A Software Framework for Reduced Basis Methods using DUNE-RB and RBMATLAB

Martin Drohmann, Bernard Haasdonk, Sven Kaulmann, and Mario Ohlberger

Abstract Many applications from science and engineering are based on parametrized evolution equations and depend on time–consuming parameter studies or need to ensure critical constraints on the simulation time. For both settings, model order reduction by the reduced basis approach is a suitable means to reduce computational time. The method is based on a projection of an underlying high–dimensional numerical scheme onto a low–dimensional function space. In this contribution, a new software framework is introduced that allows fast development of reduced schemes for a large class of discretizations of evolution equations implemented in DUNE. The approach provides a strict separation of low–dimensional and high–dimensional computations, each implemented by its own software package RBMATLAB, respectively DUNE-RB. The functionality of the framework is exemplified for a finite–volume approximation of an instationary linear convection–diffusion problem.

1 Introduction

The reduced basis methods have gained increasing attention in recent years for stationary–elliptic, instationary–parabolic problems and various systems. In this contribution, we address the task of model reduction for parametrized scalar evolution equations. The functionality of the software framework described in this paper is restricted to linear problems, but during the development of the software concept, we always bear in mind, that an extension to non–linear problems based on empirical operator interpolation is possible (cf. [3]).

M. Drohmann, S. Kaulmann, and M. Ohlberger
Institute of Computational and Applied Mathematics, University of Münster, Einsteinstr. 62, 48149 Münster, e-mail: mdrohmann, s_kaul01, ohlberger@uni-muenster.de

B. Haasdonk
Institute of Applied Analysis and Numerical Simulation, University of Stuttgart, 70569 Stuttgart
e-mail: hassdonk@mathematik.uni-stuttgart.de

Software engineering is a major problem for the development of reduced basis methods, because a reduced basis framework for a certain problem involves many steps, which can be expensive in both development and execution time. In this presentation, we show how to develop an extendible framework for linear evolution equations, based on our DUNE module DUNE-RB and the Matlab based software package RBMATLAB. This separation into two software packages serves two purposes: First, the use of a high-level programming language like Matlab with a vast library of mathematical methods accelerates the development and improvement of the abstract reduced basis methods, while the underlying discretizations can still be implemented efficiently in DUNE-RB. Second, it makes use of the decomposition of the reduced basis method into two computationally different phases, frequently denoted by the offline-/online-decomposition. During the computationally and time demanding *offline-phase* a low–dimensional function space, spanned by high–dimensional solution snapshots, is generated, which can be used to reduce the model order of the problem to only few degrees of freedom. Therefore, during the *online-phase*, reduced simulations can be computed in real-time and independently from the original discretization and high–dimensional model data. In this phase, an interactive user interface, like the one offered by Matlab, turns out to be a useful gift. We would like to mention, that a similar approach for finite–element discretizations based on the numerical software package *libmesh* has recently been developed for a variety of problems with an affine parameter dependence of the underlying data functions. [7]

Section 2 contains a very basic description of the reduced basis method for linear evolution equations. In sections 3.1 and 3.2 the two software packages RBMATLAB and DUNE-RB are introduced in detail, in terms of functionality and the specification of their interfaces. Finally, Section 4 elaborates on an example implementation for a finite–volume discretization of a linear convection–diffusion problem. We describe how to implement this scheme in DUNE-RB and how to reduce it in RBMATLAB.

2 Reduced Basis Method

In this section, we give a very short introduction to the reduced basis method for linear evolution equations. For a detailed description of the framework, including discussions on the efficiency of reduced simulations and the generation of reduced basis spaces by the "POD-Greedy" - algorithm, we refer to [6, 9].

The present study assumes problems of the following kind: For every parameter vector $\mu \in \mathscr{D} \subset \mathbb{R}^p$, we are looking for a solution trajectory $u(\mu) : [0, T_{\max}] \to \mathscr{W} \subset L^2(\Omega)$ fulfilling some linear evolution equation of the form

$$\partial_t u(t;\mu) + \mathscr{L}[u(t;\mu)] = 0, \qquad u(0;\mu) = u_0(\mu)$$

with suitable boundary conditions. Here $\mathscr{W} \subset L^2(\Omega)$ is a Hilbert space of functions on the spatial domain Ω and $\mathscr{L} : \mathscr{W} \to \mathbb{R}$ denotes a spacial differential operator.

