



INSTITUTE OF NUMERICAL AND APPLIED MATHEMATICS

DISSERTATIONSTHEMA

---

# Reduced basis model reduction for non-linear evolution equations

---

Inaugural-Dissertation  
zur Erlangung des Doktorgrades der  
Naturwissenschaften im mathematischen Fachbereich  
der Westfälischen Wilhelms-Universität Münster

vorgelegt von  
Martin Drohmann  
aus Herdecke  
- Juni 2012 -

---

Dekan:	Prof. Dr. Matthias Löwe
Erster Gutachter:	Prof. Dr. Mario Ohlberger
Zweiter Gutachter:	Prof. Dr. Bernard Haasdonk
Tag der mündlichen Prüfung:	
Tag der Promotion:	



## Zusammenfassung

In dieser Dissertation wurde eine Erweiterung der reduzierte Basis-Methode für allgemeine nicht-lineare Evolutionsprobleme entwickelt. Bisherige Veröffentlichungen auf diesem Gebiet machen Annahmen an die Art der zugrundeliegenden Nicht-Linearitäten oder an die Trennung von Parametern und Ortsvariablen. Durch die Verallgemeinerung der empirischen Interpolationsmethode für Funktionen auf diskrete Operatoren, können diese Restriktionen aufgehoben werden. Die entwickelten Methoden und algorithmischen Verbesserungen wurden anhand einiger Beispiele für skalare Evolutionsprobleme getestet und analysiert. Hierbei wurde jeweils eine zeitliche Modellreduktion um eine Größenordnung festgestellt. Als Ausblick wurde zudem ein Zwei-Phasenströmungsmodell mit der Methode der empirischen Operator-Interpolation reduziert. Auch hier kann ein Geschwindigkeitsgewinn durch die Projektion auf einen reduzierte Basis-Raum festgestellt werden.



## Acknowledgments

First of all, I would like to express my gratitude to my supervisors Prof. Dr. Mario Ohlberger and Prof. Dr. Bernard Haasdonk for their support and inspiration for this work.

Special thanks go to Felix Albrecht for proofreading of this work, and to all members of the Institute for Applied and Numerical Mathematics for the pleasant and stimulating atmosphere.

I want to appreciate the patience and encouragement of my friends and especially my parents, who have always supported me during my studies.



## Contents

Zusammenfassung	i
Acknowledgments	iii
List of Figures	vii
Chapter 1. Introduction	1
Goals and outline	2
Chapter 2. Empirical operator interpolation	5
2.1. Basis generation and Dof selection	7
2.2. Interpolation efficiency	10
2.3. Error analysis	13
2.4. Invariant properties	16
Chapter 3. Reduced basis method	19
3.1. Evolution Scheme	19
3.2. Reduced simulation scheme	25
3.3. A posteriori error estimator	32
Chapter 4. Basis generation	41
4.1. Proper orthogonal decomposition	42
4.2. Greedy algorithm	43
4.3. Combined basis generation	50
4.4. Control of error and efficiency	52
Chapter 5. Numerical experiments	59
5.1. Burgers equation	59
5.2. Porous Medium Equation	65
5.3. Example: Buckley–Leverett equation	75
Chapter 6. Immiscible two phase flow in porous media	81
6.1. Global pressure formulation	81
6.2. Finite volume discretization	84
6.3. Reduced basis scheme	87
6.4. Numerical experiments	92

Chapter 7. Abstract software framework	97
7.1. Stationary reduced basis scheme	98
7.2. Overview on RBMATLAB	102
7.3. Software concept	104
7.4. High-dimensional computations with DUNE-RB	107
7.5. Example: Poisson problem	110
7.6. Outlook: subgrid extraction for empirical interpolation	114
Chapter 8. Conclusion and outlook	115
Perspectives	116
Appendix A. Constants for finite volume operators	117
Bibliography	119
Erklärung	127

## List of Figures

3.1. Excerpt of a rectangular grid with notations used in this paper.	22
3.2. Illustration of finite volume operator evaluation on a local subgrid	31
4.3. Illustration of reduced basis error convergence with varying dimensionalities $N$ and $M$ .	52
5.1. Illustration of transport for smooth data	60
5.1. Illustration of EI quality for Burgers problem	63
5.1. Reduced basis error convergence for Burgers Problem	64
5.2. Illustration of porous medium solutions	66
5.2. Parameter selection during basis generation	68
5.2. Comparison of POD-GREEDY and PODEI-GREEDY error convergence	70
5.2. Cross-section plots of porous medium equation	71
5.2. Efficiency of a posteriori error estimates	73
5.2. Comparison of error decrease for porous medium equation	74
5.3. Illustration of Buckley–Leverett solution snapshots	77
5.3. Basis size comparison for time adaptive basis generation algorithms	79
6.4. Two phase flow data functions	92
6.4. Illustration of two phase flow solution snapshots	93
6.4. Empirical interpolation convergence and Dof selection (two phase flow)	95
7.2. Course of action in RBMATLAB	103
7.3. Call graph for main software parts in RBMATLAB	105
7.3. Call graph for main software parts in abstract software concept	106
7.5. Illustration of solution snapshots (Poisson problem)	112
7.5. Reduced basis greedy convergence	113



## CHAPTER 1

### Introduction

During the last years, numerical simulations of processes in all sciences, industry and economy have gained importance. The reason for this is a significant improvement of processor hardware and the involved algorithms for the discrete solution of partial differential equations (PDEs). In most cases, however, these simulations consume an extremely huge amount of computational power, such that even on high-performance architectures, simulations can take many hours or even days.

Therefore, *real-time* or *many-query* applications which depend on several repeated simulations are still infeasible to deal with. Real-time applications are those, which have constraints on the response time for a numerical simulation, or depend on anytime available computational power, which is not the case for current super-computers. In the many-query context, where several simulations need to be carried out on the same problem, one is usually interested in the optimization or estimation of certain parameters or tries to model stochastic behavior of the underlying system.

The reduced basis method is a means to deal with such applications for simulations based on a partial differential equation (PDE). The idea is to first introduce a parametrization of the problem which restricts the solution space to solutions of interest for the application. Then, the essential characteristics of this manifold of solution can be captured with the goal to approximate the simulation output in an efficient and reliable way. The latter implies the need for efficiently computable bounds on the error between the high-dimensional and the reduced simulation.

Then, the reduced basis method is decomposed into two parts: an *offline* and an *online* phase. During the offline phase, few high-dimensional computations are carried out, and during the online phase, we work with surrogate solvers efficiently and reliably generating low-dimensional solution outputs, such that these solvers can be used in real-time or many-query applications.

The parametrization of the underlying PDE can model e.g. initial or boundary conditions, and inertial model properties. A huge effort has also been put on the special case of shape design [64, 56], where the geometry of the spatial problem domain, gets parametrized. The quality of the parametrization of the PDE can have a big impact on the quality of the

reduced basis surrogate solver. Therefore, a focused choice of the parameter space is of great importance.

There even exist a priori convergence results for the reduced basis method, first proven for the single-parameter case [54] and recently for the multi-parameter case [7, 4, 34]. For these convergence results, it has to be assumed that the manifold  $\mathcal{S} \subset \mathcal{W}_h$  of all parametrized solutions shows an exponentially or polynomially decreasing Kolmogorov  $N$ -width in the discrete function space  $\mathcal{W}_h$ . This Kolmogorov  $N$ -width measures the error of the worst approximation by a linear function space of dimension  $N$ , by

$$d_N(\mathcal{S}, \mathcal{W}_h) := \inf_{\substack{\widehat{\mathcal{W}} \subset \mathcal{W}_h \\ \dim(\widehat{\mathcal{W}}) = N}} \sup_{v_h \in \mathcal{S}} \min_{u_h \in \widehat{\mathcal{W}}} \|v_h - u_h\|_{\mathcal{W}_h}. \quad (1.1)$$

The complete theory for the reduced basis method, including efficient reduced schemes, basis generation algorithms and a posteriori error estimates, has been developed during the last years, mainly for discretizations based on the finite element method. For linear elliptic problems, we refer to [61] and for linear parabolic problems to [33]. This was later extended to problems with quadratic non-linearities [74, 73, 58]. Most of the early publications made the assumption, that the parameter dependent contributions can be efficiently separated from the space dependent functions and operators in the original scheme. This assumption can be dropped, since the introduction of the empirical interpolation method (EIM) [1]. The method allows to interpolate parametrized functions in a suitable constructed linear function space by localized and therefore efficient computations. The integration of the EIM on the reduced basis method has been proven in several works afterwards, e.g. [32, 52, 31, 8].

### Goals and outline

The main goal of this thesis is the extension of the above described reduced basis method to the general case of non-linear, parametrized evolution equations of the form

$$\partial_t u(\boldsymbol{\mu}) - \mathcal{L}(\boldsymbol{\mu})[u(\boldsymbol{\mu})] = 0 \quad \text{on } \Omega \times [0, T_{\max}] \quad (1.2)$$

for all parameters  $\boldsymbol{\mu} \in \mathcal{M}$  from some parametrization space  $\mathcal{M} \subset \mathbb{R}^p$ . Here  $\mathcal{L}$  is a parametrized differential operator and  $u_h$  the parametrized solution trajectory. As a preliminary step, scalar partial differential equations are considered, but the long-term goal is the simulation of a system modeling two phase flow, which is a problem that is of interest mainly in the oil production context or in environmental applications.

Extending previous work on reduced basis methods, we allow all kind of non-linearities. This leads to the notion of the empirical operator interpolation, which is the main part of this work and detailed in Chapter 2. It is based on the work in [39], where it was first suggested to apply the empirical interpolation method on operators and tested for explicit discretization schemes. In this work, we generalize the idea, define derivatives for the empirical interpolants, in order to integrate them into non-linear implicit discretizations, transfer the results for a priori and a posteriori error results from [1] and discuss on the invariance of operator properties under empirical interpolation. In the latter case, we have especially finite volume operators in mind, as these are used in our numerical experiments.

The empirical interpolation of operators in numerical schemes is from a computationally point of view similar to the recently developed discrete empirical interpolation method (DEIM) [16] or to the interpolation point selection method for the “Gauß–Newton with approximated Tensors” (GNAT) method [11]. Other, than in those approaches the relationship to the function spaces of the high-dimensional problem is preserved. This simplifies the derivation of reduced basis schemes and a posteriori error estimators. In Chapter 3, we present such a scheme and a very general a posteriori error estimator for discretization of a generalized evolution scheme. As mentioned earlier this scheme allows all kind of non-linear behavior. This gives rise to a very flexible model order reduction with the reduced basis method, as it is presented in Chapter 6. There, a system of partial differential equations modeling immiscible and incompressible two phase flow in a porous media is discretized by a finite volume method proposed in [57] and shown that it can be decomposed into offline and online simulations.

In Chapter 4 the generation and quality of convergence of reduced basis spaces is discussed. We revise the most popular algorithms to find characteristic reduced basis functions which well approximate the parametrized solutions, namely proper orthogonal decomposition (POD) and different greedy algorithms. For the latter, an new variant is proposed and described in this chapter, that automatically reconstructs the optimal ratio between the quality of the reduced basis approximation and the empirical operator interpolation. This is of great importance, as the standard greedy algorithm forces the experimenter to make assumptions on this ratio a priori, which is in general infeasible.

The reason for this is the difficulty to compute the constants specifying the exact rate of the error decrease, a priori. Thus, it is impossible to predict the reduced basis size needed to reach a given error tolerance. Nonetheless, for many applications, it is realistic to assume an exponential decay of the

Kolmogorov  $N$ -width of the manifold of solution snapshots. This gives rise to smart algorithms that increase the reduced basis size incrementally and split the complexity in the parameter space if necessary. Such algorithms have been developed recently, e.g. [29, 21, 24, 35]. In Section 4.4 these smart algorithms are described by means of our own contribution splitting the time domain for the empirical interpolation.

As there always remains a risk that the reduced basis method does not lead to the anticipated performance gain, we assume, that rapid prototyping of a reduced basis framework for an existing detailed scheme is of high importance. Therefore, in Chapter 7, the software used to implement the numerical examples is described and analyzed in order to derive an abstract software concept from it. It is shown, where implementations of reduced basis methods can be split into independent components, with the goal to re-use existing approaches for the rapid development of prototypes of the reduced basis method for untested detailed schemes.

For all the theoretical and algorithmic improvements, extensive numerical experiments are carried out. The results are summarized and analyzed in Chapters 5 and 7.

The main parts, including the empirical operator interpolation, the a posteriori error estimate and the PODEI-GREEDY for a synchronized generation of reduced basis approximations and empirical operator interpolations, have already been published in [25].

Other approaches generate small reduced basis spaces for small reference domains and combine the results with domain decomposition techniques [65, 40].

## Empirical operator interpolation

As explained in the introduction, the reduced basis method depends on underlying numerical schemes which allow to separate the parameter dependent influences from the space dependent parts. Therefore, early publications on this topic considered discretizations of linear problems, e.g. [60, 66] or problems with quadratic non-linearities, e.g. [51, 73]. Furthermore, most of them required all parameter dependent functions  $g : \Omega \times \mathcal{M} \rightarrow \mathbb{R}$  to be regular, such that  $g(\cdot; \boldsymbol{\mu}) \in L^\infty(\Omega)$  for all parameters  $\boldsymbol{\mu} \in \mathcal{M} \subset \mathbb{R}^p$  and to be of a separable form

$$g(x; \boldsymbol{\mu}) = \sum_{q=1}^Q \sigma^q(\boldsymbol{\mu}) g^q(x) \quad (2.1)$$

with parameter independent functions  $g^q \in L^\infty(\Omega)$ .

As these assumptions are very restrictive, the empirical interpolation method (EIM) was introduced in [1]. This method allows to interpolate parametrized functions with a small set of data functions which are exact in certain interpolation points  $x_1, \dots, x_M \in \Omega$ , called “magic points” in the original publication. A suitable selection of these ansatz functions  $\xi_1, \dots, \xi_M \in L^\infty$  then makes the interpolants

$$\mathcal{I}_M [g(x; \boldsymbol{\mu})] = \sum_{m=1}^M g(x_M; \boldsymbol{\mu}) \xi_m \quad (2.2)$$

good approximations of the data function  $g(x; \boldsymbol{\mu})$  for all parameters  $\boldsymbol{\mu} \in \mathcal{M}$  and  $x \in \Omega$ . The interpolation costs are lower than the exact evaluation, because they depend on the evaluation of the function in the few “magic points” only.

The empirical interpolation method has been adapted to many different problem settings. These methods are published under the notions “discrete empirical interpolation” (DEIM) [16], “multi-component empirical interpolation” (MCEIM) [71], the equivalent “tensorized empirical interpolation” (TEIM) [44] and “Gauß-Newton with approximated tensors” (GNAT) [9, 11].

Other than in the original empirical interpolation approach [1], in this work, we want to define an empirical interpolation of operators  $\mathcal{L}_h(\boldsymbol{\mu}) :$

$\mathcal{W}_h \rightarrow \mathcal{W}_h$  acting on a discrete function space  $\mathcal{W}_h$ . The goal is to interpolate operator evaluations  $\mathcal{L}_h(\boldsymbol{\mu})[u_h(\boldsymbol{\mu})]$  of parametrized solution snapshots  $u_h(\boldsymbol{\mu}) \in \mathcal{W}_h$ . Therefor, the *empirical interpolant*  $\mathcal{I}_M[\mathcal{L}_h(\boldsymbol{\mu})] : \mathcal{W}_h \rightarrow \mathcal{W}_h$  of a discrete operator is defined by the interpolation of operator evaluations, such that

$$\mathcal{I}_M[\mathcal{L}_h(\boldsymbol{\mu})][\cdot] = \mathcal{I}_M[\mathcal{L}_h(\boldsymbol{\mu})[\cdot]]. \quad (2.3)$$

This concept, first introduced in [37] leads to the notion *empirical operator interpolation*. It gives us the opportunity to rewrite numerical schemes by substituting the spatial operators with their empirical interpolants and to develop reduced schemes based on the high dimensional ones, even if the operators are non-linear or cannot be separated from the parameter. For details we refer to Chapter 3. In Section 2.2.2 we show that also the Fréchet derivative of a discrete operator can be interpolated efficiently, and thus enables us to embed the operator into non-linear numerical schemes depending on the Newton-Raphson method or other optimization algorithms.

Throughout this thesis, we will demonstrate the applicability of the empirical operator interpolation on several numerical reduced basis schemes with and without non-linear behavior. The full flexibility of our approach to empirical interpolation will be demonstrated in Chapter 6, where a system of partial differential equations will be reduced with the reduced basis method and specialized empirical interpolants.

The interpolation of operators instead of model functions has two further advantages, because

- (i) it allows to deal with schemes, where the non-linearities are introduced by the discretization, like in flux based discretizations as such derived by finite volume or discontinuous Galerkin methods, and
- (ii) in many applications the reduced basis space is composed of the same solution “snapshots” as the ansatz space for the empirical interpolation. Therefor, only one space needs to be generated.

If the empirical operator interpolation is used as described in (ii), it is computationally equivalent to the “discrete empirical interpolation method” (DEIM) [16] or the “Gauss-Newton with approximated tensors” (GNAT) methods [9, 11].

In the next section, we define the empirical operator interpolation for discrete operators by specifying the generation of the ansatz functions and the interpolation Dofs which correspond to the “magic point” in the original empirical interpolation. Afterwards, we discuss the computational complexities for evaluations of empirical interpolants in Section 2.2. A priori and a posteriori error results for the original EIM can be transferred to the context

of operator interpolation, which is done in 2.3. In Section 2.4, we conclude with the discussion on linear operator properties which are retained by the empirical interpolants.

### 2.1. Basis generation and Dof selection

Before we start with the description of the empirical operator interpolation, we introduce some notation used subsequently.

**Definition 2.1.1** (Discrete function space). *We write  $\mathcal{W}_h \subset L^\infty(\Omega)$  for a finite dimensional discrete function space equipped with a norm  $\|\cdot\|_{\mathcal{W}_h}$  and defined on a closed subset  $\Omega \subset \mathbb{R}^n$  with a non empty interior and a polygonal boundary. Following the notation of a finite element by P.G. Ciarlet [19], we define the set  $\Sigma_h := \{\tau_i\}_{i=1}^H \subset \mathcal{W}'_h$  of linearly independent functionals, which are unisolvent on  $\mathcal{W}_h$ , i.e. there exist unique functions  $\psi_i \in \mathcal{W}_h, i = 1, \dots, H$  which satisfy*

$$\tau_j(\psi_i) = \delta_{ij}, \quad 1 \leq j \leq H.$$

*The linear functionals  $\tau_i, i = 1, \dots, H$  are called the degrees of freedom (DOFs) of the discrete function space  $\mathcal{W}_h$  and the functions  $\psi_i, i = 1, \dots, H$  are called basis functions. Note, that these basis functions can e.g. be finite element, finite volume or discontinuous Galerkin basis functions on a numerical grid  $\mathcal{T}_h \subset \Omega$ .*

From now on,  $\mathcal{L}_h(\boldsymbol{\mu}) : \mathcal{W}_h \rightarrow \mathcal{W}_h$  always denotes a discretized (non-linear) operator acting on an  $H$ -dimensional discrete function space  $\mathcal{W}_h$ . In order to decompose the computations in an efficient online and an offline phase for high-dimensional data, the scheme must be formulated in a separable way, i.e. the discrete operators are written as a sum of products of efficiently computable parameter dependent functionals and high-dimensional basis functions that can be precomputed during the offline phase.

Hence, we approximate the discrete operators by an empirical interpolant  $\mathcal{I}_M[\mathcal{L}_h(\boldsymbol{\mu})]$  of the form

$$\mathcal{I}_M[\mathcal{L}_h(\boldsymbol{\mu})][v_h] := \sum_{m=1}^M \tau_m^{\text{EI}}(\mathcal{L}_h(\boldsymbol{\mu})[v_h]) \xi_m \approx \mathcal{L}_h(\boldsymbol{\mu})[v_h] \quad (2.4)$$

for arguments  $v_h \in \mathcal{W}_h$ . Ingredients for the interpolation are the *collateral reduced basis*  $\boldsymbol{\xi}_M := \{\xi_m\}_{m=1}^M \subset \mathcal{W}_H$  and *interpolation Dofs*  $\tau_m^{\text{EI}} : \mathcal{W}_h \rightarrow \mathbb{R}$  which must be computable with complexity independent of  $H$ . The sum is assumed to contain few terms, i.e.  $M \ll H$ .

One can think of many reasonable choices for collateral basis functions and corresponding interpolation Dofs, but we focus on a specification resulting in the *empirical operator interpolation* specified in the next section.

**2.1.1. Empirical Operator Interpolation.** The empirical operator interpolation method can briefly be expressed based on a set of interpolation DOFs  $\Sigma_M := \{\tau_m^{EI}\}_{m=1}^M \subset \Sigma_h$  and a corresponding interpolation basis  $\xi_M$ , which is nodal in the interpolation DOFs, i.e.  $\tau_{m'}^{EI}[\xi_m] = \delta_{m,m'}$  for  $1 \leq m, m' \leq M$ . The generation process for these components works similarly to the algorithm described in the original empirical interpolation paper [1] in which point evaluations in so-called “magic points” are used as interpolation DOFs after the basis functions were selected.

The idea of the empirical interpolation data generation is sketched in Algorithm 2.1.1. First, for each parameter from a finite training set of parameters  $\mathcal{M}_{\text{train}} \subset \mathcal{M}$ , exact operator evaluations on also parametrized arguments  $u_h(\boldsymbol{\mu}) \in \mathcal{W}_h$  are computed and the empirical interpolated results of the operator evaluations are computed. Note, that the parametrization of the argument functions is important in order to define the manifold of operator evaluations that shall be approximated well by our empirical operator interpolation.

Then, the function space norm of the residual between the exact and the interpolated operator evaluations, gives a measure for the quality of the interpolation. Alternatively, we could also use the  $L^\infty$ -norm, like in the original empirical operator interpolation [1]. In this case, rigorous a posteriori error estimates can be constructed as is shown in Section 2.3. In most applications, however, better interpolation results can be expected when the interpolation error is minimized in the more “natural” norm of the discrete function space.

The general idea of a greedy algorithm is to improve the approximation with the help of the worst parameter  $\boldsymbol{\mu}_{\text{max}}$ . In case of the empirical operator interpolation, this improvement is realized by

- (1) selecting the residual  $r_M(\boldsymbol{\mu}_{\text{max}})$  as a new basis vector and
- (2) using the residual’s degree of freedom with largest absolute value as a new “magic point” after normalization in the  $L^\infty$  norm.

With this strategy, the approximation is expected to improve with growing reduced basis size  $M$ . In [1] an a priori estimate for the empirical interpolation error verifies this expectation for reasonable assumptions. The result is cited in Section 2.3.

**Remark 2.1.2.** *The nodal basis  $\xi_M := \{\xi_m\}_{m=1}^M$  introduced in equation (2.1.1) is constructed from the iteratively created basis  $\mathbf{Q}_M := \{q_m\}_{m=1}^M$  by the constructive relation*

$$\xi_i = \sum_{j=1}^M (\mathbf{B}^{-1})_{ij} q_j, \quad (2.6)$$

---

**Algorithm 2.1.1** Greedy algorithm for collateral reduced basis generation  
EI-GREEDY

---

EI-GREEDY()

– Start with empty initial basis:

$$\mathbf{Q}_0 \leftarrow \{\}$$

$$\Sigma_0 \leftarrow \{\}$$

$$M \leftarrow 0$$

**for each**  $\boldsymbol{\mu} \in \mathcal{M}_{\text{train}} \subset \mathcal{M}$  **do**

– Compute exact operator evaluations on parametrized functions

$$v_h(\boldsymbol{\mu}) \leftarrow \mathcal{L}_h(\boldsymbol{\mu})[u_h(\boldsymbol{\mu})]$$

**end for**

**repeat**

**for each**  $\boldsymbol{\mu} \in \mathcal{M}_{\text{train}} \subset \mathcal{M}$  **do**

– Compute interpolation coefficients

$$\boldsymbol{\sigma}^M(\boldsymbol{\mu}) := \left( \sigma_j^M(\boldsymbol{\mu}) \right)_{j=1}^M \in \mathbb{R}^M$$

– by solving the linear equation system at EI-Dofs  $\Sigma_M = \{\tau_i^{EI}\}_{i=1}^M$ :

$$\sum_{j=1}^M \sigma_j^M(\boldsymbol{\mu}) \tau_i^{EI} [q_j] = \tau_i^{EI} [v_h(\boldsymbol{\mu})], \quad i = 1, \dots, M \quad (2.5)$$

– Determine residuals:

$$r_M(\boldsymbol{\mu}) \leftarrow v_h(\boldsymbol{\mu}) - \sum_{j=1}^M \sigma_j^M(\boldsymbol{\mu}) q_j$$

**end for**

– Determine parameter with maximum approximation error:

$$\boldsymbol{\mu}_{\max} \leftarrow \arg \sup_{\boldsymbol{\mu} \in \mathcal{M}_{\text{train}}} \|r_M(\boldsymbol{\mu})\|_{\mathcal{W}_h}$$

(or:  $\boldsymbol{\mu}_{\max} \leftarrow \arg \sup_{\boldsymbol{\mu} \in \mathcal{M}_{\text{train}}} \|r_M(\boldsymbol{\mu})\|_{L^\infty(\Omega)}$ )

– Define the maximum residual.

$$\varepsilon_{M+1} \leftarrow r_M(\boldsymbol{\mu}_{\max})$$

– Find interpolation Dof maximizing the residual.

$$\tau_{M+1}^{EI} \leftarrow \arg \sup_{\tau \in \Sigma_h} |\tau(\varepsilon_{M+1})|$$

– Normalize to obtain a new collateral reduced basis function.

$$q_{M+1} \leftarrow (\tau_{M+1}^{EI}(r_M(\boldsymbol{\mu}_{\max})))^{-1} \cdot r_M(\boldsymbol{\mu}_{\max})$$

– extend basis data:

$$\mathbf{Q}_{M+1} \leftarrow \mathbf{Q}_M \cup \{q_{M+1}\}$$

$$\Sigma_{M+1} \leftarrow \Sigma_M \cup \{\tau_{M+1}^{EI}\}$$

$$M \leftarrow M + 1$$

**until**  $\|r_M(\boldsymbol{\mu}_{\max})\|_{\mathcal{W}_h} \leq \varepsilon_{\text{tol}}$  or  $M > M_{\max}$

– construct nodal basis as described in Remark 2.1.2.

$$\mathcal{W}_M = \text{span} \{ \xi_m \}_{m=1}^M$$


---

where the matrix  $\mathbf{B} \in \mathbb{R}^{M \times M}$  is given by its coefficients

$$\mathbf{B}_{ij} := \tau_j^{EI} [q_i]. \quad (2.7)$$

The regularity of the matrix  $\mathbf{B}$  is trivially proven by the special structure of the basis  $\mathbf{Q}_M$ , such that

$$\tau_m^{EI} [q_{m'}] = \begin{cases} 1 & \text{for } m = m', \\ 0 & \text{for } m < m' \text{ and} \\ c \in [-1, 1] & \text{else} \end{cases} \quad (2.8)$$

leading to a lower triangular matrix with ones on the diagonal. The nodal basis  $\boldsymbol{\xi}_M$  has the special structure  $\tau_m^{EI} [\xi'_m] = \delta_{mm'}$  for all  $1 \leq m, m' \leq M$ . It allows a simpler exposition of the empirical interpolation Dofs. However, in [1, 32] it is shown that the maximum norms of the nodal base functions can grow exponentially. Indeed, this fact can lead to infeasible Lebesgue constants quantifying the error ratio between the empirical interpolation and the best approximation in the collateral reduced basis space. The Lebesgue constant is detailed in Section 2.3. Due to the expected growth of the Lebesgue constant, we use the basis  $\mathbf{Q}_M$  in the implementation, but keep  $\boldsymbol{\xi}_M$  for simpler exposition in the following paragraphs.

**Remark 2.1.3.** It is worth to mention, that the loops over the training set  $\mathcal{M}_{train}$  for the computation of the operator evaluations and for the search for the worst approximation parameters in Algorithm 2.1.1 can easily be executed in parallel with hardly any communication costs.

It is noteworthy, that there exist alternative approaches of basis generation ([16, 9]) in which the reduced basis is generated by a proper orthogonal decomposition (POD) of selected solution snapshots, and afterwards a discrete empirical interpolation method on the system matrices is applied reducing the number of matrix rows, such that after projection on the reduced basis space only a low-dimensional system of linear equations needs to be solved.

As in this case the number of interpolation points (matrix rows) can exceed the reduced basis space dimension, an extension of the method is necessary, finding the best least squares approximation [9].

In Section 4.3.1, we propose a further approach for the generation of a collateral reduced basis and empirical interpolation Dofs, by improvement of the interpolation based on an error measure derived from a reduced basis scheme.

## 2.2. Interpolation efficiency

Now, we show under which conditions the empirical interpolation Dofs can be computed efficiently, i.e. with complexity independent of the discrete function space dimension  $H$ . As a further extension, we show that the

Fréchet derivative of a discrete operator can be efficiently approximated with the same collateral reduced basis as the operator itself.

An efficient evaluation of the functionals  $\tau_m^{EI}(\mathcal{L}_h(\boldsymbol{\mu})[v_h])$  for every  $\boldsymbol{\mu} \in \mathcal{M}$  and every argument function  $v_h \in \{u_h(\boldsymbol{\mu}) | \boldsymbol{\mu} \in \mathcal{M}\}$  requires them to depend on few basis functions only. This fact inspires the following definition.

**Definition 2.2.1** (*H-independent Dof dependence*). *With an H-dimensional function space  $\mathcal{W}_h$ , a discrete operator  $\mathcal{L}_h(\boldsymbol{\mu}) : \mathcal{W}_h \rightarrow \mathcal{W}_h$  fulfills an H-independent Dof dependence, if there exists a constant  $C \ll H$  independent of  $H$  such that for all  $\tau \in \Sigma_h$  a restriction operator*

$$\mathcal{R}_\tau^C : \mathcal{W}_h \rightarrow \mathcal{W}_h, v_h = \sum_{i=1}^H \tau_i(v_h) \psi_i \mapsto \sum_{j \in I_\tau} \tau_j(v_h) \psi_j \quad (2.9)$$

*exists, that restricts the operator argument to  $|I_\tau| \leq C$  degrees of freedom and the equation*

$$\tau(\mathcal{L}_h(\boldsymbol{\mu})[v_h]) = \tau(\mathcal{L}_h(\boldsymbol{\mu})[\mathcal{R}_\tau^C[v_h]]) \quad (2.10)$$

*still holds for all  $v_h \in \mathcal{W}_h$ .*

**Remark 2.2.2.** *In particular, finite element or finite volume operators fulfill the H-independent Dof dependence, as a point evaluation of an operator application only requires data of the argument on neighboring grid cells together with geometric information of this subgrid.*

**2.2.1. Evaluation of interpolation functionals  $\tau_m^{EI} \circ \mathcal{L}_h(\boldsymbol{\mu})$ .** Assuming this  $H$ -independence condition for a parametrized discrete operator, its empirical interpolant can be evaluated efficiently, i.e. independently of the dimension  $H$ . This result is summarized in the following corollary.

**Corollary 2.2.3.** *Let for each  $\boldsymbol{\mu} \in \mathcal{M}$  the discrete operators  $\mathcal{L}_h(\boldsymbol{\mu}) : \mathcal{W}_h \rightarrow \mathcal{W}_h$  fulfill the H-independent Dof dependence and let  $\Sigma_M$  and  $\boldsymbol{\xi}_M$  be determined for this operator by Algorithm 2.1.1. Then, the empirical operator interpolation  $\mathcal{I}_M$  defined by*

$$\mathcal{I}_M[\mathcal{L}_h(\boldsymbol{\mu})][v_h] := \sum_{m=1}^M \tau_m^{EI}(\mathcal{L}_h(\boldsymbol{\mu})[v_h]) \boldsymbol{\xi}_m \quad (2.11)$$

*gives a separable approximation of  $\mathcal{L}_h(\boldsymbol{\mu})$  depending on at most  $CM$  degrees of freedom for each evaluation.*

Ignoring the parameter independent collateral basis functions, an evaluation of the empirical interpolant has a complexity independent of the dimension  $H$ . This fact holds during a reduced basis simulation, because then the

collateral reduced basis functions are substituted by low-dimensional surrogates, which are computed during the offline phase. Details on this reduction can be found in the next Chapter 3.

**2.2.2. Evaluation of interpolated Fréchet derivative.** Many solvers for numerical approximations of nonlinear partial differential equations use the Newton–Raphson method to resolve non-linearities in the equation and therefore depend on derivatives of discrete operators. It is easy to observe that the Fréchet derivative can also be applied to the empirical interpolant of an operator  $\mathcal{L}_h(\boldsymbol{\mu})$  as

$$\mathbf{D}(\mathcal{I}_M[\mathcal{L}_h(\boldsymbol{\mu})|_{u_h}] [\cdot]) = \sum_{m=1}^M \mathbf{D}(\tau_m^{EI} \circ \mathcal{L}_h(\boldsymbol{\mu})|_{u_h}) [\cdot] \xi_m. \quad (2.12)$$

For an efficient usage of such an interpolation in a reduced scheme, it suffices to show that the functionals  $\mathbf{D}(\tau_m^{EI} \circ \mathcal{L}_h(\boldsymbol{\mu})|_{u_h})$  can be evaluated efficiently: Assuming the existence of the derivatives w.r.t. the degrees of freedom, we obtain with the linearity of the derivative

$$\begin{aligned} \mathbf{D}(\tau_m^{EI} \circ \mathcal{L}_h(\boldsymbol{\mu})|_{u_h}) [v_h] &= \mathbf{D}(\tau_m^{EI} \circ \mathcal{L}_h(\boldsymbol{\mu})|_{u_h}) \left[ \sum_{i=1}^H \tau_i(v_h) \psi_i \right] \\ &= \sum_{i=1}^H \mathbf{D}(\tau_m^{EI} \circ \mathcal{L}_h(\boldsymbol{\mu})|_{u_h}) [\psi_i] \tau_i(v_h) \\ &= \sum_{i=1}^H \frac{\partial}{\partial \psi_i} \tau_m^{EI}(\mathcal{L}_h(\boldsymbol{\mu})[u_h]) \tau_i(v_h) \\ &= \sum_{i \in I_{\tau_m^{EI}}} \frac{\partial}{\partial \psi_i} \tau_m^{EI}(\mathcal{L}_h(\boldsymbol{\mu})[u_h]) \tau_i(v_h). \end{aligned} \quad (2.13)$$

The reduction in the number of addends holds true, as substituting the Dof  $\tau_m^{EI}(\mathcal{L}_h(\boldsymbol{\mu})[u_h])$  by its restriction  $\tau_m^{EI}(\mathcal{L}_h(\boldsymbol{\mu})[\mathcal{R}_{\tau_m^C}^C[u_h]])$  shows that most of the directional derivatives are zero. Each summand depends on one degree of freedom of the directional function  $v_h$  and at most  $C$  degrees of freedom of  $u_h$  summing up to an overall complexity of  $C^2 M \ll H$ . Again, this result with all its prerequisites is summarized in the following corollary.

**Corollary 2.2.4.** *Let the parametrized operators  $\mathcal{L}_h(\boldsymbol{\mu})$  be as in Corollary 2.2.3 with a Fréchet derivative at the point  $u_h \in \mathcal{W}_h$ . Then, the Fréchet derivative of the empirical operator interpolation evaluated in direction  $v_h \in \mathcal{W}_h$  is a separable approximation of  $\mathbf{D}\mathcal{L}_h(\boldsymbol{\mu})|_{u_h} [v_h]$  with complexity independent of  $H$  for all  $t \in [0, T_{\max}]$  and  $\boldsymbol{\mu} \in \mathcal{M}$  if the derivatives  $\frac{\partial}{\partial \psi_i} \tau_m^{EI}(\mathcal{L}_h(\boldsymbol{\mu})[u_h])$  exist for all  $i = 1, \dots, H$  and  $m = 1, \dots, M$ .*

Note, that it might be necessary to change the greedy algorithm, in a way such that evaluations

$$\mathbf{D}\mathcal{L}_h(\boldsymbol{\mu})|_{u_h(\boldsymbol{\mu})} [v_h(\boldsymbol{\mu})] \quad (2.14)$$

are also used in order to find functions for the collateral reduced basis space.

### 2.3. Error analysis

For the analysis of the interpolation error in empirical operator interpolation, we shortly revise that the empirical operator interpolation is theoretically equivalent to the empirical interpolation of functions

$$g(x; \boldsymbol{\mu}) := \mathcal{L}_h(\boldsymbol{\mu}) [u_h(\boldsymbol{\mu})] \quad (2.15)$$

defined by the corresponding operator evaluations for all parameters  $\boldsymbol{\mu} \in \mathcal{M}$ . Therefore, in this section the most important results from [1, 53] are only shortly cited and translated to our notation. We conclude with a discussion on the a posteriori error bound which is used in Chapter 3 in order to derive an a posteriori error for a reduced basis scheme.

In the following, we always assume, that the collateral reduced basis space and the empirical interpolation Dofs are derived with Algorithm 2.1.1, where the parameter training set equals the full parameter space  $\mathcal{M}_{\text{train}} = \mathcal{M}$  and the maximum approximation error is obtained in the  $L^\infty$ -norm.

**Theorem 2.3.1** (A priori bound). *Assuming, that for the set of admissible operator evaluations  $\mathcal{U} := \{\mathcal{L}_h(\boldsymbol{\mu}) [u_h(\boldsymbol{\mu})]\}_{\boldsymbol{\mu} \in \mathcal{M}} \subset \mathcal{W}_h$ , there is a sequence of finite dimensional subspaces*

$$\mathcal{Z}_1 \subset \mathcal{Z}_2 \subset \dots \subset \mathcal{Z}_M \subset \dots \subset \text{span } \mathcal{U}, \quad \dim \mathcal{Z}_M = M, \quad (2.16)$$

and constants  $c > 0$  and  $\alpha > \log(4)$ , such that the best approximations in the subspaces are bounded by

$$\inf_{v_M \in \mathcal{Z}_M} \|u_h - v_M\|_{\mathcal{W}_h} \leq ce^{-\alpha M} \quad (2.17)$$

for all snapshots  $u \in \mathcal{U}$ , then

$$\|\mathcal{I}_M [u] - u\| \leq ce^{-(\alpha - \log(4))M}. \quad (2.18)$$

PROOF. See [53] □

This theorem proves that the greedy Algorithm 2.1.1 iteratively improves the interpolation under the reasonable assumption of a possible exponential convergence of reduced basis approximations. The next result provides us with an a priori quality measure for the selection of the interpolation points.

**Lemma 2.3.2** (Lebesgue constant). *For any  $\boldsymbol{\mu} \in \mathcal{M}$ , the empirical operator interpolation error is related to its best approximation by*

$$\begin{aligned} & \|\mathcal{L}_h(\boldsymbol{\mu})[u_h(\boldsymbol{\mu})] - \mathcal{I}_M[\mathcal{L}_h(\boldsymbol{\mu})][u_h(\boldsymbol{\mu})]\|_{L^\infty} \\ & \leq (1 + \Lambda_M) \inf_{v_h \in \mathcal{V}_M} \|v_h - \mathcal{L}_h(\boldsymbol{\mu})[u_h(\boldsymbol{\mu})]\|_{L^\infty} \end{aligned} \quad (2.19)$$

via the Lebesgue constant

$$\Lambda_M := \sup_{x \in \Omega} \sum_{m=1}^M |\xi_m(x)| \leq 2^M - 1. \quad (2.20)$$

PROOF. See [1] □

So, theoretically the best approximation in the collateral reduced basis space can be missed by the empirical interpolation by a factor depending exponentially on the number of the interpolation points. In this very unlikely case, the Greedy algorithm does not converge, as the convergence result from Theorem 2.3.1 gets “neutralized” by the interpolation error. Although the Lebesgue constant can indeed reach this bound, as shown in [53], this is very unlikely to happen and has not been observed in practical settings. Note, that the Lebesgue constant of collateral reduced basis can be computed for verification. These observations are also verified by [59] where the empirical interpolation method is numerically compared to a “best point approximation” where expensive optimization problems are solved in order to find the best interpolation Dofs.

As the construction process of the collateral reduced basis with Algorithm 2.1.1 in each extension step computes the maximum error in the  $L^\infty(\Omega)$ -norm over the the entire parameter domain, we can define the following

**Definition 2.3.3** (Overall a posteriori error bound). *For all  $M \leq M_{\max}$ , we define the overall error bound  $\varepsilon_M^*(\boldsymbol{\mu})$  for all  $u \in \mathcal{U}$  by*

$$\|u - \mathcal{I}_{M-1}[u]\|_{L^\infty} \leq \tau_M^{EI}[\varepsilon_M] =: \varepsilon_M^*(\boldsymbol{\mu}), \quad (2.21)$$

where  $\varepsilon_M$  is the maximum residual in the  $M$ -th iteration as computed in Algorithm 2.1.1

There exists a more sophisticated rigorous a posteriori error bound [28] for the maximum interpolation error over all parameters  $\boldsymbol{\mu} \in \mathcal{M}$  mainly based on the Lebesgue constant  $\Lambda_M$  from Lemma 2.3.2. The authors improve the a posteriori error by bounding the partial derivatives of the parametric function

$$\boldsymbol{\mu} \mapsto \mathcal{L}_h(\boldsymbol{\mu})[u_h(\boldsymbol{\mu})].$$

Another approach in this direction is demonstrated in [16, 14] where also an overall error bound is given introducing an assumption that again forgoes rigorousness.

We want to conclude this section with a very simple a posteriori error estimate for the empirical operator interpolation which is sensitive in the parameter. As the empirical interpolation is exact in case of

$$\dim \mathcal{W}_M = \min \{ \dim(\mathcal{W}_h), \dim(\text{span } \mathcal{U}) \}, \quad (2.22)$$

and as we can assume exponential convergence of the greedy algorithm under the assumptions in Theorem 2.3.1, it is reasonable to further assume, that the empirical operator interpolation in a large enough collateral reduced basis space and with a sufficient number of interpolation Dofs, is exact.

**Definition 2.3.4** (A posteriori error). *Under the assumption that there exists a constant  $M' > 0$  such that  $\mathcal{I}_{M+M'}[\mathcal{L}_h(\boldsymbol{\mu})][u_h(\boldsymbol{\mu})] = \mathcal{L}_h(\boldsymbol{\mu})[u_h(\boldsymbol{\mu})]$  for all  $\boldsymbol{\mu} \in \mathcal{M}$  and all  $u_h \in \mathcal{W}_h$ , the approximation error  $\eta_{M,M'}^{EI}(\boldsymbol{\mu})$  caused by the empirical operator interpolation is given by*

$$\begin{aligned} & \| \mathcal{I}_M[\mathcal{L}_h(\boldsymbol{\mu})][u_h(\boldsymbol{\mu})] - \mathcal{L}_h(\boldsymbol{\mu})[u_h(\boldsymbol{\mu})] \|_{\mathcal{W}_h} \\ &= \left\| \sum_{m=M+1}^{M+M'} \tau_m^{EI}(\mathcal{L}_h(\boldsymbol{\mu})[u_h(\boldsymbol{\mu})]) \xi_m \right\|_{\mathcal{W}_h} =: \eta_{M,M'}^{EI}(\boldsymbol{\mu}). \end{aligned} \quad (2.23)$$

This estimate has been proposed in several works, e.g. [32, 71, 39] who fixed  $M' = 1$ . We extended this to arbitrary  $M'$  in order to obtain more reliable results. In order to evaluate the a posteriori error, it is necessary to adapt the greedy Algorithm 2.1.1 to compute more basis functions  $q_{M+1}, \dots, q_{M+M'}$  and interpolation Dofs  $\tau_{M+1}^{EI}, \dots, \tau_{M+M'}^{EI}$  after the targeted interpolation error  $\varepsilon_{\text{tol}}$  has been reached. In our numerical experiments in Chapter 5, we show that already small choices for  $M'$  can produce sufficiently good estimations of this error.

Note, that both rigorousness and sensitivity in the parameter can be combined, even if we drop the assumption in Definition 2.3.4 in an error estimate

$$\eta_{M,M'}^* := \eta_{M,M'}^{EI}(\boldsymbol{\mu}) + C_{\mathcal{W}_h} \varepsilon_{M+M'}^*, \quad (2.24)$$

where  $C_{\mathcal{W}_h} > 0$  is a constant satisfying

$$\|u\|_{L^\infty(\Omega)} \leq C_{\mathcal{W}_h} \|u\|_{\mathcal{W}_h} \quad (2.25)$$

for all  $u \in \mathcal{W}_h$ . Such a constant exists, as all norms on finite dimensional vector spaces are equivalent. Such an error estimate has been used in [71].

