Efficient Schemes for Parameterized Multiscale Problems

Von der Fakultät Mathematik und Physik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von
Sven Alebrand (geb. Kaulmann)
aus Kirchen

Hauptberichter: Prof. Dr. Bernard Haasdonk
Mitberichter: Prof. Dr. Mario Ohlberger
Tag der mündlichen Prüfung: 16.12.2014

Institut für Angewandte Analysis und Numerische Simulation der Universität Stuttgart
2015

Für Leonie.

Danksagungen

Ich danke meinen Betreuern Herrn Prof. Dr. Bernard Haasdonk und Herrn Prof. Dr. Mario Ohlberger für ihre vielfältige Unterstützung in den letzten Jahren und für die vielen Impulse zur Thematik.

Weiterhin danke ich Leonie Alebrand, Stefan Girke, Patrick Henning und René Milk für Korrekturvorschläge zu dieser Arbeit und Simeon Steinig für die gute Atmosphäre in Stuttgart.

Auch meinen Eltern Rosi und Klaus Kaulmann, die mich immer, auch in schwierigen Phasen, unterstützten und antrieben, gilt mein Dank.

Meiner Frau Leonie Alebrand kann ich für die unendliche Geduld und Stärkung, die sie mir jeden Tag zeigt, gar nicht genug danken. Ohne sie wäre diese Arbeit vermutlich nicht zustande gekommen.

Contents

List of Figures

List of Tables

Acronyms

BiCGStab biconjugate gradient stabilized. 135

DEIM discrete empirical interpolation method. 33, 39

DG discontinuous Galerkin. 29, 41, 43, 45, 46, 92, 138, 152

EIM empirical interpolation method. 33, 39, 47

FE finite element. 29

FV finite volume. 29

GMsFEM generalized multiscale finite element method. 128

HMM heterogeneous multiscale method. 24, 128, 132

IMPES implicit pressure, explicit saturation. 92

LRBMS localized reduced basis multiscale method. 41, 127, 138

MMsFEM mixed multiscale finite element method. 128

MsFEM multiscale finite element method. 13, 23, 127–130, 132, 133, 135–139, 147, 149, 152

MSFV multiscale finite volume method. 127

PCA principal component analysis. 12, 13, 28, 78, 79, 84, 85, 90, 91, 99, 100, 103, 104, 107, 110–112, 114–116, 120, 123, 124, 147

PDE partial differential equation. 27, 39

PPDE parameterized partial differential equation. 27

RB reduced basis. 27, 29–31, 36, 53, 127

RBEM reduced basis element method. 41

RBHM reduced basis hybrid method. 41

RDFM reduced basis domain decomposition finite element method. 41

SCM successive constraint method. 37

VMM variational multiscale method. 23, 127, 132

Abstract

This thesis investigates efficient schemes for parameterized multiscale problems. The reduced basis method is a well-known technique for the reduction of computational effort for parameterized partial differential equations. Herein two extensions of the methodology are introduced.

First, an extension to problems with high parameter dimension is suggested. This so-called online greedy basis construction approach relies on building parameter-dependent reduced-dimensional approximation spaces during the main computational phase, the so-called online-phase. Bases constructed using the online greedy basis construction are much smaller than those constructed using conventional greedy methods and runtime improvements can be observed for certain cases.

Secondly, the reduced basis method is combined with ideas from so-called multiscale methods to make it applicable to problems with multiscale character by overcoming the lack of control over the runtime of its preparatory phase, the so-called offline-phase. It is established that a novel approach, the localized reduced basis multiscale method, allows to displace the computational effort between the two phases of the reduced basis method. This technique is applied to stationary heat diffusion problems and to two-phase flow in porous media.

Additional performance can be reached by using multiscale meth-

ods as efficient solvers during the preparatory phase of the localized reduced basis method. A parallelization concept for these methods is introduced and scaling tests for an implementation of the concept are presented.

Finally some details on our implementations of multiscale methods and of the aforementioned extensions to the reduced basis method are presented.

Zusammenfassung

Diese Dissertation untersucht effiziente Algorithmen zur Behandlung von parametrisierten Mehrskalenproblemen. Eine Standardmethode der Modellreduktion für parametrisierte partielle Differentialgleichungen ist die Reduzierte Basis Methode. In dieser Arbeit werden zwei Erweiterungen der Methodik vorgestellt.

Zuerst wird eine Erweiterung der Reduzierte Basis Methode auf Probleme mit hoher Parameterdimension eingeführt. Dazu wird ein Algorithmus zur Konstruktion eines parameterabhängigen Approximationsraumes niedriger Dimension definiert. Dieser Algorithmus – ausgeführt während der eigentlichen Simulationsphase der Reduzierte Basis Methode, der sogenannten Online-Phase – ermöglicht deutlich kleinere reduzierte Basen als herkömmliche Ansätze und unter bestimmten Voraussetzungen werden reduzierte Laufzeiten im Vergleich zur herkömmlichen Basiskonstruktion erreicht.

Weiterhin wird die Reduzierte Basis Methode mit Ideen aus dem Bereich der Mehrskalenmethoden verknüpft. Es wird damit möglich, die Laufzeit der vorbereitenden Phase der Reduzierte Basis Methode, der sogenannten Offline-Phase, zu kontrollieren. Anhand von Anwendungen im Bereich von stationärem Wärmefluss und Mehrphasenströmungen in porösen Medien wird demonstriert, dass die entstehende neue Methode, die Lokalisierte Reduzierte Basis Mehrskalenmethode eine Verschiebung des Berechnungsaufwands zwischen den beiden Phasen der Reduzierte Basis Methode erlaubt und sich

der neue Ansatz damit im Bereich von Mehrskalenproblemen anwenden lässt.

Weitere Beschleunigung der Simulation kann durch Anwendung von Mehrskalenmethoden als effiziente Löser während der Offline-Phase der Lokalisierten Reduzierte Basis Mehrskalenmethode erreicht werden. Ein Parallelisierungskonzept für diese Methoden wird eingeführt und anhand von Skalierungstests untersucht.

Abschließend werden Implementierungen dieses Parallelisierungskonzepts und der oben genannten Erweiterungen der Reduzierte Basis Methode vorgestellt.

Chapter 1

Introduction

As numerical methods find more and more application in real-world scenarios as for example oil reservoir engineering or CO_2 storage modeling, demands regarding efficiency and reliability of these methods also increase. Both oil reservoir engineering and CO_2 storage are perfect examples for scenarios with influential physical properties on different scales: both can usually be modeled by two-phase flow equations in a porous medium where variations in the small-scale quantities describing the porous medium (porosity, permeability) can influence the global properties of the flow fields heavily. This calls for discrete models of the flow resolving the small scale details which in turn easily leads to equation systems with millions of unknowns for the objective variables (usually the phase pressures and phase saturations). Additional complexity is added to the problem when the flow pattern is not only needed for one fixed setting of the physical properties and model parameters but rather in a multi-query context. This is the case for optimization scenarios in oil reservoir engineering, for example, where optimal well positions and optimal production rates are desired. This may lead to equations parameterized by boundary values, initial conditions, inflow and outflow positions and others.