There is a large number of available numerical schemes for solving problems of this kind. Many of those, which are frequently used, like finite–element, finite–volume or discontinuous Galerkin discretizations can be cast into a framework, where the solutions are approximated in a discrete function space $\mathscr{W}_h \subset \mathscr{W}$ of dimension H. First–order time discretizations then lead, for a given parameter vector μ, to a sequence of solution snapshots $u_h^k(\mu) \in \mathscr{W}_h$ for $k = 1, \dots, K$ computed by an initial projection of the initial data function

$$u_h^0(\mu) = \mathscr{P}_h\left[u_0(\mu)\right] \tag{1a}$$

with a projection operator $\mathscr{P}_h : \mathscr{W} \to \mathscr{W}_h$ and iterative solutions of the equation

$$u_h^{k+1} + \Delta t^k \mathscr{L}_h^I(\mu)\left[u_h^{k+1}(\mu)\right] = u_h^k(\mu) - \Delta t^k \mathscr{L}_h^E(\mu)\left[u_h^k(\mu)\right] \tag{1b}$$

for $k = 1, \dots, K$ with suitable time–step sizes Δt^k. In the present study, we make the assumption that both the discrete operators $\mathscr{L}_h^I(\mu)$, $\mathscr{L}_h^E(\mu)$, the initial data function $\mathscr{P}_h\left[u_0(\mu)\right]$ and the constant function $b(\mu)$ depend affinely on the parameter, i.e. they can be written in separable forms

$$\begin{aligned} \mathscr{L}_h^I(\mu) &= \sum_{q=0}^{Q_I} \sigma_I(\mu)\mathscr{L}_h^{I,q}, & \mathscr{L}_h^E(\mu) &= b(\mu) + \sum_{q=0}^{Q_E} \sigma_E(\mu)\mathscr{L}_h^{E,q}, \\ \mathscr{P}_h\left[u_0(\mu)\right] &= \sum_{q=0}^{Q_{u_0}} \sigma_{u_0}^q(\mu) u_0^q, & b(\mu) &= \sum_{q=0}^{Q_b} \sigma_b^q(\mu) b^q, \end{aligned} \tag{2}$$

with parameter dependent coefficient functions $\sigma_I^q, \sigma_E^q, \sigma_{u_0}^q, \sigma_b^q : \mathscr{D} \to \mathbb{R}$, constant functions $u_0^q, b^q \in \mathscr{W}$ and linear operators $\mathscr{L}_h^{I,q}, \mathscr{L}_h^{E,q} : \mathscr{W} \to \mathscr{W}$ depending on the space variable only.

If we assume that there is an $N \ll H$ dimensional orthonormal basis $\Phi_N := \{\varphi_n\}_{n=1}^N$ for another discrete function space $\mathscr{W}_{\text{red}} \subset \mathscr{W}_h$ which somehow is a good approximation of the manifold of sought solutions $\left\{u_h^k(\mu) \mid \mu \in \mathscr{D}, k = 0, \dots, K\right\}$, the numerical scheme (1a)-(1b) can be reduced by a Galerkin projection onto this so–called reduced basis space. For this, let $\mathscr{P}_{\text{red}} : \mathscr{W}_h \to \mathscr{W}_{\text{red}}$ be a Galerkin projection operator fulfilling the equation

$$\left\langle u - \mathscr{P}_{\text{red}}\left[u\right], \varphi \right\rangle = 0 \qquad \forall u \in \mathscr{W}_h \, \forall \varphi \in \mathscr{W}_{\text{red}}.$$

Then for each $\mu \in \mathscr{D}$, sequences of low–dimensional solution snapshots $u_{\text{red}}^k(\mu) \in \mathscr{W}_{\text{red}}$ for $k = 0, \dots, K$ are obtained through the reduced numerical scheme

$$u_{\text{red}}^0(\mu) = \mathscr{P}_{\text{red}} \circ \mathscr{P}_h\left[u_0(\mu)\right], \tag{3a}$$

$$u_{\text{red}}^{k+1} + \Delta t^k \mathscr{P}_{\text{red}} \circ \mathscr{L}_h^I\left[u_{\text{red}}^{k+1}(\mu)\right] = u_{\text{red}}^k(\mu) - \Delta t^k \mathscr{P}_{\text{red}} \circ \mathscr{L}_h^E\left[u_{\text{red}}^k(\mu)\right]. \tag{3b}$$