For the practical applications in this thesis, the bound  $\eta_{M,M'}^{EI}$  from (2.3.9) will be used, as the sensitiveness in the parameter is required for the basis

generation algorithms shown in Chapter 4 and it turns out to produce sufficiently accurate results in our experiments.

## 2.4. Invariant properties

We want to conclude this chapter by stating that operator properties with a somehow linear behavior are preserved by its empirical interpolants.

First, we want to show that the local conservation property of finite volume operators is preserved by its empirical interpolant. This result is of interest in chapters 3 and 5, as we use a finite volume schemes as a basis for our numerical experiments.

**Definition 2.4.1** (Local conservative flux operator). *In order to define the local conservation property for a flux operator  $\mathcal{L}_h : \mathcal{W}_h \rightarrow \mathcal{W}_h$ , we need a neighbor relation  $\mathcal{N} : [1, \dots, H] \rightarrow \mathcal{P}(\{1, \dots, H\})$  on the Dof indices  $1, \dots, H$  of the function space, where  $\mathcal{P}$  denotes the power set. This neighbor relation must be symmetric, such that*

$$j \in \mathcal{N}(i) \Rightarrow i \in \mathcal{N}(j). \quad (2.26)$$

*For grid based operators this usually describes the topology of the grid, i.e. the neighboring relation of cells. Then, the operator is called locally conservative for this neighbor relation  $\mathcal{N}$ , if there exist fluxes  $g_{ij} : \mathcal{W}_h \rightarrow \mathbb{R}$ , for all  $i = 1, \dots, H$  and  $j \in \mathcal{N}(i)$ , such that the operator evaluation in the  $i$ -th Dof can be written as*

$$\tau_i [\mathcal{L}_h [\cdot]] = \sum_{j \in \mathcal{N}(i)} g_{ij}(\cdot) \quad (2.27)$$

and  $g_{ij} = -g_{ji}$ .

Now, we can formulate a lemma, stating the invariance of the local conservation property for empirical interpolants of discrete operators.

**Lemma 2.4.2.** *If for all  $\boldsymbol{\mu} \in \mathcal{M}$ , the parametrized finite volume operator  $\mathcal{L}_h(\boldsymbol{\mu}) : \mathcal{W}_h \rightarrow \mathcal{W}_h$  has a locally conservative flux  $g(\boldsymbol{\mu}; \cdot)$ , then the empirical interpolant  $\mathcal{I}_M [\mathcal{L}_h(\boldsymbol{\mu})]$  inherits the local conservation property.*

**PROOF.** We assume, that the EI-GREEDY algorithm selected the parameters  $\boldsymbol{\mu}_1^{\text{EI}}, \dots, \boldsymbol{\mu}_M^{\text{EI}}$  for the extension of the collateral reduced basis space, such that the basis functions are given by

$$q_m = c_m (\mathcal{L}_h [u_h] (\boldsymbol{\mu}_m^{\text{EI}})) - \mathcal{I}_{m-1} [\mathcal{L}_h] [u_h(\boldsymbol{\mu}_m^{\text{EI}})] \quad (2.28)$$

with normalization factors  $c_m := (\tau_m^{\text{EI}}(r_m))^{-1}$  (c.f. Algorithm 2.1.1). Now, we want to show that for all  $v_h \in \mathcal{W}_h$ , the  $i$ -th Dof of the empirical operator

interpolation

$$\mathcal{I}_M [\mathcal{L}_h(\boldsymbol{\mu})] [v_h] = \sum_{i=1}^M \sigma_m(v_h) q_m \quad (2.29)$$

can be evaluated by

$$\tau_i [(\mathcal{I}_M [\mathcal{L}_h(\boldsymbol{\mu})] [v_h])] = \sum_{j \in \mathcal{N}(i)} g_{ij}^{\mathcal{I}_M}(\boldsymbol{\mu}; v_h) \quad (2.30)$$

for all  $v_h \in \mathcal{W}_h$  with a parametrized flux  $g^{\mathcal{I}_M}(\boldsymbol{\mu}; \cdot)$  recursively defined by

$$g_{ij}^{\mathcal{I}_M}(\boldsymbol{\mu}; v_h) := \sigma_M(v_h) \left( g_{ij}(\boldsymbol{\mu}_M^{\text{EI}}; u_h(\boldsymbol{\mu}_M^{\text{EI}})) - g_{ij}^{\mathcal{I}_{M-1}}(\boldsymbol{\mu}_M^{\text{EI}}; u_h(\boldsymbol{\mu}_M^{\text{EI}})) \right) + g_{ij}^{\mathcal{I}_{M-1}}(\boldsymbol{\mu}; v_h) \quad (2.31)$$

for all  $M > 0$  and  $g_{ij}^{\mathcal{I}_0}(\boldsymbol{\mu}; \cdot) := 0$ .

From equation (2.4.3), it follows by induction and with the local conservation property for all parameters that

$$\tau_i [q_M] = c_M \cdot \sum_{j \in \mathcal{N}(i)} g_{ij}(\boldsymbol{\mu}_M^{\text{EI}}; u_h(\boldsymbol{\mu}_M^{\text{EI}})) - g_{ij}^{\mathcal{I}_{M-1}}(\boldsymbol{\mu}_M^{\text{EI}}; u_h(\boldsymbol{\mu}_M^{\text{EI}})). \quad (2.32)$$

Then, from (2.4.4) follows

$$(\mathcal{I}_M [\mathcal{L}_h(\boldsymbol{\mu})] [v_h])_i = \sigma_M^M(v_h) (q_M)_i + \sum_{j \in \mathcal{N}(i)} g_{ij}^{\mathcal{I}_{M-1}}(v_h), \quad (2.33)$$

and such, after substituting (2.4.7) into (2.4.8),

$$g_{ij}^{\mathcal{I}_M}(v_h) = -g_{ji}^{\mathcal{I}_M}(v_h), \quad (2.34)$$

because  $g(\boldsymbol{\mu}; \cdot)$  and  $g^{\mathcal{I}_{M-1}}(\boldsymbol{\mu}; \cdot)$  are both conservative fluxes by assumption and by induction, respectively. Therefore, the sum of conservative fluxes stays conservative.  $\square$

Of course, the above lemma stays true for other operator properties behaving in a somehow linear way. Sometimes, however, these properties do not apply to all operators  $\mathcal{L}_h(\boldsymbol{\mu})$  in the parameter domain. In this case, one can simply generate two sets of collateral reduced basis functions and interpolation points, one for the parameters on which the property applies, and one set for the other parameters.

We conclude with a further remark on properties common to all operator evaluations.

**Remark 2.4.3.** *It is a trivial fact, but noteworthy, that the set of interpolated operator evaluations  $\cup_{\boldsymbol{\mu} \in \mathcal{M}} \mathcal{I}_M [\mathcal{L}_h(\boldsymbol{\mu})] [u_h(\boldsymbol{\mu})]$  is a subset of the convex hull of the original operators evaluations  $\mathcal{U} := \cup_{\boldsymbol{\mu} \in \mathcal{M}} \mathcal{L}_h(\boldsymbol{\mu}) [u_h(\boldsymbol{\mu})]$ . Therefore, a property which applies to all functions of this convex hull  $\text{conv}(\mathcal{U})$  is preserved*

by the empirical operator interpolation. An example is the global conservation property stating that discrete functions  $v_h \in \mathcal{U}$  have zero mean  $\int_{\Omega} v_h = 0$ .

Again, it can happen, that the property is restricted to a subset of the parameter space, but it might also happen that it apply to a local restriction of the spatial domain only. The latter case can be observed for finite volume operators modeling Dirichlet boundary conditions, for example. Here, it is preferable to split the operators in two parts  $\mathcal{L}_h(\boldsymbol{\mu}) := \mathcal{L}_h^1(\boldsymbol{\mu}) + \mathcal{L}_h^2(\boldsymbol{\mu})$ , such that the required property holds for  $\mathcal{L}_h^1(\boldsymbol{\mu})$  for all  $\boldsymbol{\mu} \in \mathcal{M}$ . Then, again the EI-GREEDY algorithm should be applied to both operators separately leading to interpolations  $\mathcal{I}_{M^1}^1$  and  $\mathcal{I}_{M^2}^2$ , such that the required operator property is conserved locally.

## CHAPTER 3

### Reduced basis method

In this chapter, we want to develop an efficient reduced basis scheme for non-linear parametrized evolution equations. These are problems which are characterized by a parameter vector  $\boldsymbol{\mu} \in \mathcal{M}$  from a set of admissible parameters  $\mathcal{M} \subset \mathbb{R}^p$ .

For a fixed  $\boldsymbol{\mu} \in \mathcal{M}$  the evolution problem consists of solving for functions  $u(x; t, \boldsymbol{\mu})$  on a bounded domain  $\Omega \subset \mathbb{R}^d$  and a finite time interval  $[0, T]$ ,  $T > 0$ , such that

$$\partial_t u(t, \boldsymbol{\mu}) + \mathcal{L}(t, \boldsymbol{\mu}) [u(t, \boldsymbol{\mu})] = 0, \quad u(0, \boldsymbol{\mu}) = u_0(\boldsymbol{\mu}), \quad (3.1)$$

and suitable boundary conditions are satisfied. Here,  $u_0(\boldsymbol{\mu})$  are the parameter dependent initial values and  $\mathcal{L}(t, \boldsymbol{\mu})$  is a parameter dependent spatial differential operator. The initial value and the solution are supposed to have some spatial regularity  $u_0(\boldsymbol{\mu}), u(\cdot; t, \boldsymbol{\mu}) \in \mathcal{W} \subset L^2(\Omega)$ .

In the following section, an abstract numerical scheme for the above problem is introduced, for which a reduced scheme with an offline-/online decomposition is demonstrated in Section 3.2. Furthermore, an example finite volume discretization for a general convection diffusion problem is presented as a basis for the implementation of the test problems used in the numerical experiments in Chapter 5

The chapter concludes with Section 3.3 on error analysis of the error between the reduced and the high dimensional solutions, with the proposal of an a posteriori error estimator, which we first published in [25]. Throughout this chapter, we assume the existence of reduced basis spaces and all necessary empirical interpolation data. The next chapter 4 comprises a detailed discussion on several ways to generate these magnitudes during the offline phase of the reduced basis method.

#### 3.1. Evolution Scheme

In this section, we define an operator based numerical scheme which can be understood as an abstract formulation for many modern discretizations of the parametrized evolution problem (3.0.10).

For the discrete solutions of the analytical problem, a finite dimensional Hilbert space  $\mathcal{W}_h \subset L^2(\Omega)$  with a suitable norm  $\|\cdot\|_{\mathcal{W}_h}$  is needed.

Considering a time discretization at parameter independent time instants  $0 = t^0 < t^1 < \dots < t^K = T$ , the evolution scheme produces discrete *solution snapshots*  $u_h^k(\boldsymbol{\mu})$  for  $k = 0, \dots, K$ .

The following Definition 3.1.1 defines an operator based numerical scheme with a first order discretization in time and a separation of the differential operator into *explicit* and *implicit* contributions operating on either the solution snapshot at the last computed time instant  $k$  or the first unknown snapshot at time instant  $k + 1$ . Both operator parts may depend in a non-linear way on the argument, where the non-linear implicit part will be treated by a Newton–Raphson method. For clarity of exposition, we fix the time step size to a constant  $\Delta t$ , but of course, it is possible to choose it problem dependent.

**Definition 3.1.1** (General parametrized evolution scheme). *Let  $\mathcal{W}_h$  be an  $H$ -dimensional discrete function space with a basis  $\{\psi_i\}_{i=1}^H$  and  $t^k := k\Delta t, k = 0, \dots, K$  be a sequence of  $K + 1$  strictly increasing time instances with a global time step size  $\Delta t > 0$ . Furthermore, there needs to exist a projection  $\mathcal{P}_h : L^2(\Omega) \rightarrow \mathcal{W}_h$  onto the discrete function space, and we assume an arbitrary space discretization operator  $\mathcal{L}_h := \mathcal{L}_I + \mathcal{L}_E$  decomposed in its implicit and explicit contributions  $\mathcal{L}_I := \mathcal{L}_I(t^k, \boldsymbol{\mu}), \mathcal{L}_E := \mathcal{L}_E(t^k, \boldsymbol{\mu}) : \mathcal{W}_h \rightarrow \mathcal{W}_h$ . For each parameter  $\boldsymbol{\mu} \in \mathcal{M}$  we define a numerical scheme for discrete solutions  $u_h^k := u_h^k(\boldsymbol{\mu}) = \sum_{i=1}^H u_{h,i}^k \psi_i \in \mathcal{W}_h$  at time instances  $t^k$  for  $k = 0, \dots, K$  by initial projection*

$$u_h^0 = \mathcal{P}_h [u_0(\boldsymbol{\mu})], \quad (3.2)$$

and subsequently solving the equations

$$F [u_h^{k+1}] := (\text{Id} + \Delta t \mathcal{L}_I) [u_h^{k+1}] - (\text{Id} - \Delta t \mathcal{L}_E) [u_h^k] = 0, \quad (3.3)$$

with the Newton–Raphson method. In each Newton step, we solve for the defect  $\delta_h^{k+1, \nu+1}$  in

$$\mathbf{D}F|_{u_h^{k+1, \nu}} [\delta_h^{k+1, \nu+1}] = -F [u_h^{k+1, \nu}], \quad (3.4)$$

where  $u_h^{k+1, 0} := u_h^k$  and  $u_h^{k+1, \nu+1} := u_h^{k+1, \nu} + \delta_h^{k+1, \nu+1}$  define the updates in each Newton step, and the solution at time instance  $t^{k+1}$  is given by  $u_h^{k+1} := u_h^{k+1, \nu_{\max}^k}$ . Here, the last Newton step index  $\nu_{\max}^k$  equals the smallest integer  $\nu$  satisfying the inequality

$$\left\| R_{h, \text{New}}^k \right\|_{\mathcal{W}_h} \leq \varepsilon^{\text{New}} \quad (3.5)$$

for the Newton residual

$$R_{h,New}^k := u_h^{k+1} - u_h^k + \Delta t \left( \mathcal{L}_I [u_h^{k+1}] + \mathcal{L}_E [u_h^k] \right) \quad (3.6)$$

and a predefined residual error bound  $\varepsilon^{New} > 0$ .

Note that, if  $\mathcal{L}_I$  is linear, a single Newton-step is sufficient and in case  $\mathcal{L}_I$  is zero, we obtain a purely explicit scheme. As a special case, the Crank–Nicolson scheme of second order is also covered.

**3.1.1. Finite volume scheme.** In this subsection, example operators for the numerical scheme from 3.1.1 are introduced, resulting in a finite volume scheme. For this, we consider a special instances of the evolution problem (3.0.10), solving the following scalar nonlinear convection–diffusion problem on a polygonal domain  $\Omega \subset \mathbb{R}^2$  with the abbreviation  $u = u(t; \boldsymbol{\mu})$  for a clearer exposition:

$$\partial_t u + \nabla \cdot (\mathbf{v}(u; \boldsymbol{\mu})u) - \nabla \cdot (d(u; \boldsymbol{\mu})\nabla u) = 0 \quad \text{in } \Omega \times [0, T_{\max}] \quad (3.7)$$

with suitable parametrized functions  $\mathbf{v}(\cdot; \boldsymbol{\mu}) \in C(\mathbb{R}, \mathbb{R}^d)$  for the convective direction and  $d(\cdot; \boldsymbol{\mu}) \in C(\mathbb{R}, \mathbb{R}_0^+)$  for the diffusion coefficient. Furthermore, we prescribe

$$u(0; \boldsymbol{\mu}) = u_0(\boldsymbol{\mu}) \quad \text{in } \Omega \times \{0\}, \quad (3.8)$$

$$u(\boldsymbol{\mu}) = u_{\text{dir}}(\boldsymbol{\mu}) \quad \text{on } \Gamma_{\text{dir}} \times [0, T_{\max}], \quad (3.9)$$

$$(\mathbf{v}(u; \boldsymbol{\mu})u - d(u; \boldsymbol{\mu})\nabla u) \cdot \mathbf{n} = u_{\text{neu}}(\boldsymbol{\mu}) \quad \text{on } \Gamma_{\text{neu}} \times [0, T_{\max}] \quad (3.10)$$

and cyclical boundary conditions on the remaining boundary  $\partial\Omega \setminus (\Gamma_{\text{dir}} \cup \Gamma_{\text{neu}})$ . Here,  $\mathbf{n}$  denotes the outer normal on the boundary. Note, that we also allow  $d \equiv 0$ .

We denote  $\mathcal{W}$  as the exact solution space with respect to the space variable that can be chosen e.g. as  $L^\infty(\Omega) \cap BV(\Omega) \subset L^2(\Omega)$ . We obtain unique entropy solutions in  $L^\infty([0, T_{\max}]; \mathcal{W})$  if the data and boundary functions fulfill adequate regularity conditions. For a discussion on well-posedness, uniqueness and existence of entropy solutions for these kind of finite volume problems, we refer to e.g. [13, 46].

*Numerical scheme.* Before we formulate the numerical scheme, we must introduce some notations.

**Definition 3.1.2** (Numerical grid). *Let  $\mathcal{T} := \{e_i\}_{i=1}^H$  denote 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 all connections of two of these points in adjacent elements are perpendicular to corresponding edges. The cell's edges are denoted by  $e_{ij}$  with  $j \in \mathcal{N}_{in}(i) \cup \mathcal{N}_{neu}(i) \cup \mathcal{N}_{dir}(i)$ . Here, the index set  $\mathcal{N}_{in}(i)$  comprises cell indices of all elements adjacent to  $e_i$  and  $\mathcal{N}_{neu}(i)$  and  $\mathcal{N}_{dir}(i)$  are enumerations of edges on which a Neumann respectively a Dirichlet condition is imposed. On each edge  $e_{ij}$ , we denote

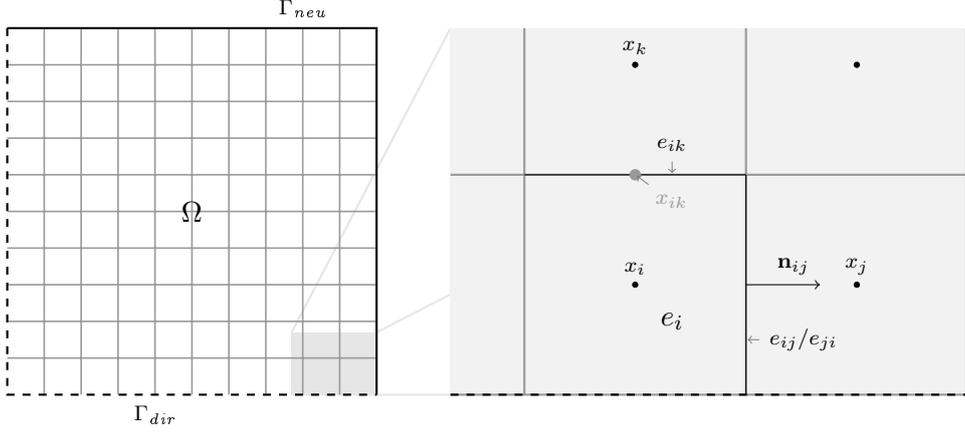


FIGURE 3.1.1. Excerpt of a rectangular grid with notations used in this paper.

the barycenter by  $x_{ij}$  and the outer unit normal by  $\mathbf{n}_{ij}$ .

**Definition 3.1.3** (Finite volume space). *Given a numerical grid  $\mathcal{T}$  like in Definition 3.1.2, on each grid cell  $e_i \in \mathcal{T}$ , we define piecewise constant indicator functions  $\psi_i := \chi_{e_i}$ , spanning a discrete finite volume space  $\mathcal{W}_h := \text{span} \{\psi_i\}_{i=1}^H$ . We denote the degrees of freedom of a finite volume function  $u_h \in \mathcal{W}_h$  by  $u_{h,i} = \tau_i(u_h) := u_h(x_i)$ . For the time interval discretization, we choose the global time step size  $\Delta t$  small enough such that a Courant–Friedrichs–Lewy (CFL) condition is fulfilled for all parameters  $\boldsymbol{\mu} \in \mathcal{M}$ .*

The implicit and explicit space discretization operator need to model the diffusive respectively the convective dynamics of the underlying partial differential equations. Therefore we define

$$\mathcal{L}_I(\boldsymbol{\mu}) := \alpha \mathcal{L}_{\text{diff}}(\boldsymbol{\mu}) + \beta \mathcal{L}_{\text{conv}}(\boldsymbol{\mu}), \quad (3.11)$$

$$\mathcal{L}_E(\boldsymbol{\mu}) := (1 - \alpha) \mathcal{L}_{\text{diff}}(\boldsymbol{\mu}) + (1 - \beta) \mathcal{L}_{\text{conv}}(\boldsymbol{\mu}) \quad (3.12)$$

with constants  $0 \leq \alpha, \beta \leq 1$  and finite volume operators  $\mathcal{L}_{\text{diff}}$  and  $\mathcal{L}_{\text{conv}}$  specified below. A judicious choice for the constants is  $\alpha = 1$  and  $\beta = 0$ , because the greater stiffness of diffusion dynamics requires implicit discretizations,

whereas for instationary problems, it is computationally more efficient to discretize convection terms explicitly. [30] Note that the operators are constant in time, but the scheme applies to time-varying operators as well.

The main idea of the finite volume method is to compute cell-wise averages over the solutions and to substitute the occurring volume integrals containing divergence terms into surface integrals with the Gauss–Ostrogradsky theorem, such that e.g.

$$\nabla \cdot \varphi \approx \frac{1}{|e_i|} \int_{e_i} \nabla \cdot \varphi = \frac{1}{|e_i|} \int_{\partial e_i} \varphi \cdot \mathbf{n}. \quad (3.13)$$

For the diffusion operator, a finite difference approximation of the normal derivative gives us the Dof-wise definition

$$\begin{aligned} (\mathcal{L}_{\text{diff}}(\boldsymbol{\mu})[u_h])_i &= -\frac{1}{|e_i|} \sum_{j \in \mathcal{N}_{\text{in}}(i)} \overline{d(u_h; \boldsymbol{\mu})_{ij}} \frac{u_{h,j} - u_{h,i}}{|x_j - x_i|} |e_{ij}| \\ &\quad - \frac{1}{|e_i|} \sum_{j \in \mathcal{N}_{\text{dir}}(i)} d(u_{\text{dir}}(x_{ij}; \boldsymbol{\mu}); \boldsymbol{\mu}) \frac{u_{\text{dir}}(x_{ij}; \boldsymbol{\mu}) - u_{h,i}}{2|x_{ij} - x_i|} |e_{ij}|, \end{aligned} \quad (3.14)$$

where  $|e_i|$  is the volume of the grid cell  $e_i$  and  $\overline{d(u_h; \boldsymbol{\mu})_{ij}}$  computes a suitable mean on the edge  $e_{ij}$ . In our experiments, we will mainly use the harmonic mean

$$\overline{d(u_h; \boldsymbol{\mu})_{ij}}^{\text{harm}} := \frac{2d(u_j; \boldsymbol{\mu})d(u_i; \boldsymbol{\mu})}{d(u_j; \boldsymbol{\mu}) + d(u_i; \boldsymbol{\mu})}. \quad (3.15)$$

In order to resolve the non-linearity of the diffusion in a numerical scheme with the Newton–Raphson method, we also need the operator’s directional derivative at a point  $u_h \in \mathcal{W}_h$

$$\begin{aligned} (\mathbf{D}\mathcal{L}_{\text{diff}}(\boldsymbol{\mu})|_{u_h}[v_h])_i &= -\frac{1}{|e_i|} \sum_{j \in \mathcal{N}_{\text{in}}(i)} \left( \mathbf{D}\overline{d(\cdot; \boldsymbol{\mu})_{ij}}|_{u_h}[v_h] \frac{u_{h,j} - u_{h,i}}{|x_j - x_i|} |e_{ij}| \right. \\ &\quad \left. + \overline{d(u_h; \boldsymbol{\mu})_{ij}} \frac{v_{h,j} - v_{h,i}}{|x_j - x_i|} |e_{ij}| \right) \\ &\quad - \frac{1}{|e_i|} \sum_{j \in \mathcal{N}_{\text{dir}}(i)} d(u_{\text{dir}}(x_{ij}; \boldsymbol{\mu}); \boldsymbol{\mu}) \frac{-v_{h,i}}{2|x_{ij} - x_i|} |e_{ij}|. \end{aligned} \quad (3.16)$$

Likewise, we define the finite volume operator for the convection term by

$$\begin{aligned} (\mathcal{L}_{\text{conv}}(\boldsymbol{\mu})[u_h])_i &= \frac{1}{|e_i|} \sum_{j \in \mathcal{N}_{\text{in}}(i)} g_{ij}(u_{h,i}, u_{h,j}; \boldsymbol{\mu}) \\ &\quad + \frac{1}{|e_i|} \sum_{j \in \mathcal{N}_{\text{dir}}(i)} g_{ij}(u_{h,i}, u_{\text{dir}}(x_{ij}); \boldsymbol{\mu}) \\ &\quad + \frac{1}{|e_i|} \sum_{j \in \mathcal{N}_{\text{neu}}(i)} \int_{e_{ij}} u_{\text{neu}}(\boldsymbol{\mu}) \end{aligned} \quad (3.17)$$

with Engquist-Osher flux functions  $g_{ij}$  leading to low numerical viscosity in this scheme. The flux functions can be expressed by setting  $c_{ij}(u; \boldsymbol{\mu}) := \mathbf{n}_{ij} \mathbf{v}(u; \boldsymbol{\mu}) u$  for all edges, defining

$$c_{ij}^+(u; \boldsymbol{\mu}) := c_{ij}(0; \boldsymbol{\mu}) + \int_0^u \max(c'_{ij}(s; \boldsymbol{\mu}), 0) ds, \quad (3.18)$$

$$c_{ij}^-(u; \boldsymbol{\mu}) := \int_0^u \min(c'_{ij}(s; \boldsymbol{\mu}), 0) ds \quad (3.19)$$

and choosing the flux as  $g_{ij}(u, v; \boldsymbol{\mu}) := |e_{ij}| \left\{ c_{ij}^+(u; \boldsymbol{\mu}) + c_{ij}^-(v; \boldsymbol{\mu}) \right\}$ , cf. [30, 48]. The corresponding directional derivative w.r.t.  $v_h$  of the Engquist-Osher flux operator is given by

$$\begin{aligned} (\mathbf{D}\mathcal{L}_{\text{conv}}(\boldsymbol{\mu})|_{u_h} [v_h])_i &= \frac{1}{|e_i|} \sum_{j \in \mathcal{N}_{\text{in}}(i)} (\partial_1 g_{ij}(u_{h,i}, u_{h,j}; \boldsymbol{\mu}) v_{h,i} \\ &\quad + \partial_2 g_{ij}(u_{h,i}, u_{h,j}; \boldsymbol{\mu}) v_{h,j}) \\ &\quad + \frac{1}{|e_i|} \sum_{j \in \mathcal{N}_{\text{dir}}(i)} \partial_1 g_{ij}(u_{h,i}, u_{\text{dir}}(x_{ij}; \boldsymbol{\mu})) v_{h,i}. \end{aligned} \quad (3.20)$$

In order to complete the scheme, we can project the initial data onto the discrete function space via a cell averaging operator  $\mathcal{P}_h : \mathcal{W} \rightarrow \mathcal{W}_h$ , Dof-wise defined by  $(\mathcal{P}_h [v_h])_i := \frac{1}{|e_i|} \int_{e_i} v_h$  and obtain a specification of the generalized numerical scheme from Definition 3.1.1.

**Remark 3.1.4** (Restriction to interpolation DOFs). *From the Dof-wise definitions of the operators (3.1.13) and (3.1.16), it follows that a constant number of flops dependent on the maximum number of cell neighbors suffices to numerically compute a single degree of freedom from the operator evaluation result. Therefore, the finite volume operators fulfill the  $H$ -independent Dof dependence condition and are suitable for empirical interpolation with the constant  $C$  bounded by one plus the maximum number of edges of an element.*

A crucial property of finite volume operators is their local conservation property assuring that everything flowing out of a cell flows into a neighboring one. For an arbitrary finite volume (update) operator in the flux formulation

$$(\mathcal{L}_h [u_h])_i = \sum_{j \in \mathcal{N}(i)} g_{ij}(u_h). \quad (3.21)$$

the local conservation property holds, iff

$$g_{ij}(u_h) = -g_{ji}(u_h). \quad (3.22)$$

In Lemma 2.4.2, it was already shown, that this property is inherited by the empirical interpolant of the operator.

As a corollary or from Remark 2.4.3, it also follows that global conservation, i.e.

$$\int_{\Omega} \mathcal{L}_h[v_h] = 0 \quad \text{for all } v_h \in \mathcal{W}_h, \quad (3.23)$$

is preserved under empirical interpolation. In case of trivial boundary conditions, the operators (3.1.13) and (3.1.16) are conservative in this sense for all  $\boldsymbol{\mu} \in \mathcal{M}$ .

For non-trivial boundary conditions, however, the operators can be split into a sum of operators acting on different domains. If, e.g. one of these addends computes the fluxes over inner edges only, it preserves the global conservation property under empirical interpolation. (c.f. Remark 2.4.3).

Note, that the interpolation procedure and the reduced scheme are identically applicable to other evolution problems, discrete function spaces and discretization operators, e.g. finite element or discontinuous Galerkin methods. Hence, for the following development of the reduced basis method, we will express the numerical scheme in terms of the more general notions from Definition 3.1.1.

### 3.2. Reduced simulation scheme

In this section, we introduce a reduced basis scheme for the evolution problem from Definition 3.1.1 and compare the computational complexity of both schemes. In order to formulate the reduced scheme, the existence of reduced bases for spaces for a Galerkin projection of the detailed data and empirical interpolation of the non-linear operators is assumed throughout the rest of this chapter. In detail, we define

- reduced basis functions  $\{\varphi_n\}_{n=1}^N \subset \mathcal{W}_h$  spanning a reduced basis space  $\mathcal{W}_{\text{red}} \subset \mathcal{W}_h$  and
- empirical interpolation data suitable for both operators, i.e. collateral reduced basis functions  $\{\xi_m\}_{m=1}^M \subset \mathcal{W}_h$  and corresponding interpolation DOFs  $\Sigma_M$ .

The question how to retrieve the above functions and interpolation DOFs algorithmically, is postponed to the next Chapter 4.

The reduced basis scheme introduced here is based on a similar formulation for explicit discretizations of evolution problems in [39, 37]. We extend these schemes by allowing non-linear or non-separable implicit operator contributions. The basic idea for the reduced basis scheme is twofold: First, the discrete evolution operators  $\mathcal{L}_E$  and  $\mathcal{L}_I$  from Definition 3.1.1 are substituted by their empirical interpolants  $\mathcal{I}_M[\mathcal{L}_E]$  and  $\mathcal{I}_M[\mathcal{L}_I]$ . Second, an orthogonal

projection of the numerical scheme onto the reduced basis space  $\mathcal{W}_{\text{red}}$  with respect to the scalar product of  $\mathcal{W}_h$  is applied.

For this purpose, we introduce the corresponding projection operator  $\mathcal{P}_{\text{red}} : \mathcal{W} \rightarrow \mathcal{W}_{\text{red}}$  satisfying

$$\langle \mathcal{P}_{\text{red}} [u], \varphi \rangle_{\mathcal{W}_h} = \langle u, \varphi \rangle_{\mathcal{W}_h} \quad \forall \varphi \in \mathcal{W}_{\text{red}}$$

and define reduced variants of the discrete operators

$$\mathcal{L}_{\text{red},E} := \mathcal{P}_{\text{red}} \circ \mathcal{I}_M \circ \mathcal{L}_E \quad \text{and} \quad \mathcal{L}_{\text{red},I} := \mathcal{P}_{\text{red}} \circ \mathcal{I}_M \circ \mathcal{L}_I. \quad (3.24)$$

For all  $\boldsymbol{\mu} \in \mathcal{M}$ , trajectories  $\{u_{\text{red}}^k(\boldsymbol{\mu})\}_{k=0}^K$  with snapshots  $u_{\text{red}}^k(\boldsymbol{\mu}) \in \mathcal{W}_{\text{red}}$  can be obtained for  $k = 0, \dots, K$  analogously to the evolution scheme described in Definition 3.1.1. The reduced initial data is given by projection of the initial data

$$u_{\text{red}}^0 := u_{\text{red}}^0(\boldsymbol{\mu}) = \mathcal{P}_{\text{red}} [u_0(\boldsymbol{\mu})]. \quad (3.25)$$

Then, for each  $k = 0, \dots, K - 1$  Newton step solutions are computed by minimizing the defects  $\delta_{\text{red}}^{k+1,\nu} := \delta_{\text{red}}^{k+1,\nu}(\boldsymbol{\mu})$  for  $\nu = 0, \dots, \nu_{\text{max}}^k(\boldsymbol{\mu})$  which solve

$$\begin{aligned} & \left( \text{Id} + \Delta t \mathbf{D} \mathcal{L}_{\text{red},I} \Big|_{u_{\text{red}}^{k+1,\nu}} \right) \left[ \delta_{\text{red}}^{k+1,\nu+1} \right] \\ & = -u_{\text{red}}^{k+1,\nu} + u_{\text{red}}^k \Delta t \left[ \mathcal{L}_{\text{red},I} \left[ u_{\text{red}}^{k+1,\nu} \right] + \mathcal{L}_{\text{red},E} \left[ u_{\text{red}}^k \right] \right], \end{aligned} \quad (3.26)$$

with  $u_{\text{red}}^{k+1,0} := u_{\text{red}}^k$ ,  $u_{\text{red}}^{k+1,\nu+1} := u_{\text{red}}^{k+1,\nu} + \delta_{\text{red}}^{k+1,\nu+1}$  for  $\nu = 1, \dots, \nu_{\text{max}}^k - 1$ , and finally assigning the reduced solution for the next time step by

$$u_{\text{red}}^{k+1} := u_{\text{red}}^{k+1,\nu_{\text{max}}^k}. \quad (3.27)$$

Here, the final Newton iteration index  $\nu_{\text{max}}^k$  is the smallest integer such that the norm of the residual defined as

$$R_{\text{red},\text{New}}^k := u_{\text{red}}^{k+1} - u_{\text{red}}^k + \Delta t \left( \mathcal{L}_{\text{red},I} \left[ u_{\text{red}}^{k+1} \right] + \mathcal{L}_{\text{red},E} \left[ u_{\text{red}}^k \right] \right) \quad (3.28)$$

drops below a given tolerance  $\varepsilon^{\text{New}}$ .

**Remark 3.2.1.** *If the implicit operator  $\mathcal{L}_I$  is linear, a single Newton-step is sufficient. If  $\mathcal{L}_I$  is zero, no Newton step at all is necessary and the numerical scheme degenerates to the purely explicit one, already discussed in [39]. Non-linear parabolic problems with finite element discretizations are also considered in [33, 8] and [32]. Similar to the empirical operator interpolation based approach presented in this work, those reduced basis methods make use of an empirical interpolation applied to specific data functions. In this case, the interpolation Dofs  $\{\tau_m^{EI}\}_{m=1}^M$  also define point evaluations at*

“magic points” for which the “ $H$ -independent Dof dependence” is trivially fulfilled with  $C = 1$ . Another approach to deal with non-linearities of low-degree polynomial form is described in [73, 70, 8]. Therefor, it can also be a good idea for some problems to approximate the non-linearities by polynomials as proposed in [32].

For easier analysis of the computational complexity during offline and online phase, we translate the above sketched reduced scheme into a vector-valued formulation based on the few degrees of freedom of the reduced solution.

**Definition 3.2.2** (Reduced basis scheme with empirical operator interpolation). *We assume a numerical scheme from Definition 3.1.1 with operators  $\mathcal{L}_E$  and  $\mathcal{L}_I$  fulfilling an  $H$ -independent Dof dependence. Hence, we can assume that an appropriate empirical interpolation operator  $\mathcal{I}_M$  is defined by means of a nodal empirical interpolation basis  $\boldsymbol{\xi}_M$  and an enumerated subset of degrees of freedom  $\Sigma_M := \{\tau_m^{EI}\}_{m=1}^M \subset \Sigma_h$ . The collateral reduced basis space shall be the same for both the operators  $\mathcal{L}_E$  and  $\mathcal{L}_I$  (c.f. Remark 3.2.3). Furthermore, there must be a reduced basis  $\Phi_N := \{\varphi_n\}_{n=1}^N$  available that spans the reduced basis space  $\mathcal{W}_{\text{red}} \subset \mathcal{W}_h$ . We define the following scheme for sequentially expressing*

- the reduced solution  $u_{\text{red}}^k(\boldsymbol{\mu}) := \sum_{n=1}^N a_n^k(\boldsymbol{\mu})\varphi_n$ ,
- intermediate Newton step solutions  $u_{\text{red}}^{k,\nu}(\boldsymbol{\mu}) := \sum_{n=1}^N a_n^{k,\nu}(\boldsymbol{\mu})\varphi_n$  and
- Newton step defects  $\delta_{\text{red}}^{k,\nu}(\boldsymbol{\mu}) := \sum_{n=1}^N d_n^{k,\nu}(\boldsymbol{\mu})\varphi_n$

by computing the coefficient vectors

$$\begin{aligned} \mathbf{a}^k &:= \mathbf{a}^k(\boldsymbol{\mu}) = \left( a_1^k(\boldsymbol{\mu}), \dots, a_N^k(\boldsymbol{\mu}) \right)^T, \\ \mathbf{a}^{k,\nu} &:= \mathbf{a}^{k,\nu}(\boldsymbol{\mu}) = \left( a_1^{k,\nu}(\boldsymbol{\mu}), \dots, a_N^{k,\nu}(\boldsymbol{\mu}) \right)^T \quad \text{and} \\ \mathbf{d}^{k,\nu} &:= \mathbf{d}^{k,\nu}(\boldsymbol{\mu}) = \left( d_1^{k,\nu}(\boldsymbol{\mu}), \dots, d_N^{k,\nu}(\boldsymbol{\mu}) \right)^T \end{aligned}$$

for  $k = 0, \dots, K$  and  $\nu = 0, \dots, \nu_{\max}^k(\boldsymbol{\mu})$ :

The initial solution vector is simply obtained by projection onto the reduced basis space. So, e.g. for an orthonormal basis  $\Phi_N$  the following computation applies:

$$\mathbf{a}^0 := \left( \langle u_0(\boldsymbol{\mu}), \varphi_1 \rangle_{\mathcal{W}_h}, \dots, \langle u_0(\boldsymbol{\mu}), \varphi_N \rangle_{\mathcal{W}_h} \right)^T. \quad (3.29)$$

Then, for each time index  $k = 0, \dots, K - 1$ , we compute Newton iterations by finding defects  $\mathbf{d}^{k+1, \nu+1}$  and residuals  $\mathbf{r}^{k+1, \nu+1}$  solving for  $\nu = 0, \dots, \nu_{\max}^k(\boldsymbol{\mu}) - 1$  the equations

$$\begin{aligned} & \left( \text{Id} + \Delta t \mathbf{C} \mathbf{I}'_I(t^k, \boldsymbol{\mu}) \left[ \mathbf{a}^{k+1, \nu} \right] \right) \left[ \mathbf{d}^{k+1, \nu+1} \right] \\ &= -\mathbf{a}^{k+1, \nu} + \mathbf{a}^{k+1, 0} \\ & \quad - \Delta t \mathbf{C} \left( \mathbf{I}_I(t^k, \boldsymbol{\mu}) \left[ \mathbf{a}^{k+1, \nu} \right] + \mathbf{I}_E(t^k, \boldsymbol{\mu}) \left[ \mathbf{a}^{k+1, 0} \right] \right), \end{aligned} \quad (3.30)$$

$$\begin{aligned} \mathbf{r}^{k+1, \nu+1}(\boldsymbol{\mu}) &:= \mathbf{a}^{k+1, \nu+1} - \mathbf{a}^{k+1, 0} \\ & \quad + \Delta t \mathbf{C} \left( \mathbf{I}_I(t^k, \boldsymbol{\mu}) \left[ \mathbf{a}^{k+1, \nu+1} \right] + \mathbf{I}_E(t^k, \boldsymbol{\mu}) \left[ \mathbf{a}^{k+1, 0} \right] \right) \end{aligned} \quad (3.31)$$

with updates

$$\begin{aligned} \mathbf{a}^{k+1, 0} &:= \mathbf{a}^k, & \mathbf{a}^{k+1, \nu+1} &:= \mathbf{a}^{k+1, \nu} + \mathbf{d}^{k+1, \nu+1}, \\ \mathbf{a}^{k+1} &:= \mathbf{a}^{k+1, \nu_{\max}^k(\boldsymbol{\mu})}. \end{aligned} \quad (3.32)$$

The number of Newton steps  $\nu_{\max}^k(\boldsymbol{\mu})$  at each time step is chosen as the smallest integer  $\nu$  such that the residual norm drops below the specified tolerance for the Newton scheme, i.e. for which

$$\left( \left( \mathbf{r}^{k+1, \nu+1}(\boldsymbol{\mu}) \right)^T \mathbf{M} \mathbf{r}^{k+1, \nu+1}(\boldsymbol{\mu}) \right)^{\frac{1}{2}} < \varepsilon^{\text{New}}$$

holds.

Here, the utilized vectors and matrices are defined as

$$(\mathbf{M})_{nn'} := \langle \varphi_n, \varphi_{n'} \rangle_{\mathcal{W}_h} = \delta_{nn'}, \quad (3.33)$$

$$(\mathbf{C})_{nm} := \langle \xi_m, \varphi_n \rangle_{\mathcal{W}_h}, \quad (3.34)$$

$$\left( \mathbf{I}'_I(t^k, \boldsymbol{\mu}) \left[ \mathbf{a}^{k+1, \nu} \right] \right)_{mn} := \sum_{i=1}^H \frac{\partial}{\partial \psi_i} \left( \tau_m^{EI} \circ \mathcal{L}_I(t^k, \boldsymbol{\mu}) \right) \left[ u_{\text{red}}^{k+1, \nu} \right] \tau_i(\varphi_n), \quad (3.35)$$

$$\left( \mathbf{I}_I(t^k, \boldsymbol{\mu}) \left[ \mathbf{a}^{k+1, \nu} \right] \right)_m := \tau_m^{EI} \left( \mathcal{L}_I(t^k, \boldsymbol{\mu}) \left[ u_{\text{red}}^{k+1, \nu} \right] \right), \quad (3.36)$$

$$\left( \mathbf{I}_E(t^k, \boldsymbol{\mu}) \left[ \mathbf{a}^k \right] \right)_m := \tau_m^{EI} \left( \mathcal{L}_E(t^k, \boldsymbol{\mu}) \left[ u_{\text{red}}^k \right] \right) \quad (3.37)$$

for  $n, n' = 1, \dots, N$  and  $m = 1, \dots, M$ .

**Remark 3.2.3** (Empirical interpolation of multiple operators). *In the reduced basis scheme, we only use one collateral reduced basis space and one set of interpolation DOFs for both the operators  $\mathcal{L}_E$  and  $\mathcal{L}_I$ . This is a feasible choice, whenever the operators implement similar “dynamics”. Separate reduced basis spaces would include large redundancies in such scenarios.*

**Remark 3.2.4** (Rigorousness). *If rigorousness is important to the reduced basis application, i.e. if the error bound must not underestimate the “true” error, but the  $M+M'$ -exactness assumption cannot be satisfied, we could also use the rigorous bound from (2.3.10). In this case, the constant  $C_{\mathcal{W}_h} \varepsilon_{M,M'}^*$  needs to be added as an error contribution in each time step. This approach has been applied in [8].*

**Remark 3.2.5** (Output functionals). *After computing reduced basis vector  $\{\mathbf{a}^k(\boldsymbol{\mu})\}_{k=0}^K$ , the reduced solutions can be reconstructed by a linear combination of the reduced basis vectors as  $u_{\text{red}}^k(\boldsymbol{\mu}) := \sum_{n=1}^N a_n^k \varphi_n$ . The computation of the latter, however, is inefficient as it depends on the high-dimensional function space  $\mathcal{W}_h$ .*

*If possible, it is preferable to initially define for every  $\boldsymbol{\mu} \in \mathcal{M}$  linear output functionals  $l(\boldsymbol{\mu}) : \mathcal{W}'_h$  for the reduced basis application, such that*

$$s_h^k(\boldsymbol{\mu}) := l(\boldsymbol{\mu}) \left[ u_h^k(\boldsymbol{\mu}) \right] \quad (3.38)$$

*defines a low dimensional “output of interest”. In this case, if the output functional is linear and separable in the parameter, its space dependent parts can be reduced during the offline phase.*

**3.2.1. Complexity analysis.** We now show that the reduced scheme from Definition 3.2.2 allows a full offline/online decomposition by summarizing the computed data fields and their theoretical complexity and size. The ability to pre-compute high-dimensional data in a single offline phase, is the key for efficient and fast online simulations.

