (mdrohmann@wwu.de)  http://morepas.org
Control structures
▶ parameter
▶ model
Visualization
Reconstruction

Offline data
▶ reduced basis space
▶ grid
▶ high dim operators

Solvers
dune
comsol
RBmatlab
others

Network

Server

Client

direct mexfunction usage

RBmatlab

Reduced simulation
Error estimators

Visualization control
Visualization

Reduced basis generation
▶ “POD-greedy”

high dim computation

communication of low dim data

low dim computation

Martin Drohmann (mdrohmann@wwu.de)  http://morepas.org
Data Structures

Matlab provides a C data structure `mxArray*` that can be used as

- Matrix
- String
- Container-Type (Struct, Cell-Array)

Dune-rb provides C++ wrappers

- `MXArray` (internal data stored as `mxArray*`)
- `RBArray` (internal data stored as `double*`, `std::string`)

derived from a common interface `SerializedArray`.
Transmission of Data

Use dune-rb as Matlab (mex) library

Use dune-rb as standalone server
Communication with mex Library

```
[ret]=
duneclient('command',args);
```

RBM Matlab

dune-rb mex library
Communication with mex Library

```
[ret]=
duneclient('command',args);
```

```
wrap 'command', args,
ret in MXArray objects
mexFunction(plhs, prhs);
unwrap 'mxArray* ret'
from return values or error
handling
```
Communication with mex Library

```matlab
[ret] = duneclient('command', args);
```

```
wrap 'command', args, ret in MXArray objects
mexFunction(plhs, prhs);
```

Martin Drohmann (mdrohmann@wwu.de)  http://morepas.org
Communication with mex Library

```
[ret] = duneclient('command', args);
```

- `duneclient` function call with 'command' and `args`
- Return value `ret` passed to `mexFunction` along with other arguments `plhs`, `prhs`
- `wrap` function wraps 'command', `args`, and `ret` in `MXArray` objects
- `mexFunction` call with `plhs`, `prhs`
- `unwrap` function unwraps `mxArray* ret` from return values or error handling

Martin Drohmann (mdrohmann@wwu.de) http://morepas.org
Communication with mex Library

```
[ret] =
duneclient('command',args);
```

```
"wait"
```

wrap 'command', args,
ret in MXArray objects

```
mexFunction(plhs, prhs);
```

unwrap 'mxArray* ret'
from return values or error
handling
Communication over TCP/IP

RBMatlab

mexclient('init_server',
struct('serverhost',
'localhost', 'port', 1234))

dune-rb server

./dunerb rb.servermode: true
rb.port: 1234;
Communication over TCP/IP

RBMatlab

mexclient('init_server', ...)

[ret] =
mexclient('command', args);

mexclient

dune-rb server

./dunerb rb.servermode ...;

“wait”

mexFunction(plhs, prhs);

 serialize data as MXArray

send return values or error handling

Martin Drohmann (mdrohmann@wwu.de) http://morepas.org
Communication over TCP/IP

RBMatlab

mexclient('init_server', ...)

[ret] =
mexclient('command', args);

mexclient

wrap 'command', args, ret
in MXArray objects

dune-rb server

=./dunerb rb.servermode...

"wait"

serialize data as RBArray objects

mexFunction(plhs, prhs);

serialize data as MXArray
send return values or error handling

Martin Drohmann (mdrohmann@wwu.de) http://morepas.org
Communication over TCP/IP

**RBMatlab**

```matlab
mexclient('init_server', ...)
```

```matlab
[ret] =
mexclient('command', args);
```

**dune-rb server**

```bash
./dunerb rb.servermode...
```

```matlab
mexclient('command', args, ret)
```

```
"wait"
```

```
"wait"
```

serialize data as `RBArray` objects

```
"wait"
```

```matlab
mexFunction(plhs, prhs);
```

wrap 'command', args, ret in `MXArray` objects

serialize data as `MXArray` objects

send return values or error handling

Martin Drohmann (mdrohmann@wwu.de)  http://morepas.org
Communication over TCP/IP

**RBMatlab**

```matlab
mexclient('init_server', ...)
```

```matlab
[ret]=
mexclient('command',args);
```

**dune-rb server**

```bash
./dunerb rb.servermode...
```

```
"wait"
mexFunction(plhs, prhs);
```

serialize data as RBArray objects

```
 serialize data as MXArray
```

send return values or error handling
dune-rb Module

Goals

- Enhance dune-fem operators with parametrization
  - Affine parametrized operators
  - Library of finite volume operators and problems
- Storage and management of reduced bases
  - discrete function lists
  - reduced basis space
- High-dimensional matrix computations
  - PCA
  - Gramian matrix computation
- Reconstruction and visualization of reduced basis solutions.

Dependencies

- dune-common, dune-grid
- dune-fem
- LAPACK (for PCA)
- Matlab (for mexclient)

Martin Drohmann (mdrohmann@wwu.de) http://morepas.org
dune-rb Module

Goals

- Enhance dune-fem operators with parametrization
- Storage and management of reduced bases
- High-dimensional matrix computations
- Reconstruction and visualization of reduced basis solutions.