So-called *multiscale methods*, such as the MsFEM [28], the vari-

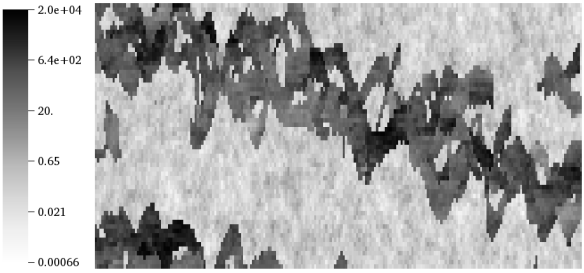


Figure 1.1 – Global channels and fine scale details in the permeability field of the SPE 10 benchmark [18].

ational multiscale method (VMM) [47] or the heterogeneous multiscale method (HMM) [26] aim at efficient solution schemes for multiscale problems by incorporating information about the fine scale details of the flow into a new basis on a coarser scale. These methods allow to break down the effort to solution for the afore-mentioned multiscale problems into several smaller parts and are hence usually a lot more efficient with respect to memory usage and have promising features regarding parallelization.

Reduced basis methods [69], as a typical example for model order reduction techniques, aim at solving the second problem: They provide means to reduce the computational effort for parameterized partial differential equations by introducing two computational phases: In a time-consuming preparatory step, the so-called *offline phase*, a reduced-dimensional surrogate space – the reduced basis space – is computed that is used as ansatz space during the now rapid *online phase*. The number of degrees of freedom during the online-phase usually does not depend on the system size arising from the underlying (finite element, discontinuous Galerkin, finite volume, ...) discretization and hence huge speed-ups over those discretizations can be reached while ensuring approximation quality via rigorous and efficiently evaluable a-posteriori error estimators.

In this thesis we study how model problems like the parameterized multiscale problems named above can be treated efficiently. As a means to this end, several schemes are investigated.

First, we introduce a new basis generation algorithm for reduced

basis methods, the *online greedy* algorithm. The aim of this algorithm is to adapt the reduced basis method to problems with high parameter dimensions and problems with high sensitivity with respect to the parameter. For both cases, the standard reduced basis approach usually shows poor performance.

Secondly, we introduce a new approach combining ideas from multiscale and model order reduction methods: The *localized reduced basis multiscale* method computes localized reduced basis spaces and decreases the workload during the offline phase which can be prohibitively large for the traditional reduced basis method when applied to multiscale problems.

Thirdly, we present a parallelization concept for multiscale methods, especially for the multiscale finite element method. As multiscale methods find more and more application in engineering science and industry, scalable computations on high-performance clusters become an interesting field of research. Also – the other way around – multiscale methods show high potential for good scaling even to *exa-scale* architectures and hence are worth studying even without having the above-named applications in mind. Here, the term *exa-scale* refers to computational systems operating at a rate of 10^{18} floating point operations per second. These systems are expected to govern high-performance computing in the next decade and pose a great challenge to all parts of a numerical simulation work-flow, especially to numerical algorithms.

Most parts of this thesis were published in journal articles and conference proceedings beforehand. In [56] we introduced the online greedy basis construction algorithm for stationary elliptic problems, in [23] we introduced a similar method for instationary convection-diffusion problems.

The concept of localized reduced basis spaces was originally introduced in [57] using different finite element schemes for the local and global problems. In [2] we applied this idea using the same discontinuous formulation on the local and global scale. Finally, in [55] we introduced a novel formulation of two-phase flow using our localized reduced basis approach.

This thesis is structured as follows: in the next chapter, we introduce the reduced basis method including error estimation in detail. Chapter 3 describes the parameterized model for two-phase

flow that will be used throughout the work and its discretization using a discontinuous Galerkin method and the multiscale finite element method. The above-mentioned online greedy basis construction algorithm is introduced in Chapter 4. Chapter 5 introduces the localized reduced basis multiscale method and its application to stationary heat diffusion and two-phase flow problems. Finally, in Chapter 7 we introduce software concepts for the implementation of reduced basis methods and for the parallelization of multiscale methods.

Chapter 2

Reduced Basis Methods

As outlined in the introduction in Chapter 1, the reduced basis (RB) method is a model order reduction technique for a special class of partial differential equations (PDEs), so-called parameterized partial differential equations (PPDEs). In these equations, typically one or more input values like boundary values, initial data or physical data like diffusion coefficients, velocity fields or others are treated as parameters.

In an optimization context these problems then, for example, need to be solved numerous times and a need for model order reduction arises that allows the numerous (probably to some extend similar) solutions to be computed at reduced cost. Another example for these so-called *many-query* contexts is the application used in Chapter 5: Here model order reduction is used to reduce the workload produced by solving the equation for the unknown phase pressure in each step of a time stepping scheme.

In *real-time* contexts, which provide another class of motivating examples for model order reduction techniques, responses from the PDE are needed ideally instantaneously. This is the case in optimal control settings, for example.

For both scenarios, the idea of the reduced basis method and its ancestors is to provide function spaces that are specifically fit to

approximate the manifold of all solutions for the equation at hand for all possible realizations of the parameters.

First studies comprising the idea of problem-specific approximation spaces for parameterized problems date back to the 1970s and 1980s, see [69] for an overview. It was then recognized that a-posteriori error estimation is an important ingredient for successful application in the context of parametric PDEs [59, 60] both for the construction of reduced bases using greedy-type algorithms [79] and for online approximation quality certification. After being applied to elliptic problems [72, 71, 69, 67] and linear and nonlinear parabolic problems [34, 74, 33], the methodology was extended to cover linear and non-linear hyperbolic problems [37, 39, 38].

The list of similar approaches includes the PCA method [52, 15, 53], which is also known as Karhunen-Loève expansion or proper orthogonal decomposition method. This method aims at finding best approximation spaces from a given data set, in this context usually a set of solution snapshots of the PDE. Other related approaches are the balanced truncation method and Krylov-subspace methods [3].

We will now introduce the RB method in its most basic form (Sections 2.1 to 2.4) and give some details on a broadly used a posteriori error estimator (Section 2.5). Finally, in Section 2.6 we will give some pointers on recent developments in the field of RB research.

2.1 Model Problem

The key model problem for the introduction of the RB method will be the parameterized partial differential equation (in the weak form) given as: For some given suitable Hilbert-space $\mathcal{X} \subset L^2(\Omega)$ on a domain Ω let $u(\mu) \in \mathcal{X}$ be given as the solution to

$$b(u(\mu), v; \mu) = l(v; \mu) \quad \forall v \in \mathcal{X}, \tag{2.1.1}$$

for $\mu \in \mathcal{P}$ where $\mathcal{P} \subset \mathbb{R}^p$ for $p \in \mathbb{N}$ denotes a compact set and $b : \mathcal{X} \times \mathcal{X} \times \mathcal{P} \to \mathbb{R}$ denotes a given parameterized bilinear form, that is: For every $\mu \in \mathcal{P}$, $b(\cdot, \cdot; \mu)$ defines a bilinear form. Correspondingly $l : \mathcal{X} \times \mathcal{P} \to \mathbb{R}$ denotes a given parameterized linear form. The compact set $\mathcal{P}$ will be referred to as the *parameter set*.

Such equations arise, for example, in the context of heat diffusion. In this case, the parameter μ could model the diffusion coefficient, boundary values or a heat source in the domain.