For the initial data projection, we assume a separable form for the analytical initial data function

$$u_0(\boldsymbol{\mu}) = \sum_{q=1}^Q \sigma_0^q(\boldsymbol{\mu}) u_0^q \quad (3.39)$$

with parameter dependent coefficient functions  $\sigma_0^q : \mathcal{M} \rightarrow \mathbb{R}$  and parameter independent functions  $u_0^q \in \mathcal{W}_h$  for  $q = 1, \dots, Q$ . In this case the parameter independent projections  $\mathcal{P}_{\text{red}}[u_0^q]$  can be pre-computed during the offline phase, and the actual parameter dependent projection is efficiently computable as

$$\mathcal{P}_{\text{red}}[u_0(\boldsymbol{\mu})] = \sum_{q=1}^Q \sigma_0^q(\boldsymbol{\mu}) \mathcal{P}_{\text{red}}[u_0^q] \quad (3.40)$$

with the already known reduced basis functions. It is noteworthy, that for non-separable initial data, which cannot be written like in (3.2.16), a classical empirical interpolation can be performed, resulting in an interpolant

$$\mathcal{I}_{M_0} [u_0(\boldsymbol{\mu})] = \sum_{m=1}^{M_0} u_0(x_0^m) q_0^m \quad (3.41)$$

with “magic points”  $x_0^m \in \Omega$  and collateral reduced basis functions  $q_0^m \in \mathcal{W}_h$  for  $m = 1, \dots, M_0$ . In this case, during the offline phase the projections  $\mathcal{P}_{\text{red}} [q_0^m]$  need to be pre-computed, such that the projection is computable efficiently as

$$\mathcal{P}_{\text{red}} [u_0(\boldsymbol{\mu})] \approx \sum_{m=1}^{M_0} u_0(x_0^m) \mathcal{P}_{\text{red}} [q_0^m] \quad (3.42)$$

with complexity  $\mathcal{O}(M)$ . Note, that the latter introduces an empirical interpolation error to which the approximation theory from Chapter 2 applies. In the rest of this thesis, including the numerical experiments, a separability of the initial data like in (3.2.16) is assumed.

For clarity of exposition, we further assume, that only one common set of collateral reduced basis functions and interpolation points for both empirically interpolated operators exists. This is certainly correct for schemes with purely implicit ( $\mathcal{L}_h \equiv \mathcal{L}_I$ ) or purely explicit operator ( $\mathcal{L}_h \equiv \mathcal{L}_E$ ) contributions. However, a single set of interpolation data can also be a suitable choice if the manifolds of operator evaluations do not differ very much. On the other hand, the reduced basis scheme is trivially extensible to a method with two different empirical operator interpolations. In Chapter 5 both cases are tested.

Efficient evaluations of the operator during the online phase, depend on

- restrictions of the reduced basis functions to  $\{\mathcal{R}_M [\varphi_n]\}_{n=1}^N$  with a restriction operator

$$\mathcal{R}_M : \mathcal{W}_h \rightarrow \mathcal{W}_h, \quad u_h = \sum_{i=1}^H \tau_i(u_h) \psi_i \mapsto \sum_{i \in I_M} \tau_i(u_h) \psi_i, \quad (3.43)$$

with a global index set  $\mathcal{I}_M := \cup_{\tau \in \Sigma_M} \mathcal{I}_\tau$ . For example, for finite volume operators, this local index sets  $\mathcal{I}_\tau$  includes all indices of Dofs corresponding to neighboring cells of the cell the Dof  $\tau \in \Sigma_h$  is associated to. (c.f. Figure 3.2.1)

- the Gramian matrix  $\mathbf{C}$  from (3.2.11), whose coefficient entries  $(\mathbf{C})_{nm} = \langle \xi_m, \varphi_n \rangle_{\mathcal{W}_h}$  depend on the *nodal* collateral reduced basis functions  $\{\xi_m\}_{m=1}^M$  that need to be generated from the functions  $\{q_m\}_{m=1}^M$  in a further pre-processing step. Note that this “basis transformation”

is very efficient because of the special form of the collateral reduced basis (c.f. Remark 2.1.2).

**Remark 3.2.6.** *In practice, discretization operators, like finite volume or finite element discretization operators are usually grid-based and the degrees of freedom correspond to distinctive points on the grid cells or its interfaces. In such case, a subgrid  $\mathcal{S}_h \subset \mathcal{T}_h \subset \Omega$  as illustrated in Figure 3.2.1, might be necessary in order to compute the local operator evaluations and the restriction operator efficiently without an otherwise needed full grid traversal. In Section 7.6 the implementation of such a grid is discussed.*

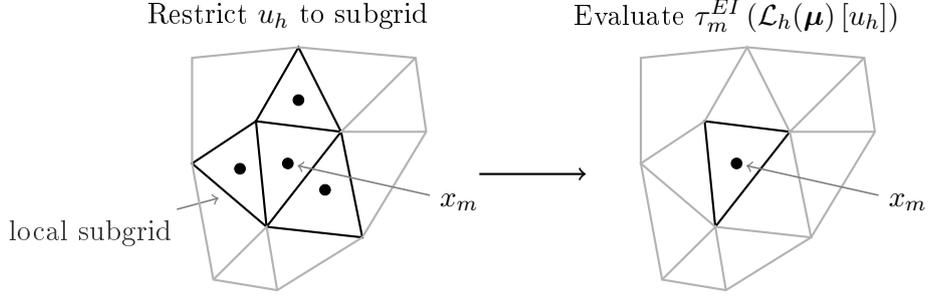


FIGURE 3.2.1. Illustration of finite volume operator evaluation on a local subgrid

Due to the  $H$ -independent Dof dependence the local operator evaluations of the restricted basis functions

$$\left( \tau_m^{EI} \circ \mathcal{L}_h(t^k, \boldsymbol{\mu}) \right) \left[ \sum_{n=1}^N a_n \varphi_n \right] = \left( \tau_m^{EI} \circ \mathcal{L}_h(t^k, \boldsymbol{\mu}) \right) \left[ \sum_{n=1}^N a_n \mathcal{R}_M[\varphi_n] \right] \quad (3.44)$$

have a complexity of  $\mathcal{O}(N|I_M|) = \mathcal{O}(NM)$  for all  $m = 1, \dots, M$ . This result can be applied to equations (3.2.13)-(3.3.12) and we observe, that each of them lies in the complexity class  $\mathcal{O}(NM^2)$ . The generation of the Jacobian from equation (3.2.12) depends on  $\mathcal{O}(N^2M^2)$  flops. This outreaches all other computations for the assembling of reduced matrices and vectors including matrix-matrix-multiplication of the reduced Jacobian with  $\mathbf{C}$  consuming  $\mathcal{O}(N^2M)$  flops. Therefore, one Newton step (3.2.7) of the reduced scheme has complexity  $\mathcal{O}(N^2M^2 + N^3)$  including the costs for the linear equation solver. The computation of the Newton residual costs  $\mathcal{O}(NM^2)$ . Unlike in the detailed simulation steps, the left hand side matrix in the linear equation system is not sparse. Because  $N$  is very small compared to the dimension of the detailed numerical scheme, we still expect the solution of the equation system to be much faster. We summarize that the reduced

	detailed simulation	reduced simulation
Initial data projection	Eqn. (3.1.1): $\mathcal{O}(H)$	Eqn. (3.2.6): $\mathcal{O}(QN)$
Assembling of operators and their derivatives in Newton step	Eqn. (3.1.3): $\mathcal{O}(H)$	Eqn. (3.2.7): $\mathcal{O}(N^2M^2)$
Solving Newton step	Eqn. (3.1.3): Complexity depending on linear solver, approximately $\mathcal{O}(H^2)$	Eqn. (3.2.7): $\mathcal{O}(N^3)$
Computing residual	Eqn. (3.1.5): $\mathcal{O}(H)$	Eqn. (3.2.8): $\mathcal{O}(NM^2)$

TABLE 3.2.1. Comparison of theoretical run-time complexities between detailed and reduced simulations.

scheme is independent of the high dimensional data size  $H$  for each parameter after the offline-phase. A detailed comparison between costs for detailed and reduced simulations is given in Table 3.2.1.

### 3.3. A posteriori error estimator

In order to efficiently find a reduced basis space  $\mathcal{W}_{\text{red}}$  that approximates all the requested parametric solutions with a given accuracy, efficient a posteriori bounds for the error between a reduced and a detailed simulation are necessary. Therefore, these are a crucial ingredient for the reduced basis method. For linear problems or polynomially non-linear problems, in which all data can be separated in space and parameter, there exist a variety of residual based and rigorous error bounds, e.g. [33, 73, 66, 58, 62]. There are also results for schemes which use empirical interpolation in order to generate a separation of parameter dependent functions, e.g. [31, 8, 39].

In this section, we propose a new error estimate for the reduced basis scheme given in Definition 3.2.2 which is based on the residual and takes into account the error contribution of the empirical operator interpolation. A simple estimator for purely explicit discretization with empirical operator interpolants has been originally introduced in [39]. Here, like in Definition 2.3.4, we assume that a higher order empirical operator interpolation of the used operators is exact. Note, that this assumption is always fulfilled for  $M + M' = H$  but for efficiency reasons in practice a much smaller value for  $M'$  is used. In our numerical experiments, we observed that already a small number of additional collateral reduced basis functions is sufficient to obtain feasible a posteriori error results. In [31], who suggested this error bound, even  $M' = 1$  is chosen for their numerical experiments.

**Theorem 3.3.1.** *Let  $\{u_h^k(\boldsymbol{\mu})\}_{k=0}^K$  and  $\{u_{\text{red}}^k(\boldsymbol{\mu})\}_{k=0}^K$  be solution trajectories obtained via the evolution schemes from Definitions 3.1.1 and 3.2.2 where the initial projection onto the reduced basis space is exact, i.e.  $u_h^0(\boldsymbol{\mu}) \in \mathcal{W}_{\text{red}}$  for all  $\boldsymbol{\mu} \in \mathcal{M}$ . Further, we make two assumptions on the discretization*

operators  $\text{Id} + \Delta t \mathcal{L}_I(\boldsymbol{\mu})$  and  $\text{Id} - \Delta t \mathcal{L}_E(\boldsymbol{\mu})$ . Firstly, the operators need to fulfill a lower respectively an upper Lipschitz continuity condition such that there exist constants  $C_{I,\Delta t}(\boldsymbol{\mu}), C_{E,\Delta t}(\boldsymbol{\mu}) > 0$ , and for all  $u, v \in \mathcal{W}_h$  and all  $\boldsymbol{\mu} \in \mathcal{M}$  the inequalities

$$\|u - v + \Delta t \mathcal{L}_I(\boldsymbol{\mu}) [u] - \Delta t \mathcal{L}_I(\boldsymbol{\mu}) [v]\|_{\mathcal{W}_h} \geq \frac{1}{C_{I,\Delta t}(\boldsymbol{\mu})} \|u - v\|_{\mathcal{W}_h} \quad (3.45)$$

$$\|u - v - \Delta t \mathcal{L}_E(\boldsymbol{\mu}) [u] + \Delta t \mathcal{L}_E(\boldsymbol{\mu}) [v]\|_{\mathcal{W}_h} \leq C_{E,\Delta t}(\boldsymbol{\mu}) \|u - v\|_{\mathcal{W}_h} \quad (3.46)$$

hold. Secondly, we assume the exactness of the empirical interpolation of the operators for a certain number of collateral reduced basis functions, i.e. there exists a positive integer  $M' > 0$ , such that

$$\mathcal{I}_{M+M'} [\mathcal{L}_I(\boldsymbol{\mu})] [u_{\text{red}}^k(\boldsymbol{\mu})] = \mathcal{L}_I(\boldsymbol{\mu}) [u_{\text{red}}^k(\boldsymbol{\mu})] \quad \text{and} \quad (3.47)$$

$$\mathcal{I}_{M+M'} [\mathcal{L}_E(\boldsymbol{\mu})] [u_{\text{red}}^k(\boldsymbol{\mu})] = \mathcal{L}_E(\boldsymbol{\mu}) [u_{\text{red}}^k(\boldsymbol{\mu})] \quad (3.48)$$

for all  $k = 0, \dots, K$  and  $\boldsymbol{\mu} \in \mathcal{M}$ .

Then, the norm of the error  $e^k(\boldsymbol{\mu}) := u_h^k(\boldsymbol{\mu}) - u_{\text{red}}^k(\boldsymbol{\mu})$  can be bounded for  $k = 0, \dots, K$  by  $\eta_{N,M,M'}^k(\boldsymbol{\mu})$  which is an efficiently computable function defined by

$$\begin{aligned} \|e^k(\boldsymbol{\mu})\|_{\mathcal{W}_h} &\leq \eta_{N,M,M'}^k(\boldsymbol{\mu}) := \\ &\sum_{i=0}^{k-1} \left[ (C_{I,\Delta t}(\boldsymbol{\mu}))^{k-i+1} (C_{E,\Delta t}(\boldsymbol{\mu}))^{k-i} \right. \\ &\quad \left. \cdot \left( \left\| \sum_{m=M+1}^{M+M'} \Delta t \theta_m^{i+1}(\boldsymbol{\mu}) \xi_m \right\| + \varepsilon^{\text{New}} + \|\Delta t R^{i+1}(\boldsymbol{\mu})\| \right) \right] \end{aligned} \quad (3.49)$$

with a residual for the error due to the projection on the reduced basis space

$$\begin{aligned} \Delta t R^{k+1}(\boldsymbol{\mu}) &:= (\text{Id} + \Delta t \mathcal{I}_M [\mathcal{L}_I]) [u_{\text{red}}^{k+1}(\boldsymbol{\mu})] \\ &\quad - (\text{Id} - \Delta t \mathcal{I}_M [\mathcal{L}_E]) [u_{\text{red}}^k(\boldsymbol{\mu})] \end{aligned} \quad (3.50)$$

and empirical interpolation coefficients  $\boldsymbol{\theta}^k(\boldsymbol{\mu}) := \{\theta_m^k(\boldsymbol{\mu})\}_{m=1}^{M+M'}$  defined by

$$\theta_m^k(\boldsymbol{\mu}) := \tau_m^{\text{EI}} \left( \mathcal{L}_I [u_{\text{red}}^k(\boldsymbol{\mu})] + \mathcal{L}_E [u_{\text{red}}^{k-1}(\boldsymbol{\mu})] \right). \quad (3.51)$$

PROOF. For clarity of the exposition, we will discard all parameters  $\boldsymbol{\mu}$  in this proof. First, we check that the residual norm  $\|\Delta t R^k\|_{\mathcal{W}_h}$  can be computed efficiently, because with Definitions (3.3.6), (3.2.10)-(3.3.12) and the empirical interpolation gram matrix  $\mathbf{X} \in \mathbb{R}^{M \times M}$  defined by

$$(\mathbf{X})_{mm'} := \langle \xi_m, \xi_{m'} \rangle, \quad (3.52)$$

it follows that

$$\begin{aligned}
\Delta t^2 \left\| R^{k+1} \right\|_{\mathcal{W}_h}^2 &= \left\langle \Delta t R^{k+1}, \Delta t R^{k+1} \right\rangle \\
&= \left( \mathbf{a}^{k+1} - \mathbf{a}^k \right)^T \mathbf{M} \left( \mathbf{a}^{k+1} - \mathbf{a}^k \right)^T \\
&\quad + 2\Delta t \left( \mathbf{l}_I \left[ \mathbf{a}^{k+1} \right] + \mathbf{l}_E \left[ \mathbf{a}^k \right] \right)^T \mathbf{C} \left( \mathbf{a}^{k+1} - \mathbf{a}^k \right) \\
&\quad + \Delta t^2 \left( \mathbf{l}_I \left[ \mathbf{a}^{k+1} \right] + \mathbf{l}_E \left[ \mathbf{a}^k \right] \right)^T \mathbf{X} \left( \mathbf{l}_I \left[ \mathbf{a}^{k+1} \right] + \mathbf{l}_E \left[ \mathbf{a}^k \right] \right).
\end{aligned}$$

So, the complexity of the residual computations summed over all time steps  $k = 1, \dots, K$  is  $\mathcal{O}(KN^2) + \mathcal{O}(KMN) + \mathcal{O}(KM^2)$  and thus independent of  $H$ . Also the empirical operator interpolation residuals which is in (3.3.5) given by  $\left\| \sum_{m=M+1}^{M+M'} \Delta t \theta_m^{i+1}(\boldsymbol{\mu}) \xi_m \right\|_{\mathcal{W}_h}$ , can be computed efficiently as

$$\left( \mathbf{l}_I^+ \left[ \mathbf{a}^{k+1} \right] + \mathbf{l}_E^+ \left[ \mathbf{a}^k \right] \right)^T \mathbf{X}^+ \left( \mathbf{l}_I^+ \left[ \mathbf{a}^{k+1} \right] + \mathbf{l}_E^+ \left[ \mathbf{a}^k \right] \right) \quad (3.53)$$

with matrices  $\mathbf{X}^+ \in \mathbb{R}^{M' \times M'}$  and vectors  $\mathbf{l}_I^+ \left[ \mathbf{a}^{k+1} \right], \mathbf{l}_E^+ \left[ \mathbf{a}^k \right] \in \mathbb{R}^{M'}$  defined by

$$(\mathbf{X}^+)_{mm'} := \langle \xi_m, \xi_{m'} \rangle \quad (3.54)$$

$$\left( \mathbf{l}_I^+ \left[ \mathbf{a}^{k+1} \right] \right)_m := \tau_m^{EI} \left( \mathcal{L}_I(t^k, \boldsymbol{\mu}) \left[ u_{\text{red}}^{k+1, \nu} \right] \right), \quad (3.55)$$

$$\left( \mathbf{l}_E^+ \left[ \mathbf{a}^k \right] \right)_m := \tau_m^{EI} \left( \mathcal{L}_E(t^k, \boldsymbol{\mu}) \left[ u_{\text{red}}^k \right] \right) \quad (3.56)$$

for  $m, m' = 1, \dots, M'$ . So, the computational complexity summed over all time step instances adds up to  $\mathcal{O}(KM'^2)$ .

Let us now derive the error bound. After each Newton iteration in the detailed numerical scheme, we obtain the equation

$$(\text{Id} + \Delta t \mathcal{L}_I) \left[ u_h^{k+1} \right] = (\text{Id} - \Delta t \mathcal{L}_E) \left[ u_h^k \right] + R_{h, \text{New}}^k \quad (3.57)$$

with Newton residual  $\left\| R_{h, \text{New}}^k \right\|_{\mathcal{W}_h} \leq \varepsilon^{\text{New}}$ .

The same can be obtained with (3.3.6) for solutions of the reduced numerical scheme

$$(\text{Id} + \Delta t \mathcal{I}_M [\mathcal{L}_I]) \left[ u_{\text{red}}^{k+1} \right] = (\text{Id} - \Delta t \mathcal{I}_M [\mathcal{L}_E]) \left[ u_{\text{red}}^k \right] + \Delta t R^{k+1}. \quad (3.58)$$

Subtracting (3.3.13) from (3.3.14) leads to

$$\begin{aligned}
&\underbrace{(\text{Id} + \Delta t \mathcal{L}_I) \left[ u_h^{k+1} \right] - (\text{Id} + \Delta t \mathcal{I}_M [\mathcal{L}_I]) \left[ u_{\text{red}}^{k+1} \right]}_{=:(I)} \\
&= \underbrace{(\text{Id} - \Delta t \mathcal{L}_E) \left[ u_h^k \right] - (\text{Id} - \Delta t \mathcal{I}_M [\mathcal{L}_E]) \left[ u_{\text{red}}^k \right]}_{=:(II)} \\
&\quad + R_{h, \text{New}}^{k+1} - \Delta t R^{k+1}.
\end{aligned} \quad (3.59)$$

After adding zeros to each of  $(I)$  and  $(II)$ , these can be decomposed into terms that can  $(Ia)$ ,  $(IIa)$  be estimated with the Lipschitz conditions and are  $(Ib)$ ,  $(IIb)$  efficiently computable terms, only depending on low dimensional data

$$(I) = \underbrace{(\text{Id} + \Delta t \mathcal{L}_I) \left[ u_h^{k+1} \right] - (\text{Id} + \Delta t \mathcal{L}_I) \left[ u_{\text{red}}^{k+1} \right]}_{=:(Ia)} + \underbrace{(\text{Id} + \Delta t \mathcal{L}_I) \left[ u_{\text{red}}^{k+1} \right] - (\text{Id} + \Delta t \mathcal{I}_M [\mathcal{L}_I]) \left[ u_{\text{red}}^{k+1} \right]}_{=:(Ib)}, \quad (3.60)$$

$$(II) = \underbrace{(\text{Id} - \Delta t \mathcal{L}_E) \left[ u_h^k \right] - (\text{Id} - \Delta t \mathcal{L}_E) \left[ u_{\text{red}}^k \right]}_{=:(IIa)} + \underbrace{(\text{Id} - \Delta t \mathcal{L}_E) \left[ u_{\text{red}}^k \right] - (\text{Id} - \Delta t \mathcal{I}_M [\mathcal{L}_E]) \left[ u_{\text{red}}^k \right]}_{=:(IIb)}. \quad (3.61)$$

Thereby, we have split the error propagation caused by the projection on the reduced basis space  $(Ia)$ ,  $(IIa)$  from the error contribution through the empirical interpolation of the explicit and implicit discretization operators  $(Ib)$ ,  $(IIb)$ . Substituting the previous equations into (3.3.15), bringing  $(Ib)$  on the right hand side and applying the Lipschitz condition (3.3.1) on it, we obtain a bound for the error  $\|e^{k+1}\|_{\mathcal{W}_h}$  by

$$\begin{aligned} \|e^{k+1}\|_{\mathcal{W}_h} &\leq C_{I,\Delta t} \left\| (\text{Id} + \Delta t \mathcal{L}_I) \left[ u_h^{k+1} \right] - (\text{Id} + \Delta t \mathcal{L}_I) \left[ u_{\text{red}}^{k+1} \right] \right\|_{\mathcal{W}_h} \\ &= C_{I,\Delta t} \left\| (\text{Id} + \Delta t \mathcal{I}_M [\mathcal{L}_I]) \left[ u_{\text{red}}^{k+1} \right] - (\text{Id} + \Delta t \mathcal{L}_I) \left[ u_{\text{red}}^{k+1} \right] \right. \\ &\quad + (\text{Id} - \Delta t \mathcal{L}_E) \left[ u_h^k \right] - (\text{Id} - \Delta t \mathcal{L}_E) \left[ u_{\text{red}}^k \right] \\ &\quad + (\text{Id} - \Delta t \mathcal{L}_E) \left[ u_{\text{red}}^k \right] - (\text{Id} - \Delta t \mathcal{I}_M [\mathcal{L}_E]) \left[ u_{\text{red}}^k \right] \\ &\quad \left. + R_{h,\text{New}}^{k+1} - \Delta t R^{k+1} \right\|_{\mathcal{W}_h} \\ &\leq C_{I,\Delta t} \left( \left\| \sum_{m=M+1}^{M+M'} \Delta t \theta_m^{k+1} \xi_m \right\|_{\mathcal{W}_h} \right. \\ &\quad \left. + C_{E,\Delta t} \|e^k\|_{\mathcal{W}_h} + \varepsilon^{\text{New}} + \left\| \Delta t R^{k+1} \right\|_{\mathcal{W}_h} \right), \end{aligned} \quad (3.62)$$

where the last inequality uses the Lipschitz continuity (3.3.2) of  $\mathcal{L}_E$ , the exactness assumptions (3.3.3) and (3.3.4) on  $(Ib)$  respectively  $(IIb)$ , the boundedness of the Newton residuals and the definition of the empirical

interpolation coefficients. Resolving the recursion in (3.3.18) with initial error  $\|e^0\|_{\mathcal{W}_h} = 0$  results in the proposed error bound.  $\square$

**Remark 3.3.2.** *The contribution of the Newton iteration error bound  $\varepsilon^{New}$  adds up with the number of time instances. This is not a problem as the bound can be chosen arbitrarily small. However, especially for problems with exponential error growth in time ( $C_{I,\Delta t}(\boldsymbol{\mu}) > 1$  for all  $\boldsymbol{\mu}$ ), it is reasonable to weigh the bound with the time steps size  $\Delta t$ .*

**Remark 3.3.3.** *If an output functional exists for the reduced basis method as proposed in Remark 3.2.5, we are only interested in the error for the output of interest*

$$\hat{\varepsilon}^k(\boldsymbol{\mu}) := \left| s_h^k(\boldsymbol{\mu}) - s_{\text{red}}^k(\boldsymbol{\mu}) \right|, \quad (3.63)$$

where the reduced output  $s_{\text{red}}^k(\boldsymbol{\mu})$  is given by

$$s_{\text{red}}^k(\boldsymbol{\mu}) := l(\boldsymbol{\mu}) \left[ u_{\text{red}}^k(\boldsymbol{\mu}) \right]. \quad (3.64)$$

The error estimate can be simply adapted to this case by

$$\hat{\eta}_{N,M,M'}^k(\boldsymbol{\mu}) := \|l(\boldsymbol{\mu})\|_{\mathcal{W}_h'} \eta_{N,M,M'}^k(\boldsymbol{\mu}) \geq \hat{\varepsilon}^k(\boldsymbol{\mu}). \quad (3.65)$$

In several examples, e.g. in [33, 8], it has been demonstrated that estimators for output functionals can even be further improved by solving a second linear problem in each time step.

It is obvious, that the error estimator respects an offline/online decomposition. A preliminary for the separation is the construction of a bigger collateral reduced basis  $\boldsymbol{\xi}_{M+M'}$ , but in the experimental section in Chapter 5, we observe that only few extra basis functions are needed for reasonable results. The evaluation of the estimator only includes low-dimensional terms or evaluations of the empirically interpolated operators. For a detailed discussion on the efficient evaluation of these quantities, we refer to Section 2.2.

The efficiency of the a posteriori error bound (3.3.5) depends on a tight selection of the Lipschitz constants  $C_{E,\Delta t}(\boldsymbol{\mu})$  and  $C_{I,\Delta t}(\boldsymbol{\mu})$ . A tight bound, especially for the implicit constant  $C_{I,\Delta t}(\boldsymbol{\mu})$  is of great importance, because the estimator grows exponentially, if the implicit lower Lipschitz constant is greater than one. Otherwise, it ceases over time, when individual snapshots can approximate the detailed simulation better because of the increasing smoothness of the solutions over time caused by diffusion. In our experiments in Chapter 5, we derived very rough constant  $C_{I,\Delta t}$  and  $C_{E,\Delta t}$ , such that

$$C_{I,\Delta t} \leq C_{I,\Delta t}(\boldsymbol{\mu}) \quad \text{and} \quad (3.66)$$

$$C_{E,\Delta t} \geq C_{E,\Delta t}(\boldsymbol{\mu}) \quad (3.67)$$

for all  $\boldsymbol{\mu} \in \mathcal{M}$ . The computations for these constants for the non-linear finite volume operators from Section 3.1.1 can be found in Appendix A.

**3.3.1. Computation of Lipschitz constants.** Now, we want to discuss the computation of operator constants, as they are needed by the a posteriori error bound given in (3.3.5). First, we show how to deal with linear problems, and how in our non-linear case, they can be derived from the operator's derivative. We conclude in Section 3.3.1.3 with a discussion on the efficient computation of these constants for all parameters in the online phase of the reduced basis method.

3.3.1.1. *Linear operators.* The computation of the Lipschitz constants is straightforward for linear operators. If we assume the operators  $\mathcal{L}_I(\boldsymbol{\mu})$  and  $\mathcal{L}_E(\boldsymbol{\mu})$  to be linear, and the function space to be a Hilbert space with a scalar product  $\langle \cdot, \cdot \rangle_{\mathcal{W}_h}$ , the Lipschitz constants  $C_{I,\Delta t}(\boldsymbol{\mu})$  and  $C_{E,\Delta t}(\boldsymbol{\mu})$  are given by the spectra of the operators. (e.g. [63]) With abbreviations  $\hat{\mathcal{L}}_{I,\Delta t}(\boldsymbol{\mu}) := \text{Id} + \Delta t \mathcal{L}_I(\boldsymbol{\mu})$  and  $\hat{\mathcal{L}}_{E,\Delta t}(\boldsymbol{\mu}) := \text{Id} - \Delta t \mathcal{L}_E(\boldsymbol{\mu})$ , these operator spectra are given by

$$\sigma\left(\hat{\mathcal{L}}_I(\boldsymbol{\mu})\right) = \left[ \sqrt{\sigma_-\left(\hat{\mathcal{L}}_{I,\Delta t}^*(\boldsymbol{\mu})\hat{\mathcal{L}}_{I,\Delta t}(\boldsymbol{\mu})\right)}, \sqrt{\sigma_+\left(\hat{\mathcal{L}}_{I,\Delta t}^*(\boldsymbol{\mu})\hat{\mathcal{L}}_{I,\Delta t}(\boldsymbol{\mu})\right)} \right] \quad \text{and} \quad (3.68)$$

$$\sigma\left(\hat{\mathcal{L}}_E(\boldsymbol{\mu})\right) = \left[ \sqrt{\sigma_-\left(\hat{\mathcal{L}}_{E,\Delta t}^*(\boldsymbol{\mu})\hat{\mathcal{L}}_{E,\Delta t}(\boldsymbol{\mu})\right)}, \sqrt{\sigma_+\left(\hat{\mathcal{L}}_{E,\Delta t}^*(\boldsymbol{\mu})\hat{\mathcal{L}}_{E,\Delta t}(\boldsymbol{\mu})\right)} \right], \quad (3.69)$$

where  $\sigma_+(\mathcal{L})$ ,  $\sigma_-(\mathcal{L})$  denotes the largest respectively the smallest eigenvalue of a linear operator  $\mathcal{L}$  and  $\mathcal{L}^*$  its adjoint w.r.t. to the scalarproduct  $\langle \cdot, \cdot \rangle_{\mathcal{W}_h}$ . Then, the Lipschitz constants are given by

$$C_{I,\Delta t}(\boldsymbol{\mu}) = \sigma_-\left(\hat{\mathcal{L}}_{I,\Delta t}^*(\boldsymbol{\mu})\hat{\mathcal{L}}_{I,\Delta t}(\boldsymbol{\mu})\right)^{-1/2} \quad \text{and} \quad (3.70)$$

$$C_{E,\Delta t}(\boldsymbol{\mu}) = \sigma_+\left(\hat{\mathcal{L}}_{E,\Delta t}^*(\boldsymbol{\mu})\hat{\mathcal{L}}_{E,\Delta t}(\boldsymbol{\mu})\right)^{1/2}. \quad (3.71)$$

Therefore, the constant  $C_{I,\Delta t}(\boldsymbol{\mu})$  is positive if the implicit operator  $\hat{\mathcal{L}}_{I,\Delta t}(\boldsymbol{\mu})$  is regular. Furthermore, we observe for symmetric and positive definite operators  $\mathcal{L}_I(\boldsymbol{\mu})$  and  $\mathcal{L}_E(\boldsymbol{\mu})$ , i.e. if the spectra of both operators are strictly positive, that the corresponding Lipschitz constants are less than 1. In this case, the residuals and empirical interpolation errors at earlier time steps do not influence the error estimator from equation (3.3.5) as much as later ones, such that a tight error bound even for discretizations over large time intervals can be computed. For example, for a linear diffusion operator, the pd-ness of the implicit operator  $\mathcal{L}_I(\boldsymbol{\mu})$  is given.

3.3.1.2. *Non-linear operators.* In the non-linear case, a feasible way to estimate lower respectively upper bounds for the Lipschitz constant for differentiable operators  $\mathcal{L}_h$  is via Taylor expansion, as there exist a  $\theta \in [0, 1]$  such that

$$\hat{\mathcal{L}}(\boldsymbol{\mu})[u_h] - \hat{\mathcal{L}}(\boldsymbol{\mu})[v_h] = \mathbf{D}\hat{\mathcal{L}}(\boldsymbol{\mu})|_{\theta u_h + (1-\theta)v_h}[u_h - v_h] \quad (3.72)$$

if the operator is locally differentiable in a region comprising  $u_h$  and  $v_h$ . Therefore,

$$C_{I,\Delta t}(\boldsymbol{\mu}) := \sup_{k=1,\dots,K} \left\| \left( \mathbf{D}\mathcal{L}_{I,\Delta t}(\boldsymbol{\mu})|_{u_h^k(\boldsymbol{\mu})} \right)^{-1} \right\|_{\mathcal{W}_h} \quad \text{and} \quad (3.73)$$

$$C_{E,\Delta t}(\boldsymbol{\mu}) := \sup_{k=0,\dots,K-1} \left\| \mathbf{D}\mathcal{L}_{E,\Delta t}(\boldsymbol{\mu})|_{u_h^k(\boldsymbol{\mu})} \right\|_{\mathcal{W}_h} \quad (3.74)$$

can be a good approximation of the required constants. Note, that we cannot expect that these constants fulfill the conditions (3.3.1) and (3.3.2), as the inequality is only true for the specific arguments  $\{u_h^k(\boldsymbol{\mu})\}_{k=0}^K$  instead of any  $u_h \in \mathcal{W}_h$ . These, however, are exactly the arguments, for which the Lipschitz inequalities are needed in our proof for (3.3.5). So, the error bound is still correct, and we can even expect to enhance the efficiency of the bound by this restriction.

3.3.1.3. *Efficient computation of Lipschitz constants.* As the computation of operator spectra is high-dimensional, it should be conducted in the offline phase only.

For parameter dependent operators which can be written in a form

$$\mathcal{L}_h(\boldsymbol{\mu}) = \sum_{q=1}^Q \sigma(\boldsymbol{\mu}) \mathcal{L}_h^q \quad (3.75)$$

the successive constraint (SCM) method, as proposed e.g. in [42, 66, 17, 50] can be applied to efficiently adapt operators constants. Here, during the offline phase for a selection of adaptively chosen parameters, the Lipschitz constant are computed exactly. During the online simulation, the constants for other parameters can be efficiently derived as a linear combination of the pre-computed ones by solving a linear problem. Unfortunately, the empirical operator interpolants cannot be written in a separate form as in (3.3.31), such that the method cannot be applied. However, in [71], where the SCM is used in order to efficiently provide coercivity constants for operators whose separable form is determined by an “multi-component empirical interpolation”, it is stated, that the SCM may provide infeasible (in this case negative) coercivity constants. Therefore, sparse grid interpolation between pre-computed operator constants is proposed as an alternative. This concept could, of

course, be transferred to the Lipschitz constants of our a posteriori error estimator.



## CHAPTER 4

### Basis generation

In this chapter, we introduce two different methods in order to construct a low-dimensional linear subspace  $\mathcal{W}$  of a compact manifold  $\mathcal{S} \subset \mathcal{W}_h$  of discrete functions. In detail, we are interested in the special manifold

$$\mathcal{S} = \mathcal{S}(\mathcal{M}) := \left\{ u_h^k(\boldsymbol{\mu}) \mid \boldsymbol{\mu} \in \mathcal{M}, k = 0, \dots, K \right\} \subset \mathcal{W}_h, \quad (4.1)$$

where  $u_h^k(\boldsymbol{\mu}) \in \mathcal{W}_h$  represent solutions from the general parametrized evolution scheme in Definition 3.1.1 for parameters  $\boldsymbol{\mu} \in \mathcal{M} \subset \mathbb{R}^p$  from a finite dimensional set of parameter vectors.

A subspace ought to minimize the angle between itself and the manifold  $\mathcal{S}$  defined by

$$\mathcal{A}(\mathcal{S}, \widehat{\mathcal{W}}) := \sup_{u \in \mathcal{S}} \inf_{\hat{u} \in \widehat{\mathcal{W}}} \|u - \hat{u}\|_{\mathcal{W}_h} \quad (4.2)$$

for any linear subspace  $\widehat{\mathcal{W}} \subset \mathcal{W}_h$ .

The optimal choice for a linear  $N$ -dimensional subspace of  $\mathcal{S}(\mathcal{M})$ , the so-called reduced basis space denoted by  $\mathcal{W}_{\text{red}} \subset \mathcal{W}_h$  in the following, has an angle equal to the Kolmogorov  $N$ -width of  $\mathcal{S}(\mathcal{M})$  defined by

$$d_N(\mathcal{S}, \mathcal{W}) := \inf_{\substack{\widehat{\mathcal{W}} \subset \mathcal{W}, \\ \dim(\widehat{\mathcal{W}}) = N}} \mathcal{A}(\mathcal{S}, \widehat{\mathcal{W}}). \quad (4.3)$$

In order to computationally generate the reduced linear subspace, we need to “discretize” the manifold  $\mathcal{S}$  by a finite subset. In the special case of parametrized solutions yielding  $\mathcal{S}(\mathcal{M})$ , this finite subset can be identified with a finite set of training parameters  $\mathcal{M}_{\text{train}} \subset \mathcal{M}$  defining training snapshots

$$\mathcal{S}_{\text{train}} := \mathcal{S}(\mathcal{M}_{\text{train}}). \quad (4.4)$$

Here, we introduce the two most popular methods to find a reduced basis space  $\mathcal{W}_{\text{red}}$  of dimension  $N \ll H$  which approximates the set  $\mathcal{S}_{\text{train}}$ : proper orthogonal decomposition (POD) and greedy algorithms. Both methods can produce reduced basis spaces in a way such that its angle to the training manifold  $\mathcal{A}(\mathcal{S}_{\text{train}}, \mathcal{W}_{\text{red}})$  is close to the Kolmogorov  $N$ -width  $d_N(\mathcal{S}_{\text{train}}, \mathcal{W}_{\text{red}})$ . For a discussion of other methods for the computational generation of reduced basis spaces we refer to [55].

In Section 4.1 the POD method is shortly summarized, and important results on convergence and optimality of the results are cited from the literature. Our focus, however, lies on the greedy approach, as this is computationally less expensive. This is true, especially for big training parameter sets  $\mathcal{M}_{\text{train}}$  which are needed for our framework. In section 4.2 the general idea of the greedy algorithm is described with an overview of the most important general results on this topic. As an example, we revise Algorithm 2.1.1 from chapter 2 and extend it in a way such that it fits in the general framework for time-dependent solution snapshots.

In Section 4.2.2, we discuss the POD-GREEDY algorithm, initially proposed in [38] for the generation of a reduced basis space and empirical interpolation of an evolution problem from Definition 3.1.1. As for the reduced basis scheme proposed in the previous chapter, both a reduced basis space and empirical interpolation data have to be generated, we discuss in Section 4.3 the combined generation of these data sequentially and in a synchronized way, respectively.

The selection of the training set  $\mathcal{M}_{\text{train}}$  can have a large impact on the computational costs and the accuracy of the outcome of the greedy algorithm. Therefore, in our experiments, we use an improvement of the greedy algorithm introduced in [36, 35], which adaptively improves the selection of training parameters. Furthermore, it is difficult to predict both the dimension and the accuracy of the generated output. As this influences the computational gain during the online phase, we conclude the chapter with a discussion on different methods to control the basis space dimensions. Here, we focus on methods, which compute several reduced bases specialized for sub-domains of the parameter space [29, 35] or the time domain [24, 21].

#### 4.1. Proper orthogonal decomposition

The proper orthogonal decomposition (POD) method, also known as Karhunen–Loève expansion or principal component analysis in the literature, is a means to find an optimal approximation space for a set of data functions. The method has been successfully applied on parametrized problems, e.g. [45] and non-parametric problems, e.g. [16, 12, 49], where the last three publications used empirical interpolation in order to deal with nonlinearities.

Given a Hilbert space  $\mathcal{W}_h$ , a set of functions  $\mathcal{S} := \{u^k\}_{k=1}^K \subset \mathcal{W}_h$ ,  $N \leq \dim(\text{span } \mathcal{S})$  basis functions  $\{\phi_i\}_{i=1}^N \subset \mathcal{W}_h$  can be obtained, that solve

---

**Algorithm 4.1.1** POD basis generation for  $N$  POD modes.

---

POD $_N(\mathcal{S} := \{u^k\}_{k=1}^K)$

– Compute a Gramian matrix  $\mathcal{G} \in \mathbb{R}^{K \times K}$  of the data functions:

$$(\mathcal{G})_{ij} \leftarrow \langle u_i, u_j \rangle_{\mathcal{X}}$$

– Apply an SVD or a similar algorithm to compute a decreasing sequence of eigenvalues  $\lambda_1 \geq \dots \geq \lambda_N$  and corresponding eigenvectors  $v^1, \dots, v^N$ , w.r.t. the Euclidian scalar product  $(\cdot, \cdot)_{\mathbb{R}^2}$ .

– Project the eigenvectors (z-scores) on the reduced basis:

$$\phi_i \leftarrow \frac{1}{\sqrt{\lambda_i}} \sum_{k=1}^K v_k^i u^k$$

**return** reduced basis:  $\{\phi_i\}_{i=1}^N$

---

the optimization problem

$$\min_{\{\phi_i\}_{i=1}^N} \sum_{i=1}^k \left\| u^k - \sum_{i=1}^N \langle u^k, \phi_i \rangle_{\mathcal{X}} \phi_i \right\|_{\mathcal{W}_h}^2, \quad (4.5)$$

$$\text{with } \langle \phi_i, \phi_j \rangle_{\mathcal{W}_h} = \delta_{ij}, \quad \text{for } 1 \leq i, j \leq N.$$

In the following, the basis functions  $\{\phi_i\}_{i=1}^N$  are denoted as *POD modes*. If constructed by Algorithm 4.1.1 detailed below, the POD modes carry an order of significance, as the first mode equals the direction of the greatest variance of all the data functions from  $\mathcal{S}$ . For further details on the method, we refer to the monograph [43].

Algorithm 4.1.1 defines the method POD $_N$ , that we use to compute the  $N$  most significant POD modes of a given set of  $K$  data functions. If applied to the data functions  $\mathcal{S}_{\text{train}}$  with the Hilbert space  $\mathcal{X} = \mathcal{W}$ , we retrieve a reduced basis space  $\mathcal{W}_{\text{red}}$ . This reduced basis space can be used in reduced simulations as defined in Definition 3.2.2. Its generation, however, is very expensive, if the number of training parameters  $|\mathcal{M}_{\text{train}}|$  is large: For every training parameter high dimensional solution snapshots need to be computed, and the construction of a large Gramian matrix  $\mathcal{G}$  and the computation of this matrix's eigenvalues, can get very inefficient as well. Therefor, we introduce the alternative method of “greedy” basis generation dealing with these drawbacks.

## 4.2. Greedy algorithm

Greedy algorithms are a strategy to build a reduced basis space for the manifold  $\mathcal{S}(\mathcal{M})$  with higher computational efficiency in comparison to the aforementioned POD method.

**Algorithm 4.2.1** Greedy basis generation

---

X-GREEDY( $\mathcal{M}_{\text{train}}, \mathbb{T}, \varepsilon_{\text{tol}}, \Upsilon_{\text{max}}$ )

– Initialize reduced basis of size  $\Upsilon_0$ :

$$\mathcal{D}_{\Upsilon_0} \leftarrow \text{X-INITBASIS}()$$

$$\Upsilon \leftarrow \Upsilon_0$$

**repeat**

– Find parameter and time instance of worst approximated snapshot:

$$(\boldsymbol{\mu}_{\text{max}}, t_{\text{max}}) \leftarrow \arg \max_{(\boldsymbol{\mu}, t) \in \mathcal{M}_{\text{train}} \times \mathbb{T}} \text{X-ERRORESTIMATE}(\mathcal{D}_{\Upsilon}, \boldsymbol{\mu}, t)$$

– Extend reduced basis by  $v$  new snapshots:

$$\mathcal{D}_{\Upsilon+v} \leftarrow \text{X-EXTENDBASIS}(\mathcal{D}_{\Upsilon}, \boldsymbol{\mu}_{\text{max}}, t_{\text{max}})$$

$$\Upsilon \leftarrow \Upsilon + v$$

– Compute maximum error on training set:

$$\varepsilon \leftarrow \max_{(\boldsymbol{\mu}, t) \in \mathcal{M}_{\text{train}} \times \mathbb{T}} \text{X-ERRORESTIMATE}(\mathcal{D}_{\Upsilon}, \boldsymbol{\mu}, t)$$

**until**  $\varepsilon \leq \varepsilon_{\text{tol}}$  or  $\Upsilon > \Upsilon_{\text{max}}$

**return** reduced basis  $\mathcal{D}_{\Upsilon}$  and maximum error  $\varepsilon$

---

The general idea is sketched in Algorithm 4.2.1. The set of reduced basis functions  $\mathcal{D}_{\Upsilon}$  of size  $\Upsilon$  is iteratively enhanced by new basis functions derived from the training manifold  $\mathcal{S}(\mathcal{M}_{\text{train}})$ . The selection of these new basis functions is controlled by an error estimate which is minimized over a finite set of parameters  $\mathcal{M}_{\text{train}} \subset \mathcal{M}$  and time instances  $\mathbb{T} := \{t^0, \dots, t^K\}$ . We will use this algorithm several times throughout this chapter by specifying the methods

- X-INITBASIS(), initializing the reduced basis,
- X-ERRORESTIMATE(), estimating the error between high dimensional and *reduced snapshots* and
- X-EXTENDBASIS(), adding solution snapshot to the reduced basis space. The number of added solution snapshots  $v$  can be greater than 1.