Dependencies

- dune-common, dune-grid
- dune-fem
- LAPACK (for PCA)
- Matlab (for mexclient)
high dim computation
communication of low dim data
low dim computation

dune
comsol
RBmatlab
others

Control structures
- parameter
- model

Offline data
- reduced basis space
- grid
- high dim operators

Visualization

dune-rb

Visualization control

Reduced basis generation
- "POD-greedy"

Network

Server

Client

RBmatlab

Reduced simulation

Error estimators

Visualization

RBmatlab

direct mexfunction usage

Martin Drohmann (mdrohmann@wwu.de) http://morepas.org
Separable Discrete Operators

\[ \mathcal{L} := \sum_{q=1}^{Q} \sigma_q^L \mathcal{L}^q \]

Implement a class for each summand as instance of \texttt{LocalParametrizedOperatorInterface}.

\begin{verbatim}
LocalParametrizedOperatorInterface
  ▶ coefficient()
  ▶ symbolicCoefficient()
  ▶ apply() [inherited]
\end{verbatim}

\begin{verbatim}
FinVol

FVLaxFriedrichs  FVDiffusion
\end{verbatim}
Separator Discrete Operators

\[ \mathcal{L} := \sum_{q=1}^{Q} \sigma_{L}^{q} \mathcal{L}^{q} \]

Implement a class for each summand as instance of 
LocalParametrizedOperatorInterface.

LocalParametrizedOperatorInterface

- coefficient()
- symbolicCoefficient()
- apply() [inherited]

FinVol

FVLaxFriedrichs FVDiffusion

Martin Drohmann (mdrohmann@wwu.de) http://morepas.org
Separable Discrete Operators

\[ \mathcal{L}[u] := \sum_{q=1}^{Q} \sigma^q \mathcal{L}^q[u] \]

Implement a class for each summand as instance of `LocalParametrizedOperatorInterface`.

- `coefficient()`
- `symbolicCoefficient()`
- `apply()` [inherited]
Separable Discrete Operators

\[ \mathcal{L} := \sum_{q=1}^{Q} \sigma_{L}^{q} \mathcal{L}^{q} \]

Implement a class for each summand as instance of LocalParametrizedOperatorInterface.

typedef Tuple<LocalParametrizedOperatorImp,...> LPOTuple;
LPOTuple lpoTuple(lpo1,...);
DiscreteDecomposedOperator<LPOTuple,
    DiscreteFunction, // type of domain
    DiscreteFunction> // type of range
    ddo(discreteFunctionSpace, lpoTuple);

ddo.coefficients();
ddo.complete();
ddo.component(i);
Reduced Basis Space (by M. Nolte)

- Derived from DiscreteFunctionSpaceDefault
Reduced Basis Space (by M. Nolte)

- Derived from DiscreteFunctionSpaceDefault
- baseFunctionSet(const Entity& en) provides restriction of the base functions to the entity en
Reduced Basis Space (by M. Nolte)

- Derived from DiscreteFunctionSpaceDefault
- baseFunctionSet(const Entity& en) provides restriction of the base functions to the entity en
- Allows manipulation of base functions:
Reduced Basis Space (by M. Nolte)

- Derived from DiscreteFunctionSpaceDefault
- baseFunctionSet(const Entity& en) provides restriction of the base functions to the entity en
- Allows manipulation of base functions:
  - addBaseFunction()
Reduced Basis Space (by M. Nolte)

- Derived from DiscreteFunctionSpaceDefault
- `baseFunctionSet(const Entity& en)` provides restriction of the base functions to the entity `en`
- Allows manipulation of base functions:
  - `addBaseFunction()`
  - `setBaseFunction()`
Reduced Basis Space (by M. Nolte)

- Derived from DiscreteFunctionSpaceDefault
- `baseFunctionSet(const Entity& en)` provides restriction of the base functions to the entity `en`
- Allows manipulation of base functions:
  - `addBaseFunction()`
  - `setBaseFunction()`
- Saves base functions in *discrete function list*
Discrete Function Lists

- Save/ load list of discrete functions

```cpp
typedef DiscreteFunctionList_xdr<DFTtype> ListType;
ListType list(discreteFunctionSpace, name);
list.push_back(function); // calls AttributeType()
list.push_back(function, attribute);
list.getFunc(i, destination);
list.getFuncByAttribute(attribute, destination);
```
Discrete Function Lists

- Save/ load list of discrete functions
  - in memory

```c
typedef DiscreteFunctionList_xdr<DType> ListType;
ListType list(discreteFunctionSpace, name);
list.push_back(function); // calls AttributeType()
list.push_back(function, attribute);
list.getFunc(i, destination);
list.getFuncByAttribute(attribute, destination);
```
Discrete Function Lists

- Save/load list of discrete functions
  - in memory
  - on hard disk

```
typedef DiscreteFunctionList_xdr<DFType> ListType;
ListType list(discreteFunctionSpace, name);
list.push_back(function);  // calls AttributeType()
list.push_back(function, attribute);
list.getFunc(i, destination);
list.getFuncByAttribute(attribute, destination);
```
Discrete Function Lists