In the sequel we will assume $b(\cdot, \cdot; \mu)$ to be symmetric. Furthermore, we assume coercivity and continuity for b:

Assumption 2.1 (Coercivity and Continuity of b)**.** *We assume b to be coercive uniformly with respect to the parameter μ, that is*

$$\alpha(\mu) := \inf_{v \in \mathcal{X} \setminus \{0\}} \frac{b(v, v; \mu)}{\|v\|^2} > 0,$$

where here and later $\| \cdot \|$ denotes the norm in $\mathcal{X}$ induced by its scalar product $\langle \cdot, \cdot \rangle$, and there exists $\alpha > 0$ such that $\alpha \leq \alpha(\mu)$ for all $\mu \in \mathcal{P}$. Furthermore we assume b to be continuous uniformly with respect to the parameter μ, that is

$$\gamma(\mu) := \sup_{v, w \in \mathcal{X} \setminus \{0\}} \frac{b(v, w; \mu)}{\|v\| \|w\|} < \infty$$

and there exists $\gamma \in \mathbb{R}$ such that $\gamma(\mu) \leq \gamma$ for all $\mu \in \mathcal{P}$.

Remark 2.2. *Existence and uniqueness of solutions to* (2.1.1) *is then guaranteed by the Lax-Milgram theorem if the right hand side form l is continuous for all parameters, that is if*

$$\sup_{v \in \mathcal{X} \setminus \{0\}} \frac{l(v; \mu)}{\|v\|} < \infty, \quad \forall \mu \in \mathcal{P}.$$

The basis for the introduction of the reduced basis (RB) method is a discrete formulation of problem (2.1.1). Different methods such as finite volume (FV), discontinuous Galerkin (DG) or finite element (FE) methods have been used as a basis for the RB method. For now we only assume a discrete space $\mathcal{V}_h \subset \mathcal{X}$ to be given and introduce the *high-dimensional* solution of (2.1.1):

Definition 2.3 (High-Dimensional Solution)**.** Given a discrete function space $\mathcal{V}_h \subset \mathcal{X}$ of dimension $\mathcal{N} = \dim(\mathcal{V}_h)$, where $\dim(\mathcal{V}_h)$ denotes the length of a basis of $\mathcal{V}_h$, the *high-dimensional solution*

$u_h(\mu) \in \mathcal{V}_h$ to Equation (2.1.1) for a parameter $\mu \in \mathcal{P}$ is given as the solution to

$$b(u_h(\mu), v; \mu) = l(v; \mu) \quad \forall v \in \mathcal{V}_h. \tag{2.1.2}$$

As usual in RB methods, we will consider Definition 2.3 a *replacement* for the weak formulation (2.1.1): We assume the error between analytical and high-dimensional solutions to be negligible and compare our reduced approximations only to the high-dimensional ones.

2.2 Reduced Basis Approximation

The RB method reduces the complexity of computing a solution to the equation at hand from $\mathcal{N} = \dim(\mathcal{V}_h)$ to $N \in \mathbb{N}$, $N \ll \mathcal{N}$, by introducing a low-dimensional surrogate X_N, $\dim(X_N) = N$ of the high-dimensional discretization space $\mathcal{V}_h$. This space X_N is the linear span of high-dimensional solutions for a given set of parameter values.

Definition 2.4 (Reduced Basis Space)**.** For a given set of parameters $\mu_1, \ldots, \mu_k \in \mathcal{P}$ stemming from the parameter set $\mathcal{P}$, the *reduced basis space* X_N is given as

$$X_N = \operatorname{span}\left(\{u_h(\mu_1), \ldots, u_h(\mu_k)\}\right).$$

Here and in the following, $\operatorname{span}(\{v_1, \ldots, v_N\})$ denotes the linear span of the set of vectors $\{v_1, \ldots, v_N\}$. Furthermore, we denote by $\Phi = \{\varphi_1, \ldots \varphi_N\}$, where N is the dimension of the linear space X_N, an orthonormal basis of X_N.

The space X_N is used to define a reduced-dimensional surrogate for Equation (2.1.2) where we look for a solution $u_N(\mu) \in X_N$ for a given parameter $\mu \in \mathcal{P}$:

$$b(u_N(\mu), v_N; \mu) = l(v_N; \mu) \quad \forall v_N \in X_N. \tag{2.2.1}$$

Remark 2.5. *As Assumption 2.1 ensures coercivity and continuity on the reduced space X_N (due to $X_N \subset \mathcal{X}$), the reduced-dimensional equation* (2.2.1) *has a unique solution: Equation* (2.2.1) *is a linear*

problem in finite space dimension. Therefore existence and uniqueness of solutions are equivalent. Let $u_1(\mu)$ *and* $u_2(\mu)$ *be two solutions to* (2.2.1)*. We then have*

$$\begin{aligned} &b(u_1(\mu) - u_2(\mu), u_1(\mu) - u_2(\mu); \mu) \\ &\quad = l(u_1(\mu) - u_2(\mu); \mu) - l(u_1(\mu) - u_2(\mu); \mu) \\ &\quad = 0. \end{aligned}$$

Therefore

$$\begin{aligned} 0 &= b(u_1(\mu) - u_2(\mu), u_1(\mu) - u_2(\mu); \mu) \\ &\geq \alpha \|u_1(\mu) - u_2(\mu)\| \end{aligned}$$

with $\alpha > 0$*. Hence we have* $\|u_1(\mu) - u_2(\mu)\| = 0$*, that is* $u_1(\mu) = u_2(\mu)$*.*

Introducing the vector $\boldsymbol{u}_N(\mu) \in \mathbb{R}^N$, the matrix-valued function $\boldsymbol{A}_N : \mathcal{P} \to \mathbb{R}^{N \times N}$ with $(\boldsymbol{A}_N(\mu))_{i,j} = b(\varphi_j, \varphi_i; \mu)$, and the vector-valued function $\boldsymbol{b}_N : \mathcal{P} \to \mathbb{R}^N$ with $(\boldsymbol{b}_N(\mu))_i = l(\varphi_i; \mu)$, the reduced dimensional equation system for Equation (2.2.1) is given as

$$\boldsymbol{A}_N(\mu) \cdot \boldsymbol{u}_N(\mu) = \boldsymbol{b}_N(\mu), \tag{2.2.2}$$

which is of much smaller size ($N \ll \mathcal{N}$) than the arising equation system for the high-dimensional discretization (Definition 2.3) but typically dense. The computational effort for solving is thus reduced from $\mathcal{O}(\mathcal{N}^2)$ to $\mathcal{O}(N^3)$. The complexity of the assembly of the reduced-dimensional system (2.2.2) still depends on $\mathcal{N}$, though. In the next paragraph we will therefore introduce a scheme that decouples the overall complexity of (2.2.2) from $\mathcal{N}$.

2.3 Offline–Online Decomposition

The main idea of the RB method is the so-called *offline–online splitting* of all computations: We introduce two phases of RB computations:

Offline phase During this phase, all $\mathcal{N}$-dependent computations are performed. In particular, the linear subspace X_N of the high-dimensional space $\mathcal{V}_h$ is built up with the aim of finding the

best approximation of the manifold of all possible solutions $\mathcal{M}(\mathcal{P}) = \{u(\mu)|\mu \in \mathcal{P}\}$. This phase may be very expensive as a certain amount of solutions $u_h(\mu_i)$ and potentially additional data like operator projections need to be computed.