The term *reduced snapshots* denotes either solution snapshots of a reduced basis numerical scheme as in Definition 3.2.2 or the projection of the high dimensional snapshot on the reduced basis space. In many cases the exact error between the high dimensional and its reduced snapshot is exactly computable, but of course with a computational complexity on the dimension  $H$ . As the error estimate has to be evaluated for all parameters from the possibly extensive training set, it is therefore preferable to use efficiently computable error estimates like the one proposed in Theorem 3.3.1 which can be decomposed in offline and online computations.

It is noteworthy that the greedy algorithm as defined in this chapter differs from those published under the same notion whose objective is to

approximate a single function by a linear combination of dictionary functions. For the latter case a thorough theory for convergence and variation results can be found for example in [68].

Recently, convergence results for Algorithm 4.2.1 have been published in [7] and [4]. These publications assume a simple specialization of Algorithm 4.2.1 that we want to refer to as BASIC-GREEDY which is detailed in the following Section 4.2.1. It iteratively generates a set of reduced basis functions  $\Phi_N := \{\varphi_i\}_{i=1}^N$ , and in [4] and it is proven that, for a considered polynomial or exponential decrease of the Kolmogorov  $N$ -width  $d_N(\mathcal{S}(\mathcal{M}))$ , the angles  $\mathcal{A}(\mathcal{S}(\mathcal{M}), \text{span } \Phi_N)$  have the same rate of decay.

**4.2.1. BASIC-greedy algorithm.** The aforementioned abstract greedy algorithm can be specified by Algorithm 4.2.1 and the following methods: First, an initial function is computed by the method BASIC-INITBASIS(), computing the function

$$\varphi_1 := \arg \max_{u_h \in \mathcal{S}(\mathcal{M}_{\text{train}})} \|u_h\|_{\mathcal{W}_h} \quad (4.6)$$

which minimizes the norm over all candidates from the training set. The extension method BASIC-EXTENDBASIS( $\{\varphi_i\}_{i=1}^N, \boldsymbol{\mu}, t^k$ ) simply returns the new basis function  $\varphi_{N+1} := u^k(\boldsymbol{\mu})$ . For the error estimation the method BASIC-ERRORESTIMATE( $\{\varphi_i\}_{i=1}^N, \boldsymbol{\mu}, t^k$ ) computes and returns an estimate  $\eta^k(\boldsymbol{\mu}) \approx \epsilon^k(\boldsymbol{\mu})$  similar to the exact error

$$\epsilon^k(\boldsymbol{\mu}) := \inf_{v_h \in \text{span } \{\varphi_i\}_{i=1}^N} \|u^k(\boldsymbol{\mu}) - v_h\|_{\mathcal{W}_h}. \quad (4.7)$$

This estimate must produce an optimal selection for the next basis function  $\varphi_{N+1}$  returned by the BASIC-EXTENDBASIS method in a way such that

$$\inf_{v_h \in \text{span } \{\varphi_i\}_{i=1}^N} \|\varphi_{N+1} - v_h\| \geq \gamma \max_{(\boldsymbol{\mu}, t^k) \in \mathcal{M}_{\text{train}} \times \mathbb{T}} \epsilon^k(\boldsymbol{\mu}) \quad (4.8)$$

for a weakness constant  $0 < \gamma \leq 1$ . This requirement (4.2.3) is fulfilled for effective estimates  $\eta^k(\boldsymbol{\mu})$ , for which there exist bounds  $c, C$  with  $0 < c \leq C$  such that

$$c\eta^k(\boldsymbol{\mu}) \leq \epsilon^k(\boldsymbol{\mu}) \leq C\eta^k(\boldsymbol{\mu}), \quad k = 0, \dots, K, \boldsymbol{\mu} \in \mathcal{M}. \quad (4.9)$$

Then, the weakness factor in (4.2.3) is given by  $\gamma = \frac{c}{C}$ . Note, that for  $c = C = 1$  the estimate is equal to the exact error computation.

**Theorem 4.2.1** (Convergence for exponential decay). *Assuming the above described BASIC-GREEDY algorithm with a weakness constant  $0 < \gamma \leq 1$  and*

$$\|\varphi_1\|_{\mathcal{W}_h} \leq M \quad (4.10)$$

for some  $M > 0$  and an exponential convergence of the Kolmogorov  $n$ -width by an exponential rate at

$$d_n(\mathcal{S}(\mathcal{M}_{train}), \mathcal{W}_h) \leq M e^{-an^\alpha} \quad (4.11)$$

for  $n > 0$  and some fixed  $a, \alpha > 0$ , then

$$\mathcal{A}(\mathcal{S}(\mathcal{M}_{train}), \text{span } \Phi_n) \leq C M e^{-cn^\beta} \quad (4.12)$$

for all  $n > 0$ , with  $\beta = \frac{\alpha}{1+\alpha}$  and for an arbitrary  $0 < \theta < 1$ , given constants  $c := \min \{|\log \theta|, (4q)^{-\alpha} a\}$ ,  $C := \max \left\{ e^{cN_0^\beta}, q^{\frac{1}{2}} \right\}$ ,  $q := \lceil 2(\gamma\theta)^{-1} \rceil$  and  $N_0 := \lceil (8q)^{\alpha+1} \rceil$ .

PROOF. See [4]. □

**4.2.2. POD-greedy.** The state-of-the-art ansatz for the generation of reduced bases for evolution problems with greedy algorithms differs slightly from the previously described BASIC-GREEDY: As the changes in solution snapshots from one time step to the next are small in many applications, the POD-GREEDY algorithm has been introduced in [38] which combines the strengths of the POD and the greedy basis generation methods. Here, in each extension step, a POD is performed on the residual between the entire trajectory  $\{u_h^k(\boldsymbol{\mu}_{\max})\}_{k=0}^K$  and its projection on the reduced basis space. The extension method then returns the  $l$  most significant modes, where  $l$  can be adapted in each extension, but in our experiments we only used the most significant mode and fixed  $l = 1$ .

Like in the BASIC-GREEDY algorithm, we assume that the quality of reduced simulation trajectories  $\{u_{\text{red}}^k(\boldsymbol{\mu})\}_{k=0}^K$  which are obtained by the reduced numerical scheme introduced in Definition 3.2.2, can be assessed by a posteriori error bounds. These error estimates, denoted by  $\eta_{N,M,M'}^k : \mathcal{M} \rightarrow \mathbb{R}$ , bound the reduction error  $\|u_{\text{red}}^k(\boldsymbol{\mu}) - u_h^k(\boldsymbol{\mu})\|_{\mathcal{W}_h}$  for every  $k = 0, \dots, K$ , different dimensions  $N$  and  $M$  of the reduced basis respectively the collateral reduced basis space, and an accuracy constant  $M' \geq 1$ . Note, that the proposed estimation method POD-ERRORESTIMATE returns a cumulative error independent of the time step argument  $t^k$ , because we are only interested in finding bad approximated trajectories. This fact is import for the error analysis of the algorithm discussed below. An example of such an error estimate is given in Theorem 3.3.1. In our experiments, the error estimate was known to grow monotonically with increasing time steps, which is why in our implementations, we returned the error estimate for the last time step  $t^K$  only. It is important to note, that the reduced simulation and the error estimate as a consequence of this, depend on the existence of empirical interpolation data. The combined generation of reduced basis spaces

**Algorithm 4.2.2** Methods for the POD–GREEDY algorithm

---

 POD–INITBASIS()

**return** initial reduced basis functions:  $\{\varphi_n\}_{n=1}^{N_0}$ 


---

 POD–ERRORESTIMATE( $\{\varphi_n\}_{n=1}^N, \boldsymbol{\mu}, t^k$ )

**return** cumulative error estimate for a fixed  $M' \geq 1$ :

$$\eta_{N,M,M'}(\boldsymbol{\mu}) := \sqrt{\sum_{k=0}^K \Delta t \left( \eta_{N,M,M'}^k(\boldsymbol{\mu}) \right)^2} \geq \sqrt{\sum_{k=0}^K \|u_{\text{red}}^k(\boldsymbol{\mu}) - u_h^k(\boldsymbol{\mu})\|_{\mathcal{W}_h}^2}$$


---

 POD–EXTENDBASIS( $\{\varphi_n\}_{n=1}^N, \boldsymbol{\mu}_{\text{max}}, t$ )

– Compute trajectory

$$\left\{ u_h^k(\boldsymbol{\mu}_{\text{max}}) \right\}_{k=0}^K$$

with scheme from Definition 3.2.2

 – Compute new basis function with Galerkin projection  $\mathcal{P}_{\text{red}}$  projecting onto span  $\{\varphi_n\}_{n=1}^N$ 

$$\varphi_{N+l} \leftarrow \text{POD}_l \left( \left\{ u_h^k(\boldsymbol{\mu}_{\text{max}}) - \mathcal{P}_{\text{red}} \left[ u_h^k(\boldsymbol{\mu}_{\text{max}}) \right] \right\}_{k=0}^K \right)$$

**return** extended reduced basis:  $\{\varphi_n\}_{n=1}^{N+l}$ 


---

and empirical interpolation data is discussed in section 4.3. Of course, it is also possible to use the exact error directly. However, a direct evaluation of the error depends on a large number of inefficient computations. Therefore, more efficient a posteriori estimators allowing to deal with a large number of training samples, are preferable.

Details on the POD–greedy algorithm are given in Algorithm 4.2.2. Note, that it is not specified how the initial basis shall be composed. In order to fulfill the requirements of Theorem 3.3.1, the reduced basis given by the span of the initial reduced basis functions  $\Phi_{N_0} = \{\varphi_n\}_{n=1}^{N_0}$  should include projections of the initial values  $\mathcal{P}_h[u_0(\boldsymbol{\mu})]$  for all  $\boldsymbol{\mu} \in \mathcal{M}$ . If the initial condition has a parametric separable form like in (3.2.16), the initial basis functions should be selected as an orthonormalization of the projections  $\{\mathcal{P}_h[u_0^q]\}_{q=1}^{Q_0}$ . Otherwise, an empirical interpolation as in (3.2.18) can be performed and the optimal choice for the initial reduced basis is  $\{q_0^m\}_{m=1}^{M_0}$ . Then, for rigor in the error estimation a term  $(C_{I,\Delta t} C_{E,\Delta t})^{k+1} \varepsilon^0$  should be added, where  $\varepsilon^0$  bounds the maximum interpolation error

$$\max_{\boldsymbol{\mu} \in \mathcal{M}} \left\| u_0(\boldsymbol{\mu}) - \sum_{m=1}^M u_0(x_0^m) q_0^m \right\|_{\mathcal{W}_h}. \quad (4.13)$$

See section 2.3 for a discussion on how to compute this bound.

4.2.2.1. *Convergence analysis.* Recently, in [34], the convergence results for the BASIC–GREEDY algorithm have been extended to the POD–GREEDY

algorithm as well. As in the case of evolution problems, the accuracy of projections of entire trajectories needs to be optimized, a different measure for the angle between the reduced basis space  $\mathcal{W}_{\text{red}} := \text{span } \Phi_N \subset \mathcal{W}_h$  and the manifold  $\mathcal{S}(\mathcal{M}_{\text{train}}) \subset \mathcal{W}_h$  of solution snapshots is defined by

$$\begin{aligned} & \mathcal{A}^{\text{POD}}(\mathcal{S}(\mathcal{M}_{\text{train}}), \mathcal{W}_{\text{red}}) \\ & := \sup_{\{u_h^k(\boldsymbol{\mu})\}_{k=0}^K \subset \mathcal{S}(\mathcal{M}_{\text{train}})} \sqrt{\sum_{k=0}^K \Delta t \|u_h^k(\boldsymbol{\mu}) - \mathcal{P}_{\text{red}}[u_h^k(\boldsymbol{\mu})]\|^2}. \end{aligned} \quad (4.14)$$

Here,  $\mathcal{P}_{\text{red}} : \mathcal{W}_h \rightarrow \mathcal{W}_{\text{red}}$  denotes the projection operator onto the reduced basis space. Like for the greedy algorithm without POD compression, the error estimate must comply with an accuracy constraint: The trajectory  $\{u_h(\boldsymbol{\mu}^*)\}_{k=1}^K$  with parameter

$$\boldsymbol{\mu}^* := \arg \max_{\boldsymbol{\mu} \in \mathcal{M}_{\text{train}}} \eta_{N,M,M'}(\boldsymbol{\mu}) \quad (4.15)$$

selected by the error estimate as the worst approximated one, may deviate from the optimal choice only by a factor  $0 < \gamma \leq 1$ , such that

$$\sqrt{\sum_{k=0}^K \Delta t \|u_h^k(\boldsymbol{\mu}^*) - \mathcal{P}_{\text{red}}[u_h^k(\boldsymbol{\mu}^*)]\|^2} \leq \gamma \mathcal{A}^{\text{POD}}(\mathcal{S}(\mathcal{M}_{\text{train}}), \mathcal{W}_{\text{red}}). \quad (4.16)$$

Then, it is shown, that the result from Theorem 4.2.1 can be extended to the POD–GREEDY algorithm, by proving that exponential convergence of the Kolmogorov  $n$ -width  $d_n(\mathcal{S}(\mathcal{M}_{\text{train}}), \mathcal{W}_h)$  is inherited by the new angle  $\mathcal{A}^{\text{POD}}(\mathcal{S}(\mathcal{M}_{\text{train}}), \text{span } \Phi_n)$ .

**Theorem 4.2.2** (Exponential convergence of POD–GREEDY). *Assuming the above described POD–GREEDY algorithm with a weakness constant  $0 < \gamma \leq 1$  for the error estimate  $\eta_{N,M,M'}$  and*

$$d_0(\mathcal{S}(\mathcal{M}_{\text{train}}), \mathcal{W}_h) := \max_{u_h \in \mathcal{S}(\mathcal{M}_{\text{train}})} \|u_h\|_{\mathcal{W}_h} \leq M \quad (4.17)$$

for some  $M > 0$  and an exponential convergence of the Kolmogorov  $n$ -width by an exponential rate at

$$d_n(\mathcal{S}(\mathcal{M}_{\text{train}}), \mathcal{W}_h) \leq M e^{-an^\alpha} \quad (4.18)$$

for  $n > 0$  and some fixed  $a, \alpha > 0$ , then

$$\mathcal{A}^{\text{POD}}(\mathcal{S}(\mathcal{M}_{\text{train}}), \text{span } \Phi_{N_0+n}) \leq C M e^{-cn^\beta} \quad (4.19)$$

for all  $n > 0$ , with  $\beta = \frac{\alpha}{1+\alpha}$  and for an arbitrary  $0 < \theta < 1$ , given constants  $c := \min\{|\log \theta|, (4q)^{-\alpha} a\}$ ,  $C := \max\{e^{cN_0^\beta} \sqrt{T}, \sqrt{qT}\}$ ,  $q := \lceil 2(\gamma\theta)^{-1} \sqrt{K+1} \rceil^2$  and  $N_0 := \lceil (8q)^{\alpha+1} \rceil$ .

PROOF. See [34] with  $w_k := \Delta t$  for  $k = 0, \dots, K$ . As a further modification to the original result, we ignored the first  $N_0$  basis functions in the convergence, because these need to be selected before all requirements of the error estimate from Theorem 3.3.1 are fulfilled.  $\square$

**4.2.3. EI-greedy.** The X-GREEDY Algorithm 4.2.1 can also be specialized for the generation of the empirical interpolation data, i.e. collateral reduced basis functions  $\mathbf{Q}_M = \{\mathbf{q}_m\}_{m=1}^M$  and interpolation Dofs  $\Sigma_M = \{\tau_m^{EI}\}_{m=1}^M$ .

---

**Algorithm 4.2.3** Methods for collateral reduced basis generation EI-GREEDY

---

EI-INITBASIS()

**return** *empty initial basis*:  $\mathcal{D}_0 \leftarrow \{\}$

---

EI-ERRORESTIMATE( $(\mathbf{Q}_M, \Sigma_M), \boldsymbol{\mu}, t^k$ )

– *Compute exact operator evaluation*

$$v_h \leftarrow \mathcal{L}_h(t^k, \boldsymbol{\mu})[u_h^k(\boldsymbol{\mu})]$$

– *Compute interpolation coefficients*

$$\boldsymbol{\sigma}^M(v_h) := (\sigma_j^M(v_h))_{j=1}^M \in \mathbb{R}^M \quad (4.20)$$

– *by solving the linear equation system*

$$\sum_{j=1}^M \sigma_j^M(v_h) \tau_i^{EI}[q_j] = \tau_i^{EI}[v_h], \quad i = 1, \dots, M \quad (4.21)$$

**return** *approximation error*:  $\|v_h - \sum_{j=1}^M \sigma_j^M(v_h) q_j\|_{\mathcal{W}_h}$

EI-EXTENDBASIS( $(\mathbf{Q}_M, \Sigma_M), \boldsymbol{\mu}, t^k$ )

– *Compute exact operator evaluation*

$$v_h \leftarrow \mathcal{L}_h(t^k, \boldsymbol{\mu})[u_h^k(\boldsymbol{\mu})]$$

– *and interpolation coefficients  $\boldsymbol{\sigma}^M(v_h)$  as in (4.2.15) and (4.2.16).*

– *Compute the residual between  $v_h$  and its current interpolant.*

$$r_M \leftarrow v_h - \sum_{j=1}^M \sigma_j^M(v_h) q_j$$

– *Find interpolation Dof maximizing the residual.*

$$\tau_{M+1}^{EI} \leftarrow \arg \sup_{\tau \in \Sigma_h} |\tau(r_M)|$$

– *Normalize to obtain a new collateral reduced basis function.*

$$q_{M+1} \leftarrow (\tau_{M+1}^{EI}(r_M))^{-1} \cdot r_M$$

**return** *extended basis data*:  $\mathcal{D}_{M+1} \leftarrow \left( \{q_m\}_{m=1}^{M+1}, \{\tau_m^{EI}\}_{m=1}^{M+1} \right)$

---

The pertinent methods are given in Algorithm 4.2.3 and the resulting Greedy algorithm, denoted by EI-GREEDY in the following is almost equivalent to Algorithm 2.1.1 from chapter 2. The only difference to the algorithm

defined earlier is the use of the time parameter, as now evaluations of time dependent operators on solution snapshots derived from an evolution ?? scheme are considered.

### 4.3. Combined basis generation

As mentioned in the previous section, the error estimate of the POD-GREEDY algorithm depends on empirical interpolation data. Naturally, one would therefore first execute Algorithms 4.2.1+2.1.1 for all discrete operators in the scheme. Afterwards, the required collateral reduced basis and empirical interpolation Dofs can be used to build a reduced basis space with the POD-GREEDY algorithm. This approach of subsequently executing two greedy algorithms, however, has some drawbacks:

- (i) There is no efficiently computable error estimate for the empirical operator interpolation, i.e. the error estimation method `EI-ERRORESTIMATE()` depends on high dimensional computations for each parameter and time step tested. This can be very inefficient for large parameter sets  $\mathcal{M}_{\text{train}}$ , and eats up the computational gain by the efficient error estimate in the POD-GREEDY, later.
- (ii) The empirical interpolation bases are generated such that an artificial interpolation error is reduced for which it is not clear, how it relates to the error estimates  $\eta_{N,M}^k$  used in the POD-GREEDY algorithm. Therefore, it is impossible to determine a priori the optimal correlation between the reduced basis space and the collateral reduced basis space.
- (iii) In our experiments, the reduced basis generation can be improved if the training parameter set  $\mathcal{M}_{\text{train}}$  is adapted during the basis generation. This allows to begin with a small parameter set and to reduce the computation time for the basis generation. The POD-GREEDY algorithm finds a good training parameters set, but this set is unknown at the stage when the collateral reduced basis is generated. Details on this adaptive parameter sampling are given in Section 4.4.1.

**4.3.1. PODEI-greedy basis generation.** In this section we will introduce another greedy algorithm for a synchronized execution of POD greedy and empirical interpolation basis generation (PODEI). This algorithm is again based on Algorithm 4.2.1 but generates the reduced basis and the collateral reduced basis spaces in parallel and overcomes the drawbacks of the subsequent execution of the algorithms EI-GREEDY and POD-GREEDY. The methods for the PODEI-GREEDY algorithm are sketched in Algorithm

**Algorithm 4.3.1** Methods for the PODEI-GREEDY algorithm

---

PODEI-INITBASIS()

– Generate small empirical interpolation basis with  $M_{\text{small}} \geq M'$ :

$$(\mathbf{Q}_{M_{\text{small}}}, \Sigma_{M_{\text{small}}}) \leftarrow \text{EI-GREEDY}(\mathcal{M}_{\text{train}}^{\text{coarse}}, 0, M_{\text{small}})$$

– Compute initial reduced basis:

$$\{\varphi_n\}_{n=0}^{N_0} \leftarrow \text{POD-INITBASIS}()$$

**return** initial bases data:  $\mathcal{D}_1 \leftarrow \{\varphi_n\}_{n=1}^{N_0} \cup (\mathbf{Q}_{M_{\text{small}}}, \Sigma_{M_{\text{small}}})$ 


---

PODEI-ERRORESTIMATE( $\mathcal{D}_\Upsilon, \boldsymbol{\mu}, t^k$ )

**return** reduced basis error estimate:  $\eta_{N,M,M'}^k(\boldsymbol{\mu})$ 


---

PODEI-EXTENDBASIS( $\mathcal{D}_\Upsilon, \boldsymbol{\mu}_{\text{max}}, t^k$ )

Reduced data  $\mathcal{D}_\Upsilon$  comprises  $\mathcal{D}_N^{\text{RB}} := \{\varphi_n\}_{n=1}^N$  and  $\mathcal{D}_M^{\text{EI}} := (\mathbf{Q}_M, \Sigma_M)$ 

– Extend EI basis:

$$\mathcal{D}_{M+1}^{\text{EI}} \leftarrow \text{EI-EXTENDBASIS}(\mathcal{D}_M^{\text{EI}}, \boldsymbol{\mu}_{\text{max}}, t^k)$$

– Extend RB basis:

$$\mathcal{D}_{N+1}^{\text{RB}} \leftarrow \text{POD-EXTENDBASIS}(\mathcal{D}_N^{\text{RB}}, \boldsymbol{\mu}_{\text{max}}, t^k)$$

– Discard extended RB if error increases:

**if**  $\eta_{N-1,M-1,M'}^k(\boldsymbol{\mu}_{\text{max}}) \leq \max_{(\boldsymbol{\mu}, t) \in \mathcal{M}_{\text{train}}} \eta_{N,M,M'}^k(\boldsymbol{\mu})$  **then**
**return** extended basis data:  $\mathcal{D}_{\Upsilon+1} \leftarrow (\mathcal{D}_N^{\text{RB}}, \mathcal{D}_{M+1}^{\text{EI}})$ 
**else**
**return** extended basis data:  $\mathcal{D}_{\Upsilon+1} \leftarrow (\mathcal{D}_{N+1}^{\text{RB}}, \mathcal{D}_{M+1}^{\text{EI}})$ 
**end if**


---

4.3.1. A similar approach of a synchronized generation of reduced basis spaces has recently been published in [71].

Our proposed algorithm uses the error estimates  $\eta_{N,M,1}^k$  for a greedy search in the parameter samples  $\mathcal{M}_{\text{train}} \subset \mathcal{M}$  and attempts to extend both reduced spaces in each step. Unlike in the POD–ErrorEstimate method, the time step maximizing the error estimate is of interest here, as we want to add only this single snapshot to the collateral reduced basis space. In previous applications of the empirical operator interpolation to non–linear discrete operators in reduced basis schemes [39, 23], it was observed that numerical schemes become unstable if the accuracy of the empirical interpolation is too bad with respect to the accuracy of the reduced basis space. Figure 4.3.1 depicts an exemplary illustration of this effect. The error  $\eta_{N,M,\infty}$  of the reduced simulation is plotted with varying dimensions of reduced basis and empirical interpolation data dimension  $N$  and  $M$ . We observe that for growing reduced basis size, the error estimate “explodes” at some point, where the empirical interpolation is too inaccurate. In order to prevent this behavior during the basis generation, in each extension of the PODEI–GREEDY algorithm, it is checked whether the estimated error increased w.r.t

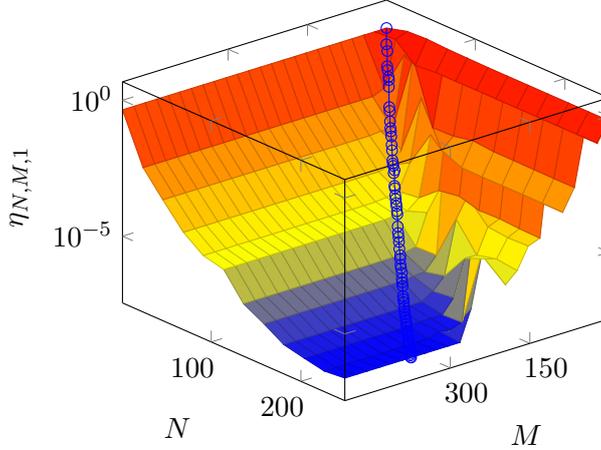


FIGURE 4.3.1. Illustration of reduced basis error convergence with varying dimensionalities  $N$  and  $M$ .

the last extension step and discard newly computed reduced basis functions in this case. This leads to an automatic control of the  $M$ – $N$  correlation between the dimensions of the two basis spaces.

A similar approach for this automatic control of the two basis sizes is presented in [71]. This approach is based on the same idea, but identifies so-called *EIM plateaus* on which the model reduction error is dominated by the projection error made by the projection on the reduced basis space. On these EIM plateaus, a further extension of empirical interpolation basis functions is useless, which is also observable in Figure 4.3.1.

It is noteworthy, that the initial collateral basis is generated by a full EI-GREEDY algorithm generating a small initial basis on a coarser parameter sampling set  $\mathcal{M}_{\text{train}}^{\text{coarse}}$ . This is necessary, because the error estimate depends on a larger set of interpolation Dofs and collateral reduced basis functions. Therefore, the initial collateral reduced basis must be of dimension  $M'$  at least.

#### 4.4. Control of error and efficiency

Greedy algorithm are an efficient means to find good approximation spaces for a finite set of functions. In our setting of parametrized evolution equations, we have to deal with continuous parameter spaces, and therefore, with an infinite amount of functions that have to be approximated by surrogates from a linear function space. In order to apply any of the aforementioned approaches for basis generation, the parameter space needs to be discretized by a finite set of parameters  $\mathcal{M}_{\text{train}} \subset \mathcal{M}$ . This need of parameter space discretization and the requirement of efficiency arouses two crucial problems of the greedy basis generation:

- (i) If the set of training parameters  $\mathcal{M}_{\text{train}}$  comprises too few parameters, the greedy algorithm might “overfit” the generated basis on the training sample, such that

$$\eta_{N,M,M'}(\boldsymbol{\mu}^*) \gg \max_{\boldsymbol{\mu} \in \mathcal{M}_{\text{train}}} \eta_{N,M,M'}(\boldsymbol{\mu}) \quad (4.22)$$

for some  $\boldsymbol{\mu}^* \in \mathcal{M} \setminus \mathcal{M}_{\text{train}}$ .

- (ii) Usually, it is thus unknown, how fast the greedy algorithms converge, and such there is not method to predict the needed dimensions of reduced basis spaces in order to get approximations with the desired error tolerance  $\varepsilon_{\text{tol}}$ . Many applications, however, have constraints on both the error tolerance and the performance of the simulation implied by the reduced basis dimensions.

In this chapter, we discuss solutions for the above two problems. In order to deal with (i), [36, 35] propose to adaptively refine the set of training parameters, if an overfit is detected on the basis of a set of validation parameters. As this extension of the greedy algorithm is applied on all our numerical experiments in Chapter 5, we detail it in the following Section 4.4.1.

The control of the reduced simulations efficiency can be assured by the generation of specialized approximation spaces for sub-domains of the parameter space [29, 27, 35] or sub-intervals of the time domain [21, 24]. As this thesis deals with time dependent evolution problems, we want to focus on adaptive methods to partition the time domain. Especially for convection dominant problems, this approach can lead to good results, since a lot of empirical interpolation Dofs or reduced basis functions are necessary to capture convective flows. Exemplarily, in Section 4.4.2, the time-adaptive generation of empirical operator interpolation data from [24] is presented as an extension to existing publications in this field. This method has been applied for a simple Buckley–Leverett problem in Section 5.3, where we also discuss on its practicability. If control of the reduced simulations efficiency is important in applications, however, combinations of time and parameter space partitions might be necessary. The most extensive publication on the generation of specialized basis space for certain parameter space partitions is [29]. The authors propose a hierarchical reduced basis method, called *hp* certified reduced basis method for affinely parameter dependent problems. It is combined with a method to adaptively increases the number of training parameters in order to prevent overfitting, such that the maximum approximation error of solutions which are not in the set of training parameters can be bounded as well.

**4.4.1. Parameter space adaptation.** In this section, we want to revise the method proposed in [36, 35] which adaptively refines the set of training parameters. For this, we have to change the abstract greedy algorithm defined in Algorithm 4.2.1 by adding a set of validation parameters, and another termination condition, triggering when the “overfit” equation (4.4.1) is fulfilled for some  $\boldsymbol{\mu}^* \in \mathcal{M}_{\text{val}}$ . Here,  $\mathcal{M}_{\text{val}} \subset \mathcal{M}$  is further (finite) set of randomly selected parameter vectors. If the greedy algorithm stops early because of a detected “overfit”, the training set of parameters  $\mathcal{M}_{\text{train}}$  must be refined. After the refinement, the greedy algorithm can be restarted. Of course, it is possible to reuse the previously generated reduced basis as a starting point for the new greedy algorithm execution. Algorithms 4.4.1&4.4.2 sketch the above idea in pseudo-code. This extension is applicable to all of the previously defined greedy algorithms, leading to new versions POD-ADAPTIVEGREEDY, EI-ADAPTIVEGREEDY and PODEI-ADAPTIVEGREEDY.

---

**Algorithm 4.4.1** Adaptive parameter sampling algorithm

---

```

X-ADAPTIVEGREEDY( $\mathcal{M}_0, \mathbb{T}, \varepsilon_{\text{tol}}, \rho_{\text{tol}}, \Upsilon_{\text{max}}$  )
- Initialize reduced basis of size  $\Upsilon_0$ :
   $\mathcal{D}_{\Upsilon_0} \leftarrow \text{X-INITBASIS}()$ 
- Initialize extension step index
   $s \leftarrow 0$ 
repeat
- Compute greedy algorithm for current training set Algorithm 4.4.2
   $(\mathcal{D}_{\Upsilon_{s+1}}, \varepsilon) \leftarrow \text{X-ESGREEDY}(\mathcal{D}_{\Upsilon_s}, \mathcal{M}_s, \mathbb{T}, \mathcal{M}_{\text{val}}, \varepsilon_{\text{tol}}, \rho_{\text{tol}}, \Upsilon_{\text{max}})$ 
if  $\varepsilon \geq \varepsilon_{\text{tol}}$  then
- Refine training set
   $\mathcal{M}_{s+1} \leftarrow \text{REFINESAMPLING}(\mathcal{M}_s, \mathcal{D}_s, \varepsilon)$ 
   $s \leftarrow s + 1$ 
end if
until  $\varepsilon < \varepsilon_{\text{tol}}$  or  $\Upsilon_s > \Upsilon_{\text{max}}$ 
return Final reduced basis:  $\mathcal{D}_{\Upsilon_s}$ 

```

---

The only remaining unknown in the new algorithms is the definition of the refinement algorithm REFINESAMPLING. We present two possible choices in the following:

4.4.1.1. *Mesh refinement.* In our implementation, we used the approach proposed in [36], where for all adaptation steps  $s$ , the parameter set  $\mathcal{M}_s \subset \mathcal{M}$  is given by the vertices  $\mathcal{M}_s = V(\mathcal{G}_s)$  of a Cartesian mesh  $\mathcal{G}_s$  in the parameter space. Each element  $e \in \mathcal{G}_s$  of this mesh can then be associated with an error function

$$\bar{\eta}(e) := \max_{\boldsymbol{\mu} \in V(e) \cup \{c(e)\}} \text{X-ErrorEstimate}(\mathcal{D}_{\Upsilon_s}, \boldsymbol{\mu}), \quad (4.23)$$

**Algorithm 4.4.2** Early-Stopping greedy basis generation

---

X-ESGREEDY( $\mathcal{D}_\Upsilon, \mathcal{M}_{\text{train}}, \mathbb{T}, \mathcal{M}_{\text{val}}, \varepsilon_{\text{tol}}, \rho_{\text{tol}}, \Upsilon_{\text{max}}$ )
**repeat**

– Find parameter and time instance of worst approximated snapshot:

$$(\boldsymbol{\mu}_{\text{max}}, t_{\text{max}}) \leftarrow \arg \max_{(\boldsymbol{\mu}, t) \in \mathcal{M}_{\text{train}} \times \mathbb{T}} \text{X-ERRORESTIMATE}(\mathcal{D}_\Upsilon, \boldsymbol{\mu}, t)$$

– Extend reduced basis by  $v$  new snapshots:

$$\begin{aligned} \mathcal{D}_{\Upsilon+v} &\leftarrow \text{X-EXTENDBASIS}(\mathcal{D}_\Upsilon, \boldsymbol{\mu}_{\text{max}}, t_{\text{max}}) \\ \Upsilon &\leftarrow \Upsilon + v \end{aligned}$$

– Compute maximum error on training set:

$$\varepsilon \leftarrow \max_{(\boldsymbol{\mu}, t) \in \mathcal{M}_{\text{train}} \times \mathbb{T}} \text{X-ERRORESTIMATE}(\mathcal{D}_\Upsilon, \boldsymbol{\mu}, t)$$

– Compute maximum validation ratio:

$$\rho \leftarrow \max_{(\boldsymbol{\mu}, t) \in \mathcal{M}_{\text{val}} \times \mathbb{T}} \text{X-ERRORESTIMATE}(\mathcal{D}_\Upsilon, \boldsymbol{\mu}, t) / \varepsilon$$

**until**  $\varepsilon \leq \varepsilon_{\text{tol}}$  or  $\rho \geq \rho_{\text{tol}}$  or  $\Upsilon > \Upsilon_{\text{max}}$ **return** reduced basis  $\mathcal{D}_\Upsilon$  and error  $\varepsilon$ 


---

where  $V(e)$  and  $c(e)$  denote the mesh vertices adjacent to the element  $e$  respectively its barycenter. This gives an orientation of where the training set is too coarse, i.e. where new elements need to be added to the mesh. So in each adaptation step, a fraction  $\theta \in (0, 1]$  of the mesh elements is chosen for refinement, by selecting the ones with the highest error function  $\eta(e)$ . In Cartesian meshes as used in our experiments the refinement of an element in a  $p$  dimensional mesh leads to  $2^p$  congruent child elements.

Since we would like to begin with a very coarse mesh, a variation of the above element-wise error function is used in our algorithms, given by

$$\eta(e) := |e|s(e) + \frac{\bar{\eta}(e)}{\varepsilon}, \quad (4.24)$$

where  $\varepsilon$  is the maximum error estimate over all vertices of the current mesh,  $|e|$  denotes the size of a mesh element  $e$  and  $s(e)$  counts the number of refinement steps in which the element  $e$  was not refined. This penalty enforces the refinement of very coarse cells in order to detect all local maxima of the error.

This refinement method is used in all our experiments in Section 5. For each numerical example, we comment on the number of refinement step and discuss the automatic selection of parameter samples which are identified to be very important for the reduced basis approximation.

4.4.1.2. *Random refinement.* The construction of a Cartesian mesh for the parameter space can be infeasible for high dimensional parameter spaces  $\mathcal{M}$  (dimension higher than 4 or 5). In those cases, one usually prefers to select the training parameters randomly in the parameter space. In [29], in

each refinement step the number of training parameters is doubled by adding the same number of new random points. A more directed approach would be to again take the position of the worst approximated errors in account, and then use random number generators that generate a fixed number of new training parameters which have high probability to be in the vicinity of previously overfitted parameters. For example, random number generators could generate numbers with Gaussian distribution centered at the parameters with highest error estimates by methods as described in [69].

---

**Algorithm 4.4.3** Time adaptive EI-GREEDY algorithm
 

---

```

EI-TIMEADAPTIVE( $\mathbb{T}, \varepsilon_{\text{tol}}, M_{\text{max}}$ )
- Initialize empirical interpolation data:
  ( $\mathbf{Q}_{M_{\mathbb{T}}}^{\mathbb{T}}, \Sigma_{M_{\mathbb{T}}}^{\mathbb{T}}; \epsilon$ )  $\leftarrow$  EI-GREEDY( $\mathcal{M}_{\text{train}}, \mathbb{T}, \varepsilon_{\text{tol}}, M_{\text{max}}$ )
if  $\text{card}(\mathbb{T}) \leq 2c_{\text{min}}$  then
  - basis space cannot be split, so extend it with  $M_{\text{max}} = \infty$ 
  ( $\mathbf{Q}_{M_{\mathbb{T}}}^{\mathbb{T}}, \Sigma_{M_{\mathbb{T}}}^{\mathbb{T}}; \epsilon$ )  $\leftarrow$  EI-GREEDY( $\mathcal{M}_{\text{train}}, \mathbb{T}, \varepsilon_{\text{tol}}, \infty$ )
end if
if  $\varepsilon \leq \varepsilon_{\text{tol}}$  then
  return empirical interpolation data:  $\left\{ \left( \mathbf{Q}_{M_{\mathbb{T}}}^{\mathbb{T}}, \Sigma_{M_{\mathbb{T}}}^{\mathbb{T}} \right) \right\}$ 
else // if:  $M > M_{\text{max}}$ 
  - Split time interval  $\mathbb{T}$ 
   $\mathbb{T}_1, \mathbb{T}_2 \leftarrow$  SPLITTIMEINTERVAL( $\mathbb{T}, \mathbf{Q}_{M_{\mathbb{T}}}^{\mathbb{T}}, \Sigma_{M_{\mathbb{T}}}^{\mathbb{T}}$ )
  - Compute time interval specific interpolation data
   $\mathcal{D}_1 \leftarrow$  EI-TIMEADAPTIVE( $\mathbb{T}_1, \varepsilon_{\text{tol}}, M_{\text{max}}$ )
   $\mathcal{D}_2 \leftarrow$  EI-TIMEADAPTIVE( $\mathbb{T}_2, \varepsilon_{\text{tol}}, M_{\text{max}}$ )
  return Final empirical interpolation data:  $\mathcal{D}_1 \cup \mathcal{D}_2$ 
end if

```

---

**4.4.2. Time-adaptive empirical operator interpolation.** If the reduced simulations need to fulfill some requirements w.r.t. speed and/or memory of the conducted computations or the generated solution data, it is important to control the maximum size of the reduced basis space and the empirical interpolation data. As most applications also have requirements on the accuracy of the simulations, the aforementioned algorithms can be infeasible, if either the dimension of the generated reduced basis spaces  $\mathcal{W}_{\text{red}}, \mathcal{W}_M$  and the empirical interpolation Dofs  $\Sigma_M$  is too high for efficient reduced simulations, or the maximum certified reduction error is above the error tolerance  $\varepsilon_{\text{tol}}$ .

The dimensions of the reduced basis spaces and the empirical interpolation Dofs can be reduced by splitting the parameter space or the time intervals for which the reduced spaces are generated into smaller sub-domains.

However, it is very difficult or even impossible to predict the required basis size a priori. Therefore, the correct splitting needs to be carried out adaptively during the basis generation algorithms.

Exemplarily, we describe a method published in [24] for a time adaptive generation of empirical interpolation data. The general idea of the so-called EI-TIMEADAPTIVE algorithm is sketched in Algorithm 4.4.3. It produces a splitting of the original discretized time domain  $\mathbb{T} := \{t^k\}_{k=0}^K$  into  $L$  non-overlapping sub-domains  $\tau_1, \dots, \tau_L \subset \mathbb{T}$  such that  $\cup_{l=1}^L \tau_l = \mathbb{T}$  and  $\tau_k \cap \tau_l = \emptyset$  for all  $1 \leq k, l \leq L$ . Each of these sub-domains  $\tau_l$  is connected with a tuple

$$\left( \mathbf{Q}_{M_{\tau_l}}^{\tau_l}, \Sigma_{M_{\tau_l}}^{\tau_l} \right) \quad (4.25)$$

of empirical interpolation basis functions and interpolation points. First, the algorithm executes one of the above defined algorithms EI-GREEDY or EI-ADAPTIVEGREEDY in order to generate a tuple on the entire time domain. If the error tolerance  $\varepsilon_{\text{tol}}$  cannot be reached for the requested basis size  $M_{\text{max}}$ , the time domain is split into two sub-domains, and on each of them, the time adaptive empirical interpolation greedy algorithm recursively computes a new splitting. The recursion ends, if either both constraints on the reduced basis size and the error tolerance are met, or no splitting is possible anymore. The latter condition is true, if a splitting of some time domain  $\tau$  would yield two sub-domains with less than  $c_{\text{min}} \geq 1$  time steps. So, the constant  $c_{\text{min}}$  controls the minimum dimension of the reduced basis spaces. In Section 5.3, the algorithm is tested for a simple discretization of a Buckley–Leverett equation. It is shown, that — at the cost of a more expensive offline-phase — the reduced simulation time can be effectively reduced in this example. However, we do not have a guarantee to fulfill both the accuracy and the online time constraints, as the latter is bounded by a partitioning with time domains consisting of one time index only.

4.4.2.1. *Splitting of the time interval.* The natural choice for the method call to  $\text{SPLITTIMEINTERVAL}(\mathbb{T}, \mathbf{Q}_{M_{\mathbb{T}}}^{\mathbb{T}}, \Sigma_{M_{\mathbb{T}}}^{\mathbb{T}})$  divides the given time domain  $\mathbb{T} = \{t_i\}_{i=1}^{|\mathbb{T}|}$  at its center  $c(\mathbb{T}) := \lceil |\mathbb{T}|/2 \rceil$  into two equally sized intervals  $\mathbb{T}_1 := \{t_i\}_{i=1}^{c(\mathbb{T})}$  and  $\mathbb{T}_2 := \{t_i\}_{i=c(\mathbb{T})+1}^{|\mathbb{T}|}$ .

4.4.2.2. *Reduced simulations.* Of course, the reduced basis scheme from Definition 3.2.2 needs a revision if multiple basis spaces are existent. Concerning the time adaptive generation of empirical interpolation data, the changes are very straight-forward: The EI-TIMEADAPTIVE algorithm, yields the sub-domains  $\mathcal{K} := \{\tau_l\}_{l=1}^L$  and for each  $\tau \in \mathcal{K}$ , a set of empirical basis functions  $\mathbf{Q}_{M_{\tau}}^{\tau}$  with empirical interpolation Dofs  $\Sigma_{M_{\tau}}^{\tau}$ . As explained in Remark 2.1.2, a set of nodal base functions  $\Xi_{M_{\tau}}^{\tau} := \{\xi_m^{\tau}\}_{m=1}^{M_{\tau}}$  can be generated out of the set of collateral reduced basis functions as well. During the offline

phase, we then need to compute further Gramian matrices  $\mathbf{C}^\tau \in \mathbb{R}^{N \times M}$  with components

$$(\mathbf{C}^\tau)_{nm} := \langle \xi_m^\tau, \varphi_n \rangle \quad (4.26)$$

. Then, the empirical operator interpolations given by  $\mathbf{CI}(t^k, \boldsymbol{\mu})$  in equations (3.2.7) and (3.2.8) need to be substituted by the specialized variants  $\mathbf{C}^{\tau(t^k)}\mathbf{I}(t^k, \boldsymbol{\mu})$ , where  $\tau(t^k)$  maps to the unique time domain  $\tau \in \mathcal{K}$  comprising the time index  $t^k$ .