These reduced solutions approximate the actual numerical solution in the higher dimensional discrete function space. In order to understand, why the above numerical scheme (3a)-(3b) can be computed efficiently, we switch to a vector based formulation of the numerical scheme by identifying the reduced solution sequences $u^k_{\mathrm{red}}(\mu) = \sum_{n=1}^{N} a^k_n(\mu)\varphi_n$ with their coefficient vectors

$$\mathbf{a}^k(\mu) := \left(a^k_1(\mu), \ldots, a^k_N(\mu)\right)^t \in \mathbb{R}^N$$

and compute these vectors for each parameter vector $\mu \in \mathscr{D}$ with the scheme

$$\mathbf{a}^0(\mu) = \mathbf{u}_0(\mu), \tag{4a}$$

$$\mathbf{a}^{k+1}(\mu) + \Delta t \mathbf{L}^I(\mu)\left[\mathbf{a}^{k+1}(\mu)\right] = \mathbf{a}^k(\mu) - \Delta t^k \mathbf{L}^E(\mu)\left[\mathbf{a}^k(\mu)\right], \tag{4b}$$

where the N-dimensional vectors $\mathbf{u}_0(\mu)$ and $\mathbf{b}(\mu)$ and the $N \times N$ sized matrices $\mathbf{L}^I$ and $\mathbf{L}^E$ are given by linear combinations of precomputed vectors or matrices, respectively.

$$\mathbf{L}^I(\mu) = \sum_{q=1}^{Q_I} \sigma^q_I(\mu)\mathbf{L}^q_I, \qquad \mathbf{L}^E(\mu) = \mathbf{b}(\mu) + \sum_{q=1}^{Q_E} \sigma^q_E(\mu)\mathbf{L}^q_E,$$

$$\mathbf{u}_0(\mu) = \mathbf{b}(\mu) + \sum_{q=1}^{Q_{u_0}} \sigma^q_{u_0}(\mu)\mathbf{u}^q_0, \qquad \mathbf{b}(\mu) = \sum_{q=1}^{Q_b} \sigma^q_b(\mu)\mathbf{b}^q.$$

The entries of the precomputed vectors $\mathbf{u}^q_0$ and $\mathbf{b}^q$ are computed as the projections onto the basis functions

$$\left(\mathbf{u}^q_0\right)_n := \int_\Omega u^q_0 \varphi_n, \qquad \left(\mathbf{b}^q_0\right)_n := \int_\Omega b^q \varphi_n \tag{5a}$$

and the matrices $\mathbf{L}^q_I$ and $\mathbf{L}^q_E$ are weighted gramian matrices with entries

$$\left(\mathbf{L}^q_I\right)_{nn'} := \int_\Omega \mathscr{L}^{I,q}_h[\varphi_{n'}]\,\varphi_n, \qquad \left(\mathbf{L}^q_E\right)_{nn'} := \int_\Omega \mathscr{L}^{E,q}_h[\varphi_{n'}]\,\varphi_n \tag{5b}$$

In order to gain control of the approximation error induced by the Galerkin projection onto reduced basis space, efficiently computable a posteriori error estimators are necessary. For linear problems, such estimators $\eta(\mu)$ assessing the error between reduced and detailed simulations $\max_{k=0}^{K} \left\|u^k_h(\mu) - u^k_{\mathrm{red}}(\mu)\right\| \leq \eta(\mu)$ can be deduced in a way such that they integrate into the concept of offline-/online decomposition.

For details on a posteriori error estimators and the generation of a reduced basis space $\mathscr{W}_{\mathrm{red}}$ with the so–called "POD–Greedy" algorithm, we refer to [6].

3 Software Concept

As mentioned in the introduction, the generation of the reduced basis method is split into two parts. The low–dimensional computations and abstract algorithms for the generation of the reduced basis methods are implemented in RBMATLAB, whereas DUNE-RB provides algorithms and data structures for the manipulation of the high–dimensional reduced basis space, and implements interfaces to several classes of parametrized high–dimensional problems, e.g. to linear evolution equations of the form (1a)-(1b). The two software packages can communicate by

1. either compiling DUNE-RB as a mex-library, which allows it to be called directly from the Matlab prompt, or
2. via TCP/IP communication over sockets.