Online phase During this phase, the equation at hand is solved in the reduced space X_N for a given parameter μ. This phase is ideally totally independent of $\mathcal{N}$, and therefore usually very fast.

While the construction of X_N can clearly be done during the offline phase, the assembly of the equation system (2.2.2) needs to be done for every new given parameter μ during the online phase. As this requires evaluations of b at the solutions $u_h(\mu_i)$, the online phase would depend on $\mathcal{N}$. We thus make the assumption of parameter-separability:

Assumption 2.6. *Assume b, l to be parameter-separable, that is for all $u, v \in \mathcal{V}_h$ we assume*

$$b(u, v; \mu) = \sum_{q=1}^{Q_b} \Theta_b^q(\mu) b^q(u, v),$$
$$l(u; \mu) = \sum_{q=1}^{Q_l} \Theta_l^q(\mu) l^q(u)$$

for a set of parameter-dependent functions $\Theta_b^q, \Theta_l^q : \mathcal{P} \to \mathbb{R}$, a set of parameter-independent bilinear and, respectively, linear forms $b^q : \mathcal{V}_h \times \mathcal{V}_h \to \mathbb{R}$, $l^q : \mathcal{V}_h \to \mathbb{R}$ and given numbers $Q_b, Q_l \in \mathbb{N}$.

Using Assumption 2.6, the assembly of the system (2.2.2) can be done in two steps: During the offline phase, after computing the reduced basis space X_N, project the parameter-independent components of b and l to X_N:

$$\begin{aligned} (\boldsymbol{A}_N^q)_{i,j} &= b^q(\varphi_j, \varphi_i), \quad 1 \le q \le Q_b, 1 \le i, j \le N, \\ (\boldsymbol{b}_N^q)_i &= l^q(\varphi_i), \quad 1 \le q \le Q_l, 1 \le i \le N. \end{aligned} \tag{2.3.1}$$

During the online phase, it then only remains to sum up the precomputed components:

$$\boldsymbol{A}_N(\mu) = \sum_{q=1}^{Q_b} \Theta_b^q(\mu)\boldsymbol{A}_N^q, \qquad \boldsymbol{b}_N(\mu) = \sum_{q=1}^{Q_l} \Theta_l^q(\mu)\boldsymbol{b}_N^q. \tag{2.3.2}$$

If parameter separability can not be fulfilled for the equation at hand, approximation techniques like the empirical interpolation method (EIM) [8], the empirical operator interpolation method [39, 24] or the discrete empirical interpolation method (DEIM) [14] can be applied, see Section 2.6. For the sake of simplicity of the exposition of our key points we will nevertheless henceforth assume that an affine separation as in Assumption 2.6 is possible for the given bilinear forms.

2.4 Reduced Basis Construction

It remains to clarify how the parameters $\mu_1, \dots, \mu_k$ for the construction of the reduced space X_N in Definition 2.4 are chosen. In the following we introduce a greedy-type algorithm [79] that is frequently used in the construction of RB spaces and give some hints on approximation quality using this algorithm.

2.4.1 Greedy Algorithm

Let the *training set* $\mathcal{P}_{\text{tr}}$, $\mathcal{P}_{\text{tr}} \subset \mathcal{P}$, be a finite subset of the parameter set. The training set can for example be comprised of a set of random points in $\mathcal{P}$ or the set of vertices of a structured mesh in the parameter set. Furthermore we assume an efficient and reliable a-posteriori error estimator $\Delta_{X_i} : \mathcal{P} \to [0, \infty)$ with

$$\|u_h(\mu) - u_N(\mu)\| \leq \Delta_{X_i}(\mu)$$

for $\mu \in \mathcal{P}$ to be given. Details on the estimator used throughout this work are given in Section 2.5.

Algorithm 2.7 (Greedy Basis Construction)**.** *Choose an initialization parameter $\mu_{init} \in \mathcal{P}$, a maximum basis size $N_{\max} \in \mathbb{N}$,*

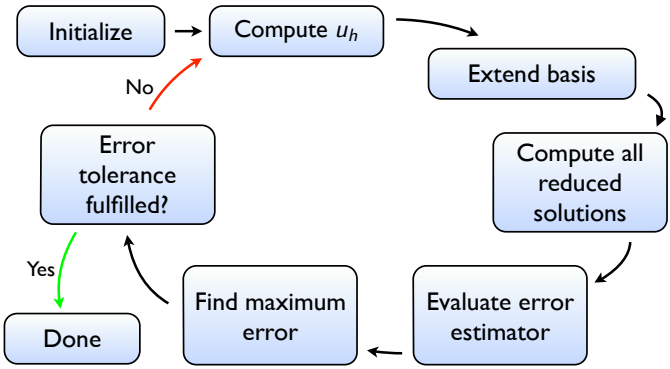


Figure 2.1 – Illustration of the greedy basis construction method (Algorithm 2.7).

$N_{\max} > 0$, a tolerance $\epsilon > 0$ and a training set $\mathcal{P}_{tr} \subset \mathcal{P}$. Furthermore, let a projection $\omega_\Psi : \mathcal{V}_h \to \mathcal{V}_h$ for a subset $\Psi \subset \mathcal{V}_h$ be given. The basis construction algorithm then proceeds as follows:

1. *Initialize the reduced basis: Set $\Phi_0 = \{\omega_\Phi(u_h(\mu_{init}))\}$ and $X_0 = \operatorname{span}(\Phi_0)$*
2. *For $i = 0, \dots, N_{\max}$:*
   a) *Compute the reduced-dimensional solutions $u(\mu) \in X_i$ for all parameters $\mu \in \mathcal{P}_{tr}$ (see Equation (2.2.1)).*
   b) *Evaluate the error estimator for all parameters in the training set:*

$$e_\mu = \Delta_{X_i}(\mu), \quad \forall \mu \in \mathcal{P}_{tr}.$$

   If $\max_{\mu \in \mathcal{P}_{tr}} e_\mu < \epsilon$: Set $N = i$, $X_N = X_i$ and break.
   c) *Find the parameter giving the worst approximation in the current basis:*

$$\mu_{\max} = \arg\max_{\mu \in \mathcal{P}_{tr}} e_\mu.$$

   If multiple parameters fulfill this criterion, select one randomly.

d) Compute the high-dimensional solution $u_h(\mu_{\max})$ *for the parameter* $\mu_{\max}$ *using* (2.1.2).

e) Extend the reduced basis with $u_h(\mu_{\max})$*:*

$$\Phi_{i+1} = \Phi_i \cup \{\omega_{\Phi_i}(u_h(\mu_{\max}))\}$$

f) Set $X_{i+1} = \operatorname{span}(\Phi_{i+1})$*. If* $i = N_{\max}$*: Set* $N = i$*,* $X_N = X_i$*.*

A widely used example for the mapping $\omega_\Psi : \mathcal{V}_h \to \mathcal{V}_h$ is the Gram-Schmidt orthogonalization in conjunction with an normalization with respect to a certain norm. Figure 2.1 illustrates the course of action of Algorithm 2.7.