- Save/ load list of discrete functions
  - in memory
  - on hard disk
- Save attribute (of arbitrary serializable type) for each function

```cpp
typedef DiscreteFunctionList_xdr<DFType> ListType;
ListType list(discreteFunctionSpace, name);
list.push_back(function); // calls AttributeType()
list.push_back(function, attribute);
list.getFunc(i, destination);
list.getFuncByAttribute(attribute, destination);
```
Discrete Function Lists

- Save/load list of discrete functions
  - in memory
  - on hard disk
- Save attribute (of arbitrary serializable type) for each function
- Access function by index or attribute

```cpp
typedef DiscreteFunctionList_xdr<DFType> ListType;
ListType list(discreteFunctionSpace, name);
list.push_back(function); // calls AttributeType()
list.push_back(function, attribute);
list.getFunc(i, destination);
list.getFuncByAttribute(attribute, destination);
```
Gramian Pipeline

- Efficient computation of Gramian-matrices with entries like $G_{ij} = (\mathcal{L} [\varphi_i], \varphi_j)$
- Few grid iterations
- Efficient memory management (optimization of hard disk access)

GramianPipeline pipe(DFList&)
OpHandle hId = pipe.getIdentityHandle();
OpHandle hL = pipe.registerDiscreteOperator(\mathcal{L});
pipe.addGramMatrixComputation(hL,hId, G);
... // further computations added
pipe.run();
Gramian Pipeline

- Efficient computation of Gramian-matrices with entries like $G_{ij} = \langle \mathcal{L} \varphi_i, \varphi_j \rangle$
- few grid iterations
- efficient memory management (optimization of hard disk access)

```cpp
GramianPipeline pipe(DFList&)
OpHandle hId = pipe.getIdentityHandle();
OpHandle hL = pipe.registerDiscreteOperator(\mathcal{L});
pipe.addGramMatrixComputation(hL, hId, G);
... // further computations added
pipe.run();
```
Principal Component Analysis

- Principal Component Analysis (PCA) of discrete function lists
- Uses LAPACK::dsyev
- Usage: pca(U, pcomps, ratio)

Prospects: Empirical Interpolation

The Empirical Interpolation puts arbitrary discrete operators into “separable” form.
Prospects: Empirical Interpolation

The Empirical Interpolation puts arbitrary discrete operators into “separable” form.
Prospects: Empirical Interpolation

The Empirical Interpolation puts arbitrary discrete operators into “separable” form.

References: [Barrault et al., 2004], [Haasdonk et al., 2008]
high dim computation
communication of low dim data

Control structures
▶ parameter
▶ model

Visualization

Reconstruction

Offline data
▶ reduced basis space
▶ grid
▶ high dim operators

Solvers

dune
comsol
RBmatlab
others

Network

Server

direct mexfunction usage

Client

low dim computation

Reduced simulation

Error estimators

Visualization control

Reduced basis generation
▶ “POD-greedy”

RBmatlab

Martin Drohmann (mdrohmann@wwu.de) http://morepas.org
RBMatlab

```matlab
model_data = gen_model_data(model)
  ▶ generate grid

detailed_data = gen_detailed_data(model, model_data)
  ▶ compute high dim simulations
  ▶ generate reduced basis space

reduced_data = gen_reduced_data(model, detailed_data)
  ▶ generate reduced matrices and vectors for reduced simulation and error estimation

sim_data = rb_simulation(model, reduced_data)
  ▶ perfom reduced simulation
  ▶ evaluate error estimators
```

Martin Drohmann (mdrohmann@wwu.de)  http://morepas.org
Further Capabilities of RBMatlab

- Different basis generation algorithms
Further Capabilities of RBMatlab

- Different basis generation algorithms
  - adaptive/fixed training set search in parameter space
Further Capabilities of RBMatlab

- Different basis generation algorithms
  - adaptive/fixed training set search in parameter space
  - multiple basis generation

Martin Drohmann (mdrohmann@wwu.de)  http://morepas.org
Further Capabilities of RBMatlab

- Different basis generation algorithms
  - adaptive/fixed training set search in parameter space
  - multiple basis generation
- Visualization

Martin Drohmann (mdrohmann@wwu.de)  http://morepas.org
Further Capabilities of RBMatlab

- Different basis generation algorithms
  - adaptive/fixed training set search in parameter space
  - multiple basis generation
- Visualization
- Post processing
Further Capabilities of RBMatlab

- Different basis generation algorithms
  - adaptive/fixed training set search in parameter space
  - multiple basis generation
- Visualization
- Post processing
- Empirical interpolation

Martin Drohmann (mdrohmann@wwu.de) http://morepas.org
Further Capabilities of RBMatlab

- Different basis generation algorithms
  - adaptive/fixed training set search in parameter space
  - multiple basis generation
- Visualization
- Post processing
- Empirical interpolation
- Finite volume discretization of parametrized PDEs
Thank you for your attention!