Both methods rely on an encapsulation of Matlab data structures. For the first choice the memory handling of these data structures is kept in Matlab, whereas in the second case, data needs to be copied between both processes and re-instantiated on the DUNE-RB side as C++ data structures. Nevertheless, the second approach is assessed as more advantageous by the authors, because the compilation as a mex-library sets certain requirements on compilers and compiler options, as well as library dependencies. Furthermore, the TCP/IP interface allows to easily run the high–dimensional and the low–dimensional computations on different hardware platforms, each suitably chosen for its needs (server/client model).

In this section, the structure and the capabilities of both software packages are presented with a focus on the user interface methods for an already implemented numerical scheme. The implementation of a particular numerical scheme and the realization of a resulting reduced basis method for a parametrized linear evolution equation is then described in Section 4.

3.1 RBMATLAB

The Matlab based software package RBMATLAB can be seen as a user interface for both the offline and the online phase of a reduced–basis–method execution cycle. Figure 1 illustrates the course of action for all reduced basis method implementations independent from the underlying problem. The headers in the boxes are code snippets as they are executed in RBMATLAB. Gray shaded boxes indicate interface functions where the arguments and return values are communicated to DUNE-RB via TCP/IP sockets and the actual computations are executed with DUNE. Steps 1 and 3 gather high–dimensional information needed for the generation of the reduced basis and step 4 finally reduces this data for use with efficient simulations by the reduced scheme. The actual implementation for these high–level functions is specified by the central object called `model`. The crucial attributes, this objects specifies, are *(i)* the problem type of the underlying parametrized partial differential equations

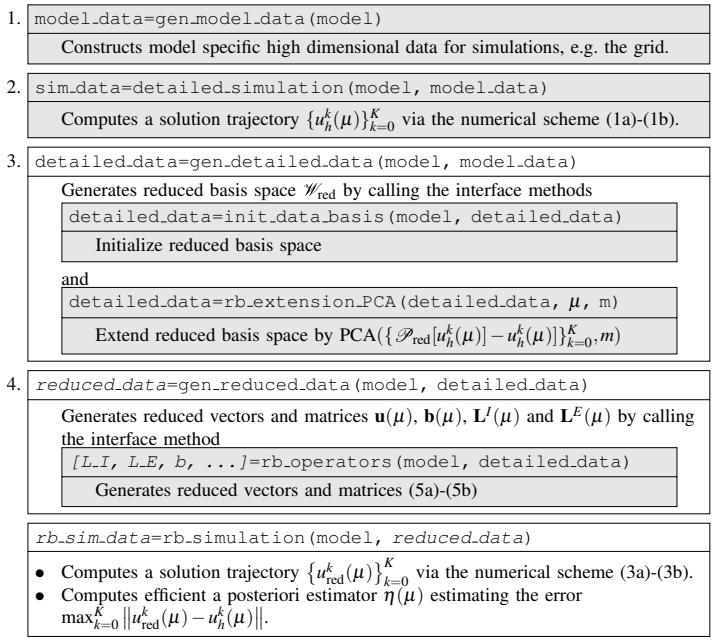

Fig. 1 Illustration of the course of action for reduced basis methods in RBMATLAB. Method calls delegated to DUNE-RB are surrounded with gray shaded boxes. The data structures `model_data`, `sim_data` and `detailed_data` are high–dimensional and depend on $\mathscr{O}(H)$, `reduced_data`, `[L_I, L_E, b, ...]` and `rb_sim_data` are low–dimensional data structures depending on $\mathscr{O}(N)$.

and their discretizations, *(ii)* the chosen reduced basis generation strategy and *(iii)* the link to the numerical scheme in DUNE-RB.

Apart from linear evolution problems of the form (1a)-(1b) there is also support for non–linear problems and discretization schemes where the data fields do not depend affinely on the parameter as required in (2). For this purpose, RBMATLAB provides the method of empirical operator interpolation proposed in [3].

For the reduced basis generation, RBMATLAB implements some variants, which are all an adaptation of the aforementioned "POD–Greedy"-algorithm. Noteworthy are the possibilities to adaptively refine the subset of the parameter space $\mathscr{D}$ from which the reduced basis functions are chosen [5] and to generate smaller reduced basis spaces for different subspaces of the parameter space [4].