4.4.2.3. *Differences to other adaptive basis generation algorithms.* As mentioned in the introduction of this section, the approach to generate multiple specialized reduced basis spaces has also been applied to reduced basis spaces specialized for sub-domains of the time interval [21] and of the parameter space [29, 27, 35]. The general idea of these algorithms is the same, as for the aforementioned EI-TIMEADAPTIVE algorithm, but there are some differences: In the time adaptive case [21], (i) the splitting heuristic balances the error contribution over time instead of simply splitting each domain in the middle, and (ii) the reduced simulation needs to be extended by projections of solution snapshots between the different reduced basis spaces on the boundary of the time intervals.

For parameter space adaptation of parameter spaces with more than one dimension, the partitioning of the parameter space  $\mathcal{M} \subset \mathbb{R}^p$  is more complex. In [35], the partitioning is represented by a Cartesian mesh, with cells that can be refined into  $2^p$  congruent sub-cells, whereas [29, 27] choose a hierarchical tessellation where on each level of the hierarchy, two tessellation cells exists, comprising all parameters which are nearest neighbor of either of two corresponding reference parameters.

Furthermore, [35, 27] introduce a two-step approach in which they forecast the right partition after computing one with less accuracy  $\varepsilon_{\text{tol},1} > \varepsilon_{\text{tol}}$ . This increases the offline time, as less reduced basis spaces must be discarded.

## CHAPTER 5

### Numerical experiments

In this chapter, we demonstrate experiments for the presented reduced basis scheme. We numerically prove the applicability of the reduced basis scheme from Definition 3.2.2, compare the basis generation algorithms from Chapter 4 and analyze the efficiency of the a posteriori error estimate (3.3.5).

Three main problems are considered that all fit into the setting of example (3.1.6)-(3.1.9). The first one is a Burgers problem with a purely implicit discretization and smooth solution snapshots. Preliminary results on this example without the new a posteriori error estimator and with explicit discretization have been presented in [37]. The second problem is based on a nonlinear non-stationary diffusion equation also with a purely implicit discretization. As initial data function we chose a discontinuous one and, by the choice of non-linearity, this discontinuity should be preserved over time for certain parameters. This problem setting was chosen in order to validate, how well the reduced basis method can deal with discontinuities. Such problems are popular applications for finite volume methods and considered to be a difficult case for Galerkin projection methods like the reduced basis method.

The third problem can be understood as an intermediate step to the two phase flow problem discussed in the next chapter. It also models the flow of two phases in a porous medium, but with a fixed velocity field, neglecting the effects aroused by capillary pressure. In one dimension, this problem is known as the Buckley–Leverett equation in literature [6]. The results of this problem were first published in [24] in order to validate the time adaptive empirical interpolation generation described in Section 4.4.2.

All three problems are of non-linear type, but degenerate into linear ones for specific parameter configurations.

All implementation are based on our MATLAB software package *RB-matlab* [75, 22], which is further detailed in Chapter 7.

#### 5.1. Burgers equation

In a first example, we demonstrate the applicability to a nonlinear convection problem

$$\partial_t u - \nabla \cdot f(u; \boldsymbol{\mu}) = 0 \tag{5.1}$$

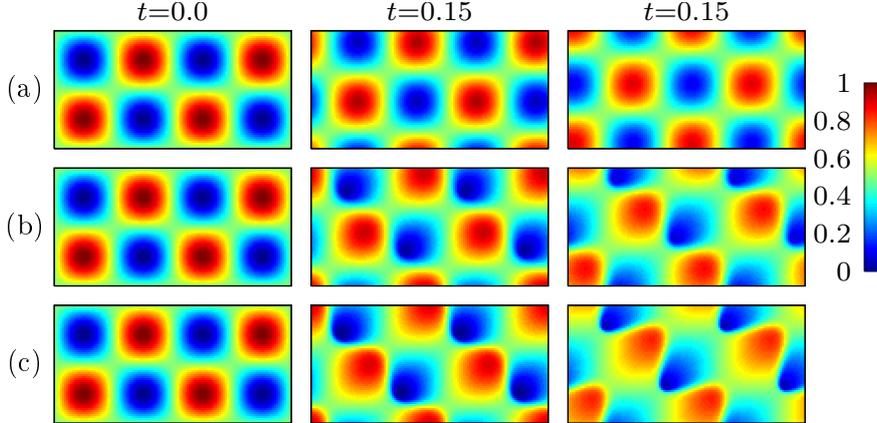


FIGURE 5.1.1. Illustration of transport for smooth data. Snapshots at different time instants for (a)  $\mu_1 = 1$  and (b)  $\mu_1 = 1.5$  and (c)  $\mu_1 = 2$ .

with smooth initial data and a single parameter.

We choose  $\Omega = [0, 2] \times [0, 1]$  with purely cyclical boundary conditions and fix the end time  $T = 0.3$ . We prescribe the nonlinear flux function

$$f(u; \boldsymbol{\mu}) := \mathbf{v}u^{\mu_1} \quad (5.2)$$

with exponent  $\mu_1$  and space- and time-constant velocity field

$$\mathbf{v} = (1, 1)^T, \quad (5.3)$$

the initial data is a smooth function

$$u_0(x) = \frac{1}{2}(1 + \sin(2\pi x_1) \sin(2\pi x_2)) \quad (5.4)$$

for  $x = (x_1, x_2)^T \in \Omega$ .

Overall, we consider the single parameter  $\boldsymbol{\mu} = (\mu_1) \in \mathcal{M} := [1, 2]$  for the exponent in the flux of the evolution equation. We choose a  $120 \times 60$  rectangular grid for decomposing  $\Omega$  and  $K = 100$  time-steps which satisfies the CFL-condition.

Figure 5.1.1 illustrates the time evolution of the solutions for different parameters, in particular the initial data which is independent of the parameter  $\mu_1$  and the final state for  $\mu_1 = 1$  respectively  $\mu_1 = 2$ . The transition between linear convection ( $\mu_1 = 1$ ) and the nonlinear non-viscous Burgers equation ( $\mu_1 = 2$ ) can nicely be observed. In the latter case shock discontinuities emerge over time.

**Offline phase.** For this model setting, the reduced spaces and the empirical interpolation Dofs are generated by

- (A) subsequent execution of the EI- and the POD-ADAPTIVEGREEDY algorithms and
- (B) the PODEI-ADAPTIVEGREEDY algorithm which combines the generation of the reduced basis space and the empirical interpolation data.

All algorithms are defined in Chapter 4. In the first case (A), the collateral reduced basis space is extended to its maximum size, i.e. until the reduction of the interpolation error stagnates due to machine precision and numerical constraints.

All computations are carried out on the PALMA cluster of the University of Münster using 24 cores each running at 2.67 GHz. The implementation of the algorithms makes use of the simple parallelization technique mentioned in Remark 2.1.3. For the POD-ADAPTIVEGREEDY and the PODEI-ADAPTIVEGREEDY algorithms, we also apply the adaptation technique described in Section 4.4.1, i.e. we begin with an initial uniform parameter set  $\mathcal{M}_{\text{train}}^0 \subset \mathcal{M}$  and refine this set, when indicated by a bad ratio between the maximum errors over the training parameters and another set of validation parameters. This algorithm is discussed in Section 4.4.1. In the experiments, we aim to assure a ratio of  $\rho_{\text{tol}} := 1.1$ . The initial parameter sampling set  $\mathcal{M}_{\text{train}}^0$  for all three greedy algorithms consists of 26 uniformly distributed parameters in the parameter space interval  $[1, 2]$ .

In case (A) the generation of the collateral reduced basis with the EI-ADAPTIVEGREEDY algorithm takes 36 minutes and terminates with  $M = 499$  basis functions after the tolerance of  $10^{-10}$  for the interpolation error has been reached. The computation of the detailed simulations for all parameters takes only 100 seconds because of parallelization of these computations. The POD-ADAPTIVEGREEDY algorithm terminates after a bit less than 3 hours and produces a reduced basis space of dimension  $N_{\text{max}} = 249$ . Note, that the main part of the offline run-time costs, namely the search for new basis functions, depends linearly on the number of available cores and therefor, the offline time can be controlled by using more processors. It is noteworthy, that the collateral reduced basis space is generated much faster than the reduced basis space. This is firstly because of the expensive POD step that needs to be computed in every POD-ADAPTIVEGREEDY extension step, and secondly, because the empirical interpolation errors are computed very fast after all necessary operator evaluations of detailed simulations have been cached.

In case (B) the initial collateral reduced basis is generated based on the coarse training sample  $\mathcal{M}_{\text{train}}^{\text{coarse}} = \{1, 2\}$  including only the external points of

the parameter domain, and stops after three minutes with  $M_{\text{small}} = 20$  generated basis functions. As the PODEI-ADAPTIVEGREEDY algorithm starts with a very small initial collateral reduced basis, the total offline time is smaller than in case (A) — especially for small reduced basis space dimensions. (c.f. Table 5.1.1) The algorithm stops after 3.27 hours. In both cases, the initial training sets  $\mathcal{M}_{\text{train}}^0$  are not refined.

Next, we analyze the quality of the model order reduction induced by the reduced basis spaces. Figure 5.1.2(a) shows the growth of the Lebesgue constant defined in (2.3.6) with respect to the increasing collateral reduced basis size  $M$ . It can be clearly seen, that the increase is linear with a maximum Lebesgue value of 182. Figure 5.1.2(b)+(c) help to understand how the empirical interpolation algorithm works. It illustrates the cell midpoints corresponding to the selected interpolation DOFs  $\Sigma_M$  and visualizes the selection order of the empirical interpolation algorithm by plotting points selected earlier in darker shades. It is visually comprehensible from the illustration that the algorithm realizes an obvious space compression, because it recognizes the space symmetry of the solution, such that the selected cell midpoints are all located in the lower left quarter of the domain. This is because the EI-EXTENDBASIS method selects the Dof with the lowest index in case of several equal maxima.

**Online phase.** In order to get a notion of the reduced simulations accuracy, in Figure 5.1.3 we illustrate the error convergence for the resulting reduced simulation scheme. We select a set  $\mathcal{M}_{\text{test}} \subset \mathcal{M}$  of 100 random values for  $\boldsymbol{\mu}$  not used during basis generation and determine the maximum error

$$\max_{\boldsymbol{\mu} \in \mathcal{M}_{\text{test}}} \|u_{\text{red}}(\boldsymbol{\mu}) - u_h(\boldsymbol{\mu})\|_{L^\infty([0,T]; \mathcal{W}_h)} \quad (5.5)$$

between the reduced and the detailed simulations for different dimensionalities  $N$  and  $M$ . The resulting maximum error is plotted in logarithmic scale. Figure 5.1.3(b) nicely shows, how a simultaneous increase of  $N$  and  $M$  reveals almost exponential convergence along a selected diagonal of the plot. This simultaneous increase is important: If  $M$  is fixed at a low value, increase of  $N$  over a certain limit can give an error increase induced by incorrectly approximated operator evaluations. If  $N$  is fixed, raising  $M$  gives no error improvement at some point.

The main goal of RB-methods is an accurate approximation under largely reduced simulation time. To assess these computation times, we determine the detailed and reduced simulation times over a sample of 100 random parameters and report the average run-times. These efficiency results are summarized in Tables 5.1.1(A)+(B) for different reduced basis sizes and

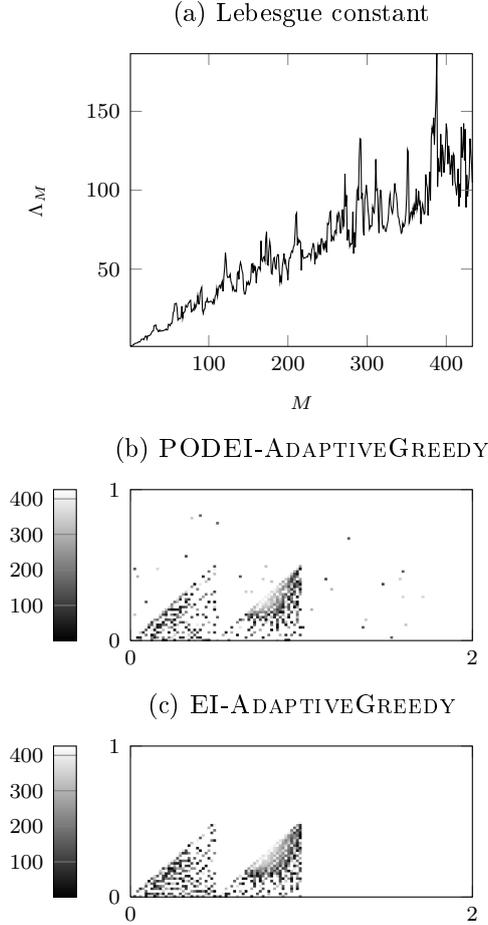


FIGURE 5.1.2. Illustration of (a) growth of Lebesgue constant and (b)+(c) the order of interpolation Dof selection for the Burgers problem. DOFs corresponding to darker points are selected first.

for (A) a subsequent generation of reduced basis and collateral reduced basis space and (B) the synchronized generation of both using the PODEI-ADAPTIVEGREEDY algorithm. In the first case the ratio between the dimensions for the empirical interpolation and the reduced basis are determined from the maximum basis sizes  $499/249 \approx 2$  at which the algorithms stopped. Note, that this ratio cannot be assumed to be a good choice for smaller basis dimensions. In contrast, the  $M$ - $N$  correlations in the second table are taken as inferred from the PODEI-ADAPTIVEGREEDY algorithm which sequentially expands both reduced basis spaces. It can be nicely observed, that this approach leads to better ratios, i.e. reaches smaller maximum errors with smaller basis spaces and therefore faster reduced simulation times. An exception is the second row of the tables with small dimensions  $(N, M) = (42, 72)$ . For reduced simulations with small reduced basis spaces,

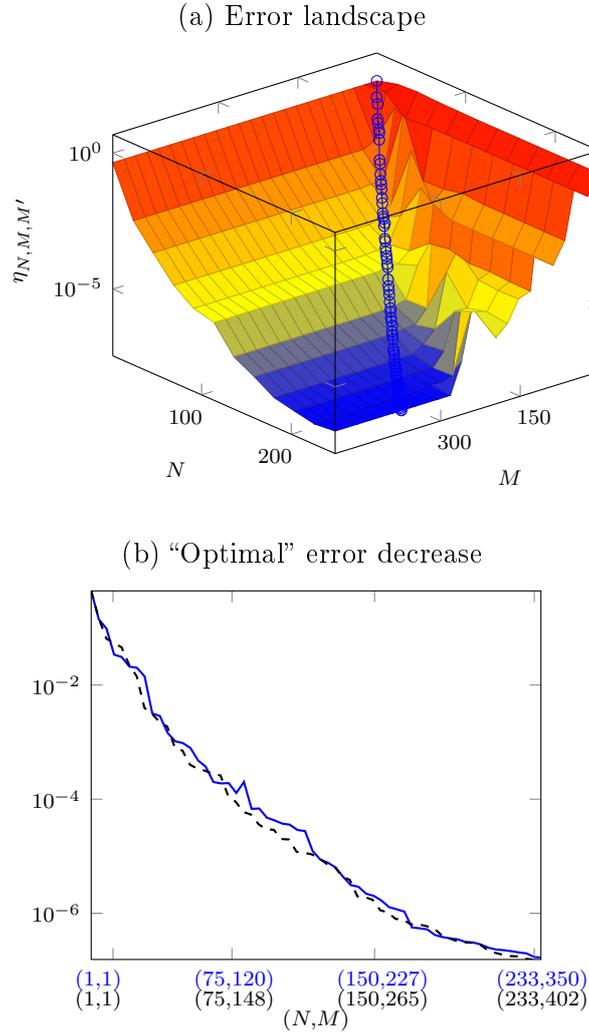


FIGURE 5.1.3. (a) Illustration of reduced basis error convergence with varying dimensionalities  $N$  and  $M$ . (b) Error plot for simultaneously increased bases sizes  $N$  and  $M$  for the “optimal” ratio visually derived from the landscape (a) and the error curve derived from the PODEI-ADAPTIVEGREEDY algorithm (dashed line).

the PODEI-ADAPTIVEGREEDY suffers from the bad initial collateral reduced basis which needs a few extension steps to stabilize.

In general, however, the PODEI-ADAPTIVEGREEDY algorithm finds a very good choice for the  $M$ – $N$  correlation. This observations is emphasized by the Figure 5.1.3(b) comparing the maximum error decrease of the PODEI-ADAPTIVEGREEDY with an “optimal” error decrease curve manually derived from the neighboring error landscape plot.

		N	M	$\emptyset$ -run-time[s]	max. error	offline time[h]
(A) EL- ADAPTIVEGREEDY	+ POD-	$H = 7200$	–	90.01	0.00	0
		42	83	4.42	$1.15 \cdot 10^{-3}$	0.96
		83	166	6.23	$6.03 \cdot 10^{-5}$	1.34
		125	250	8.99	$7.43 \cdot 10^{-6}$	1.74
		166	333	11.6	$8.33 \cdot 10^{-7}$	2.23
		208	416	15.64	$2.47 \cdot 10^{-7}$	2.78
		249	499	19.56	$2.38 \cdot 10^{-7}$	3.4
		N	M	$\emptyset$ -run-time[s]	max. error	offline time[h]
(B) PODEL- ADAPTIVEGREEDY		$H = 7200$	–	90.01	0.00	0
		42	72	4.44	$1.73 \cdot 10^{-3}$	0.54
		83	144	6.04	$5.74 \cdot 10^{-5}$	1.09
		125	216	8.37	$7.30 \cdot 10^{-6}$	1.55
		167	288	11.92	$7.63 \cdot 10^{-7}$	2.08
		208	360	15.08	$2.31 \cdot 10^{-7}$	2.69
		233	402	16.48	$1.55 \cdot 10^{-7}$	3.27

TABLE 5.1.1. Run-time comparison for detailed simulation with reduced simulations of varying reduced dimensionalities. The average run-times and maximum errors are obtained over a test sample  $\mathcal{M}_{\text{test}} \subset \mathcal{M}$  of size 100. The maximum error  $\|u_h^k(\boldsymbol{\mu}) - u_{\text{red}}^k(\boldsymbol{\mu})\|_{\mathcal{W}_h}$  is obtained over all tuples  $(\boldsymbol{\mu}, k) \in \mathcal{M}_{\text{test}} \times [0, \dots, K]$  involving high dimensional error computations.

It can be seen nicely, that we obtain acceleration factors of 4.7 – 20 depending on the dimensionalities of the reduced simulation. The acceleration factors obtained by the two different basis generation methods hardly differ.

## 5.2. Porous Medium Equation

In this section, we consider the porous medium equation given by the nonlinear diffusion problem

$$\partial_t u - m \Delta u^{\mu_1} = 0 \quad \text{in } \Omega \times [0, T_{\max}], \quad (5.6)$$

$$u = c_0 + u_0 \quad \text{on } \partial\Omega \times [0, T_{\max}], \quad (5.7)$$

$$u(\cdot, 0) = c_0 + u_0 \quad \text{on } \Omega \times \{0\}, \quad (5.8)$$

on a rectangular domain  $\Omega = [0, 1]^2$ . The end time is fixed at  $T_{\max} = 1.0$ . The initial data function  $u_0$  is a field of symmetrically arranged bar shaped concentrations illustrated in Figure 5.2.1(a). This gives us a non-smooth initial concentration depending on the initial parameter  $c_0$ .

The parameter vector is chosen as

$$\boldsymbol{\mu} = (\mu_1, m, c_0) \in \mathcal{M} := [1, 5] \times [0, 0.01] \times [0, 0.2] \quad (5.9)$$

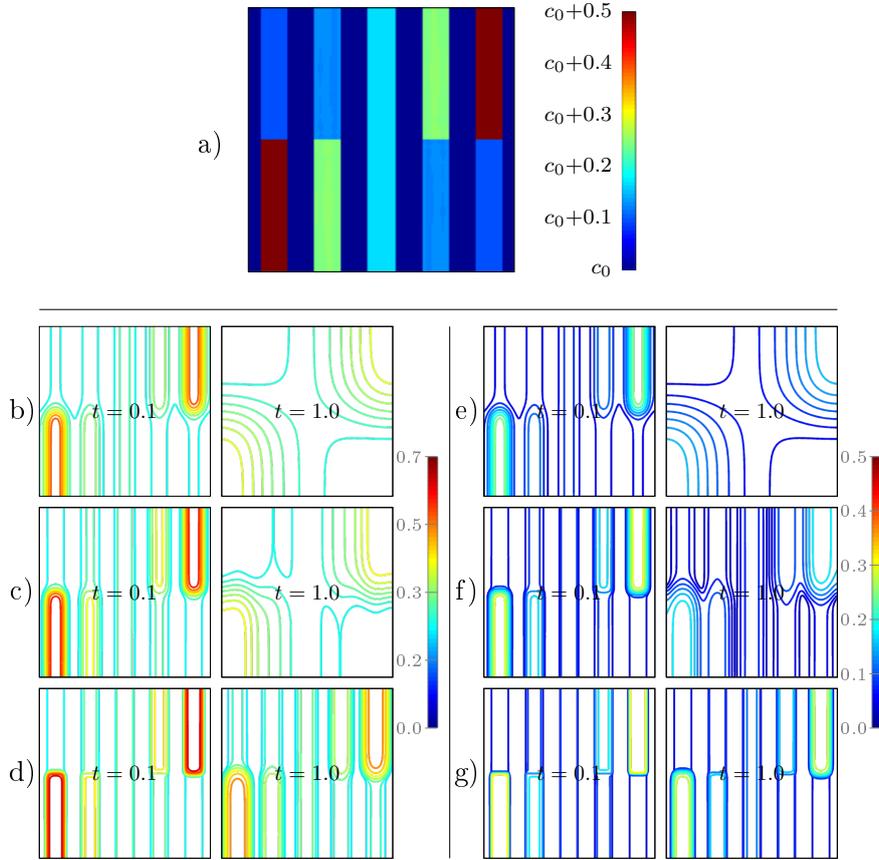


FIGURE 5.2.1. Plot (a) depicts a color shading of the initial data. Below, isolines of reduced solutions are given at time instances  $t = 0.1$  and  $t = 1.0$  for different parameter vectors: (b)  $\boldsymbol{\mu} = (1, 0.01, 0.2)$ , (c)  $\boldsymbol{\mu} = (2, 0.01, 0.2)$ , (d)  $\boldsymbol{\mu} = (4, 0.01, 0.2)$ , (e)  $\boldsymbol{\mu} = (1, 0.01, 0.0)$ , (f)  $\boldsymbol{\mu} = (2, 0.01, 0.0)$  and (g)  $\boldsymbol{\mu} = (4, 0.01, 0.0)$ .

such that for  $\mu_1 = 2$  we get the isothermal porous medium equation and for  $\mu_1 > 2$  a porous medium equation with adiabatic flow. For  $\mu_1 = 1$  it degenerates into the linear heat equation. Note also, that for  $c_0$  close to zero, diffusion outside the bars is turned off in the non-linear case ( $\mu_1 > 1$ ).

This effect can be observed in Figure 5.2.1(f)+(g) showing large diffusion effects inside the bars with high concentration after short time periods already, but almost none outside. Furthermore, Figure 5.2.1 clearly illustrates the nonlinear effects, as the diffusion is larger (more contour lines) in the bars with high initial concentration. An exception, of course, are the reduced solutions in the upper row, modeling linear diffusion where the diffusivity stays the same globally. For the discretization, we chose again the finite volume scheme from Section 3.1.1 on a  $100 \times 100$  grid for decomposing

$\Omega$  and  $K = 80$  time steps for the discretization of the time interval  $[0, 1]$ . The diffusivity is discretized implicitly, such that its non-linearities are to be resolved with the Newton–Raphson method. If we make the reasonable assumption that the domain of the operator  $\mathcal{L}_I(\boldsymbol{\mu})$  as defined in Section 3.1.1 stays in the range  $[0, 1]$ , the operator fulfills the Lipschitz condition (3.3.1) for all  $\boldsymbol{\mu} \in \mathcal{M}$  with  $C_{I,\Delta t} = 1$ . For details on this, we refer to Appendix ??.

**5.2.1. Offline phase.** Like in the previous example, we compute the reduced basis spaces on the PALMA cluster using 24 cores for

- (A) subsequent execution of EI- and POD-ADAPTIVEGREEDY algorithms,
- (B) the PODEI-ADAPTIVEGREEDY algorithm, and
- (C) This time a third run with a variation of (A), but where the “true” error indicator

$$\eta(\boldsymbol{\mu}) = \|u_h^K(\boldsymbol{\mu}) - u_{\text{red}}^K(\boldsymbol{\mu})\|_{\mathcal{W}_h}$$

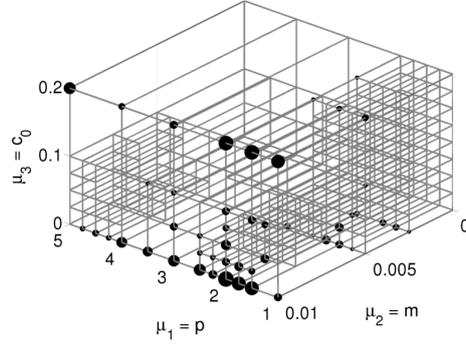
instead of the error estimator  $\eta_{N,M,M'}^K(\boldsymbol{\mu})$  is executed.

With the latter, we want to assess the suitability of the error estimator for the basis generation. This question is discussed in Section 5.2.3 below.

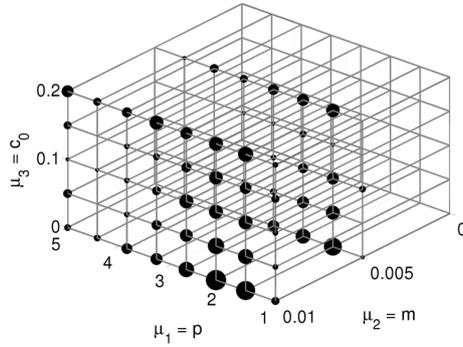
Again, we apply the adaptation technique described in Section 4.4.1 on the parameter space. The parameter sampling set for the EI-ADAPTIVEGREEDY algorithm consists of 120 parameter vectors in order to get a good operator interpolation needed for reduced basis generation later. The distribution of training parameters is illustrated in Figure 5.2.2(b). The vertices of the drawn grid match with the training parameters. For the POD-ADAPTIVEGREEDY algorithm the same initial parameter set  $\mathcal{M}_{\text{train}}^0$  is chosen, whereas the PODEI-ADAPTIVEGREEDY algorithm starts with 30 training parameters. The result of the adaptive refinement procedures is illustrated in Figures 5.2.2(a)+(c). The training set in case (A) has been refined once with a final number of 209 parameter vectors, and in case (B) where we started with a coarser grid, has been refined three times resulting in a set of size 305. Furthermore, Figure 5.2.2 shows that parameters which are often selected for basis extension correlate with the refined parts of the grids. Here, we nicely observe two facts: First, solutions that show a complex evolution over time, are selected more frequently until they are approximated well enough, and second for parameters with solutions evolving more linear in time, few or even zero snapshots are sufficient, because these can be approximated by linear combinations of other basis functions.

In case (A) the empirical interpolation algorithm takes 38.5 minutes until it reaches the final number of 425 basis functions and the computation of

(a) Parameter selection PODEI-ADAPTIVEGREEDY



(b) Parameter selection EI-GREEDY



(c) Parameter selection POD-ADAPTIVEGREEDY

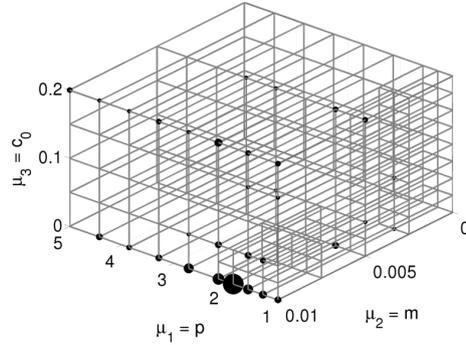


FIGURE 5.2.2. Illustration of the final training parameter sets  $\mathcal{M}_{\text{train}}^{m_{\text{ref}}}$  after (a)  $m_{\text{ref}} = 3$ , (b)  $m_{\text{ref}} = 0$  and (c)  $m_{\text{ref}} = 1$  refinement steps. The vertices match with the parameters in the training sets and the overlaying bubble plots illustrates how frequently a parameter is picked for basis extension.

the detailed simulations for all training set parameters takes 4 minutes. The reduced basis space generation terminates after 2.1 hours and 99 generated reduced basis functions. It does not reach the targeted error of  $\varepsilon_{\text{tol}} = 10^{-4}$  because after 99 basis extensions no snapshots can be found which reduce

the maximum error estimate over the training parameter set. Figure 5.2.3(b) illustrating the decrease of the error estimates during the basis extension suggest that the reason for the stagnation comes from a bad estimation of the empirical operator interpolation: The stagnation begins after the refinement step which is indicated by the strong peak in the error curve, i.e. when the current training set differs from the one used for the EI-ADAPTIVEGREEDY algorithm.

In case (B) the synchronized generation of basis functions takes about 4 hours, but reaches the a priori given error tolerance  $\varepsilon_{\text{tol}} = 10^{-4}$ . The initial collateral reduced basis is generated in about two minutes with a coarse training set  $\mathcal{M}_{\text{train}}^{\text{coarse}}$  containing only the extremal points of the parameter space and after  $M_{\text{small}} = 20$  basis functions have been generated.

The Lebesgue constant  $\Lambda_M$  of the collateral reduced basis space (c.f. (2.3.6) grows in both cases linearly with a maximum of 153 in case (A) and 203 in case (B).

For details on the offline computation times, we refer to Table 5.2.1. The table nicely shows that for case (B) very reasonable  $M$ - $N$  correlations are inferred from the synchronized basis extension and even a basis of better quality is produced: Comparing lines with similar maximum errors over the validation set of 100 parameters, the reduced basis from (B) needs less basis vectors which have been generated in a shorter offline phase. For example, the maximum error of  $3.54 \cdot 10^{-5}$  is reached with basis dimensions  $(N, M) = (93, 358)$  which can be generated in 2.72 hours, whereas Table 5.2.1(A) shows that the worse error of  $4.06 \cdot 10^{-5}$  needs basis size  $(N, M) = (99, 425)$  and a basis generation time of 3.3 hours.

Figure 5.2.3(a) illustrates how often and how frequently basis functions are discarded during the execution of the PODEI-ADAPTIVEGREEDY algorithm. 5.2.3(b) compares the error estimation decrease during POD-ADAPTIVEGREEDY and the PODEI-ADAPTIVEGREEDY extension. The latter needs more extension steps in order to reach a certain basis space dimension, because some basis functions are discarded as indicated by the crossed marks. Note also that both error curves have intermediate peaks because of the adaptation of the parameter training set. As the maximum error is computed over the current training parameter set, it grows after such a refinement step, when more parameters are added.

**5.2.2. Online phase.** Figure 5.2.4 shows cross-section plots of detailed and reduced simulation snapshots of the two worst solutions from a set of 100 randomly chosen solution trajectories. Visually, there are no differences between the dashed curve of the detailed and the solid curve of the reduced

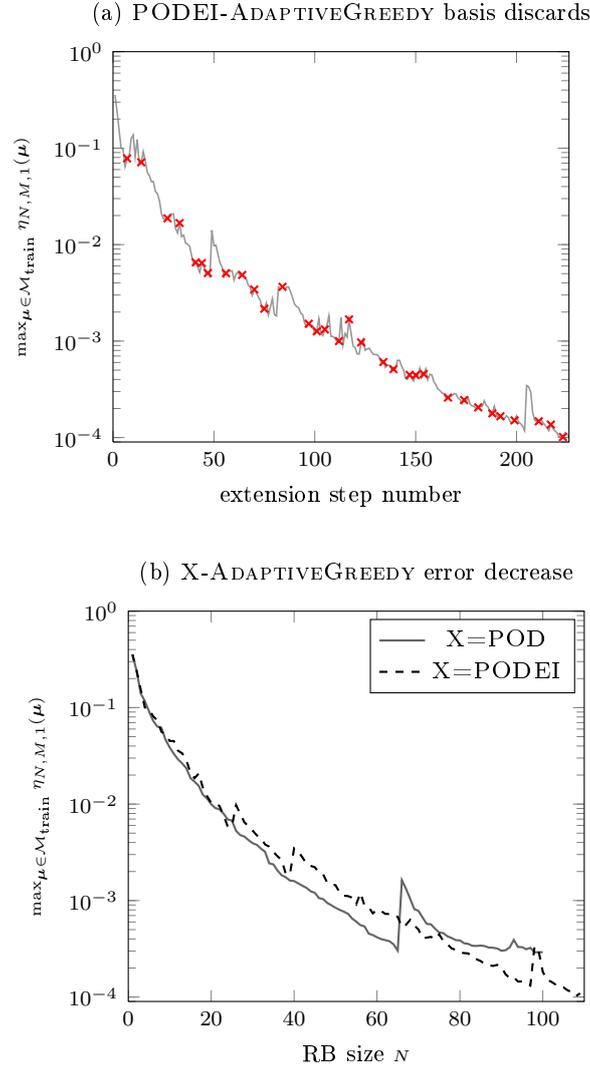


FIGURE 5.2.3. Illustration of (a) the extension steps during PODEI-ADAPTIVEGREEDY at which reduced basis functions were discarded (marked with a cross) and (b) the error decrease during basis extension with growing reduced basis size.

solution snapshots. This indicates that dispersion effects arising from the additional approximation are negligible for this example.

To quantify the reduced simulations quality, we proceed exactly as we did in the previous section for the Burgers problem. We again pick a test sample  $\mathcal{M}_{\text{test}} \subset \mathcal{M}$  of 100 randomly chosen values from the parameter space, measure the error  $\|u_h(\boldsymbol{\mu}) - u_{\text{red}}(\boldsymbol{\mu})\|_{L^\infty([0, T_{\text{max}}]; \mathcal{W}_h)}$  for all  $\boldsymbol{\mu} \in \mathcal{M}_{\text{test}}$  and compare the computation times of detailed and reduced simulations. Results for different magnitudes of the reduced bases dimensions  $M$  and  $N$  are shown

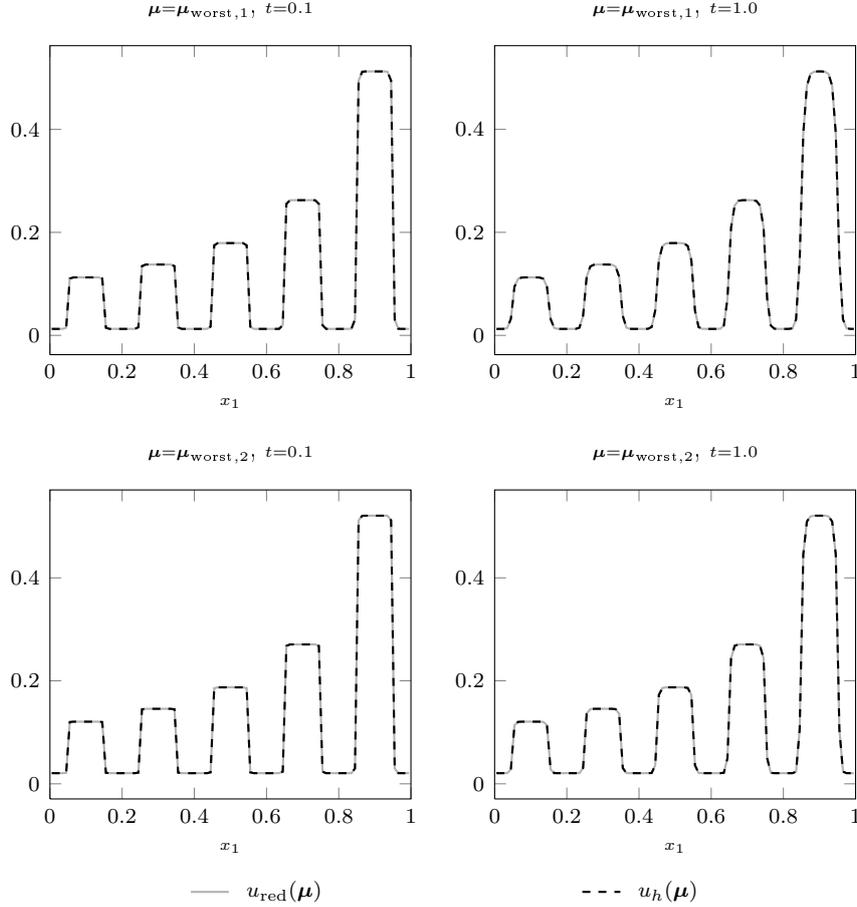


FIGURE 5.2.4. Comparison of cross-section plots at  $x_2 = 0.6$  between detailed and reduced simulation snapshots for different time instances and parameters with worst and seconds worst error from a random test set of 100 parameters:  $\boldsymbol{\mu}_{\text{worst},1} = (1.349, 7.293 \cdot 10^{-5}, 0.013)$  and  $\boldsymbol{\mu}_{\text{worst},2} = (1.737, 5.758 \cdot 10^{-5}, 0.021)$ .

in Table 5.2.1. Note, that the run-times are averaged over the test parameter set, and actually show a high deviation from this mean by factors up to 10, because the number of Newton steps that are needed to proceed between time-steps varies noticeably. For linear problems one Newton step is enough, whereas up to a maximum of 30 Newton steps for stronger non-linearities, i.e.  $\mu_2 > 1$  are necessary. Consequently, the acceleration factors for reduced simulations with maximum reduced basis dimensions also differ notably.

**5.2.3. A posteriori error estimator.** The a posteriori error estimator (3.3.5) has two main purposes: It should first give a tight and rigorous bound on the real error made through the model order reduction, and second improve the run-time of basis generation algorithms by providing an efficient

	N	M	$\phi$ -run-time[s]	max. error	offline time[h]
(A) EI- + POD- ADAPTIVEGREEDY, $\eta_{N,M,1}$	$H = 10000$	–	55.38	0.00	0
	17	71	1.57	$3.56 \cdot 10^{-3}$	1.35
	33	142	1.95	$8.33 \cdot 10^{-4}$	1.67
	50	213	2.51	$2.08 \cdot 10^{-4}$	2.07
	66	283	3.19	$5.88 \cdot 10^{-5}$	2.43
	83	354	4.07	$5.55 \cdot 10^{-5}$	2.88
	99	425	5.3	$4.06 \cdot 10^{-5}$	3.3
	N	M	$\phi$ -run-time[s]	max. error	offline time[h]
(B) PODEI- ADAPTIVEGREEDY, $\eta_{N,M,1}$	$H = 10000$	–	55.38	0.00	0
	19	72	1.61	$3.01 \cdot 10^{-3}$	0.16
	37	143	2.07	$7.90 \cdot 10^{-4}$	0.45
	56	215	2.67	$1.66 \cdot 10^{-4}$	1.01
	74	286	3.6	$6.36 \cdot 10^{-5}$	1.69
	93	358	4.83	$3.54 \cdot 10^{-5}$	2.72
	111	429	6.55	$1.96 \cdot 10^{-5}$	4.02
	N	M	$\phi$ -run-time[s]	max. error	offline time[h]
(C) EI- + POD- ADAPTIVEGREEDY, $\ \cdot\ $ $\eta$	$H = 10000$	–	55.38	0.00	0
	17	71	1.46	$2.97 \cdot 10^{-3}$	1.26
	33	142	1.83	$5.87 \cdot 10^{-4}$	1.91
	50	213	2.31	$1.24 \cdot 10^{-4}$	2.64
	67	283	3.32	$6.30 \cdot 10^{-5}$	3.49
	83	354	4.06	$3.19 \cdot 10^{-5}$	4.31
	100	425	5.99	$1.34 \cdot 10^{-5}$	5.64

TABLE 5.2.1. Run-time comparison for detailed simulation with reduced simulations of varying reduced dimensionalities. The average run-times and maximum errors are obtained over a test sample  $\mathcal{M}_{\text{test}} \subset \mathcal{M}$  of size 100. The maximum error  $\|u_h^k(\boldsymbol{\mu}) - u_{\text{red}}^k(\boldsymbol{\mu})\|_{\mathcal{W}_h}$  is obtained over all tuples  $(\boldsymbol{\mu}, k) \in \mathcal{M}_{\text{test}} \times [0, \dots, K]$ . Subtable (C) is based on a reduced basis generated with a “true” error indicator  $\eta(\boldsymbol{\mu}) = \|u_h^K(\boldsymbol{\mu}) - u_{\text{red}}^K(\boldsymbol{\mu})\|_{\mathcal{W}_h}$ .

and trust-worthy error indicator. In this section, we evaluate how well both these tasks are resolved.

In a first test, we check the efficiency of the error estimator

$$\lambda(\boldsymbol{\mu}) := \frac{\eta_{N,M,M'}^K(\boldsymbol{\mu})}{\|u_h^K(\boldsymbol{\mu}) - u_{\text{red}}^K(\boldsymbol{\mu})\|_{\mathcal{W}_h}} \quad (5.10)$$

for a random sample of 100 parameters and different values for the extra collateral basis functions  $M'$  used to estimate the interpolation error. We expect  $\lambda(\boldsymbol{\mu})$  to be greater than one, meaning the estimator is rigorous, i.e. does not underestimate the real error. On the other hand, it is desirable that the efficiency is very close to one.

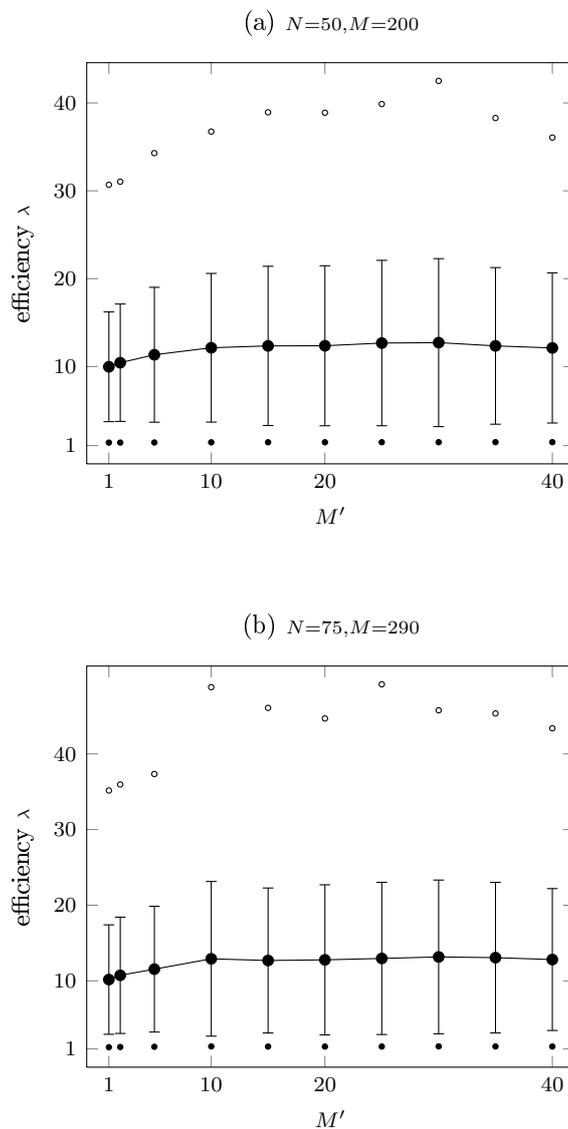


FIGURE 5.2.5. Error bar plot showing mean and standard deviation of error estimator efficiency over a sample of 100 random parameters for different values of  $M'$ . The dots indicate the minimum and maximum efficiency.