2.4.2 Approximation Quality

Using a straightforward computation one can show a result similar to Céa's lemma:

$$\|u_h(\mu) - u_N(\mu)\| \leq \left(\frac{\gamma(\mu)}{\alpha(\mu)}\right)^{1/2} \inf_{v_h \in X_N} \|u_h(\mu) - v_h\|.$$

In other words: the reduced basis scheme (2.2.1) computes a quasi best-approximation in the reduced basis space X_N. The question of approximation quality of the reduced basis method is therefore brought down to the question of optimal construction of the space X_N. In [13, 12] the notion of the Kolmogorov n-width is used to answer this question. The Kolmogorov n-width $d_n(\mathcal{M})$ of a compact subset $\mathcal{M}$ of a Hilbert space $\mathcal{X}$ is defined as

$$d_n(\mathcal{M}) = \inf_{Y \subset \mathcal{X}, \dim(Y) = n} \sup_{v \in \mathcal{M}} \inf_{w \in Y} \|v - w\|,$$

where $Y \subset \mathcal{X}$ denotes a linear subspace. It can then be demonstrated that if the reduced basis is constructed using Algorithm 2.7 and if the Kolmogorov n-width of $\mathcal{M}(\mathcal{P}) = \{u(\mu) | \mu \in \mathcal{P}\}$ decays polynomially, the reduced basis method converges polynomially: If

$$d_k(\mathcal{M}(\mathcal{P})) \leq Mk^{-a},$$

for $a > 0$ and $M > 0$, then for a given approximation quality ε of the high-dimensional scheme ($\|u(\mu) - u_h(\mu)\| \leq \varepsilon$):

$$\|u(\mu) - u_k(\mu)\| \leq C \max\left\{Mk^{-a}, \varepsilon\right\} \quad \forall \mu \in \mathcal{P},$$

where $C := (1 + 2^a)q^{1/2}(4q)^a$, $q := \lceil 2^{a+2}\tau^{-1}\rceil^2$ and $u_k(\mu)$ denotes the reduced solution in the RB space of dimension k. Here, for a given real number $r \in \mathbb{R}$, $\lceil r \rceil$ denotes the smallest natural number $m \in \mathbb{N}$ such that $r \leq m$ and the constant $\tau \in (0, 1]$ depends on the training set $\mathcal{P}_{\text{tr}}$ and the estimator Δ_{X_i}.

A similar statement holds if the convergence of the Kolmogorov n-width is exponential, see [12].

These results were extended to basis construction using the *POD-Greedy* method [39] for instationary problems in [35].

2.5 Error Estimation

One crucial ingredient of the reduced basis method is *a posteriori* error estimation. Error estimation is used for basis construction during the offline phase in Algorithm 2.7 and for certification by approximation quality control during the online phase. In our work we use a residual-based estimator which is shortly outlined in this paragraph. For more details we refer to [69, 2].

Definition 2.8. For b given from (2.1.1) we denote by $\|\cdot\|_\mu : \mathcal{V}_h \to [0, \infty)$ the parameter-dependent *energy* norm

$$\|u\|_\mu = b(u, u; \mu)^{1/2}.$$

Next we introduce the residual and its representation via the Riesz representation theorem.

Definition 2.9. For a given function $u \in \mathcal{V}_h$ let the *residual* $r_h[u] : \mathcal{V}_h \times \mathcal{P} \to \mathbb{R}$ be given by

$$r_h[u](v; \mu) := l(v; \mu) - b(u, v; \mu) \quad \forall v \in \mathcal{V}_h.$$

As b defines a scalar product on $\mathcal{V}_h$, the residual can be represented by $r_u(\mu) \in \mathcal{V}_h$, via the Riesz representation theorem, given a parameter $\overline{\mu} \in \mathcal{P}$

$$b(r_u(\mu), v; \overline{\mu}) = r_h[u](v; \mu) \quad \forall v \in \mathcal{V}_h.$$

We can now state the residual-based a posteriori error estimator that will be used in the rest of this work.

Theorem 2.10 (A Posteriori Error Estimate)**.** *Assume* $\Theta_b^q(\mu) > 0$, $b^q(u,u) \geq 0$ *for all* $\mu \in \mathcal{P}$, $u \in \mathcal{X}$ *and* $1 \leq q \leq Q_b$ *and denote*

$$\Delta_{X_N}(\mu) = \frac{1}{\alpha_{\overline{\mu}}(\mu)} \left\| r_{u_N(\mu)}(\mu) \right\|_{\overline{\mu}}$$

with $\alpha_{\overline{\mu}}(\mu), \gamma_{\overline{\mu}}(\mu) \in \mathbb{R}$,

$$\alpha_{\overline{\mu}}(\mu) := \min_{1 \leq q \leq Q_b} \frac{\Theta_b^q(\mu)}{\Theta_b^q(\overline{\mu})}, \quad \gamma_{\overline{\mu}}(\mu) := \max_{1 \leq q \leq Q_b} \frac{\Theta_b^q(\mu)}{\Theta_b^q(\overline{\mu})}. \tag{2.5.1}$$

Then for parameters $\mu, \overline{\mu} \in \mathcal{P}$, *we have*

$$\frac{1}{\gamma_{\overline{\mu}}(\mu)} \left\| r_{u_N(\mu)}(\mu) \right\|_{\overline{\mu}} \leq \| u_h(\mu) - u_N(\mu) \|_{\overline{\mu}} \leq \Delta_{X_N}(\mu),$$

that is: The energy norm of the representation of the residual for the reduced approximation $u_N(\mu)$ *provides an efficient and rigorous estimate for the discrepancy to the high-dimensional solution* $u_h(\mu)$.

Proof. For a proof see [69]. □

Remark 2.11. *Equation* (2.5.1) *represents the so-called min-*Θ *approach for providing computable lower (upper) bounds to the coercivity (continuity) constant* α *(*γ*) [69]. Another approach is the successive constraint method (SCM) [48].*

Efficient Evaluation of the Error Estimator

As mentioned before, the estimator will not only be used in the basis construction during the offline phase but also for certification purposes during the online phase. We will therefore describe in this section how evaluations of the error estimator can be decomposed into computationally intensive steps that are independent of the parameter but only depend on the degrees of freedom of the discretization space $\mathcal{V}_h$ and steps dependent on the parameter that are computationally inexpensive. The idea here again is to make use of the affine decomposition (Assumption 2.6) of the underlying system.