For rapid prototyping, RBMATLAB contains a library of finite–volume discretizations for partial differential equations, which is based on triangular or rectangular grids in two dimensions and implements the interface for communication with the reduced basis algorithms in RBMATLAB. This interface can actually be implemented by arbitrary software packages providing numerical schemes for partial differential equations. In the next section, we focus on the software package DUNE-RB bringing the flexibility and power of DUNE to the reduced basis world.

3.2 DUNE-RB

The DUNE module DUNE-RB for the high–dimensional computations is based on the discretization module DUNE-FEM providing a layer of abstraction for discrete function spaces and differential operators [2]. DUNE-RB extends the concepts from DUNE-FEM and provides interface classes for *(i)* solution trajectories $\left\{u_h^k(\mu)\right\}_{k=0}^K$, *(ii)* reduced basis spaces $\mathscr{W}_{\text{red}}$ and *(iii)* decomposed discrete operators of the form $\mathscr{L}_h[\cdot] = \sum_{q=1}^Q \sigma^q(\mu)\mathscr{L}_h^q[\cdot]$, where the notations are adopted from Section 2. This section gives an overview of the available implementations for these interfaces in DUNE-RB. In the end of this section, we elaborate on the implementation of the communication interface to RBMATLAB and further low–level functionality.

Interface classes: *(i)* The data structure for storing either a solution trajectory or reduced basis functions can in both cases be seen as a list of discrete functions from the high–dimensional function space $\mathscr{W}_h$. Such lists are given as implementations of an abstract `DiscreteFunctionListInterface` class. At the moment DUNE-RB provides the two implementations `DiscreteFunctionList_xdr` for efficient storage of the discrete functions on the harddrive in the xdr format and `DiscreteFunctionList_mem` holding all the discrete functions in main memory.

(ii) The reduced basis space class `ReducedBasisSpace` is derived from the DUNE-FEM interface `DiscreteFunctionSpaceInterface` with a base function list implemented as a `DiscreteFunctionListInterface`. In contrast to classical finite–element spaces implemented in DUNE-FEM, this class offers additional methods for manipulating the space by adding and changing global basis functions.

(iii) DUNE-FEM introduces the concept of local operators which are defined by iteratively applying the operator to local representations of the argument's or destination's functions living on a grid entity and its neighbours only. This concept allows to implement matrix–free discrete operators saving memory and bandwidth, and to concatenate these operators without the overhead of further grid iterations, which are very expensive at least for unstructured grids. DUNE-RB extends this concept by `LocalParametrizedOperatorInterface` classes, enriching the local operator implementation by the method `coefficient()` scaling the operator's output by the parameter dependent coefficient function. A sum of such parametrized and locally implemented operators is managed by decomposed operators implementing the `DecomposedOperatorInterface`. Such instances are actually discrete operators by their own, evaluating to the entire sum, but also providing access to the coefficients– and parameter–independent operator parts, which are needed for construction of reduced matrices.

Decomposed operators are implemented for general finite–volume operators for various numerical flux implementations, like the Lax–Friedrichs flux. The decomposition of these operators is derived automatically from the affinely parameter dependent decomposition of the used data functions. This will be described in more detail, in Section 4.2.

Communication interface: The communication interface to RBMATLAB is simply achieved by instantiating an `RBSocksServer` object provided with a template parameter derived from `RBMatlabBase`. The class given as the template parameter needs to call the derived method `registerFunction("op", mp)` for each operation `"op"` that shall be available for triggering from the other side, binding it to a program entry point given by the second argument `mp`. Calling the method `run()` on the server object then puts the program into "listening" mode awaiting orders from RBMATLAB.

Furthermore, this DUNE module provides a singleton class `Parameter`, which globally manages the parameter vector $\mu \in \mathscr{D}$ in such a way that variables depending on the parameter vector simply delegate the evaluation of these magnitudes to the `Parameter` singleton instance.