Recall our assumption, that a large enough collateral reduced basis allows interpolated operator evaluations to be almost exact. This gives rise to assess the empirical operator interpolation error with basis dimension  $M$  by comparing it to the finer interpolation with basis dimension  $M + M'$ . One question that needs to be answered empirically here, is whether this assumption is valid and if yes, how big  $M'$  needs to be chosen. The results of our experiments are illustrated in Figure 5.2.5: The plot shows statistical data of the measured effectivities for different error estimators, i.e. different

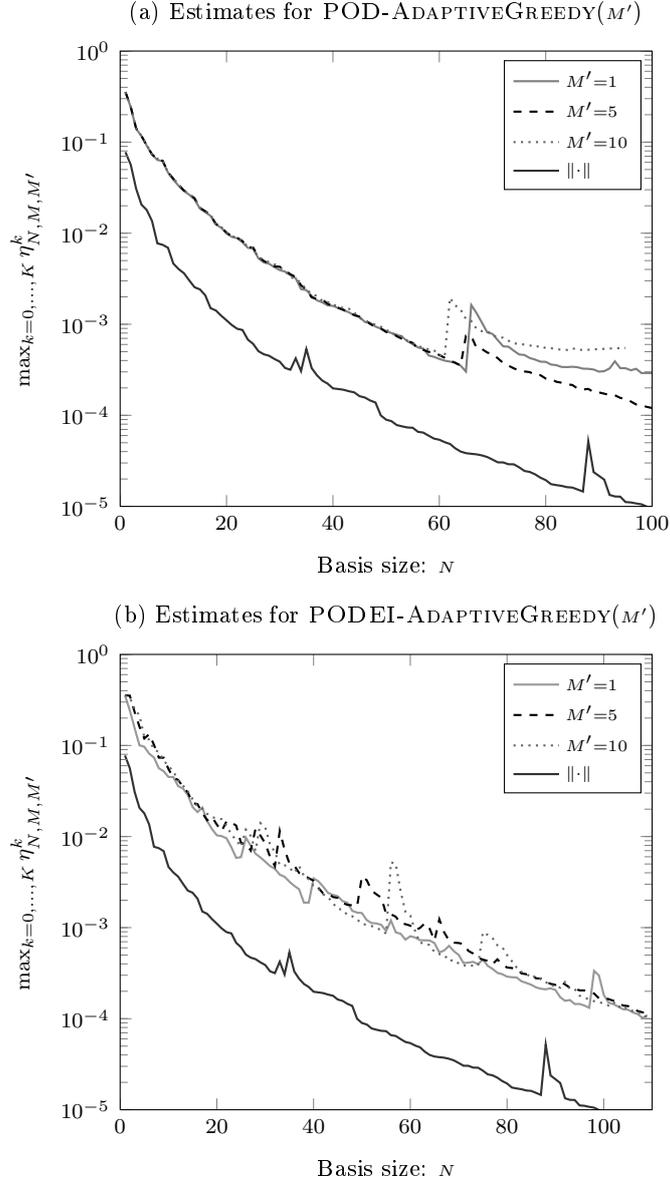


FIGURE 5.2.6. Comparison of estimated error decrease during basis generation with (a) POD-ADAPTIVEGREEDY and (b) PODEI-ADAPTIVEGREEDY algorithm. Different error indicators are used in order to select the worst approximated trajectories (c.f. Algorithm 4.2.2). The error indicators vary in the number of collateral reduced basis functions  $M'$  used in order to approximate the empirical interpolation error. The lower indicator curves depict the error decrease during a POD-ADAPTIVEGREEDY with “true” error indicator  $\eta(\boldsymbol{\mu}) = \|u_{\text{red}}^K(\boldsymbol{\mu}) - u_h^K(\boldsymbol{\mu})\|_{\mathcal{W}_h}$ .

values for  $M'$ . We observe that the mean effectivity is slightly above 10 for  $M' = 1$  and stabilizes at about 12 for small  $M'$  already. The latter gives rise to our assumption that the empirical interpolation error is well approximated by a small set of extra basis functions.

As the standard deviation of the error estimator's efficiency is still in a reasonable range for the sample parameters in this test, we can expect the estimator to have a good qualification as an error indicator for the POD-ADAPTIVEGREEDY and the PODEI-ADAPTIVEGREEDY algorithms. This is confirmed in further test runs where both algorithms are run several times with different choices of  $M'$  in the error indicator for the greedy search. The result are shown in Figure 5.2.6.

The plots show the maximum error estimates for all parameters from the training set at each reduced basis extension step during the greedy search algorithm. Here, the lower black line corresponds to the error curve of the reference run (C) where the greedy extension algorithm is used with the “true” error  $\|u_{\text{red}}^K(\boldsymbol{\mu}) - u_h^K(\boldsymbol{\mu})\|_{\mathcal{W}_h}$  as an indicator. We observe, that, in general, all plots show an error decrease at a rate similar to the reference plot. Only the POD-ADAPTIVEGREEDY algorithm makes an exception — after the last adaptation of the training parameter set at about  $N = 60$ . The reason for this behavior can be the non-adaptive training parameter set used for the collateral reduced basis space. No matter what value has been chosen for  $M'$ , the runs show no qualitative deviation. Table 5.2.1 shows that the error reduction obtained with reduced basis spaces generated with greedy search algorithms based on the error estimator is of comparable quality to the “optimal” values of Table 5.2.1(C), while the offline time is reduced significantly. The effect is especially salient for small basis spaces generated with the PODEI-ADAPTIVEGREEDY algorithm. All this confirms our assumption that the estimator is a valid error indicator for the greedy search — already with a small number of extra collateral basis functions.

### 5.3. Example: Buckley–Leverett equation

As a third example, we consider a highly non-linear convection–diffusion problem, that we use primarily in order to validate the newly proposed time adaptive empirical interpolation algorithm described in Section 4.4.2.

Again, we want to solve for solutions  $u = u(x, t; \boldsymbol{\mu})$  satisfying the equations (3.1.6)–(3.1.9). Here, we consider a specialization, which can be seen as a preliminary step to the two phase flow problem in the next chapter. We choose a Buckley–Leverett type problem in two space dimensions modeling two–phase flow, where the velocity field is prescribed. The unknown can be

interpreted as the concentration of a single fluid, e.g. water. For more details on two phase flow models, we refer to the next Chapter 6.

We choose a rectangular domain  $\Omega := [0, 1]^2$  and  $T_{\max} := 1$ . The initial data function is given by fixed rectangular concentration function, depending on one parameter  $c_{\text{low}} > 0$  given by

$$u_0(\boldsymbol{\mu}) = c_{\text{low}} + (1 - c_{\text{low}})\chi_{[0.2, 0.6] \times [0.25, 0.75]}. \quad (5.11)$$

Furthermore, we define the velocity vector

$$\mathbf{v}(u; \boldsymbol{\mu}) = (0, 1)^t f(u; \boldsymbol{\mu}) \quad (5.12)$$

and the diffusion coefficient

$$d(u; \boldsymbol{\mu}) = KD(s; \boldsymbol{\mu}). \quad (5.13)$$

Here

$$f(u; \boldsymbol{\mu}) = \frac{u^3}{5} \cdot \left( \frac{u^3}{5} \right) + \frac{(1-u)^{3-1}}{3} \quad (5.14)$$

denotes the fractional flow rate, and

$$D(u; \boldsymbol{\mu}) = \frac{(1-u)^3}{3} f(u; \boldsymbol{\mu}) p'_c(u; \boldsymbol{\mu}) \quad (5.15)$$

is the capillary diffusion for a capillary pressure

$$p_c(u; \boldsymbol{\mu}) = u^{-\lambda}. \quad (5.16)$$

The variable parameters are chosen as  $\boldsymbol{\mu} := (K, c_{\text{low}}, \lambda)$  and the parameter space is given by

$$\mathcal{M} := [1, 2] \times [0, 0.1] \times [0.1, 0.4]. \quad (5.17)$$

At the boundary of the domain a Dirichlet condition applies with  $u_{\mathcal{N}_{\text{dir}}}(\boldsymbol{\mu}) = c_{\text{low}}$ .

Like all our numerical examples, the problem is discretized with a standard finite volume scheme comprising an explicitly computed Engquist–Osher flux for the convective terms and an implicit discretization of the diffusive terms. The underlying grid is very coarse and has a dimension of  $H = 25 \times 25$  grid cells. The time interval  $[0, T_{\max}]$  is discretized by 60 uniformly distributed time steps.

Figure 5.3.1 illustrates solution snapshots for two different parameters with different diffusion levels  $K = 0$  respectively  $K = 2$ . The cross-section plots in the last column show the expected behavior of combinations of rarefaction waves and smoothed shocks.

**5.3.1. Offline phase.** In order to assess the effects of the adaptation algorithms, the reduced basis algorithms are run three times. We always use the typical subsequent combination of empirical interpolation reduced basis

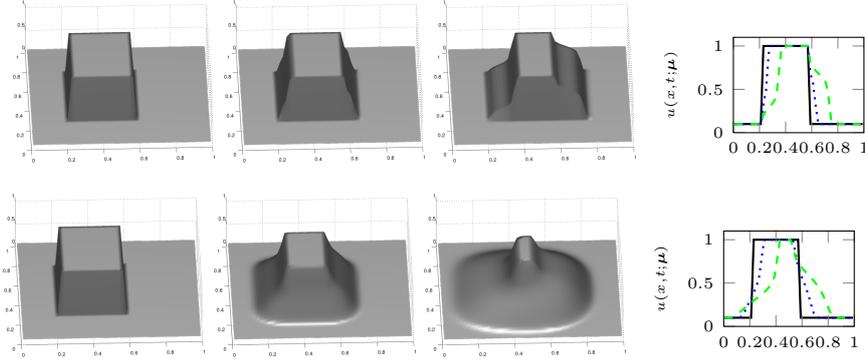


FIGURE 5.3.1. Detailed simulation solution snapshots at time instants  $t = 0.0$  (first column),  $t = 0.1$  (second column),  $t = 0.3$  (third column) and for different parameters  $\boldsymbol{\mu} = (0, 0.1, 0.4)$  (first row) and  $\boldsymbol{\mu} = (2, 0.1, 0.4)$  (second row). The last column shows the reduced solution on cross-sections at  $y = 0.5$  for the time instants  $t = 0$  (solid line),  $t = 0.1$  (dotted line),  $t = 0.3$  (dashed line).

adaptation	no. of bases	$\phi$ -dim(CRB)	offline time[h]	$\phi$ -runtime[s]	max. error
no	1	350	1.47	6.79	$5.88 \cdot 10^{-4}$
yes, $c_{\min} = 5$	11	223.09	2.08	4.06	$5.80 \cdot 10^{-4}$
yes, $c_{\min} = 1$	26	198.42	8.40	3.38	$5.75 \cdot 10^{-4}$

TABLE 5.3.1. Comparison of the number of bases, the reduced basis sizes averaged over sub-intervals, offline time, averaged online reduced simulation times and maximum errors for non-adaptive and adaptive runs with threshold  $c_{\min} = 5$ , and  $c_{\min} = 1$ . The average online run-times and maximum errors are obtained from 20 simulations with randomly selected parameters  $\boldsymbol{\mu}$ .

generation. Once we used the EI-GREEDY + POD-ADAPTIVEGREEDY combination and the other two times the EI-TIMEADAPTIVE algorithm for the generation of the collateral reduced basis, followed by a POD-ADAPTIVEGREEDY algorithm for the efficient search for reduced basis functions. Here, we used two different thresholds  $c_{\min} = 1, 5$ , which bound the minimal time interval size from below. The results concerning reduced basis sizes, offline and reduced simulation time, are summarized in Table 5.3.1.

In order to assure that the generated reduced basis leads to equally small reduction errors for all parameters of the parameter space, the parameter training set is being adapted with a validation set of randomly chosen parameters as described in Section 4.4.1. In the test runs, after three refinement

steps the training parameter set comprises 255 elements, and the chosen validation ratio of 1.4 is assured after the maximum error for the training parameters has fallen below the targeted level of  $5 \cdot 10^{-4}$ . The target interpolation error for the empirical interpolation was set to  $10^{-6}$  in all runs. This error is reached with an average number of 198 respectively 223 basis functions in the adaptive cases, and 350 basis functions without adaptation. In the adaptive runs, the time interval has been decomposed into 11 respectively 26 sub-intervals (c.f. Figure 5.3.2(a)+(b)). Figure 5.3.2(c) illustrates the error decrease during the generation of the reduced spaces for selected time intervals (dashed lines) for the run with  $c_{\min} = 1$ . It can be observed that the slopes for the error graphs are much steeper than in the non-adaptive case illustrated with a dashed line. Because of the larger variation of the solutions for larger time steps, however, the basis on the last interval  $[0.29, 0.30]$  still shows the slowest error decrease. Figure 5.3.2(a+b) show that for both adaptive runs the bases dimensions for all intervals stay significantly below the non-adaptive basis size of 350.

Thus, we observed that the adaptive search in the time domain can lead to faster reduced simulations. However, the costs of 26 generated basis spaces for an average dimension reduction by a factor of approximately 0.56 turned out to be very expensive. So, the general applicability of the algorithm is proven, but we suggest to combine the time domain search with a parameter domain search or a further time domain split in the reduced basis space as in [21] to obtain a further improvement of the method.

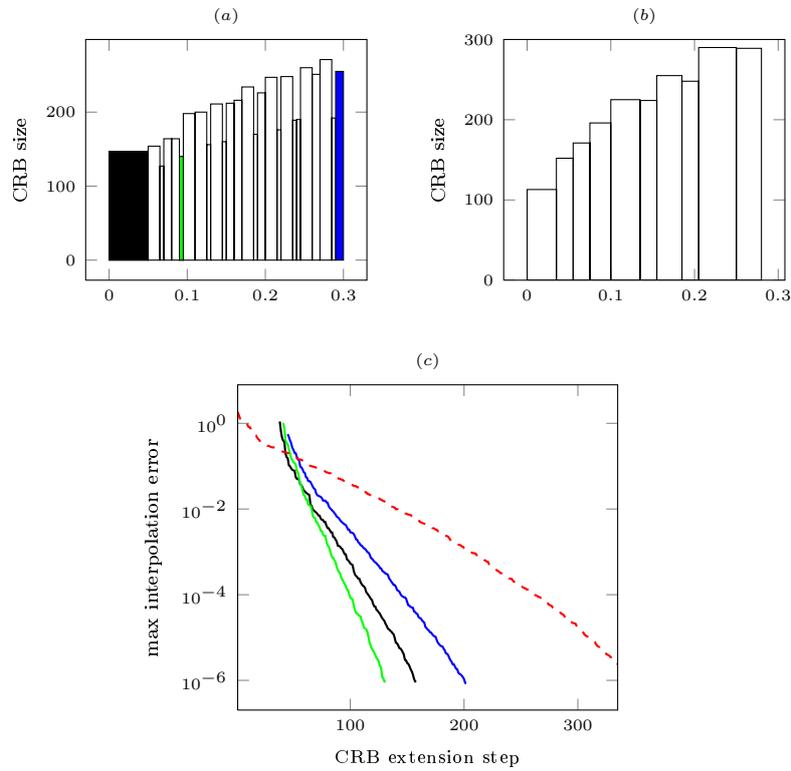


FIGURE 5.3.2. Illustration of basis sizes on time intervals after adaptation with (a)  $c_{\min} = 1$  and (b)  $c_{\min} = 5$ . Plot (c) illustrates the error decrease during generation of bases on three intervals marked with the same color in plot (a). The dashed line graph shows the slower decrease for a single basis without adaptation.



## CHAPTER 6

### Immiscible two phase flow in porous media

The reduced basis method has been applied to different problem classes like stationary, linear problems with affine dependence on the parameter on the one end (e.g. [61, 36]) and lately also to non-linear, non-stationary systems of partial differential equations on the other end (c.f. [10], [15]). In this chapter, we want to deal with the latter problem class. We apply the empirical operator interpolation and the reduced basis framework as developed in the previous chapter to a system of two non-linear coupled partial differential equations, handling the model reduction of the function spaces for all physical unknowns separately. The equations considered model the flow of two immiscible, incompressible fluids in a porous medium. For example, this can be used in reservoir simulation, in order to simulate the production of oil in a well.

In section 6.1, we introduce the general model for immiscible, incompressible two phase flow in porous media and derive the global pressure formulation from this.

The analysis and modeling of the high-dimensional numerical scheme is mainly taken from [57]. We used the proposed finite volume scheme in order to discretize the two phase flow equation, as described in Section 6.2. The scheme comprises two non-linear operators and we show in Section 6.3.1 how they can be empirically interpolated. The reduced basis space is simply obtained by a proper orthogonal decomposition, because we pass on a parametrization of the problem in our experiments. In Section 6.3 the reduced scheme derived by a Galerkin projection onto the reduced basis space and the offline/online decomposition of this scheme is described. We conclude this chapter with experiments for the reduced basis generation and compare the run-times of high-dimensional and reduced simulations.

#### 6.1. Global pressure formulation

In this section, we introduce the general idea on how to model two phase flow in porous media with a system of equations for conservation of mass and the Darcy flow of the two phases. Furthermore, we shortly derive the equivalent so-called *global pressure formulation* which is preferred in numerical

simulations, as the number of unknowns and equations in this formulation is minimized.

The problem is situated on a bounded spatial domain  $\Omega \subset \mathbb{R}^2$  modeling a porous medium and the time interval  $[0, T_{\max}]$ . The saturation of the two fluid are denoted by  $s_w, s_o$  and measure the ratio by which the two phases fill the void space of the porous medium in a point  $x \in \Omega$ .

Thus, they should add up to one

$$s_w + s_o = 1, \quad (6.1)$$

Therefor, it suffices to consider the saturation of the wetting phase  $s := s_w$  only. The flow can be described by a combination of the mass conservation law and Darcy flow

$$\frac{\partial (\phi \rho_\alpha s_\alpha)}{\partial t} = \nabla \cdot (\rho_\alpha \mathbf{u}_\alpha) + q_\alpha \quad \text{in } \Omega \times [0, T_{\max}] \quad (6.2)$$

$$\mathbf{u}_\alpha = -\frac{k_\alpha}{\mu_\alpha} \mathbf{k} (\nabla p_\alpha - \rho_\alpha \mathbf{g} \nabla z) \quad \text{in } \Omega \times [0, T_{\max}] \quad (6.3)$$

for  $\alpha = w, o$ . Here the variables  $\phi, \rho_\alpha, \mathbf{u}_\alpha, \mathbf{k}, k_\alpha, \mu_\alpha, q_\alpha$  and  $p_\alpha$  denote the porosity, the fluid density, Darcy's velocity, the absolute permeability, the relative permeability, viscosity, sources and pressure for each phase. The constant  $\mathbf{g}$  models gravitational effects. Intrinsically the physics for both phases are coupled by the capillary pressure which can be expressed as the pressure difference

$$p_c = p_o - p_w. \quad (6.4)$$

One can empirically determine functions for  $p_c$  with respect to the saturation  $s$ . These curves can be modeled for example by the Brooks-Corey [5] or the van Genuchten-Mualem model [72].

The number of equations in (6.1.2)-(6.1.3) can be reduced by adding the two equations in (6.1.2) for  $\alpha = o, w$  resulting in  $\partial_t (\phi (s_o + s_w)) = \partial_t (\phi \cdot 1)$ . On the right hand side, because of (6.1.1), one obtains

$$\nabla \cdot \mathbf{u} = -\frac{\partial \phi}{\partial t} + \frac{q_w}{\rho_w} + \frac{q_o}{\rho_o} \quad (6.5)$$

for a *global velocity*  $\mathbf{u} := u_w + u_o$ . It is noteworthy, that the "true" velocity fields  $\mathbf{u}_w$  and  $\mathbf{u}_o$  can be reconstructed from the global velocity, e.g.

$$\mathbf{u}_w = f(s) \mathbf{u} + \mathbf{k} \frac{k_o}{\mu_o} f(s) (\nabla (p_c(s)) + (\rho_w - \rho_o) \mathbf{g} \nabla z), \quad (6.6)$$

with fractional flow rates  $f = f_w := (M \mu_w)^{-1} k_w$ ,  $f_o = (M \mu_o)^{-1} k_o$  and the total mobility  $M = \mu_w^{-1} k_w + \mu_o^{-1} k_o$ . Corresponding to this global velocity

field, a *global pressure* is defined by

$$\psi := p_o + \int^{p_c(s)} f(p_c^{-1}(\xi)) d\xi, \quad (6.7)$$

such that a Darcy flow relation between the two unknowns can be obtained by

$$\mathbf{u} = -\mathbf{k}\lambda(\nabla\psi - (\rho_w f + \rho_o f_o) \mathbf{g}\nabla z). \quad (6.8)$$

For the derivation of (6.1.8) the equations  $s = p_c^{-1}(p_o - p_w)$  and (6.1.3) have been used.

Usually one can assume that the porosity of the medium does not change over time, such that  $\partial_t \phi = 0$ . If we further neglect the gravity ( $\mathbf{g} = 0$ ), we end up with the following system of equations for the unknowns  $(s, \mathbf{u}, \psi)$

$$\phi \frac{\partial s}{\partial t} + \nabla \cdot (f(s)\mathbf{u} - \mathbf{K}(s)\nabla s) = q_1, \quad \text{in } \Omega \times [0, T], \quad (6.9)$$

$$\nabla \cdot \mathbf{u} = q_1 + q_2, \quad \text{in } \Omega \times [0, T], \quad (6.10)$$

$$\mathbf{u} = -\mathbf{k}M(s)\nabla\psi, \quad \text{in } \Omega \times [0, T], \quad (6.11)$$

where the diffusion coefficient is given by

$$\mathbf{K} : [0, 1] \rightarrow \mathbb{R}, s \mapsto \mathbf{k}(s)k_w(s)f(s)p'_c(s), \quad (6.12)$$

and the source terms  $q_1 := \frac{q_w}{\rho_w}$  and  $q_2 := \frac{q_o}{\rho_o}$ , respectively.

In order to close the system (6.1.9-6.1.11), we introduce Neumann boundary conditions  $\nabla s \cdot \mathbf{n} = 0$  and  $\nabla \psi \cdot \mathbf{n} = 0$  on  $\partial\Omega \times [0, T]$ , prescribe an initial saturation

$$s(\cdot, 0) = s_0 \quad \text{in } \Omega \quad (6.13)$$

and define the global pressure to have zero mean

$$\int_{\Omega} \psi(x, \cdot) dx = 0 \quad \text{in } \Omega \times [0, T], \quad (6.14)$$

as it is defined up to a constant only, otherwise. For the first test of a reduced scheme presented in this work, we refrain from a parametrization of the problem.

In order to fulfill the prerequisites of the two-phase problem as it is stated in [57], the source terms must be of the form

$$q_1(s) = c\bar{q} - s\underline{q} \quad \text{and} \quad q_2(s) = (1-c)\bar{q} - (1-s)\underline{q} \quad (6.15)$$

modeling injection or production wells, where  $c \in [0, 1]$  is an injection constant.

For details on the derivation of the global pressure formulation of two-phase flow problems, we refer to [18].

## 6.2. Finite volume discretization

In this section, we summarize the fully implicit finite volume scheme from [57] reformulated in a notation suitable for the model reduction in the following sections. First, we need to introduce a new admissible mesh on  $\Omega$ , on which we can store vector field information, and two finite volume function spaces defined on the grid, respectively its associated skeleton.

**Definition 6.2.1** (Numerical grid). *Let  $\mathcal{T} := \{e_i\}_{i=1}^H$  denote a numerical grid consisting of  $H$  disjoint and convex control volumes, a family of faces  $\mathcal{F}$  with  $\sigma \subset \bar{\Omega}$  for all  $\sigma \in \mathcal{F}$  and a family of control volume centers  $\{x_i\}_{i=1}^H$  with  $x_i \in e_i$  for  $i = 1, \dots, H$ . Further properties of the mesh are:*

- (1)  $\cup_{i=1}^H \bar{e}_i = \bar{\Omega}$ ,
- (2) For any face  $\sigma \in \mathcal{F}$ , we denote by  $x_\sigma$  its barycenter, and by  $m(\sigma)$  its one dimensional Hausdorff measure.
- (3) For any  $e, f \in \mathcal{T}$  with  $e \neq f$ , either the face measure  $m(\bar{e} \cap \bar{f}) = 0$  or  $\bar{e} \cap \bar{f} = \bar{\sigma}$  for a  $\sigma \in \mathcal{F}$ . The set of neighboring control volumes is denoted by  $\mathcal{N}(e) := \{f \in \mathcal{T} \mid m(\bar{e} \cap \bar{f}) \neq 0\}$ .
- (4) For any cell  $e \in \mathcal{T}$  its boundary is given by a set of faces  $\mathcal{F}(e) \subset \mathcal{F}$ , i.e.  $\partial e = \cup_{\sigma \in \mathcal{F}(e)} \bar{\sigma}$ .
- (5) The connection of two points  $x_i, x_j$  is orthogonal to the face  $\sigma_{ij} := \bar{e}_i \cap \bar{e}_j$  and its length is denoted by  $d_{ij} := |x_i - x_j|$ . The length of the connection from  $x_i$  up to the intersection with the face is denoted by  $d_{i,\sigma}$ .

Furthermore, we define the mesh skeleton by

$$\mathcal{E} := \cup_{\sigma \in \mathcal{F}} \bar{\sigma}. \quad (6.16)$$

**Definition 6.2.2** (Function spaces). *Given an admissible mesh on  $\Omega$ , we define the two discrete Hilbert spaces  $\mathcal{W}_h^T$  and  $\mathcal{W}_h^E$ . These spaces are spanned by basis functions  $\{\chi_i\}_{i=1}^H$  and  $\{\chi_\sigma\}_{\sigma \in \mathcal{E}}$  which are each constant and equal to one on one grid control volume and face of the mesh, respectively. The discrete functions in the finite volume function spaces are then piecewise constant, and functions  $f^T \in \mathcal{W}_h^T, f^E \in \mathcal{W}_h^E$  can be identified by their degrees of freedom*

$$f_{h,i}^T = \tau_i^T(f_h^T) := f_h^T(x_i) \quad \text{and} \quad (6.17)$$

$$f_{h,\sigma}^E = \tau_\sigma^E(f_h^E) := f_h^E(x_\sigma), \quad (6.18)$$

with the Dof functionals  $\Sigma_h^{\mathcal{T}} := \{\tau_i\}_{i=1}^H$  and  $\Sigma_h^{\mathcal{E}} := \{\tau_\sigma^{\mathcal{E}}\}_{\sigma \in \mathcal{E}}$ . The spaces are equipped with scalar products

$$\langle u_h, v_h \rangle_{\mathcal{W}_h^{\mathcal{T}}} := \int_{\Omega} u_h v_h \quad \text{and} \quad (6.19)$$

$$\langle u_h, v_h \rangle_{\mathcal{W}_h^{\mathcal{E}}} := \int_{\mathcal{S}} u_h v_h. \quad (6.20)$$

Furthermore, we define the projection operator  $\mathcal{P}_h^{\mathcal{T}} : L^2(\Omega) \rightarrow \mathcal{W}_h^{\mathcal{T}}$  by

$$\tau_i(\mathcal{P}_h^{\mathcal{T}}[u]) := \frac{1}{m(e_i)} \int_{e_i} u \quad (6.21)$$

with  $m(e_i)$  being the measure of a control volume  $e_i$  for  $i = 1, \dots, H$ .

Now, in order to discretize equations (6.1.9)-(6.1.11), we define two non-linear finite volume operators

$$\mathcal{L}_h^s : \mathcal{W}_h^{\mathcal{T}} \times \mathcal{W}_h^{\mathcal{E}} \rightarrow \mathcal{W}_h^{\mathcal{T}}, \quad (6.22)$$

$$\mathcal{L}_h^u : \mathcal{W}_h^{\mathcal{T}} \times \mathcal{W}_h^{\mathcal{T}} \rightarrow \mathcal{W}_h^{\mathcal{E}} \quad (6.23)$$

and one linear operator

$$\mathcal{L}_h^\psi : \mathcal{W}_h^{\mathcal{E}} \rightarrow \mathcal{W}_h^{\mathcal{T}} \quad (6.24)$$

discretizing the equations (6.1.9)-(6.1.11).

The saturation operator is given Dof-wise by

$$\begin{aligned} (\mathcal{L}_h^s[s_h, \mathbf{u}_h])_i &= \sum_{\sigma \in \mathcal{E}(e_i)} g_\sigma(\mathbf{u}_h, s_h) \\ &\quad - \sum_{e_j \in \mathcal{N}(e_i)} \{\mathbf{K}(s_h)\}_{ij} \frac{m(\sigma_{ij})}{d_{ij}} (s_{h,j} - s_{h,i}) \end{aligned} \quad (6.25)$$

where

$$\{f_h\}_{ij} := \frac{d_{ij}}{\frac{d_{i,\sigma_{ij}}}{f_{h,i}} + \frac{d_{j,\sigma_{ij}}}{f_{h,j}}} \quad (6.26)$$

is the harmonic mean on the edge  $\sigma_{ij}$  and  $g_\sigma : \mathcal{W}_h^{\mathcal{E}} \times \mathcal{W}_h^{\mathcal{T}} \rightarrow \mathbb{R}$  is an upwind finite volume flux given as

$$g_{\sigma_{ij}}(\mathbf{u}_h, s_h) := \begin{cases} \mathbf{u}_{h,\sigma_{ij}} f(s_{h,i}) & \text{if } \mathbf{u}_{h,\sigma_{ij}} > 0 \\ \mathbf{u}_{h,\sigma_{ij}} f(s_{h,j}) & \text{if } \mathbf{u}_{h,\sigma_{ij}} \leq 0. \end{cases} \quad (6.27)$$

The velocity operator is defined by

$$(\mathcal{L}_h^u[s_h, \psi_h])_{ij} = \{\mathbf{k}M(s_h)\}_{\sigma_{ij}} \frac{m(\sigma_{ij})}{d_{ij}} (\psi_{h,j} - \psi_{h,i}) \quad (6.28)$$

and the pressure operator by

$$\left(\mathcal{L}_h^\psi[\mathbf{u}_h]\right)_i = \sum_{\sigma \in \mathcal{E}(e_i)} \mathbf{u}_{h,\sigma}. \quad (6.29)$$

**Definition 6.2.3** (Finite volume scheme for two phase flow). *Given a discretization of the time interval  $[0, T]$  by a sequence of  $K+1$  strictly increasing time instances  $t^k := k\Delta t$ ,  $k = 0, \dots, K$  with a global time step size  $\Delta t > 0$ , we are searching for discrete solutions*

$$u_h^k := (s_h^k, \mathbf{u}_h^k, \psi_h^k) \in \mathcal{W}_h := \mathcal{W}_h^T \times \mathcal{W}_h^\mathcal{E} \times \mathcal{W}_h^T. \quad (6.30)$$

These are computed by an initial projection

$$(s_h^0, \mathbf{u}_h^0, \psi_h^0) = (\mathcal{P}_h^T[s_0], 0, 0) \quad (6.31)$$

and subsequently solving the equation

$$\mathcal{L}_h \left[ s_h^{k+1}, \mathbf{u}_h^{k+1}, \psi_h^{k+1} \right] = 0 \quad (6.32)$$

with the Newton–Raphson method. In each Newton step, we solve for the defect  $\delta^{k+1, \nu+1}$  in

$$\mathbf{D}\mathcal{L}_h|_{u_h^{k+1, \nu}} \left[ \delta^{k+1, \nu+1} \right] = -\mathcal{L}_h \left[ u_h^{k+1, \nu} \right], \quad (6.33)$$

where

$$u_h^{k+1, 0} := u^k \quad \text{and} \quad (6.34)$$

$$u_h^{k+1, \nu+1} := u_h^{k+1, \nu} + \delta^{k+1, \nu+1} \quad (6.35)$$

define the updates in each Newton step, and the solution at each time instance  $t^k$  is given by  $u_h^{k+1} := u_h^{k+1, \nu_{\max}^k}$ . The last Newton step index  $\nu_{\max}^k$  equals the smallest integer  $\nu$  satisfying

$$\left\| \mathcal{L}_h \left[ u_h^{k+1, \nu+1} \right] \right\|_{\mathcal{W}_h^s \times \mathcal{W}_h^u \times \mathcal{W}_h^\psi \times \mathbb{R}} \leq \varepsilon^{\text{New}}. \quad (6.36)$$

Here,  $\mathcal{L}_h[s_h^{k+1}, \mathbf{u}_h^{k+1}, \psi_h^{k+1}]$  evaluates to

$$\begin{pmatrix} \frac{1}{\Delta t} (s_h^{k+1} - s_h^k) - \mathcal{L}_h^s [s_h^{k+1}, \mathbf{u}_h^{k+1}] - \mathcal{P}_h^T [q_1] \\ \mathcal{L}_h^u [s_h^{k+1}, \psi_h^{k+1}] - \mathbf{u}_h^{k+1} \\ \mathcal{L}_h^\psi [\mathbf{u}_h^{k+1}] - \mathcal{P}_h^T [q_1 + q_2] \\ \int_{\Omega} p_h^{k+1} \end{pmatrix}, \quad (6.37)$$

with the row entries corresponding to the discretizations of equations (6.1.9)–(6.1.11) and (6.1.14) with operators defined in (6.2.10), (6.2.13) and (6.2.14),

and the norm  $\|\cdot\|_{\mathcal{W}_h^s \times \mathcal{W}_h^{\mathbf{u}} \times \mathcal{W}_h^\psi \times \mathbb{R}}$  given by

$$\|(s_h, \mathbf{u}_h, \psi_h, c)^t\| = \|s_h\|_{\mathcal{W}_h^s} + \|\mathbf{u}_h\|_{\mathcal{W}_h^{\mathbf{u}}} + \|\psi_h\|_{\mathcal{W}_h^\psi} + |c|. \quad (6.38)$$

### 6.3. Reduced basis scheme

In this section, we want to show how to reduce the Newton scheme defined in Definition 6.2.3. This reduced basis scheme is decomposable into an *offline* phase with computations depending on the high dimension of the original function space  $\mathcal{W}_h$  and into an *online* phase with memory efficient and fast reduced simulations. In order to make this decomposition clear, we first define the necessary reduced basis spaces and empirical operator interpolants. Afterwards, we define the reduced basis scheme and analyze its computational complexity.

**6.3.1. Basis generation.** In the following, we assume the existence of three reduced basis spaces, one for saturation  $\mathcal{W}_{\text{red}}^s \subset \mathcal{W}_h^T$ , one for the pressure  $\mathcal{W}_{\text{red}}^\psi \subset \mathcal{W}_h^T$  and one for the velocity  $\mathcal{W}_{\text{red}}^{\mathbf{u}} \subset \mathcal{W}_h^{\mathcal{E}}$ . These shall be spanned by reduced bases  $\Phi^* := \{\varphi_n^*\}_{n=1}^{N_*}$  with  $* = \{s, \mathbf{u}, \psi\}$ .

Furthermore, the two non-linear operators need to be substituted by empirical interpolants. Therefor, we assume the existence of two empirical interpolations  $\mathcal{I}_{M_s}^s$  and  $\mathcal{I}_{M_{\mathbf{u}}}^{\mathbf{u}}$  for the empirical operator interpolation of the non-linear operators  $\mathcal{L}_h^s$  and  $\mathcal{L}_h^{\mathbf{u}}$ . The pressure operator  $\mathcal{L}_h^s$  does not need to be interpolated, as it is linear in  $\psi$ .

As we forgo parametrization in this problem, the reduced basis spaces shall be obtained by a POD of the “snapshots”  $s_h^{k,\nu}$ ,  $\mathbf{u}_h^{k,\nu}$  and  $\psi_h^{k,\nu}$  for all  $(k, \nu) \in \mathbf{T}$ , where

$$\mathbf{T} := \{(k, \nu) \in 0, \dots, K \times \mathbb{N}_+ \mid \nu \leq \nu_{\max}\}. \quad (6.39)$$

Hence, the entire trajectory of solution snapshots and the intermediate solutions generated by the Newton method are taken into account for basis generation. With the  $\text{POD}_N$  algorithm defined in Algorithm 4.1.1, the reduced bases are given by

$$\Phi_{N_*}^* := \text{POD}_{N_*} \left( \left\{ *_{h}^{k,\nu} \mid (k, \nu) \in \mathbf{T} \right\} \right) \quad (6.40)$$

for  $* = s, \mathbf{u}, \psi$ . Here the bases sizes  $N_s$ ,  $N_{\mathbf{u}}$  and  $N_\psi$  are selected, such that the smallest eigenvalue  $\lambda_{N_*}$  selected in the POD algorithm, is the only one below a given tolerance  $\varepsilon_{\text{tol}}^{\text{POD}}$ .

The empirical interpolation data for the two operators is obtained by the EI-GREEDY algorithm as described in Algorithm 4.2.3. Note, however, that the operators in our case are not parametrized, such that as for the greedy algorithms, only the time steps and intermediate Newton steps are used for

training of the empirical interpolation. For example, for the operator  $\mathcal{L}_h^s(\boldsymbol{\mu})$ , the training is done with operator evaluations

$$\left\{ \mathcal{L}_h^s \left[ s_h^{k,\nu}, \mathbf{u}_h^{k,\nu} \right] \right\}_{(k,\nu) \in \mathbf{T}}. \quad (6.41)$$

The resulting collateral reduced bases and empirical interpolation Dofs shall be denoted by

$$\begin{aligned} \boldsymbol{\xi}_{M_s}^s &:= \{\zeta^m\}_{m=1}^{M_s}, & \boldsymbol{\xi}_{M_u}^u &:= \{\zeta^m\}_{m=1}^{M_u}, \\ \Sigma_{M_s}^s &:= \{\tau_{s,m}^{EI}\}_{m=1}^{M_s} \subset \Sigma_{\mathcal{T}_h}^s, & \Sigma_{M_u}^u &:= \{\tau_{\mathbf{u},m}^{EI}\}_{m=1}^{M_u} \subset \Sigma_h^{\mathcal{E}}. \end{aligned} \quad (6.42)$$

**6.3.2. Reduced scheme.** The derivation of the reduced basis scheme is now straight-forward. With the help of reduced basis spaces defined above, we can generate the reduced operator

$$\mathcal{L}_{\text{red}} : \mathcal{W}_{\text{red}}^s \times \mathcal{W}_{\text{red}}^u \times \mathcal{W}_{\text{red}}^\psi \rightarrow \mathcal{W}_{\text{red}}^s \times \mathcal{W}_{\text{red}}^u \times \mathcal{W}_{\text{red}}^\psi \times \mathbb{R}. \quad (6.43)$$

The numerical scheme in Definition 6.2.3 can then be reduced by searching for reduced solutions

$$u_{\text{red}} := \left( s_{\text{red}}^k, \mathbf{u}_{\text{red}}^k, \psi_{\text{red}}^k \right) \in \mathcal{W}_{\text{red}}^s \times \mathcal{W}_{\text{red}}^u \times \mathcal{W}_{\text{red}}^\psi \quad (6.44)$$

for which in each time instance the residual of the reduced operator evaluation

$$\mathcal{L}_{\text{red}}[s_{\text{red}}^{k+1}, \mathbf{u}_{\text{red}}^{k+1}, \psi_{\text{red}}^{k+1}] \quad (6.45)$$

gets minimized. This operator is defined similar to its high-dimensional counterpart in (6.2.22) by

$$\left( \begin{array}{c} \frac{1}{\Delta t} \left( s_{\text{red}}^{k+1} - s_{\text{red}}^k \right) - \mathcal{L}_{\text{red}}^s \left[ s_{\text{red}}^{k+1}, \mathbf{u}_{\text{red}}^{k+1} \right] - \mathcal{P}_{\text{red}}^{\mathcal{T}}[q_1] \\ \mathcal{L}_{\text{red}}^u \left[ s_{\text{red}}^{k+1}, \psi_{\text{red}}^{k+1} \right] - \mathbf{u}_{\text{red}}^{k+1} \\ \mathcal{L}_{\text{red}}^\psi \left[ \mathbf{u}_{\text{red}}^{k+1} \right] - \mathcal{P}_{\text{red}}^{\mathcal{T}}[q_1 + q_2] \\ \int_{\Omega} p_{\text{red}}^{k+1} \end{array} \right), \quad (6.46)$$

where the operators  $\mathcal{L}_h^s$ ,  $\mathcal{L}_h^u$  and  $\mathcal{L}_h^\psi$  are replaced by reduced surrogates defined as

$$\mathcal{L}_{\text{red}}^s := \mathcal{P}_{\text{red}}^s \circ \mathcal{I}_{M_s}[\mathcal{L}_h^s], \quad (6.47)$$

$$\mathcal{L}_{\text{red}}^u := \mathcal{P}_{\text{red}}^u \circ \mathcal{I}_{M_u}[\mathcal{L}_h^u] \text{ and} \quad (6.48)$$

$$\mathcal{L}_{\text{red}}^\psi := \mathcal{P}_{\text{red}}^\psi \circ \mathcal{L}_h^\psi \quad (6.49)$$

with orthogonal projection operators  $\mathcal{P}_{\text{red}}^s : \mathcal{W}_h^s \rightarrow \mathcal{W}_{\text{red}}^s$ ,  $\mathcal{P}_{\text{red}}^\psi : \mathcal{W}_h^\psi \rightarrow \mathcal{W}_{\text{red}}^\psi$  and  $\mathcal{P}_{\text{red}}^u : \mathcal{W}_h^u \rightarrow \mathcal{W}_{\text{red}}^u$ .

For easier analysis of the computational costs during offline and online-phase, we want to formulate the above sketched scheme in a vector-valued form based on the few degrees of freedom of the reduced solutions.

**Definition 6.3.1** (Reduced two-phase flow scheme). *We assume a numerical scheme as defined in Definition 6.2.3 with operators  $\mathcal{L}_h^s$  and  $\mathcal{L}_h^u$  fulfilling an  $H$ -independent Dof dependence and two empirical interpolation operators  $\mathcal{I}_{M_s}^s$  and  $\mathcal{I}_{M_u}^u$  with associated collateral reduced bases  $\xi_{M_s}^s$ ,  $\xi_{M_u}^u$  and empirical interpolation Dofs  $\Sigma_{M_s}^s$  and  $\Sigma_{M_u}^u$ . Furthermore, there are three orthonormal reduced bases  $\Phi_{N_*}^*$  with  $*$  =  $s, u, \psi$ .*

*Then, we define the following numerical scheme, for sequentially expressing*

- the reduced solutions

$$s_{\text{red}}^k = \sum_{n=1}^{N_s} a_n^k \varphi_n^s, \quad \mathbf{u}_{\text{red}}^k = \sum_{n=1}^{N_u} b_n^k \varphi_n^u, \quad \psi_{\text{red}}^k = \sum_{n=1}^{N_\psi} c_n^k \varphi_n^\psi, \quad (6.50)$$

- the intermediate Newton step solutions

$$s_{\text{red}}^{k,\nu} = \sum_{n=1}^{N_s} a_n^{k,\nu} \varphi_n^s, \quad \mathbf{u}_{\text{red}}^k = \sum_{n=1}^{N_u} b_n^{k,\nu} \varphi_n^u, \quad \psi_{\text{red}}^k = \sum_{n=1}^{N_\psi} c_n^{k,\nu} \varphi_n^\psi, \quad (6.51)$$

- and the Newton step defects

$$\sum_{n=1}^{N_s} \tilde{a}_n^{k,\nu} \varphi_n^s, \quad \sum_{n=1}^{N_u} \tilde{b}_n^{k,\nu} \varphi_n^u, \quad \sum_{n=1}^{N_\psi} \tilde{c}_n^{k,\nu} \varphi_n^\psi \quad (6.52)$$

by computing the coefficient vectors

$$\mathbf{a}^k := \left( a_n^k \right)_{n=1}^{N_s}, \quad \mathbf{b}^k := \left( b_n^k \right)_{n=1}^{N_u}, \quad \mathbf{c}^k := \left( c_n^k \right)_{n=1}^{N_\psi}, \quad (6.53)$$

$$\mathbf{a}^{k,\nu} := \left( a_n^{k,\nu} \right)_{n=1}^{N_s}, \quad \mathbf{b}^{k,\nu} := \left( b_n^{k,\nu} \right)_{n=1}^{N_u}, \quad \mathbf{c}^{k,\nu} := \left( c_n^{k,\nu} \right)_{n=1}^{N_\psi}, \quad (6.54)$$

$$\tilde{\mathbf{a}}^{k,\nu} := \left( \tilde{a}_n^{k,\nu} \right)_{n=1}^{N_s}, \quad \tilde{\mathbf{b}}^{k,\nu} := \left( \tilde{b}_n^{k,\nu} \right)_{n=1}^{N_u}, \quad \tilde{\mathbf{c}}^{k,\nu} := \left( \tilde{c}_n^{k,\nu} \right)_{n=1}^{N_\psi}, \quad (6.55)$$

for  $(k, \nu) \in \mathbf{T}$ .