As a preparation of evaluations of the error estimator in a reduced space X_N we compute the components $r_l^q \in \mathcal{V}_h$, $q \in \{1, \dots, Q_l\}$

$$b(r_l^q, v; \overline{\mu}) = l^q(v) \quad \forall v \in \mathcal{V}_h,$$

and the components $r_b^q \in \mathcal{V}_h$, $q \in \{1, \dots, Q_b \cdot N\}$ with

$$b(r_b^{(j-1)\cdot N+i}, v; \overline{\mu}) = b^j(\varphi_i, v) \quad \forall v \in \mathcal{V}_h,$$

where $\Phi = \{\varphi_i | 1 \leq i \leq N\}$ denotes a basis of X_N, see Section 2.1. For the sake of simplicity of the following presentation, we collect the energy products of all components r_l^q, r_b^q in one matrix $\mathbf{G} \in \mathbb{R}^{Q_r \times Q_r}, Q_r = Q_l + Q_b N$:

$$\mathbf{G} = \begin{pmatrix} G_1 & G_2 \\ G_3 & G_4 \end{pmatrix}, \tag{2.5.2}$$

where

$$(G_1)_{i,j} = b(r_l^i, r_l^j; \overline{\mu}), \quad (G_2)_{i,j} = b(r_l^i, r_b^j; \overline{\mu}),$$
$$(G_3)_{i,j} = b(r_b^i, r_l^j; \overline{\mu}), \quad (G_4)_{i,j} = b(r_b^i, r_b^j; \overline{\mu}).$$

Finally, we define the parameter vector $\mathbf{\Theta}_r(\mu, u_N) \in \mathbb{R}^{Q_r}$ for a given parameter $\mu \in \mathcal{P}$ and a given reduced function $u_N \in X_N$:

$$(\mathbf{\Theta}_r(\mu, u_N))_k = \begin{cases} \Theta_l^k(\mu), & \text{if } k \leq Q_l, \\ -(u_N)_i \Theta_b^j(\mu), & \text{else,} \end{cases}$$

where $i = ((k - Q_l) \mod N)$, $j = \frac{k - Q_l - i}{N} + 1$. The evaluation $\Delta_{X_N}(\mu)$ of the error estimator is then given as

$$\Delta_{X_N}(\mu) = \frac{1}{\alpha_{\overline{\mu}}(\mu)} \sqrt{\mathbf{\Theta}_r(\mu, u_N)^\intercal \cdot \mathbf{G} \cdot \mathbf{\Theta}_r(\mu, u_N)}. \tag{2.5.3}$$

Here we used the notation $v^\intercal$ to denote the transpose of a vector $v \in \mathbb{R}^m$ for $m \in \mathbb{N}$.

By using this scheme, the computational complexity of evaluations of the error estimator is now independent of $\mathcal{N}$ but only polynomial in $Q_r \ll \mathcal{N}$. Furthermore, Q_r does not depend on the size of the underlying mesh but is a constant determined by the modeling process and therefore does not change when the mesh is refined.

2.6 Recent Developments

As mentioned in Section 2.3 the assumption of separable parameter dependence of data functions and physical properties is not crucial as affine parameter dependence of the bilinear and linear forms in Assumption 2.6 can be introduced using different techniques.

In [8] an *empirical* interpolation method was introduced that enables efficient, fully offline–online decoupled computations for problems with non-separable parameter dependence. For a given non-separably parameter-dependent data function $g(\cdot;\mu) \in L^\infty(\Omega) \cap C^0(\Omega)$, the idea of this method is to iteratively build up a so-called *collateral reduced basis* $q_1, \ldots, q_{M_c} : \Omega \to \mathbb{R}$ such that $g_M : \Omega \times \mathcal{P} \to \mathbb{R}$, $g_M(x;\mu) = \sum_{m=1}^{M_c} \varphi_{Mm}(\mu) q_m(x)$ is a good approximation of $g(x;\mu)$. Here, the parameter-independent collateral basis functions q_i are, broadly speaking, residuals normalized at iteratively selected parameter values that are computed during the offline phase. The parameter-dependent coefficients φ_{Mm} are solutions to a lower-triangular system of size M_c that needs to be solved during the online phase of the RB algorithm.

The gain of this approach is an online phase complexity independent of $\mathcal{N}$. The additional cost introduced by solving the equation system for the coefficients for the collateral basis is usually acceptable as $M_c \ll \mathcal{N}$. It was shown that in some cases $M_c \approx N$ is a good choice [33] where N denotes the size of the reduced basis introduced in the last sections.

The empirical interpolation method and the similar discrete empirical interpolation method were applied in different non-linear and non-separable contexts as for example in a two-phase flow setting [25] and especially for the simulation of non-linear miscible viscous fingering in porous media [15].

Equations with high sensitivity with respect to the parameter and large extent of the parameter domain usually yield large reduced bases and hence reduced gain by application of the reduced basis method. Different methods were proposed to overcome this problem. In [30, 31] an extension to the RB method, the so-called "*hp*" reduced basis method was introduced for elliptic and parabolic PDEs. In this approach, the parameter space is bisected during the basis construction (h-refinement) and one reduced space is build per

parameter subdomain (p-refinement). In the online phase, given a parameter, the reduced simulation space is then selected via a binary search. In [36] a similar approach is suggested that allows to alleviate the problem of *over-fitting* — a problem that occurs when the training set $\mathcal{P}_{\text{tr}}$ is does not cover the parameter set well enough and the approximation errors during the online phase hence get unfeasibly large. Towards this end an *early-stopping greedy* algorithm is introduced that uses a second set of parameters (in addition to $\mathcal{P}_{\text{tr}}$) to detect over-fitting. If over-fitting is detected, the training parameter set is refined in order to enhance the approximation quality. Moreover an adaptive parameter domain partitioning, based on the early-stopping greedy algorithm is introduced to overcome the problem of impractically large reduced bases. In [22] the time domain of evolution problems is partitioned to construct several reduced bases for the different intervals.

A further effort of overcoming the obstacle of sensible choice of the training set $\mathcal{P}_{\text{tr}}$ was introduced in [78]. Therein an alternative to the maximization step in the greedy algorithm is proposed that consists in performing nonlinear optimization of the parameter set.

In [42] an alternative sampling strategy for the construction of the training set is proposed: Using the assumption of (nearly) monotone error decrease during the greedy algorithm, parameters fulfilling the error tolerance are discarded and new random ones are added. This way a much better coverage of the parameter set can be reached and the probability of the occurrence of over-fitting is reduced.

Another approach to overcome problems arising from large extent or high dimension of the parameter domain are local greedy approximations as introduced in [62]. Herein, given a parameter, a local approximation space is constructed during the online phase selecting parameters from a training set by proximity in space. Our own approach, the online greedy basis construction [56] (also see Section 4) uses function similarity (measured by a posteriori error estimation) as selection criterion in a greedy-type algorithm run during the online phase. The latter approach was also applied to evolution problems in [23].

A further range of extension to the reduced basis method was introduced to master impractical offline runtimes due to overly expensive snapshot computations: In its basic form introduced above, the

RB method does not allow to control the workload introduced by the computation of high-dimensional snapshots. This can be prohibitive for models leading to very detailed high-dimensional discretizations like two-phase flow applications in porous media, for example. This challenge was answered by different extensions to the RB method in the last years.

The reduced basis element method (RBEM) [58, 61, 16] computes multiple reduced bases during the offline phase, one for each of a set of representative domains. The computational domain Ω is then composed from those representative domains during the online phase (possibly using deformation mappings) and a reduced-dimensional approximation space is constructed from the precomputed spaces using Lagrange multipliers. The reduced basis hybrid method (RBHM) [50] combines this concept with global finite element solutions on a coarse mesh to guarantee continuity of velocities and stresses across interfaces. The reduced basis domain decomposition finite element method (RDFM) [49] introduces a domain decomposition approach coupling reduced basis approximation for the domain spaces and with finite element approximation for the interface. A related approach using RB methods in the context of heterogeneous domain decomposition was introduced in [63]. Finally, our own approach, the localized reduced basis multiscale method (LRBMS) [57, 2, 55] (also see Chapter 5) makes use of local reduced basis spaces, stemming from localizations of global snapshots, glued together via a DG discretization to reduce the number of necessary snapshot computations during the offline phase.