4 Example: Linear evolution equation

In this section, we analyze the reduced basis method for a linear instationary convection–diffusion problem implemented in DUNE for two and three dimensions. The considered problem looks as follows: For each parameter vector $\mu \in \mathscr{D}$, we are looking for solutions $u(t;\mu) \in BV(\Omega) \cap L^\infty(\Omega) \subset L^2(\Omega)$ fulfilling the equations

$$\partial_t u(t;\mu) + \mathbf{v}(\mu)\nabla u(t;\mu) - D(\mu)\Delta u(t;\mu) = 0 \qquad \text{in } \Omega \times [0,T_{\max}] \tag{6a}$$

$$u(0;\mu) = u_0 \qquad \text{in } \Omega \times \{0\} \tag{6b}$$

$$u(t;\mu) = u_{\text{dir}} \qquad \text{on } \Gamma_{\text{dir}} \times [0,T_{\max}] \tag{6c}$$

$$(\mathbf{v}(\mu)\nabla u(t;\mu)) \cdot \mathbf{n} = 1 \qquad \text{on } \Gamma_{\text{neu}} \times [0,1] \tag{6d}$$

on a rectangular domain $\Omega := [0,1]^d$ with $d \in \{2,3\}$ and $T_{\max} = 1.0$. In our computations, we consider a three dimensional parameter space $\mathscr{D} := [0,0.001] \times [0.3,1] \times [0.3,1]$ and the parametrized data functions are given by $D(x;\mu) = \mu_1$ and $\mathbf{v}(\mu) = (\mu_2, \mu_3, 0.1)$. The Dirichlet boundary function is given by $u_{\text{dir}} = 0.01$ and the initial data function $u_0(x) = \exp(-10\|x-0.5\|_2)\chi_{\|x-0.5\|_2 \leq 0.2}$ implements a non–smooth circle–shaped concentration. The domain's boundary is separated into an "inflow" boundary $\Gamma_{\text{dir}} := \{0\} \times [0,1] \cup [0,1] \times \{0\}$ with a Dirichlet condition and an "outflow" Neumann boundary $\Gamma_{\text{neu}} := \partial\Omega \setminus \Gamma_{\text{dir}}$.

4.1 Finite–volume discretization

The problem is discretized with a purely explicit finite–volume scheme, where the convection part is discretized by a Lax–Friedrichs flux. For a discretization of the form (1a)-(1b), we therefore need to specify the operator $\mathscr{L}_h^E(\mu)$. In our example, the implicit operator $\mathscr{L}_h^I(\mu)$ is set to zero. For the time discretization we choose a

global time–step size Δt, which is small enough such that a CFL–type condition is fulfilled for all parameters $\mu \in \mathscr{D}$. Before we define the implemented discretization operator, however, we need to fix some notations concerning the tessellation of the domain: $\mathscr{T} := \{e_i\}_{i=1}^{H}$ denotes a numerical grid consisting of H disjoint polygonal elements forming a partition of the domain $\bar{\Omega} = \bigcup_{i=1}^{H} \bar{e}_i$. For each element $e_i, i = 1,\dots,H$, we assume that there exist

- certain points x_i lying inside the element e_i, such that points in adjacent elements are perpendicular to the corresponding edges and
- a set of indices $\mathscr{N}(i) := \mathscr{N}_{\text{in}}(i) \cup \mathscr{N}_{\text{dir}}(i) \cup \mathscr{N}_{\text{neu}}(i)$ counting the element's edges e_{ij} for $j \in \mathscr{N}(i)$, where $\mathscr{N}_{\text{in}}$ corresponds to edges between inner elements of the domain or elements adjacent by cyclical boundary conditions, $\mathscr{N}_{\text{dir}}$ includes those edges on the Dirichlet boundary and $\mathscr{N}_{\text{neu}}$ those ones on the Neumann boundary of the domain.

On each edge e_{ij}, we denote their barycenters by x_{ij} and their outer unit normals by $\mathbf{n}_{ij}$.

Furthermore, we define the finite–volume space $\mathscr{W}_h$ as the span of base functions $\psi_i \in \mathscr{W}, i = 1,\dots,H$ being piecewise constant on the i-th cell e_i and vanishing elsewhere. For a function $u_h \in \mathscr{W}_h$ its degrees of freedom (DoFs) are given by $u_{h,i} = u_h(x_i)$, allowing a DoF–wise definition of the explicit discretization operator