The initial vector is simply obtained by projection onto the reduced basis spaces. As all bases are orthonormal, the following applies:

$$\mathbf{a}^0 := \left( \langle s_0, \varphi_1^s \rangle_{\mathcal{W}_h^s}, \dots, \langle s_0, \varphi_{N_s}^s \rangle_{\mathcal{W}_h^s} \right)^t \quad (6.56)$$

$$\mathbf{b}^0 := 0 \quad \mathbf{c}^0 := 0. \quad (6.57)$$

Then, for each time index  $k = 0, \dots, K-1$ , we compute Newton iterations by finding the defects  $(\tilde{\mathbf{a}}^{k,\nu}, \tilde{\mathbf{b}}^{k,\nu}, \tilde{\mathbf{c}}^{k,\nu})^t$  and solving for  $\nu = 0, \dots, \nu_{\max}(k) - 1$

the equation

$$\begin{aligned} \tilde{\mathbf{a}}^{k+1,\nu+1} + \Delta t \mathbf{J} \left( \mathbf{a}^{k+1,\nu}, \mathbf{b}^{k+1,\nu}, \mathbf{c}^{k+1,\nu} \right) \left[ \tilde{\mathbf{a}}^{k+1,\nu+1}, \tilde{\mathbf{b}}^{k+1,\nu+1}, \tilde{\mathbf{c}}^{k+1,\nu+1} \right] \\ = -\mathbf{R}^k \left( \mathbf{a}^{k+1,\nu}, \mathbf{b}^{k+1,\nu}, \mathbf{c}^{k+1,\nu} \right) \end{aligned} \quad (6.58)$$

and computing the residual

$$\mathbf{R}^{k+1,\nu+1} := \mathbf{R}^k \left( \mathbf{a}^{k+1,\nu}, \mathbf{b}^{k+1,\nu}, \mathbf{c}^{k+1,\nu} \right) \quad (6.59)$$

with updates

$$*^{k+1,0} := *^k, \quad *^{k+1,\nu+1} := *^{k+1,\nu} + \tilde{*}^{k+1,\nu+1}, \quad (6.60)$$

$$*^{k+1} := *^{k+1,\nu_{\max}(k)}, \quad (6.61)$$

for  $* = \mathbf{a}, \mathbf{b}, \mathbf{c}$ . The number of Newton steps  $\nu_{\max}(k)$  in each time step is chosen as the smallest integer  $\nu$ , such that the residual norm drops below the specified tolerance for the Newton scheme, i.e.

$$\left\| \mathbf{R}^{k+1,\nu+1} \right\| \leq \varepsilon^{\text{New}}. \quad (6.62)$$

Here, the right hand side vector  $\mathbf{R}^k(\mathbf{a}, \mathbf{b}, \mathbf{c}) \in \mathbb{R}^N$ ,  $N := N_s + N_{\mathbf{u}} + N_{\psi}$  is defined by

$$\begin{pmatrix} \mathbf{a} - \mathbf{a}^k + \mathbf{C}^s \mathbf{l}^s(\mathbf{a}, \mathbf{b}) - \mathbf{z}^s \\ \mathbf{C}^u \mathbf{l}^u(\mathbf{a}, \mathbf{c}) - \mathbf{b} \\ \mathbf{L}^{\psi} \mathbf{b} - \mathbf{z}^{\psi} \\ (\mathbf{i}^{\psi})^t \mathbf{c} \end{pmatrix} \quad (6.63)$$

and the Jacobian  $\mathbf{J}(\mathbf{a}, \mathbf{b}, \mathbf{c}) \in \mathbb{R}^{N+1 \times N}$  is defined by

$$\begin{pmatrix} \mathbf{C}^s \mathbf{l}^{s,s}(\mathbf{a}, \mathbf{b}) & \mathbf{C}^s \mathbf{l}^{s,u}(\mathbf{a}, \mathbf{b}) & 0 \\ \mathbf{C}^u \mathbf{l}^{u,s}(\mathbf{a}, \mathbf{c}) & -\text{Id} & \mathbf{C}^u \mathbf{l}^{u,\psi}(\mathbf{a}, \mathbf{c}) \\ 0 & \mathbf{L}^{\psi} & 0 \\ 0 & 0 & (\mathbf{i}^{\psi})^t \end{pmatrix}. \quad (6.64)$$

The matrices and vectors in the above equations can be split into offline and online components. During the offline phase, the matrices  $\mathbf{C}^s \in \mathbb{R}^{N_s \times M_s}$ ,  $\mathbf{C}^u \in \mathbb{R}^{N_u \times M_u}$ ,  $\mathbf{L}^{\psi} \in \mathbb{R}^{N_u \times N_{\psi}}$ , and the vectors  $\mathbf{z}^s \in \mathbb{R}^{N_s}$ ,  $\mathbf{z}^{\psi} \in \mathbb{R}^{N_{\psi}}$ ,  $\mathbf{i}^{\psi} \in \mathbb{R}^{N_{\psi}}$  need to be computed, each component-wise defined by

$$(\mathbf{C}^s)_{n,m} := \langle \xi_m^s, \varphi_n^s \rangle_{\mathcal{W}_h^s}, \quad (6.65)$$

$$(\mathbf{C}^u)_{n,m} := \langle \xi_m^u, \varphi_n^u \rangle_{\mathcal{W}_h^u}, \quad (6.66)$$

$$\left( \mathbf{L}^{\psi} \right)_{nn'} := \left\langle \varphi_{n'}^u, \left[ \varphi_n^{\psi} \right] \right\rangle_{\mathcal{W}_h^{\psi}}, \quad (6.67)$$

$$(\mathbf{z}^s)_n := \langle q_1, \varphi_n^s \rangle_{\mathcal{W}_h^s}, \quad (6.68)$$

$$\left( \mathbf{z}^{\psi} \right)_n := \left\langle q_1 + q_2, \varphi_n^{\psi} \right\rangle_{\mathcal{W}_h^{\psi}} \quad (6.69)$$

and

$$\left(\mathbf{i}^\psi\right)_n := \int_{\Omega} \varphi_n^\psi. \quad (6.70)$$

In (6.3.29), the jump operator  $[\cdot] : \mathcal{W}_h^T \rightarrow \mathcal{W}_h^\mathcal{E}$  given by  $[u_h]_{\sigma_{ij}} := u_{h,i} - u_{h,j}$  for all  $\sigma_{ij} \in \mathcal{F}$  is used. It measures the jump of a finite volume function over the mesh faces.

During the online-phase, the Dof-wise operator evaluations  $\mathbf{I}^* \in \mathbb{R}^{M^*}$  for  $* = s, \mathbf{u}$ ,  $\mathbf{I}^{s,*} \in \mathbb{R}^{M_s \times N^*}$  for  $* = s, \mathbf{u}$  and  $\mathbf{I}^{\mathbf{u},*} \in \mathbb{R}^{M_{\mathbf{u}} \times N^*}$  for  $* = s, \psi$ , need to be computed. They are Dof-wise defined by

$$\left(\mathbf{I}^s(\mathbf{a}, \mathbf{b})\right)_m := \tau_{s,m}^{EI} \left( \mathcal{L}_h^s \left[ \sum_{i=1}^{N_s} a_i \varphi_i^s, \sum_{i=1}^{N_{\mathbf{u}}} b_i \varphi_i^{\mathbf{u}} \right] \right), \quad (6.71)$$

$$\left(\mathbf{I}^{\mathbf{u}}(\mathbf{a}, \mathbf{c})\right)_m := \tau_{s,m}^{EI} \left( \mathcal{L}_h^{\mathbf{u}} \left[ \sum_{i=1}^{N_s} a_i \varphi_i^s, \sum_{i=1}^{N_{\mathbf{u}}} c_i \varphi_i^\psi \right] \right), \quad (6.72)$$

$$\left(\mathbf{I}^{s,s}(\mathbf{a}, \mathbf{b})\right)_{mn} := \sum_{j=1}^H \frac{\partial}{\partial \chi_j^s} \tau_{s,m}^{EI} \left( \mathcal{L}_h^s \left[ \sum_{i=1}^{N_s} a_i \varphi_i^s, \sum_{i=1}^{N_{\mathbf{u}}} b_i \varphi_i^{\mathbf{u}} \right] \right) \tau_j(\varphi_n^s), \quad (6.73)$$

$$\left(\mathbf{I}^{s,\mathbf{u}}(\mathbf{a}, \mathbf{b})\right)_{mn} := \sum_{\sigma \in \mathcal{E}} \frac{\partial}{\partial \chi_\sigma^{\mathbf{u}}} \tau_{s,m}^{EI} \left( \mathcal{L}_h^s \left[ \sum_{i=1}^{N_s} a_i \varphi_i^s, \sum_{i=1}^{N_{\mathbf{u}}} b_i \varphi_i^{\mathbf{u}} \right] \right) \tau_\sigma(\varphi_n^{\mathbf{u}}), \quad (6.74)$$

$$\left(\mathbf{I}^{\mathbf{u},s}(\mathbf{a}, \mathbf{c})\right)_{mn} := \sum_{j=1}^H \frac{\partial}{\partial \chi_j^s} \tau_{\mathbf{u},m}^{EI} \left( \mathcal{L}_h^{\mathbf{u}} \left[ \sum_{i=1}^{N_s} a_i \varphi_i^s, \sum_{i=1}^{N_\psi} c_i \varphi_i^\psi \right] \right) \tau_j(\varphi_n^s), \quad (6.75)$$

$$\left(\mathbf{I}^{\mathbf{u},\psi}(\mathbf{a}, \mathbf{c})\right)_{mn} := \sum_{j=1}^H \frac{\partial}{\partial \chi_j^\psi} \tau_{\mathbf{u},m}^{EI} \left( \mathcal{L}_h^{\mathbf{u}} \left[ \sum_{i=1}^{N_s} a_i \varphi_i^s, \sum_{i=1}^{N_\psi} c_i \varphi_i^\psi \right] \right) \tau_j(\varphi_n^\psi). \quad (6.76)$$

We refer to Chapter 2 for a discussion on why all these coefficients can be computed efficiently at constant cost  $\mathcal{O}(1)$ .

(6.3.27)	(6.3.28)	(6.3.29)	(6.3.30)	(6.3.31),(6.3.32)
$\mathcal{O}(N_s M_s H)$	$\mathcal{O}(M_{\mathbf{u}} N_{\mathbf{u}} H)$	$\mathcal{O}(N_{\mathbf{u}} N_\psi H)$	$\mathcal{O}(N_s H)$	$\mathcal{O}(N_\psi H)$

TABLE 6.3.1. Costs of offline matrix computations

The costs for pre-computing the offline matrices (6.3.27)–(6.3.32) are summarized in Table 6.3.1.

The assembling of the residual (6.3.21) has complexity  $\mathcal{O}(N_s M_s + N_{\mathbf{u}} M_{\mathbf{u}} + N_\psi N_{\mathbf{u}})$  which is assumed to be significantly more efficient than the detailed operator evaluation (6.2.22) of complexity  $\mathcal{O}(H)$ . The assembling of the Jacobian (6.3.26) has complexity  $\mathcal{O}(N_s(N_s + N_{\mathbf{u}})M_s + N_{\mathbf{u}}(N_s + N_\psi)M_{\mathbf{u}} + N_\psi N_{\mathbf{u}})$ . In this respect, the reduced scheme might struggle to compete with its detailed counterpart, where the Jacobian assembly still takes  $\mathcal{O}(H)$ , but

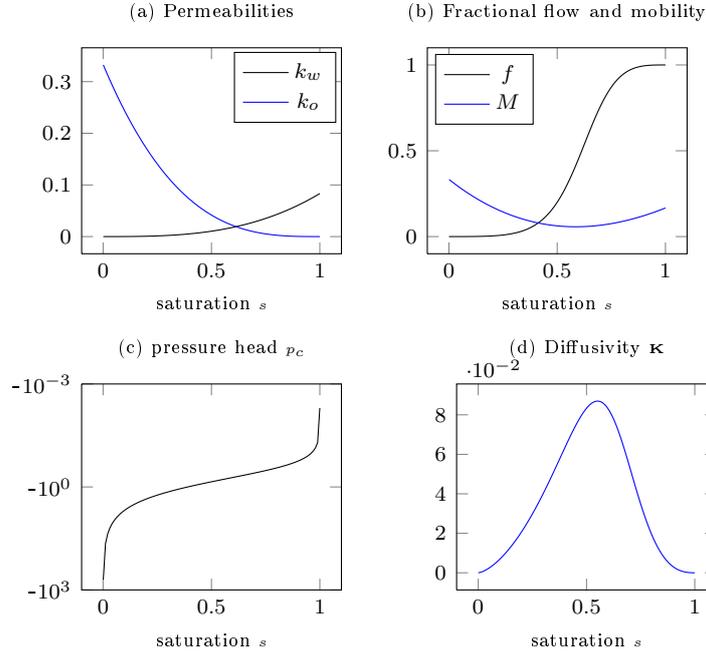


FIGURE 6.4.1. Plots of chosen data functions: (a)  $k_w(s)$ ,  $k_o(s)$ , (b)  $f(s)$ ,  $M(s)$ , (c)  $p_c(s)$  and (d)  $\kappa(s)$ .

with a larger constant factor. However, the most costly computations in the detailed scheme is the solution of the linear equation system, which adds up to  $\mathcal{O}(N^3)$  for the reduced scheme and in the detailed scheme  $\mathcal{O}(H^3)$  in case of Gaussian elimination or  $\mathcal{O}(HN_{\text{iter}})$  for an iterative solver with a maximum number of iterations  $N_{\text{iter}}$ .

#### 6.4. Numerical experiments

In our experiments with the reduced basis method for the two-phase flow problem (6.1.9)-(6.1.11), we use similar data functions like in [57]:

$$\begin{aligned}
 \mathbf{k} &= 1, \\
 k_w(s) &= \frac{s^3}{12}, & k_o(s) &= \frac{(1-s)^3}{3}, \\
 M(s) &= k_w(s) + k_{nw}(s), & f(s) &= \frac{k_w(s)}{M(s)} \quad \text{and} \\
 p_c(s) &= -0.5\sqrt{\frac{1-s}{s}}.
 \end{aligned} \tag{6.77}$$

Some of the above data functions are depicted in Figure 6.4.1. Here, especially the diffusivity, the fractional flow and the mobility functions are interesting in order to understand the non-linear behavior of the equation.

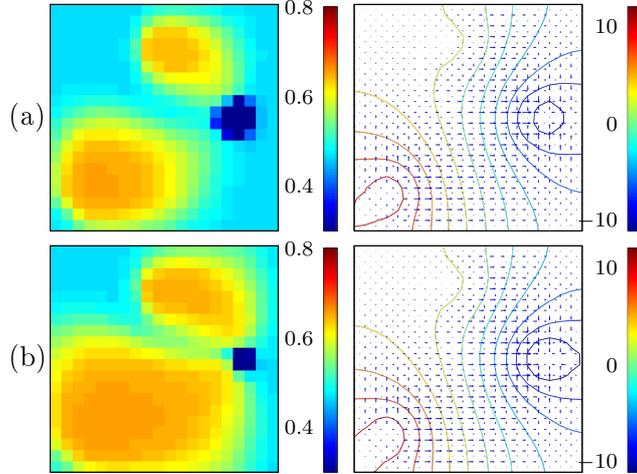


FIGURE 6.4.2. Illustration of saturation concentration, and contour plot of pressure field with velocity flux at time instants (a)  $t = 0.25$  and (b)  $t = 0.5$ . The snapshots are reconstructed from a reduced simulation.

The source and sink terms (6.1.15) are given as characteristic functions on circles  $B(\hat{x}, r) := \{x \in \mathbb{R}^2 \mid \|x - \hat{x}\| \leq r\}$  by

$$\begin{aligned}\bar{q} &= 10 \mathbf{1}_{B((0.5,0.8),0.01)} + 20 \mathbf{1}_{B((0.2,0.2),0.01)} \\ \underline{q} &= 30 \mathbf{1}_{B((0.8,0.5),0.01)}\end{aligned}$$

and the injection constant  $c = 0.8$ . The initial saturation is set to 0.5 on the entire domain  $\Omega := [0, 1]^2$ . The final time  $T$  is set to 0.5.

**6.4.1. Discretization.** For the finite volume discretization a rectangular mesh with 400 control volumes and a time step length of  $\Delta t = 0.01$  is used. Solution snapshots reconstructed from reduced simulations of the three unknowns at  $t = 0.25$  and  $t = 0.5$  are depicted in Figure 6.4.2. We observe non-linear flow from the two sources at the top and the lower left of the domain to the sink on the right.

**6.4.2. Reduced basis method.** After computing the trajectory of solution snapshots with the scheme described in Definition 6.2.3, a POD is applied to compute the three reduced bases  $\Phi^s$ ,  $\Phi^u$  and  $\Phi^\psi$ . Discarding all POD modes with eigenvectors less than  $\varepsilon_{\text{tol}}^{\text{POD}} := 10^{-6}$ , the resulting basis sizes are  $N_s = 28$ ,  $N_u = 72$  and  $N_\psi = 34$ .

The empirical interpolation basis and interpolation points for the non-linear operators  $\mathcal{L}_h^s$  and  $\mathcal{L}_h^u$  are derived with the EI-GREEDY algorithm for  $\varepsilon_{\text{tol}} = 10^{-8}$  resulting in basis sizes  $M_s = 387$  and  $M_u = 386$ . The error convergence and the selected interpolation points are illustrated in Figures

$(N_s, N_{\mathbf{u}}, N_\psi)$	$(M_s, M_{\mathbf{u}})$	$\ s_h - s_{\text{red}}\ $	$\ \psi_h - \psi_{\text{red}}\ $	time
(28,72,34)	(387,386)	$6.2 \cdot 10^{-5}$	$4.11 \cdot 10^{-4}$	30.15
(28,72,34)	(75,75)	$1.03 \cdot 10^{-4}$	$2.11 \cdot 10^{-3}$	21.56
(28,72,34)	(75,125)	$7.59 \cdot 10^{-5}$	$8.69 \cdot 10^{-4}$	20.61
(28,72,34)	(125,125)	$8.31 \cdot 10^{-5}$	$8.26 \cdot 10^{-4}$	21.37
(23,58,28)	(75,125)	$2.47 \cdot 10^{-4}$	$2.55 \cdot 10^{-3}$	18.24

TABLE 6.4.1. Error and timings of reduced simulations with different basis sizes.

6.4.3. We observe that the EI-GREEDY chooses the interpolation Dofs at different grid cells for the two operators resulting in two substantially different sub-grids. Both empirical interpolation algorithm show an exponential convergence. Table 6.4.1 illustrates the reduced simulation computation time and the  $L^2(\Omega)$  error between high dimensional and reduced discrete solutions for saturation and pressure. As the computation time of a high dimensional simulation is about 52 seconds, the computational gain is about 2.5. We observe that the empirical interpolation basis is oversized, as a reduced simulation with basis sizes  $(M_s, M_{\mathbf{u}}) = (75, 125)$  achieves similarly good results as with all basis vectors. So, a combined generation of basis space as proposed in Algorithm 4.3.1 for scalar equations, is desirable in the future.

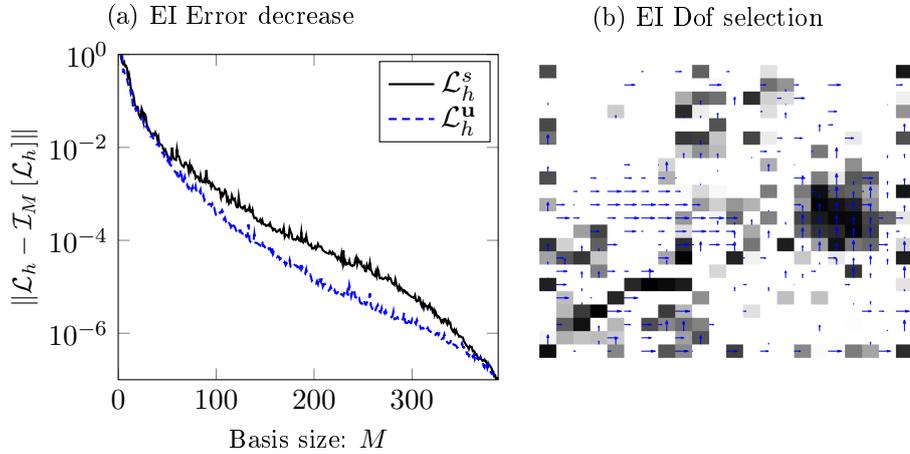


FIGURE 6.4.3. (a) Illustration of error convergence of EI-GREEDY algorithm for operators  $\mathcal{L}_h^s$  and  $\mathcal{L}_h^u$ . (b) Illustration of selected interpolation DOFs for operators  $\mathcal{L}_h^s$  (control volumes) and  $\mathcal{L}_h^u$  (fluxes over faces). Darker shades of marked control volumes and longer arrows represent earlier selected DOFs.

## Abstract software framework

In this chapter, we derive and define an abstract software concept, which can be used to efficiently implement a reduced basis scheme based on existing software code for numerical solvers of high-fidelity problems. It includes the entire execution cycle of the reduced basis method, including basis generation, offline-/ online decomposition and rapid reduced simulations with efficient error estimates.

The key properties of the software concept shall be the following:

- (1) *hardly intrusive extension*: It shall be possible to rapidly extend a programmer's favorite code of a solver for parametrized partial differential equations, such that it can be used as a reduced basis scheme.
- (2) *software decomposition*: High-dimensional computations and low-dimensional computations can be separated from each other, such that they can be executed on different computer architectures.
- (3) *reduced basis library*: The entire reduced basis machinery as described in the previous chapters shall be integrated.

The first property is the most important one: The necessary changes to an existing numerical solver code should be *hardly intrusive*. It is noteworthy, that during the offline phase at least access to the system matrices is necessary. Therefore, a non-intrusive black-box approach, where the code of the underlying high-dimensional scheme does not need to be changed, is in most cases impossible to realize. Especially commercial software packages often regard matrices and data functions to be private and do not allow to hand it over to the user. In open source software solutions, however, existing implementations can be adapted. Reduced basis methods for the open source software LIBMESH, e.g., have already been implemented in [47].

The second point deals with the fact, that the hardware and development requirements for high-dimensional and low-dimensional computations differ significantly. On the one hand, high-dimensional numerical solvers often demand high performance, parallel hardware architectures, and therefore often have a code base developed in C++ and Fortran. On the other hand, the reduced computations can be implemented on less demanding hardware devices

which provide higher mobility and less development effort. For example, in [41] a reduced basis application running on smartphones is introduced.

The outline of this chapter is as follows: We first want to analyze the general demands of a reduced basis software. For this, we first introduce a simple example for a general stationary partial differential equation in Section 7.1. This serves as a useful base for the identification and exemplification of the core reduced basis method components, because it is liberated from most technical peculiarities. Afterwards, we describe the functionality of the software package RBMATLAB which was originally developed by Bernard Haasdonk. All numerical examples in this thesis are developed with this Matlab based software, and such, we want to use it as a basis for our general software concept introduced in Section 7.3. We want to exemplify this concept by the means of another C++ based software package, called DUNE-RB. In Section 7.4 the functionality of DUNE-RB is described and we discuss which parts of our software concept it implements. We conclude with a proof of concept by running the entire reduced basis method execution cycle on a simple Poisson problem in Section 7.5 and by a discussion on implementational extensions which are necessary for the handling of non-linear problems in Section 7.6.

### 7.1. Stationary reduced basis scheme

Although the goal of this chapter is to describe an abstract framework for all kind of reduced basis problems, on occasion, we want to refer to a simple linear example. For a clearer exposition, we omit technical difficulties like non-linear operators and time dependence. In this section, we define a parametrized problem based on a stationary partial differential equation. Implementation details for the handling of non-linear operators with empirical operator interpolation are commented on in Section 7.6.

In this section, we define the abstract high-dimension discretization scheme, together with its corresponding reduced basis scheme and a posteriori error estimate. Based on these definitions, a proof of concept for the abstract framework is given in Section ?? for a finite volume discretization of a parametrized elliptic partial differential equation.

**Definition 7.1.1** (Analytical problem formulation). *Given a Hilbert space  $\mathcal{W}$ , a parameter space  $\mathcal{M} \subset \mathbb{R}^p$  of dimension  $p$ , for each parameter  $\boldsymbol{\mu} \in \mathcal{M}$ , we want to solve for solutions  $u(\boldsymbol{\mu}) \in \mathcal{W}$ , such that*

$$\mathcal{L}(\boldsymbol{\mu}) [u(\boldsymbol{\mu})] = b(\boldsymbol{\mu}), \quad (7.1)$$

*with a linear, regular differentiable operator  $\mathcal{L}(\boldsymbol{\mu}) : \mathcal{W} \rightarrow \mathcal{W}$  and a function  $b(\boldsymbol{\mu}) \in \mathcal{W}$ .*

In order to discretize the above problem, we restrict the Hilbert space  $\mathcal{W}$  to a discrete space  $\mathcal{W}_h \subset \mathcal{W}$  of dimension  $H$ .

**Definition 7.1.2** (Numerical scheme). *Given a discrete Hilbert space  $\mathcal{W}_h := \text{span} \{ \psi_i \}_{i=1}^H \subset \mathcal{W}$  spanned by basis functions  $\psi_i, i = 1, \dots, H$ , and a projection operator  $\mathcal{P}_h : \mathcal{W} \rightarrow \mathcal{W}_h$ , for each parameter  $\boldsymbol{\mu} \in \mathcal{M}$ , we want to solve for discrete solutions  $u_h(\boldsymbol{\mu}) \in \mathcal{W}_h$ , such that*

$$\mathcal{L}_h(\boldsymbol{\mu}) [u_h(\boldsymbol{\mu})] = b_h(\boldsymbol{\mu}), \quad (7.2)$$

with a linear discrete operator  $\mathcal{L}_h(\boldsymbol{\mu}) : \mathcal{W}_h \rightarrow \mathcal{W}_h$  approximating the analytical one  $\mathcal{L}_h \approx \mathcal{L}|_{\mathcal{W}_h}$  and projected functions  $b_h(\boldsymbol{\mu}) := \mathcal{P}_h \circ b(\boldsymbol{\mu})$ .

We also define matrix representations  $\underline{\mathcal{L}}_h(\boldsymbol{\mu}) \in \mathbb{R}^{H \times H}$  and  $\underline{b}_h(\boldsymbol{\mu}) \in \mathbb{R}^H$ , such that finding  $\underline{u}_h(\boldsymbol{\mu}) \in \mathbb{R}^H$ , with

$$\underline{\mathcal{L}}_h(\boldsymbol{\mu}) [\underline{u}_h(\boldsymbol{\mu})] = \underline{b}_h(\boldsymbol{\mu}) \quad (7.3)$$

is equivalent to (7.1.2) via  $u_h(\boldsymbol{\mu}) = \sum_{i=1}^H \underline{u}_h^i(\boldsymbol{\mu}) \psi_i$ . Here  $\underline{u}_h^i(\boldsymbol{\mu})$  denotes the  $i$ -th component of the Dof vector of  $u_h(\boldsymbol{\mu})$ .

Now, if we substitute the discrete function space by a reduced one  $\mathcal{W}_{\text{red}} \subset \mathcal{W}_h$  of dimension  $N \ll H$ , we can obtain the reduced scheme via

**Definition 7.1.3** (Reduced basis scheme). *We assume a reduced basis space  $\mathcal{W}_{\text{red}} := \text{span} \{ \varphi_n \}_{n=1}^N \subset \mathcal{W}_h$  with an orthonormal basis  $\Phi := \{ \varphi_n \}_{n=1}^N$  w.r.t. to the scalar product  $\langle \cdot, \cdot \rangle_{\mathcal{W}_h}$  and a Galerkin projection  $\mathcal{P}_{\text{red}} : \mathcal{W}_h \rightarrow \mathcal{W}_{\text{red}}$  satisfying*

$$\langle \mathcal{P}_{\text{red}}[u], v \rangle_{\mathcal{W}_h} = \langle u, v \rangle_{\mathcal{W}_h} \quad \text{for all } v \in \mathcal{W}_{\text{red}} \quad (7.4)$$

for all  $u \in \mathcal{W}_h$ . Then, for each parameter  $\boldsymbol{\mu} \in \mathcal{M}$ , we want to solve for reduced solutions  $u_{\text{red}}(\boldsymbol{\mu}) \in \mathcal{W}_{\text{red}}$ , such that

$$\mathcal{L}_{\text{red}}(\boldsymbol{\mu}) [u_{\text{red}}(\boldsymbol{\mu})] = b_{\text{red}}(\boldsymbol{\mu}) \quad (7.5)$$

with projections  $\mathcal{L}_{\text{red}}(\boldsymbol{\mu}) := \mathcal{P}_{\text{red}} \circ \mathcal{L}_h(\boldsymbol{\mu})$  and  $b_{\text{red}} := \mathcal{P}_{\text{red}} \circ b_h(\boldsymbol{\mu})$ . The corresponding matrices  $\underline{\mathcal{L}}_{\text{red}}(\boldsymbol{\mu}) \in \mathbb{R}^{N \times N}$  and  $\underline{b}_{\text{red}}(\boldsymbol{\mu}) \in \mathbb{R}^N$  can be computed by

$$\underline{\mathcal{L}}_{\text{red}}(\boldsymbol{\mu}) := (\underline{\mathcal{L}}_h(\boldsymbol{\mu}) \Phi)^t W \Phi \quad \text{and} \quad (7.6)$$

$$\underline{b}_{\text{red}}(\boldsymbol{\mu}) := (\underline{b}_h(\boldsymbol{\mu}))^t W \Phi, \quad (7.7)$$

with a reduced basis matrix  $\Phi \in \mathbb{R}^{H \times N}$  with column matrices set to the Dofs of the reduced basis functions  $\varphi_n$  and a weighting matrix  $W \in \mathbb{R}^{H \times H}$  such that

$$\underline{u}^t W \underline{v} = \langle u, v \rangle_{\mathcal{W}_h} \quad \text{for all } u, v \in \mathcal{W}_h. \quad (7.8)$$

Furthermore, an error estimate for the reduction error in the norm  $\|\cdot\|_{\mathcal{W}_h}$  can be easily obtained just like in [38].

**Lemma 7.1.4** (Error estimate). *We want to bound the approximation error between solutions  $u_h(\boldsymbol{\mu}) \in \mathcal{W}_h$  of the scheme from Definition 7.1.2 and reduced solutions  $u_{\text{red}}(\boldsymbol{\mu}) \in \mathcal{W}_{\text{red}}$  from Definition 7.1.3. If, for every  $\boldsymbol{\mu} \in \mathcal{M}$  there is a constant  $C(\boldsymbol{\mu}) > 0$  with*

$$\left\| (\mathcal{L}_h(\boldsymbol{\mu}))^{-1} \right\|_{\mathcal{W}_h} \leq C(\boldsymbol{\mu}), \quad (7.9)$$

*the approximation error can be bounded by*

$$\|u_h(\boldsymbol{\mu}) - u_{\text{red}}(\boldsymbol{\mu})\|_{\mathcal{W}_h} \leq \eta(\boldsymbol{\mu}) := C(\boldsymbol{\mu}) \|R(\boldsymbol{\mu})\| \quad (7.10)$$

*with the residual*

$$R(\boldsymbol{\mu}) := \mathcal{L}_h(\boldsymbol{\mu}) [u_{\text{red}}(\boldsymbol{\mu})] - b_h. \quad (7.11)$$

PROOF. With  $e(\boldsymbol{\mu}) := u_h(\boldsymbol{\mu}) - u_{\text{red}}(\boldsymbol{\mu})$  from the linearity of the operators  $\mathcal{L}_h(\boldsymbol{\mu})$ , it follows that

$$\begin{aligned} \mathcal{L}_h(\boldsymbol{\mu}) [e(\boldsymbol{\mu})] &= \mathcal{L}_h(\boldsymbol{\mu}) [u_h(\boldsymbol{\mu})] - \mathcal{L}_h(\boldsymbol{\mu}) [u_{\text{red}}(\boldsymbol{\mu})] \\ &= \underbrace{\mathcal{L}_h(\boldsymbol{\mu}) [u_h(\boldsymbol{\mu})] - b_h(\boldsymbol{\mu})}_{=0} + R(\boldsymbol{\mu}). \end{aligned} \quad (7.12)$$

Thus, (7.1.9) leads to the proposition

$$\|e(\boldsymbol{\mu})\|_{\mathcal{W}_h} = \left\| (\mathcal{L}_h(\boldsymbol{\mu}))^{-1} \right\|_{\mathcal{W}_h} \|R(\boldsymbol{\mu})\|_{\mathcal{W}_h} \leq C(\boldsymbol{\mu}) \|R(\boldsymbol{\mu})\|_{\mathcal{W}_h}. \quad (7.13)$$

□

In order to obtain an efficient offline-/online decomposition, we want to assume that the operators  $\mathcal{L}_h(\boldsymbol{\mu})$  and the functions  $b_h(\boldsymbol{\mu})$  are separable in the parameter, i.e. there exist decompositions

$$\mathcal{L}_h(\boldsymbol{\mu}) = \sum_{q=1}^{Q_L} \sigma_L(\boldsymbol{\mu}) \mathcal{L}_h^q \quad \text{and} \quad (7.14)$$

$$b_h(\boldsymbol{\mu}) = \sum_{q=1}^{Q_b} \sigma_b(\boldsymbol{\mu}) b_h^q. \quad (7.15)$$

with coefficient functions  $\sigma_L, \sigma_b : \mathcal{M} \rightarrow \mathbb{R}$ , linear operators  $\mathcal{L}_h^q : \mathcal{W}_h \rightarrow \mathcal{W}_h, q = 1, \dots, Q_L$  and functions  $b_h^q \in \mathcal{W}_h, q = 1, \dots, Q_b$ . In this case the required parameter dependent reduced matrices and vectors can be obtained by a linear combination of parameter independent reduced matrices and vectors. We want to summarize this offline-/online decomposition in the following by the next definition and lemma.

**Definition 7.1.5** (Offline-Phase). *During the offline phase for the scheme, the following reduced matrices and vectors need to be computed:*

$$\underline{\mathcal{L}}_{\text{red}}^q := \left( \underline{\mathcal{L}}_h^q \underline{\Phi} \right)^t W \underline{\Phi} \quad \text{for } q = 1, \dots, Q_L, \quad (7.16)$$

$$\underline{b}_{\text{red}}^q := \left( \underline{b}_h^q \right)^t W \underline{\Phi} \quad \text{for } q = 1, \dots, Q_b, \quad (7.17)$$

$$\underline{\mathcal{K}}_{\text{red}}^{q,q'} := \left( \underline{\mathcal{L}}_h^q \underline{\Phi} \right)^t W \underline{\mathcal{L}}_h^{q'} \underline{\Phi} \quad \text{for } q, q' = 1, \dots, Q_L, \quad (7.18)$$

$$\underline{m}_{\text{red}}^{q,q'} := \left( \underline{\mathcal{L}}_h^q \underline{\Phi} \right)^t W \underline{b}_h^{q'} \quad \text{for } q = 1, \dots, Q_L, q' = 1, \dots, Q_b \text{ and} \quad (7.19)$$

$$\underline{n}_{\text{red}}^{q,q'} := \left( \underline{b}_h^q \right)^t W \underline{b}_h^{q'} \quad \text{for } q, q' = 1, \dots, Q_b. \quad (7.20)$$

The dimensions of all these matrices depend on the dimension of the reduced basis space  $N$ .

**Lemma 7.1.6** (Online-Phase). *Given the matrices from Definition 7.1.5, the parameter dependent reduced matrices and vectors can be efficiently computed for all  $\boldsymbol{\mu} \in \mathcal{M}$  as*

$$\underline{\mathcal{L}}_{\text{red}}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_L} \sigma_L^q(\boldsymbol{\mu}) \underline{\mathcal{L}}_{\text{red}}^q, \quad (7.21)$$

$$\underline{b}_{\text{red}}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_b} \sigma_b^q(\boldsymbol{\mu}) \underline{b}_{\text{red}}^q, \quad (7.22)$$

$$\underline{\mathcal{K}}_{\text{red}}(\boldsymbol{\mu}) := \sum_{q,q'=1}^{Q_L} \sigma_L^q(\boldsymbol{\mu}) \sigma_L^{q'}(\boldsymbol{\mu}) \underline{\mathcal{K}}_{\text{red}}^{q,q'}, \quad (7.23)$$

$$\underline{m}_{\text{red}}(\boldsymbol{\mu}) := \sum_{q=1}^{Q_L} \sum_{q'=1}^{Q_b} \sigma_L^q(\boldsymbol{\mu}) \sigma_b^{q'}(\boldsymbol{\mu}) \underline{m}_{\text{red}}^{q,q'} \quad \text{and} \quad (7.24)$$

$$\underline{n}_{\text{red}}(\boldsymbol{\mu}) := \sum_{q,q'=1}^{Q_b} \sigma_b^q(\boldsymbol{\mu}) \sigma_b^{q'}(\boldsymbol{\mu}) \underline{n}_{\text{red}}^{q,q'}. \quad (7.25)$$

The residual norm  $\|R(\boldsymbol{\mu})\|_{\mathcal{W}_h}$  from Lemma 7.1.4 is also efficiently computable by

$$\|R(\boldsymbol{\mu})\|_{\mathcal{W}_h}^2 = \left( \underline{u}_{\text{red}}(\boldsymbol{\mu}) \right)^t \underline{\mathcal{K}}_{\text{red}}(\boldsymbol{\mu}) \underline{u}_{\text{red}}(\boldsymbol{\mu}) - 2 \left( \underline{u}_{\text{red}}(\boldsymbol{\mu}) \right)^t \underline{m}_{\text{red}}(\boldsymbol{\mu}) + \underline{n}_{\text{red}}(\boldsymbol{\mu}). \quad (7.26)$$

PROOF. The efficient computability of (7.1.21)-(7.1.25) follows from the fact that all matrices from Definition 7.1.5 are low dimensional. Thus, also equation (7.1.26) is efficiently computable. Furthermore, it can be easily

seen, that

$$\begin{aligned}
\|R(\boldsymbol{\mu})\|_{\mathcal{W}_h}^2 &= \langle R(\boldsymbol{\mu}), R(\boldsymbol{\mu}) \rangle_{\mathcal{W}_h} \\
&= \langle \mathcal{L}_h(\boldsymbol{\mu})[u_{\text{red}}(\boldsymbol{\mu})] - b_h(\boldsymbol{\mu}), \mathcal{L}_h(\boldsymbol{\mu})[u_{\text{red}}(\boldsymbol{\mu})] - b_h(\boldsymbol{\mu}) \rangle_{\mathcal{W}_h} \\
&= (\underline{u}_{\text{red}}(\boldsymbol{\mu}))^t \underline{\mathcal{K}}_{\text{red}}(\boldsymbol{\mu}) \underline{u}_{\text{red}}(\boldsymbol{\mu}) - 2 (\underline{u}_{\text{red}}(\boldsymbol{\mu}))^t \underline{m}_{\text{red}}(\boldsymbol{\mu}) + \underline{n}_{\text{red}}(\boldsymbol{\mu}). \quad (7.27)
\end{aligned}$$

□

## 7.2. Overview on RBMATLAB

The Matlab based software package RBMATLAB is a library of functions useful for reduced basis method implementations. For rapid prototyping, it comes with numerical schemes for several finite volume and finite element discretizations of parametrized partial differential equations in one or two dimensions based on rectangular or triangular grids. For all these schemes a reduced counterpart exists, which can be used to efficiently compute solutions on a reduced basis space. For the generation of the reduced basis spaces and empirical interpolation data, all the algorithms described in Chapter 4 are implemented.

In this chapter, we do not want to delve into the details of the above implementations, but analyze the main parts of the software and how these interact with each other. From these observations, we will derive necessary interfaces for other software packages in the next section.

First, we concentrate on the general course of action in RBMATLAB that can be regarded as a user interface for both the offline phase of a reduced basis method execution cycle and the efficient simulations during the online phase. Figure 7.2.1 illustrates the main methods in the order as they are called and describes them shortly.

The first step is optional and can be used to construct parameter independent high-dimensional data structures which is important, if e.g. a complex grid structure is needed for every detailed simulation. In this case, it can pay off to externalize the initialization into a parameter independent call. The method `gen_detailed_data` in step 3 handles the generation of reduced basis spaces and empirical interpolation data as described in Chapter 4. In the function body, no high-dimensional computations shall be processed. Instead all these computations shall be encapsulated in specialized method calls, as sketched in Figure 7.2.1 for the example of an abstract greedy algorithm. The same holds true for the next step, in which low-dimensional, but still parameter independent Gramian matrices and vectors are assembled. The time and memory demanding assembling is delegated to the method `rb_operators`. In Table 7.2.1 possible calls of this method are

1.	<pre>model_data=gen_model_data(model)</pre> <p>Constructs parameter independent high dimensional data for simulations, e.g. a mesh of the domain.</p>
2.	<pre>sim=detailed_simulation(model, model_data)</pre> <p>Computes parameter dependent high dimensional solutions <math>u_h(\boldsymbol{\mu})</math> as in Definition 7.1.2.</p>
3.	<pre>detailed_data=gen_detailed_data(model, model_data)</pre> <p>Generates reduced basis space for example by an algorithm from Chapter 4. For the example from Section 7.1, we would use the BASIC-GREEDY algorithm by calling high dimensional methods:</p> <pre>detailed_data=init_data_basis(model, detailed_data)</pre> <p>Initialize reduced basis space</p> <pre>detailed_data=rb_extension(detailed_data, <math>\boldsymbol{\mu}</math>)</pre> <p>Extend reduced basis space by <math>u_h(\boldsymbol{\mu})</math>.</p> <p>and one that evaluates the error estimate <math>\eta(\boldsymbol{\mu})</math>. (c.f. Algorithm 4.2.1)</p>
4.	<pre>reduced_data=gen_reduced_data(model, detailed_data)</pre> <p>Generates reduced vectors and matrices as in Definition 7.1.5 using the method</p> <pre>[Gramians]=rb_operators(model, detailed_data, op, 1)</pre> <p>C.f. Table 7.2.1 for details.</p>
5.	<pre>rb_sim=rb_simulation(model, reduced_data)</pre> <p>Computes a reduced solution Dofs <math>u_{\text{red}}(\boldsymbol{\mu})</math> and the a posteriori error estimate <math>\eta(\boldsymbol{\mu})</math>, e.g. as in Definition 7.1.3 and Lemma 7.1.4.</p>
6.	<pre>sim=rb_reconstruction(model, detailed_data, rb_sim)</pre> <p>Computes a reduced solution <math>u_{\text{red}}(\boldsymbol{\mu})</math>.</p>
7.	<pre>p=plot_sim_data(model, detailed_data, sim)</pre> <p>Visualizes a vector given by <code>sim</code>, this can be obtained by a detailed simulation <math>u_{\text{red}}(\boldsymbol{\mu})</math> or as a reconstructed reduced simulation <math>u_{\text{red}}(\boldsymbol{\mu})</math>.</p>

FIGURE 7.2.1. Illustration of the course of action for reduced basis methods in RBMATLAB. Method calls involving high-dimensional computations are surrounded with gray shaded boxes. The data structures `model_data`, `sim_data` and `detailed_data` are high-dimensional and depending on  $\mathcal{O}(H)$ , `reduced_data`, `Gramians` and `rb_sim` are low-dimensional data structures depending on  $\mathcal{O}(N)$ .

illustrated for the general example from Section 7.1. Finally, in step 5 the reduced basis scheme and the error estimator can be efficiently computed. Here, it is also possible to compute an output functional. If the user wants to visualize the reduced simulation, it needs to be reconstructed from the computed low dimensional Dof vector. Thus, the visualization takes two additional steps.

<code>rb_operators(model, detailed_data, decomp_mode, opstring)</code>		
<code>opstring</code>	<code>decomp_mode</code>	
	1	2
<code>&lt;Φ, Φ&gt;_W</code>	$\{\underline{\Phi}^t W \underline{\Phi}\}$	$\{1\}$
<code>&lt;L[Φ], Φ&gt;_W</code>	$\left\{(\underline{\mathcal{L}}_h^q \underline{\Phi})^t W \underline{\Phi}\right\}_{q=1}^{Q_L}$	$\{\sigma_L^q(\boldsymbol{\mu})\}_{q=1}^{Q_L}$
<code>&lt;L[Φ], L[Φ]&gt;_W</code>	$\left\{(\underline{\mathcal{L}}_h^q \underline{\Phi})^t W \underline{\mathcal{L}}_h^{q'} \underline{\Phi}\right\}_{q,q'=1}^{Q_L}$	$\{\sigma_L^q(\boldsymbol{\mu}) \cdot \sigma_L^{q'}(\boldsymbol{\mu})\}_{q,q'=1}^{Q_L}$
<code>&lt;L[Φ], b&gt;_W</code>	$\left\{(\underline{\mathcal{L}}_h^q \underline{\Phi})^t W \underline{b}_h^{q'}\right\}_{\substack{1 \leq q \leq Q_L \\ 1 \leq q' \leq Q_b}}$	$\{\sigma_L^q(\boldsymbol{\mu}) \cdot \sigma_b^{q'}(\boldsymbol{\mu})\}_{\substack{1 \leq q \leq Q_L \\ 1 \leq q' \leq Q_b}}$
<code>&lt;b, Φ&gt;_W</code>	$\left\{(\underline{b}_h^q)^t W \underline{\Phi}\right\}_{q=1}^{Q_b}$	$\{\sigma_b^q(\boldsymbol{\mu})\}_{q=1}^{Q_b}$
<code>&lt;b, b&gt;_W</code>	$\left\{(\underline{b}_h^q)^t W \underline{b}_h^{q'}\right\}_{q,q'=1}^{Q_b}$	$\{\sigma_b^q(\boldsymbol{\mu}) \sigma_b^{q'}(\boldsymbol{\mu})\}_{q,q'=1}^{Q_b}$

TABLE 7.2.1. Illustration for the computation of Gramian matrices with the `rb_operators` method. The table shows possible return values for various arguments `decomp_mode` and `opstring`.