Chapter 3

Two-Phase Flow in Porous Media

In this chapter we introduce the mathematical model for two-phase flow that will be used in the remainder of the work. Also, we describe a discontinuous Galerkin (DG) discretization for the full two-phase flow problem. Parts of this chapter were originally introduced in [55].

3.1 Model Problem

Our model problem is the flow of two phases in a spatio-temporal domain $\Omega \times (0,T) \subset \mathbb{R}^d \times \mathbb{R}^+$ where $d \in \{1,2,3\}$ denotes the space dimension. We use a global pressure, total velocity formulation for two incompressible, immiscible fluids that includes gravity but no capillary effects. For a given source term $q_1 : \Omega \times (0,T) \to \mathbb{R}$ this yields the equation for the unknown pressure p

$$-\nabla \cdot \Big(\lambda(s)\kappa\nabla p - \kappa\big[\lambda_w(s)\varrho_w + \lambda_o(s)\varrho_o\big]G \Big) = q_1 \quad \text{in } \Omega \times (0,T), \tag{3.1.1}$$

where ϱ_w and ϱ_o denote the densities for the wetting and non-wetting phases, respectively, and G is the gravitational force vector

$$G \in \mathbb{R}^d, \quad (G)_d = -g, \quad (G)_j = 0 \text{ for } j \neq d,$$

with g being the gravitational acceleration. Furthermore, κ denotes the total permeability and the functions λ_w, λ_o, $\lambda : [0,1] \to \mathbb{R}^+$ denote the wetting, non-wetting and total mobility, given by

$$\lambda_w(s) = \frac{k_{\mathrm{rw}}(s)}{\eta_w}, \quad \lambda_o(s) = \frac{k_{\mathrm{ro}}(s)}{\eta_o}, \quad \lambda(s) = \lambda_w(s) + \lambda_o(s), \tag{3.1.2}$$

where η_w and η_o is the viscosity of the wetting and non-wetting phase, respectively and the relative permeabilities k_{rw} and k_{ro} are given functions of the saturation s of the wetting phase, which will be specified in the subsequent sections.

Throughout this work we will assume uniform boundedness of $\lambda\kappa$, that is: there exist $k_1, k_2 \in \mathbb{R}$, $k_2 > k_1 > 0$ such that for all $v \in \mathbb{R}^d$ we have

$$k_1 v \cdot v \leq \lambda(x)\kappa(x)v \cdot v \leq k_2 v \cdot v \quad \forall x \in \Omega. \tag{3.1.3}$$

Using Darcy's law, the total velocity u is given by

$$u = -\lambda(s)\kappa\nabla p + \kappa\big[\lambda_w(s)\varrho_w + \lambda_o(s)\varrho_o\big]G \quad \text{in } \Omega \times [0,T], \tag{3.1.4}$$

and enters the transport equation for the saturation s,

$$\phi\,\partial_t s + \nabla \cdot \Big(f_w(s)\left[u + \kappa\lambda_o(s)(\varrho_w - \varrho_o)G\right]\Big) = q_2 \quad \text{in } \Omega \times [0,T]. \tag{3.1.5}$$

Here, $q_2 : \Omega \times [0,T] \to \mathbb{R}$ is a source term, ϕ denotes the porosity of the porous medium and the fractional flow of water f_w is given by

$$f_w(s) = \frac{\lambda_w(s)}{\lambda_w(s) + \lambda_o(s)}. \tag{3.1.6}$$

The above three equations for the pressure (3.1.1), velocity (3.1.4) and saturation (3.1.5) are equipped with the boundary conditions

$$\begin{aligned} s &= s_D \quad \text{in } \partial\Omega_{s,d} \times [0,T], \\ -\lambda(s)\kappa\nabla p \cdot \mathrm{n} &= v_N \quad \text{in } \partial\Omega_{p,n} \times [0,T], \\ p &= p_D \quad \text{in } \partial\Omega_{p,d} \times [0,T], \end{aligned} \tag{3.1.7}$$

on the inflow boundary for the saturation $\partial\Omega_{s,d} \subset \partial\Omega$ and on the Dirichlet and Neumann boundaries for the pressure $\partial\Omega_{p,d}$, $\partial\Omega_{p,n}$ where $\partial\Omega_{p,d} \cap \partial\Omega_{p,n} = \emptyset$, $\partial\Omega_{p,d} \cup \partial\Omega_{p,n} = \partial\Omega$. Further, we impose initial conditions

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

3.2 Discontinuous Galerkin Discretization

In this section we introduce the *high-dimensional* discretization of problem (3.1.1)–(3.1.7). The term high-dimensional is used here to indicate the antonym of the reduced (*low-dimensional*) discretization, cf. Section 2.2.

Different schemes are frequently used to discretize (3.1.1)–(3.1.7). A tendency towards finite-volume-type schemes can be observed in the literature because they are motivated by local conservation properties. As it will prove advantageous in the analysis of our reduced method, we will use a DG discretization with arbitrary local polynomial degree. In particular, we have found the Symmetric Weighted Interior Penalty (SWIP) DG method [32], including the total velocity reconstruction presented therein, to be very robust and it is therefore our method of choice in the following.

Discretization

As a first step towards a discrete version of Equations (3.1.1)–(3.1.7), we introduce a tessellation $\mathcal{T}_h$ of the computational domain Ω. We assume $\mathcal{T}_h$ to be admissible in the sense that

- the interior of all elements $e \in \mathcal{T}_h$ is nonempty ($\text{int}(e) \neq \emptyset$),
- the mesh covers the whole domain Ω ($\bar{\Omega} = \bigcup_{e \in \mathcal{T}_h} e$); this assumption usually restricts the choice of Ω to domains with boundaries that are polygonal chains,
- for any $e_1, e_2 \in \mathcal{T}_h$ with $e_1 \neq e_2$ the intersection $f = e_1 \cap e_2$ is either empty or a joint vertex, edge or (in the case of $d = 3$) face.

An intersection of codimension one of two elements e_1, e_2 or one element e and the boundary $\partial\Omega$ is denoted by f and its width by $h_f = \operatorname{diam}(f)$. Further, we introduce the element width $h_e = \operatorname{diam}(e)$, the grid size $h = \max_{e\in\mathcal{T}_h} h_e$, the d-dimensional Lebesgue measure $|e|$ of an element $e \in \mathcal{T}_h$ and the $d-1$-dimensional Lebesgue measure $|f|$ of an intersection f.

With each intersection f we associate a unique normal vector n_f, for which the subscript f will be dropped whenever no ambiguity arises. We let $\Gamma_{p,d}$ and $\Gamma_{p,n}$ denote all boundary intersections of $\mathcal{T}_h$ where Dirichlet and Neumann conditions for the pressure shall be implied, respectively, and $\Gamma_{s,d}$ denote all boundary intersections on which we impose a fixed saturation. Finally, by Γ_i we denote the inner intersections and by $\Gamma_a = \Gamma_{p,n} \cup \Gamma_{p,d} \cup \Gamma_i$ the set of all intersections. For convenience we denote the set of all intersection where jump and mean values of functions will be defined by $\Gamma_j = \Gamma_i \cup \Gamma_{p,d}$.