$$\begin{aligned}\left(\mathscr{L}_h^E(\mu)\,[u_h]\right)_i = &\frac{1}{|e_i|}\sum_{j\in\mathscr{N}_{\text{in}}(i)} g_{ij}^{\text{lf}}(u_{h,i},u_{h,j}) + g_{ij}^{\text{diff}}(u_{h,i},u_{h,j}) \\ &+ \frac{1}{|e_i|}\sum_{j\in\mathscr{N}_{\text{dir}}(i)} g_{ij}^{\text{lf}}(u_{h,i},u_{\text{dir}}(x_{ij})) + g_{ij}^{\text{diff}}(u_{h,i},u_{\text{dir}}(x_{ij})) \\ &+ \frac{1}{|e_i|}\sum_{j\in\mathscr{N}_{\text{neu}}(i)} \int_{e_i} u_{\text{neu}}\end{aligned} \tag{7}$$

with numerical fluxes, c.f. [8]

$$g_{ij}^{\text{lf}}(u,v) := \frac{1}{2}\left(\mathbf{v}(\mu)\cdot\mathbf{n}_{ij}(u+v) + \frac{1}{\lambda}(u-v)\right) \text{ and} \tag{8}$$

$$g_{ij}^{\text{diff}}(u,v) := \frac{|e_{ij}|}{|x_i - x_j|} D(x_{ij};\mu)(u-v). \tag{9}$$

Remark 1. In order to fullfil the requirements on the separable form of the operator as stated in (2), it suffices to require the data functions u_0, u_{dir}, D and $\mathbf{v}$ to be affinely parameter dependent. It can be easily observed that a decomposition into a sum of products of purely parameter and space dependent data functions is inherited by these linear numerical fluxes and, therefore, leads to the desired decomposition of the discrete operator.

4.2 Implementation

A major design principle of DUNE-RB is to enable quick development of new numerical schemes without knowledge of the reduced basis method framework. For this purpose, DUNE-RB provides code snippets for easy construction of programs run as a TCP/IP server controlled by RBMATLAB.

Listing 1 (Excerpt from a typical main file)

```
typedef DefaultDescr
           < LinEvolExplicitDiscretization,
             ConvDiffModel >             Description;

typedef ProblemTraits
          < LinEvolDefault, LinEvolFacade,
            RBSocksServer, Description > ServerTraits;


%#include <dune/rb/matlab/duneserver/main.inc>
#include <main.inc>
```

Listing 1 shows an example code file for such a server, where the developer only needs to provide two Traits class declarations. The `ServerTraits` class provides the full declaration of the server by specifying the interface methods made available in RBMATLAB (in this case `LinEvolFacade`) and the actual entry points of these commands (`LinEvolDefault`). The latter one, of course, depends on the problem definition and the discretization method, specified by the typedef `Description` This class again consists of

- an abstract problem type `LinEvolExplicitDiscretization` implementing the numerical scheme (1a)-(1b) in this case, and
- the declaration for the spatial operators $\mathscr{L}_h^I(\mu)$ and $\mathscr{L}_h^E(\mu)$ provided by the model class `ConvDiffModel`.

As described in Remark 1, the concept of inheriting the decomposed structure of discrete operators from data functions is also implemented in DUNE-RB, which further simplifies the definition of the aforementioned model class.

4.3 Results

In Figure 2 we demonstrate the behaviour of the error estimator during the construction of the reduced basis via the "POD–Greedy" algorithm. Plotting the error bound in logarithmic scale against the size of the reduced basis, varying from one to 30, we are able to see an approximately exponential decay of the error, as expected from the theory.

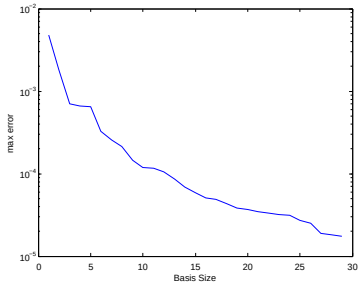

Fig. 2 Maximum of error bound over a set of 216 parameters for different reduced basis sizes in logarithmic scale.