### 7.3. Software concept

Now, that we have understood the main functions in RBMATLAB and got an overview on the methods involving high-dimensional computations, we want to define an abstract concept based on the main software parts that we can identify in any reduced basis application.

These parts are

- a *detailed scheme*, providing a solver for parametrized solutions, and also visualization routines,
- *basis space manipulation*, i.e. the storage handling and the algorithmic extension of reduced basis spaces and empirical interpolation data,
- a *reduced matrix assembler*, which efficiently computes reduced matrices as illustrated in Table 7.2.1,

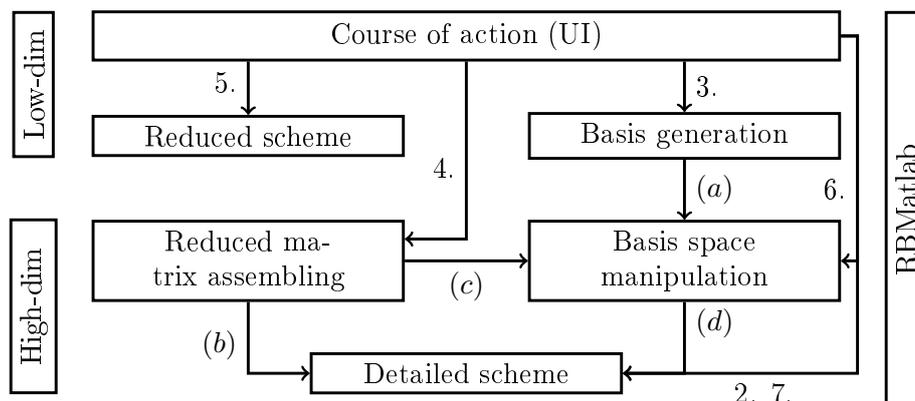


FIGURE 7.3.1. Call graph for main software parts in RB-MATLAB. The numbers refer to the steps in the course of action in Figure 7.2.1

- a *basis generation* algorithm, like the greedy algorithms defined in Chapter 4,
- a *reduced scheme* conducting all reduced simulations, including the computation of output functionals and error estimates, and
- the *user interface* as described in the previous section.

Figure 7.3.1 illustrates the call graph between these software parts as it can be observed in RBMATLAB. From these calls, we can derive the necessary interfaces the software parts need to provide. The numbers next to the arrows indicate the step numbers in the usual course of action as described in the previous chapter and in Figure 7.2.1. The internal calls which are not accessible through user interface are denoted with small letters. In the following, we want to define these internal calls and analyze them.

Call (a) depends on the selected basis generation algorithm. It is very difficult to generalize, because the requirements of basis generation algorithms can differ significantly. For greedy algorithms based on the X-GREEDY from Algorithm 4.2.1, however, the *basis space manipulation* part needs to provide methods for the initialization of the reduced basis and for an incremental extension.

The *reduced matrix assembler* needs via (b) access to all operators and functions which are separable in the parameter and utilized by the *detailed scheme*. So, it requires an interface for operators and functions which controls access to their parameter independent components and defines a unique identifier, such that the assembler can be used as in Table 7.2.1. On the other hand, the matrix assembler needs access to the reduced basis functions via (c), preferably returned as a matrix of column vectors.

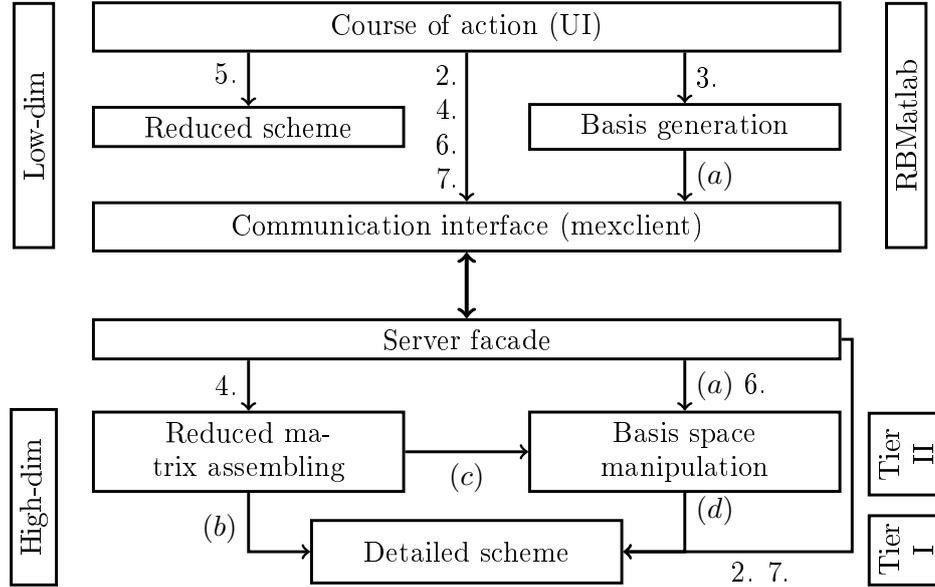


FIGURE 7.3.2. Call graph for main software parts in our abstract software concept. High-dimensional parts can be strictly separated from the low-dimensional ones. The numbers refer to the steps in the course of action in Figure 7.2.1

As during basis generation the reduced basis spaces are enriched with solution snapshots from the *detailed scheme*, the *basis space manipulation part* needs to initiate such high-dimensional simulations. This call (*d*), however, is equivalent to the user interface call (2.).

The figure also nicely depicts the strict separation between high-dimensional and low-dimensional computations. As the goal is to strictly separate these two, in order to make use of specialized hardware architectures and software tools, we split the parts and such allow them to exist in two different programs with a client-server communication part in between. The result is illustrated in Figure 7.3.2. All calls bridging the boundary between the low- and the high-dimensional software parts, i.e. (2.,4.,6.,7.,*a*), need to communicate their requests via a communication interface which forwards the request to a server. Note, that only low-dimensional data is communicated.

Additionally, we suggest to split the high-dimensional software into two tiers in the face of rapid re-usability of existing detailed schemes. Tier I is the problem dependent part that needs to be adjusted for every discretization by the implementation of interfaces for (2.,7.,*b*), whereas tier II can be considered as more robust and needs to be adjusted for new basis generation algorithms only. Note, that the data containers utilized in the two tiers may

differ significantly as well. Whereas tier I usually deals with linear algebra on sparse matrices and vectors, tier II requires dense linear algebra routines and containers.

In the next section, we focus on the software package DUNE-RB which implements many methods in tier II, provides interfaces and example implementations for tier I and provides methods for TCP/IP communication with RBmatlab. Thus, it brings the flexibility and power of Dune to the reduced basis world.

#### 7.4. High-dimensional computations with DUNE-RB

The C++ software Dune [3, 2] has been developed for rapid and efficient implementation of numerical solvers for partial differential equations which are based on grid-based methods. Many existing solver implementations can be run efficiently on high performance hardware architectures. Due to its interface based design, pursuing the goals of re-usability and exchangeability of code parts, Dune is a perfect extension for our reduced basis framework.

Dune comes as a collection of smaller software units packaged as modules. The most important modules include

- DUNE-GRID providing a common interface for structured and unstructured, conforming and non-conforming, parallel grids, and implementations of those,
- DUNE-ISTL for solvers of linear and non-linear equation system and containers for efficient storage of sparse matrices and vectors,
- DUNE-LOCALFUNCTIONS provides a library of so-called local basis functions which are defined on a generic reference domain only, and from which global basis functions can be constructed in an abstract and efficient way, and
- DUNE-COMMON comprising basic data functions, like dense matrices and vectors, and auxiliary scripts that help to automatically resolve dependencies between Dune modules.

Furthermore, there exist different modules with implementations of discretizations for partial differential equations which are based on the listed core modules. Examples for this discretization modules are DUNE-FEM, DUNE-PDELAB and DuMu<sup>x</sup>, which are based on the above core modules.

Our own Dune module DUNE-RB shall be build on top of such discretization modules. In [26], we showed how a simple linear evolution problem implemented in DUNE-FEM can be turned into a reduced basis aware scheme. In this presentation, however, we will conclude with numerical experiments for a simple elliptic problem based on a DUNE-PDELAB implementation. The DUNE-PDELAB module suits our needs very well, as its implementations are

operator based, and allow a simple substitution of linear solvers and matrix/vector containers. Especially, the last fact is of high interest for us, because as discussed earlier, the requirements on a linear algebra library differ crucially between the two high-dimensional software tiers I and II. Whereas numerical schemes usually depend on sparse operators and linear algebra algorithms, the assembling of the reduced matrices and vectors and the reduced basis generation require efficient dense matrices and routines for tasks like eigenvalue computations or orthogonalization. One software library that comes with efficient implementations for both tiers, is the linear algebra package EIGEN which is used in our numerical experiment.

**7.4.1. Interfaces.** In the following, we describe the most important interface classes that ensure its implementations provide the necessary functionality in order to make all the calls depicted in Figure 7.3.2.

7.4.1.1. *Interfaces in tier I.* In tier I, we have two main interfaces: First, the `Detailed::Solver::Connector::Interface` comprises methods

- `solve()` for calls (d) and (2.), computing a detailed simulation for a specified parameter vector and returning the resultant Dof vector,
- `visualize()` for call (7.), writing a vtk file to the harddrive which can be visualized by external programs then and
- `getSystemMatrix()` and `getSystemVector()` in order to return problem dependent matrices and vectors that need to be reduced. In the example from Section 7.1, the operators  $\mathcal{L}_h(\boldsymbol{\mu})$  and the functions  $b_h(\boldsymbol{\mu})$  are returned here.

If separable, the system matrices implement interface classes denoted by `LA::SeparableParametric::IMatrix` and `::IVector`. These classes have methods

- `component()`, returning the  $q$ -th parameter independent components of a separable decomposition, e.g.  $\underline{\mathcal{L}}_h^q$ ,
- `coefficients()`, for the parameter dependent coefficient functions, e.g.  $(\sigma_L^q)_{q=1}^{Q_L}$ , and
- `symbolicCoefficients()`, returning strings that can be evaluated in Matlab as function handles such that during a reduced simulation, we can omit to communicate with the *detailed scheme*.

Furthermore, every separable matrix or vector needs to have a unique identifier string. The above interfaces standardize the call (b).

7.4.1.2. *Interfaces in tier II.* In tier II, we have two main classes: the reduced basis space `Offline::Space::RB` and a tool to assemble reduced matrices for linear evolution problems `Offline::Generator::LinStat`.

The reduced basis space comes with a method

- `reconstruct()`, which reconstructs a low dimensional Dof vector in the original high dimensional space by a linear combination of the basis functions. This is for call (6.).
- several methods in order to add, remove or return basis functions. The latter is needed for call (a) and (c). The other only for (a). However, depending on the implementation of the basis generation algorithm, it might be necessary to extend the implementation.

The other class derived from `Offline::Generator::Iterator` provides exactly the functionality as the RBMATLAB function `rb_operators`, and it is needed for call (4.).

7.4.1.3. *Communication interface.* As the reduced basis method is split into two independent units, data needs to be transferred between RBMATLAB and DUNE-RB efficiently. For this, DUNE-RB comprises implementations of serializable data container interface `MatlabComm::SerializedArray`. Its two implementations `RBMatrix` and `MXMatrix` both mimic the behavior of the classical Matlab data containers “matrix”, “struct”, “cell array” and “string”. `RBMatrix` utilizes the STL data containers and Eigen matrices for the underlying data storage, whereas the data in `MXMatrix` is stored in the `mxArray` data structure provided by Matlab. As both implementations can be serialized they can be interchanged via TCP/IP connections or over the hard-drive.

We have two possibilities in order to connect the two software units RBMATLAB and DUNE-RB:

- (1) Either we compile the program based on DUNE-RB as a mex-library, which allows it to be called directly from the Matlab prompt, or
- (2) we compile it as a server waiting for TCP/IP communication from the RBMATLAB side.

In both cases, calls bridging the barrier between both units are initiated from RBMATLAB by a method call of the form

```
[ret1,ret2]=mexclient('operation',arg1,arg2)
```

where `operation` is the name of an interface function, e.g. `rb_operators`. The arguments and return values are wrapped as `std::vector<MXMatrix *>` objects by the mex library. For the client-server case, these arguments are then serialized as `std::vector<RBMatrix *>` objects on the server side. For this reason, the server does not depend on the Matlab mex library. This is also why we prefer the second alternative, because the compilation of a complex C++ program as a mex library can be very difficult as certain constraints on the compiler versions and options apply.

The main classes for the communication between the two software units are either `MatlabComm::Server::MexLibrary` for compiling a program as a mex library or `MatlabComm::Server::Sockets` for compilation as a TCP/IP server. Both of these classes are singletons which must be bound to a so-called server facade. A server facade object is always derived from the interface class `MatlabComm::Facade::Base` and has an entry point

```
void mexFunction(std::vector<RBMatrix*> largs,
                 std::vector<RBMatrix*> rargs);
```

from which methods defined by the `operation` string, stored in `rargs[0]` can be called. Thus, the server facade can be seen as a second user interface, as it gathers all the methods a DUNE-RB program provides and makes the high-dimensional software parts from Figure 7.3.2 available to Matlab.

For example, for the general stationary problem defined in Section 7.1, the following methods are defined in our facade `MatlabComm::Facade::LinStat` implemented in DUNE-RB:

- `gen_model_data`,
- `detailed_simulation`,
- `init_data_basis`,
- `rb_extension`,
- `rb_operators`,
- `rb_reconstruction` and
- `get_rb_size`.

### 7.5. Example: Poisson problem

In this section, we provide a proof of concept for our abstract concept described above by implementing a finite volume discretization of a parametrized elliptic problem. Earlier, in [22], we used DUNE-RB with a linear evolution problem based on the discretization module DUNE-FEM [20].

Here, we want to solve a Poisson problem on a bounded domain  $\Omega \subset \mathbb{R}^d$ ,  $d = 2, 3$ , and find functions  $u \in BV(\Omega) \cap L^\infty(\Omega) \subset L^2(\Omega)$ , such that the partial differential equation

$$-k\Delta u - mu = 1 \quad \text{in } \Omega \quad (7.28)$$

$$u = 0 \quad \text{on } \partial\Omega \quad (7.29)$$

with zero Dirichlet boundary conditions and parameters  $k, m > 0$  is fulfilled. The parametrization of the problem is realized by parameter vectors  $\boldsymbol{\mu} := (k, m) \in \mathcal{M} := [1, 10] \times [0, 0.2]$ .

For discretization, a finite volume scheme as in Section 3.1.1 is used, based on a discrete function space  $\mathcal{W}_h$  spanned by  $H$  basis functions whose support is bounded to elements of an underlying admissible grid  $\mathcal{T}$  as given in Definition 3.1.2. Then, we obtain a numerical scheme as in Definition 7.1.2 by specifying the separable discretization operator

$$\mathcal{L}_h(\boldsymbol{\mu}) := k\mathcal{L}_{\text{diff}} + m\mathcal{L}_{\text{id}} \quad (7.30)$$

with the negative identity operator component  $\mathcal{L}_{\text{id}} : \mathcal{W}_h \rightarrow \mathcal{W}_h, u_h \mapsto -u_h$  and the Laplace operator component  $\mathcal{L}_{\text{diff}} : \mathcal{W}_h \rightarrow \mathcal{W}_h$  defined by

$$(\mathcal{L}_{\text{diff}}[u_h])_i = -\frac{1}{|e_i|} \sum_{j \in \mathcal{N}_{\text{in}}(i)} \frac{u_{h,j} - u_{h,i}}{|x_j - x_i|} |e_{ij}|, \quad i = 1, \dots, H. \quad (7.31)$$

The right hand side function is parameter independent and fixed to the constant function

$$b_h(\boldsymbol{\mu}) = 1. \quad (7.32)$$

Figure 7.5.1 depicts solution snapshots for various parameter configurations. For  $m = 0$ , the solutions are scaled by the first parameter  $k$ , such that a single reduced basis functions is enough to approximate these functions. The parameter  $m$  increases the steepness of the solutions close to the boundary of the domain. Due to the generality of the grid implementations in Dune, the computations can be carried out in two and three dimensions.

As described in Section 7.1, the reduced scheme from Definition 7.1.3 and the error estimate from Lemma 7.1.4 can be applied. For this reason, we can create a reduced basis space  $\mathcal{W}_{\text{red}} \subset \mathcal{W}_h$  spanned by basis functions  $\{\varphi_n\}_{n=1}^N$  with a greedy algorithm. In this case, we use the BASIC-GREEDY algorithm as defined in Section 4.2.1.

**7.5.1. Implementation.** The numerical scheme including the discrete operators are implemented based on the discretization module DUNE-PDELAB which provides classes for the assembling of operators based on local operator restrictions.

As described in Section 7.3, for the extension of the scheme in DUNE-RB only the tier I software parts are necessary. In our case, the numerical scheme is wrapped by a class called `Detailed::Solver::Connector::PoissonCCFV` exporting the detailed simulation, the separable operators and the right hand side function, and a method to write out a vtk [67] file for visualization of a given solution function. The separable containers are represented by classes `SeparableLaplaceOperator` and `SeparableLaplaceResidual` for the operator  $\mathcal{L}_h(\boldsymbol{\mu})$  respectively the right hand side function  $b_h(\boldsymbol{\mu})$ .

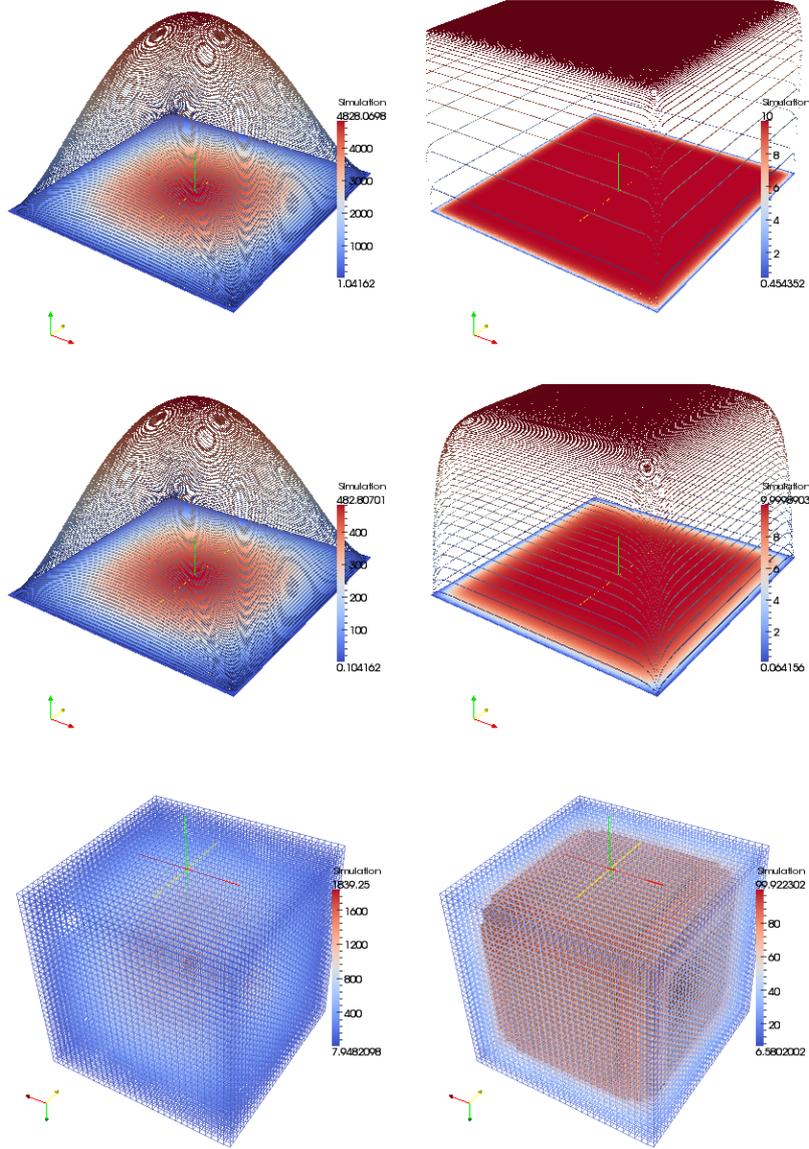


FIGURE 7.5.1. Solution snapshots for different parameter vectors. First row:  $\boldsymbol{\mu} = (1,0)$  and  $\boldsymbol{\mu} = (1,0.1)$ , Second row:  $\boldsymbol{\mu} = (10,0)$  and  $\boldsymbol{\mu} = (10,0.1)$ , Third row: 3D problem  $\boldsymbol{\mu} = (1,0)$  and  $\boldsymbol{\mu} = (1,0.01)$

**7.5.2. Numerical results.** In Figure 7.5.2 we demonstrate the behavior of the error estimator during the construction of the reduced basis via the BASIC-GREEDY algorithm. The error bound in logarithmic scale is plotted against the size of the reduced basis, varying from one to the maximum number of generated reduced basis function, which in this experiment is about 10. Note, that independent of the high dimensional problem size, we observe the same rate of exponential error decay. So, the time gain factor induced by the reduced basis method increases for higher dimensional problems. This

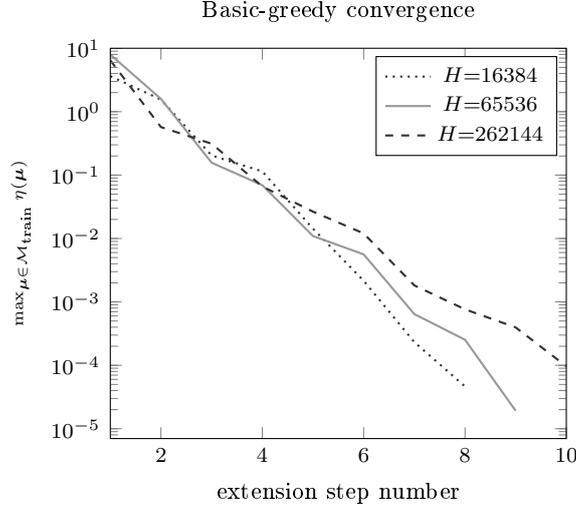


FIGURE 7.5.2. Maximum of error bound over a set of 200 randomly chosen training parameters in logarithmic scale for different high dimensional problem sizes.

$H$	$N$	max. error	$\phi$ -time[s]			offline-time[s]
			detailed	reduced	reconstr.	
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 7.5.1. Average run-times and maximum error estimates for a set of 100 random test parameters. The last row refers to a reduced simulation of a problem in three dimensions.

fact is very noticeable in Table 7.5.1 in which average run-times and maximum error estimates over 100 test parameters are shown. The time gain ratio between a detailed and a reduced simulation varies from  $2.25 \cdot 10^5$  for the smallest problem size to  $6.65 \cdot 10^5$  for the largest problem size. We also observe significant costs for the reconstruction of the reduced solutions in the original discrete function space. Therefore, the use of output functionals instead of reconstructed solutions, is highly recommended.

## 7.6. Outlook: subgrid extraction for empirical interpolation

So far, we have only shown how linear problems with operators and data functions which are separable in the parameter can be integrated into our software concept. For non-linear problems, however, this is not applicable, as the empirical interpolation method needs to be implemented as well. For

this, the *reduced scheme* needs to efficiently compute operator evaluations at empirical interpolation Dofs. Thus, two implementations of the same operator are needed.

In the setting of Figure 7.3.1 where the *detailed scheme* is implemented in RBMATLAB, no duplication of the operators is necessary, as the discrete operators are grid based and the grid implementations in RBMATLAB enable us to construct sub-grids which are efficient w.r.t. memory consumption and access to geometric or topological information. If during the offline-phase such a sub-grid is computed that holds all necessary information in order to compute operator evaluations at the empirical interpolation Dofs, the same operator implementation used in the detailed scheme can be re-used in the reduced one. This approach is similar to the “sample mesh” method introduced in [11].

The sub-grid extraction can be surely transferred to the setting in Figure 7.3.2 with the further difficulty that the two variants of the grid and the non-linear operators are located in two different software units, and possibly implemented in different programming languages. In order to deal with this, the tier II software layer should be extended by an `RBmatlabGrid` implementation, that can be initialized with Matlab data structures comprising all geometry and topology information of a sub-grid. Then, the operators in the *detailed scheme* software part can be compiled as a library method based on an `RBmatlabGrid` instance and deployed for efficient usage in the reduced scheme.

## Conclusion and outlook

In this thesis, we have developed a reduced basis method, that can be applied to a broad class of parametrized evolution problems. This method includes an offline-/online decomposition, efficient a posteriori error estimates, and quickly convergent basis generation algorithm.

We do not need to make assumptions on the non-linearity of the underlying solver. The main ingredient which makes us achieve these goals, is the empirical interpolation of operators, introduced in 2. We transferred results on error analysis from the original empirical interpolation method for functions [1] to our new setting, and extended it by discussions on the behavior of empirical operator interpolants under differentiation and on invariant operator properties in Section.2.4.

In order to efficiently generate reduced basis spaces and empirical interpolants, a variant of the classical greedy algorithms was proposed that automatically synchronizes the quality of both basis spaces. In numerical experiments, we observed that it could reconstruct the optimal ratio between the two basis spaces, and observed a time gain in the offline phase at small basis sizes.

Furthermore, in the field of basis generation algorithms, we demonstrated extensions to the greedy algorithms, that adaptively generate multiple basis spaces or empirical operator interpolations based on a split in the time domain.

In our numerical experiments, all the proposed algorithms and schemes proved to be applicable to three different partial differential equations, and showed a reduction in time of about one order of magnitude. The numerical experiments were chosen in order to gain experience on our path to the final goal, the simulation of immiscible two phase flow in a porous medium. This helped to finally construct a reduced scheme for such system of partial differential equations, whose simulation time could be reduced significantly in Chapter 6.

The developed a posteriori error bound turned out to be a very good choice in order to construct the reduced basis spaces by the developed greedy algorithm, and already produced fairly good efficiency results although the needed Lipschitz constants were not optimized as discussed in Chapter 3.

Finally, we analyzed our software packages used for numerical experiments, in order to develop an abstract software concept, that allows rapid prototyping of reduced basis methods for already available detailed schemes.

### **Perspectives**

The work presented in this thesis, gives rise to some obvious extension steps. The error bound efficiency should be improved by a more parameter sensitive computation of the used Lipschitz constants. This might lead to an improvement in the greedy generation of the reduced basis spaces, but at least enhances necessary reliability statements about reduced solutions.

Furthermore, the entire framework developed for the scalar evolution equations should be transferred to the two phase flow problem. Thus, in a next step, a parametrization could be introduced, and the greedy algorithms for the basis generation need to be adapted to the multi-variable case. This will also depend on efficient a posteriori error estimates.

In order to enhance the possibility to construct reduced basis method prototypes for existing Dune implementations, it is desirable to extend the existing software package DUNE-RB by methods for efficient empirical operator interpolation on this side. A solution was already proposed in Section 7.6.

## APPENDIX A

### **Constants for finite volume operators**

sdfasdf asdfa asdf

asdf



## Bibliography

- [1] M. Barrault, Y. Maday, N.C. Nguyen, and A.T. Patera. 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] P. Bastian, M. Blatt, A. Dedner, C. Engwer, R. Klöfkorn, R. Kornhuber, M. Ohlberger, and O. Sander. A Generic Grid Interface for Parallel and Adaptive Scientific Computing. Part II: Implementation and Tests in DUNE. *Computing*, 82(2–3):121–138, 2008.
- [3] P. Bastian, M. Blatt, A. Dedner, C. Engwer, R. Klöfkorn, M. Ohlberger, and O. Sander. A Generic Grid Interface for Parallel and Adaptive Scientific Computing. Part I: Abstract Framework. *Computing*, 82(2–3):103–119, 2008.
- [4] Peter Binev, Albert Cohen, Wolfgang Dahmen, Ronald DeVore, Guergana Petrova, and Przemyslaw Wojtaszczyk. Convergence rates for greedy algorithms in reduced basis methods. *SIAM J. Math. Anal.*, 43(3):1457–1472, 2011.
- [5] R.H. Brooks and A.T. Corey. Hydraulic properties of porous media. *Hydrology Papers*, 1964.
- [6] S.E. Buckley and M.C. Leverett. Mechanism of fluid displacement in sands. *Petroleum Technology*, 146(1337):107–116, 1942.
- [7] Annalisa Buffa, Yvon Maday, Anthony T. Patera, Christophe Prud'homme, and Gabriel Turinici. A priori convergence of the greedy algorithm for the parametrized reduced basis. *ESAIM-Math. Model. Numer. Anal.*, 46(3):595–603, 2012. Special Issue in honor of David Gottlieb.
- [8] C. Canuto, T. Tonn, and K. Urban. A-posteriori error analysis of the reduced basis method for non-affine parameterized nonlinear pde's. *SIAM J. Numer. Anal.*, 47(e):2001–2022, 2009.
- [9] K. Carlberg. *Model Reduction of Nonlinear Mechanical Systems via Optimal Projection and Tensor Approximation*. PhD thesis, Stanford University, 2011.

- [10] K. Carlberg, C. Bou-Mosleh, and C. Farhat. Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations. *International Journal for Numerical Methods in Engineering*, 86(2):155–181, 2011.
- [11] K. Carlberg, J. Cortial, D. Amsallem, M. Zahr, and C. Farhat. The GNAT nonlinear model reduction method and its application to fluid dynamics problems. In *6th AIAA Theoretical Fluid Mechanics Conference, Honolulu, Hawaii, June 27–30*, AIAA Paper 2011-3112, 2011.
- [12] K. Carlberg and C. Farhat. A low-cost, goal-oriented ‘compact proper orthogonal decomposition’ basis for model reduction of static systems. *International Journal for Numerical Methods in Engineering*, 86(3):381–402, 2011.
- [13] J. Carrillo. Entropy solutions for nonlinear degenerate problems. *Arch. Ration. Mech. Anal.*, 147(4):269–361, 1999.
- [14] S. Chaturantabut and D. C. Sorensen. A state space error estimate for POD–DEIM nonlinear model reduction. Technical report, CAAM, Rice U., December 2010. Technical Report TR10-32.
- [15] S. Chaturantabut and D. C. Sorensen. Application of POD and DEIM to dimension reduction of nonlinear miscible viscous fingering in porous media. *Math. Comput. Model. Dyn. Syst.*, 17(4):337–353, 2011.
- [16] S. Chaturantabut and D.C. Sorensen. Discrete empirical interpolation for nonlinear model reduction. *SIAM J. Sci. Comp.*, 32(5):2737–2764, 2010.
- [17] Y. Chen, J.S. Hesthaven, Y. Maday, and J. Rodríguez. A monotonic evaluation of lower bounds for inf-sup stability constants in the frame of reduced basis approximations. *Comptes Rendus Mathématique*, 346(23-24):1295–1300, 2008.
- [18] Zhangxin Chen, Guanren Huan, and Yuanlen Ma. *Computational methods for multiphase flows in porous media*. SIAM, 2006.
- [19] P.G. Ciarlet. *The finite element method for elliptic problems*. North-Holland, 1978.
- [20] A. Dedner, R. Klöforn, M. Nolte, and M. Ohlberger. A Generic Interface for Parallel and Adaptive Scientific Computing: Abstraction Principles and the DUNE-FEM Module. *Computing*, 90(3–4):165–196, 2010.
- [21] M. Dihlmann, M. Drohmann, and B. Haasdonk. Model reduction of parametrized evolution problems using the reduced basis method with adaptive time partitioning. In D. Aubry and P. Diez, editors, *International Conference on Adaptive Modeling and Simulation ADMOS 2011*, 2011.

- [22] M. Drohmann, B. Haasdonk, S. Kaulmann, and M. Ohlberger. A software framework for reduced basis methods using DUNE-RB and RB-matlab. In A. Dedner, B. Flemisch, and R. Klöfkorn, editors, *Advances in DUNE*. Springer, to appear.
- [23] M. Drohmann, B. Haasdonk, and M. Ohlberger. Reduced basis method for finite volume approximation of evolution equations on parametrized geometries. In *Proceedings of ALGORITMY 2009*, pages 111–120, 2008.
- [24] M. Drohmann, B. Haasdonk, and M. Ohlberger. Adaptive reduced basis methods for nonlinear convection-diffusion equations. In J. Fort et al., editor, *Finite Volumes for Complex Applications VI - Problems & Perspectives*, volume 1 of *Springer Proceedings in Mathematics 4*, pages 369–377. Springer, 2011.
- [25] M. Drohmann, B. Haasdonk, and M. Ohlberger. Reduced basis approximation for nonlinear parametrized evolution equations based on empirical operator interpolation. *SIAM J. Sci Comp*, 34:A937–A969, 2011. submitted.
- [26] Martin Drohmann, Bernard Haasdonk, Sven Kaulmann, and Mario Ohlberger. A software framework for reduced basis methods using `dune-rb` and `rbmatlab`. In Andreas Dedner, Bernd Flemisch, and Robert Klöfkorn, editors, *Advances in DUNE*, pages 77–88. Springer Berlin Heidelberg, 2012. 10.1007/978-3-642-28589-9\_6.
- [27] J. Eftang, D. Huynh, D. Knezevic, and A. Patera. A two-step certified reduced basis method. *Journal of Scientific Computing*, 51:28–58, 2012. 10.1007/s10915-011-9494-2.
- [28] Jens L. Eftang, Martin A. Grepl, and Anthony T. Patera. A posteriori error bounds for the empirical interpolation method. *Comptes Rendus Mathématique*, 348(9–10):575–579, May 2010.
- [29] Jens L. Eftang, David J. Knezevic, and Anthony T. Patera. An "hp" certified reduced basis method for parametrized parabolic partial differential equations. *Mathematical and Computer Modelling of Dynamical Systems*, 17(4):395–422, 2011.
- [30] Robert Eymard, Thierry Gallouët, and Raphaële Herbin. Finite volume methods. In *Handbook of numerical analysis, Vol. VII*, Handb. Numer. Anal., VII, pages 713–1020. North-Holland, Amsterdam, 2000.
- [31] M.A. Grepl. *Reduced-basis Approximations and a Posteriori Error Estimation for Parabolic Partial Differential Equations*. PhD thesis, Massachusetts

- Institute of Technology, May 2005.
- [32] M.A. Grepl, Y. Maday, N.C. Nguyen, and A.T. Patera. Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations. *M2AN, Math. Model. Numer. Anal.*, 41(3):575–605, 2007.
  - [33] M.A. Grepl and A.T. Patera. A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations. *M2AN, Math. Model. Numer. Anal.*, 39(1):157–181, 2005.
  - [34] B Haasdonk. Convergence rates of the pod-greedy method. Technical Report 23, SimTech Preprint 2011, University of Stuttgart, 2011.
  - [35] B. Haasdonk, M. Dihlmann, and M. Ohlberger. A training set and multiple bases generation approach for parametrized model reduction based on adaptive grids in parameter space. *Mathematical and Computer Modelling of Dynamical Systems*, 17(4):423–442, 2011.
  - [36] B. Haasdonk and M. Ohlberger. Adaptive basis enrichment for the reduced basis method applied to finite volume schemes. In *Proc. 5th International Symposium on Finite Volumes for Complex Applications*, pages 471–478, 2008.
  - [37] B. Haasdonk and M. Ohlberger. Reduced basis method for explicit finite volume approximations of nonlinear conservation laws. In *Proc. 12th International Conference on Hyperbolic Problems: Theory, Numerics, Application*, 2008.
  - [38] B. Haasdonk and M. Ohlberger. Reduced basis method for finite volume approximations of parametrized linear evolution equations. *M2AN, Math. Model. Numer. Anal.*, 42(2):277–302, 2008.
  - [39] B. Haasdonk, M. Ohlberger, and G. Rozza. A reduced basis method for evolution schemes with parameter-dependent explicit operators. *Electronic Transactions on Numerical Analysis*, 32:145–161, 2008.
  - [40] D. B. P. Huynh, D. J. Knezevic, and A. T. Patera. A static condensation reduced basis element method: Approximation and a posteriori error estimation. Technical report, (online), 2011.
  - [41] DBP Huynh, DJ Knezevic, JW Peterson, and AT Patera. High-fidelity real-time simulation on deployed platforms. *Computers & Fluids*, 43(1):74–81, 2011.
  - [42] D.B.P. Huynh, G. Rozza, S. Sen, and A.T. Patera. A successive constraint linear optimization method for lower bounds of parametric coercivity and inf-sup stability constants. *C. R. Math. Acad. Sci. Paris Series I*, 345:473–478, 2007.
  - [43] I.T. Jolliffe. *Principal component analysis*, volume 2. Springer-Verlag, 2002.
  - [44] Nadine Jung. *Error Estimation for Parametric Model Order Reduction and its Application*. PhD thesis, Technische Universität München, 2011.

- [45] M. Kahlbacher and S. Volkwein. Galerkin proper orthogonal decomposition methods for parameter dependent elliptic systems. *Discussiones Mathematicae: Differential Inclusions, Control and Optimization*, 27:95–117, 2007.
- [46] K.H. Karlsen and N.H. Risebro. On the uniqueness and stability of entropy solutions of nonlinear degenerate parabolic equations with rough coefficients. *Discrete Contin. Dyn. Syst.*, 9(5):1081–1104, 2003.
- [47] DJ Knezevic and JW Peterson. A high-performance parallel implementation of the certified reduced basis method. computer methods in applied mechanics and engineering. Technical report, preprint submitted to CMAME, 2010.
- [48] D. Kröner. *Numerical Schemes for Conservation Laws*. John Wiley & Sons and Teubner, 1997.
- [49] Oliver Lass and Stefan Volkwein. Adaptive pod basis computation for parametrized nonlinear systems using optimal snapshot location. Technical report, University of Konstanz, 2012.
- [50] T. Lassila, A. Manzoni, and G. Rozza. On the approximation of stability factors for general parametrized partial differential equations with a two-level affine decomposition. Technical report, MATHICSE, 2011. report 8.2011.
- [51] L. Machiels, A.T. Patera, J. Peraire, and Y. Maday. A general framework for finite element a posteriori error control: application to linear and nonlinear convection-dominated problems. In *Proc. ICFD Conference on Numerical Methods for Fluid Dynamics*, 1998.
- [52] Y. Maday. Reduced basis method for the rapid and reliable solution of partial differential equations. In European Mathematical Society, editor, *Proceedings of International Conference of Mathematicians*, pages 1–17, 2006.
- [53] Y. Maday, N.C. Nguyen, A.T. Patera, and G.S.H. Pau. A general, multipurpose interpolation procedure: the magic points. *Communications on Pure and Applied Analysis*, 8(1):383–404, January 2009.
- [54] Yvon Maday, Anthony T. Patera, and Gabriel Turinici. A priori convergence theory for reduced-basis approximations of single-parameter elliptic partial differential equations. *Journal of Scientific Computing*, 17:437–446, 2002. 10.1023/A:1015145924517.
- [55] A. Manzoni, A. Quarteroni, and G. Rozza. Computational reduction for parametrized pdes: strategies and applications. Technical Report Report 05.2012, MATHICSE, EPFL, Lausanne, CH, 2012.
- [56] Andrea Manzoni, Alfio Quarteroni, and Gianluigi Rozza. Model reduction techniques for fast blood flow simulation in parametrized geometries. *International Journal for Numerical Methods in Biomedical Engineering*, 28(6-7):604–625, 2012.
- [57] A. Michel. A finite volume scheme for two-phase immiscible flow in porous media. *SIAM Journal on Numerical Analysis*, 41(4):1301–1317, 2004.

- [58] N. C. Nguyen, G. Rozza, and A.T. Patera. Reduced basis approximation and a posteriori error estimation for the time-dependent viscous burgers' equation. *Calcolo*, 46(3):157–185, 2009.
- [59] NC Nguyen, AT Patera, and J. Peraire. A ‘best points’ interpolation method for efficient approximation of parametrized functions. *International journal for numerical methods in engineering*, 73(4):521–543, 2008.
- [60] N.C. Nguyen, K. Veroy, and A.T. Patera. Certified real-time solution of parametrized partial differential equations. In S. Yip, editor, *Handbook of Materials Modeling*, pages 1523–1558. Springer, 2005.
- [61] A.T. Patera and G. Rozza. *Reduced Basis Approximation and a Posteriori Error Estimation for Parametrized Partial Differential Equations*. MIT, 2007. Version 1.0, Copyright MIT 2006-2007, to appear in (tentative rubric) MIT Pappalardo Graduate Monographs in Mechanical Engineering.
- [62] Jan Pomplun and Frank Schmidt. Accelerated a posteriori error estimation for the reduced basis method with application to 3d electromagnetic scattering problems. *SIAM Journal on Scientific Computing*, 32(2):498–520, 2010.
- [63] A. Quarteroni, R. Sacco, and F. Saleri. *Numerical mathematics*. Texts in Applied Mathematics. Springer, Berlin Heidelberg, 2007.
- [64] C.W. Rowley. Model reduction for fluids, using balanced proper orthogonal decomposition. *Int. J. Bifurcat. Chaos*, 15(3):997–1013, 2005.
- [65] G. Rozza. *Shape design by optimal flow control and reduced basis techniques: Applications to bypass configurations in haemodynamics*. PhD thesis, École Polytechnique Fédérale de Lausanne, November 2005.
- [66] G. Rozza, D.B.P. Huynh, and A.T. Patera. Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations. *Arch. Comput. Meth. Eng.*, 15(3):229–275, 2007.
- [67] Will Schroeder, Ken Martin, and Bill Lorensen. *The Visualization Toolkit: An Object-Oriented Approach To 3D Graphics*. Kitware, Inc., 4th edition edition, 2006.
- [68] V. N. Temlyakov. Greedy approximation. *Acta Numerica*, 17:235–409, 2008.
- [69] D.B. Thomas, W. Luk, P.H.W. Leong, and J.D. Villaseñor. Gaussian random number generators. *ACM Computing Surveys (CSUR)*, 39(4):11, 2007.
- [70] T. Tonn and K. Urban. A reduced-basis method for solving parameter-dependent convection-diffusion problems around rigid bodies. In P. Wesseling, E. Onate, and J. Periaux, editors, *ECCOMAS CFD 2006 Proceedings*, 2006.
- [71] Timo Tonn. *Reduced-Basis Method (RBM) for Non-Affine Elliptic Parametrized PDEs*. PhD thesis, University of Ulm, 2011.

- [72] M.Th. van Genuchten. A Closed-form Equation for Predicting the Hydraulic Conductivity of Unsaturated Soils. *Soil Sci. Soc. Am. J.*, 44:892–898, 1980.
- [73] K. Veroy and A.T. Patera. Certified real-time solution of the parametrized steady incompressible Navier-Stokes equations: Rigorous reduced-basis a posteriori error bounds. *Int. J. Numer. Meth. Fluids*, 47:773–788, 2005.
- [74] K. Veroy, C. Prud’homme, and A.T. Patera. Reduced-basis approximation of the viscous Burgers equation: rigorous a posteriori error bounds. *C. R. Math. Acad. Sci. Paris Series I*, 337:619–624, 2003.
- [75] <http://morepas.org/software/>.



## Erklärung

Hiermit versichere ich, Martin Drohmann, dass ich die Arbeit selbstständig und nur mit den angegebenen Quellen und Hilfsmitteln erstellt habe. Inhaltlich oder wörtlich übernommene Stellen, wurden als solche gekennzeichnet. Ich erkläre außerdem, dass die von mir vorgelegte Dissertation in keinem anderen Promotionsverfahren im In- und Ausland in dieser oder ähnlicher Form vorgelegt wurde.

Münster, den 24. Juni 2012

---

gez. Martin Drohmann