	High-dim. Solution (s)	RB Generation (s)	Gen. of Online Matrices (s)	Reduced Sim. (s)	Reconstruction (s)	Grid Cells	$L^\infty - L^2$-Error
2D Transport (25 Base Functions)	11	71	6.69	0.11	0.33	1,024	$1.42 \cdot 10^{-3}$
2D Transport (50 Base Functions)	11	2,250	21	0.15	0.42	1,024	$4.64 \cdot 10^{-4}$
3D Transport (50 Base Functions)	944	$1.57 \cdot 10^5$	4,659	0.15	26	32,768	$9.11 \cdot 10^{-4}$

Table 1 Numerical results for a transport problem in 2D and 3D with non–divergence–free velocity. The given error is the mean error between full and reduced simulation, tested with 250 randomized parameter values

In Table 1 we present runtimes for the model problem described in Section 4 in two and three space dimensions as generated by our implementation. Column one gives the time for the computation of one trajectory using a `YaspGrid` as a grid manager with the numbers of cells as indicated in column six. Columns two and three give the important numbers for the offline part of the reduced basis algorithm: The time for the "POD–Greedy" algorithm and the Galerkin projection step, that is, the computation of the low–dimensional matrices. In columns four and five we see the runtimes for the online part of the RB method: the solution of the reduced system and the reconstruction of a high–dimensional function from a reduced one, respectively.

With increasing size of the reduced basis from row one to two, and as well, with growing world dimensions, one can observe a significant growth in the runtime for the offline algorithm. Still, the rather time consuming offline phase, with a total runtime of about 45 hours in the 3D case, pays off with a speedup–factor of about 25 and 20 in the 2D cases and even 36 for the 3D case (online phase of the RB method compared to one high–dimensional solution). All these factors consider the reconstruction step to be part of the online phase, which is not the case if one is only interested in the value of an output functional, for example. Considering only the run–times for one high– and ones low–dimensional solution, the speedup–factor grows to 6293 in the 3D case. In all these cases we deal with a relative error of 10^{-3} to 10^{-4}, which is quite acceptable.

5 Conclusion and Outlook

We developed a software framework for the reduced basis methods simplifying both the development of new reduced basis algorithms and the implementation of new model problems and numerical schemes. An example for a finite–volume discretization of a linear evolution equation has been implemented in our software module DUNE-RB demonstrating the flexibility of DUNE by using the same model description on domains with different dimensions.

Future work will deal with the implementation of a library of more complex test cases and the extension of the software framework to non–linear problems and to systems of parametrized partial differential equations. Furthermore, new ideas for the generation of reduced basis spaces will be implemented in RBMATLAB and evaluated with the aforementioned test cases. Theoretically, most of the algorithms implemented in DUNE-RB can be run in parallel because of the parallel capabilities of DUNE, but practical tests of this functionality are still outstanding.

Finally, we plan to publish both software packages with installation instructions and documentation on our project homepage http://morepas.org

References

1. Barrault, M., Maday, Y., Nguyen, N., Patera, A.: An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. C. R. Math. Acad. Sci. Paris Series I **339**, 667–672 (2004)
2. Dedner, A., Klöfkorn, R., Nolte, M., Ohlberger, M.: A generic interface for parallel and adaptive discretization schemes: abstraction principles and the DUNE-FEM module. Computing **90**, 165–196 (2010)
3. Drohmann, M., Haasdonk, B., Ohlberger, M.: Reduced Basis Approximation for Nonlinear Parametrized Evolution Equations based on Empirical Operator Interpolation. Tech. rep., FB10, University of Münster (2010)
4. Haasdonk, B., Dihlmann, M., Ohlberger, M.: A training set and multiple bases generation approach for parametrized model reduction based on adaptive grids in parameter space. Tech. rep., University of Stuttgart (submitted) (2010)
5. Haasdonk, B., Ohlberger, M.: Adaptive basis enrichment for the reduced basis method applied to finite volume schemes. In: Proc. 5th International Symposium on Finite Volumes for Complex Applications, pp. 471–478 (2008)
6. Haasdonk, B., Ohlberger, M.: Reduced basis method for finite volume approximations of parametrized linear evolution equations. M2AN, Math. Model. Numer. Anal. **42**(2), 277–302 (2008)
7. Knezevic, D., Petterson, J.: A high-performance parallel implementation of the certified reduced basis method. Submitted to CMAME. (2010)
8. Kröner, D: Numerical Schemes for Conservation Laws. John Wiley & Sons and Teubner (1997)
9. Patera, A., Rozza, G.: Reduced Basis Approximation and a Posteriori Error Estimation for Parametrized Partial Differential Equations. MIT (2007). http://augustine.mit.edu/methodology/methodology_bookPartI.htm. Version 1.0, Copyright MIT 2006-2007, to appear in (tentative rubric) MIT Pappalardo Graduate Monographs in Mechanical Engineering