In addition to the spatial discretization, we introduce a temporal discretization $t^0, \dots, t^{N_T} \in [0,T]$, $t^i = i \cdot \Delta_t$ and space-time-discrete approximations $s_h^n \approx s(\cdot, t^n)$, $p_h^n \approx p(\cdot, t^n)$ of the saturation and the pressure, both stemming from the so-called DG space.

Definition 3.1 (Discontinuous Galerkin Space)**.** We call the space of piecewise polynomials of maximum local order k

$$\mathcal{V}_h = \left\{ v \in L^2(\Omega) \,\middle|\, v_{|e} \in \mathbb{P}_k(e) \quad \forall e \in \mathcal{T}_h \right\}$$

the *discontinuous Galerkin* space.

In the following we will use $k = 1$. Functions in $\mathcal{V}_h$ are two-valued on intersections $f = e_1 \cap e_2$. Here we assume the unique normal n_f introduced above to point from e_1 to e_2. We define jump $[\![\cdot]\!]_f$ and (weighted) mean $\{\!\{\cdot\}\!\}_f$ values for $w \in \mathcal{V}_h$, i.e.,

$$\begin{aligned} [\![w]\!]_f &= w_{|e_1} - w_{|e_2}, \\ \{\!\{w\}\!\}_f &= \tau_{e_1,f} w_{|e_1} + \tau_{e_2,f} w_{|e_2}, \end{aligned} \tag{3.2.1}$$

where we used

$$\tau_{e_\ell,f} = \frac{a_{\ell,f}}{a_{1,f} + a_{2,f}}, \quad a_{\ell,f} = \|(\kappa)_{|e_\ell}\|_{L^\infty(f)}, \ \ell = 1,2.$$

For the sake of readability, the subscript f in $[\![\cdot]\!]_f$ and $\{\!\{\cdot\}\!\}_f$ will be dropped in the following whenever no ambiguity arises.

Pressure Equation

Following the ideas in [32], we introduce discrete formulations of Equations (3.1.1)–(3.1.7). For a given general mobility function $\gamma : \Omega \to \mathbb{R}^+$ the bilinear form $b_h(\cdot,\cdot;\gamma) : \mathcal{V}_h \times \mathcal{V}_h \to \mathbb{R}$ is defined as

$$\begin{aligned} &b_h(v,w;\gamma) \\ &\quad= \sum_{e\in\mathcal{T}_h} \int_e \gamma\kappa\nabla v\nabla w \,\mathrm{d}x + \sum_{f\in\Gamma_i\cup\Gamma_{p,d}} \frac{\sigma_f}{h_f} \int_f [\![v]\!]\,[\![w]\!] \,\mathrm{d}S \\ &\qquad- \sum_{f\in\Gamma_i\cup\Gamma_{p,d}} \int_f [\{\!\{\gamma\kappa\nabla v\cdot \mathrm{n}_f\}\!\}\,[\![w]\!] + \{\!\{\gamma\kappa\nabla w\cdot \mathrm{n}_f\}\!\}\,[\![v]\!]] \,\mathrm{d}S. \end{aligned} \tag{3.2.2}$$

Here, the penalty parameter σ_f is given as

$$\sigma_f = c_f \frac{2a_{1,f}a_{2,f}}{a_{1,f}+a_{2,f}}, \tag{3.2.3}$$

where $c_f > 0$ is a constant that has to be chosen larger than a minimal threshold depending on the regularity of the mesh.

Remark 3.2. *Note that in [32], $a_{\ell,f} = \|(\gamma\kappa)_{|e_\ell}\|_{L^\infty(f)}$ was used. As the penalty parameter σ_f does not allow an affine decomposition in that case, we refrain from using the mobility in the penalties and instead replace the original constant c_f by $c_f/\max\{\eta_w,\eta_o\}$. Usage of the original weighted averages and penalties would be possible using the empirical interpolation method [8, 24].*

The right-hand side for the discrete formulation of the pressure equation is for $s \in \mathcal{V}_h$, $\gamma_o, \gamma_w : \Omega \to \mathbb{R}^+$, $\gamma = \gamma_o + \gamma_w$ given via the

linear form $l_h(\cdot;\gamma_o,\gamma_w):\mathcal{V}_h\to\mathbb{R}$,

$$\begin{aligned} l_h(w;\gamma_o,\gamma_w) = &\sum_{e\in\mathcal{T}_h}\int_e q_1 w + (\gamma_o\varrho_o+\gamma_w\varrho_w)\kappa\, G\cdot\nabla w\,\mathrm{d}x \\ &- \sum_{f\in\Gamma_i\cup\Gamma_{p,d}}\int_f \{\!\{(\gamma_o\varrho_o+\gamma_w\varrho_w)\kappa\, G\cdot \mathrm{n}_f\}\!\}\,[\![w]\!]\,\mathrm{d}S \\ &+ \sum_{f\in\Gamma_{p,d}}\int_f \left(\frac{\sigma_f}{h_f}w - \gamma\kappa\nabla w\cdot\mathrm{n}_f\right) p_D\,\mathrm{d}S \\ &+ \sum_{f\in\Gamma_{p,n}}\int_f v_N w\,\mathrm{d}S. \end{aligned} \tag{3.2.4}$$

At this point we are ready to introduce the high-dimensional discretization of (3.1.1).

Definition 3.3 (DG Pressure Discretization)**.** For given functions $\gamma_w,\gamma_o:\Omega\to\mathbb{R}^+$ and $\gamma=\gamma_w+\gamma_o$, we call $p_h\in\mathcal{V}_h$ the *high-dimensional pressure solution* if it satisfies

$$b_h(p_h,w;\gamma) = l_h(w;\gamma_o,\gamma_w)\quad\forall w\in\mathcal{V}_h.$$

Remark 3.4. *Problem 3.3 has a unique solution by the Lax-Milgram theorem since b_h is continuous and coercive and l_h is continuous and bounded if the data functions and the local penalty function are chosen accordingly [7].*

Remark 3.5. *Time-dependency enters the DG pressure discretization only via the mobility which usually depends on the saturation. On occasion we will use the discretization 3.3 in a stationary context as well, where the mobility function γ is a given function depending only on the position in space.*

Total Velocity Reconstruction

From a given pressure p, which may be either p_h or the solution of a reduced-dimensional pressure discretization later on, we compute the total velocity u_h using the conservative reconstruction from [32],

which yields continuity of the normal components. For $k = 1$, i.e., piecewise linear pressure, the velocity is computed in the lowest-order Raviart-Thomas space

$$\mathrm{RT} = \left\{ u \in H(\mathrm{div}) \,\middle|\, u\big|_T \in [\mathbb{P}_0(T)]^d + \mathbf{x}\mathbb{P}_0(T)\ \forall T \in \mathcal{T}_h \right\},$$

where $H(\mathrm{div}) = \{ v \in [L^2(\Omega)]^d \,|\, \nabla \cdot v \in L^2(\Omega) \}$ and $\mathbf{x} \in \mathbb{R}^d$.

Additionally, for $v \in \mathcal{V}_h$ (in the following $v = s_h^n$ or $v = p$) with Dirichlet boundary data v_D we introduce

$$[\![v]\!]_f^* = \begin{cases} [\![v]\!]_f, & f \in \Gamma_i, \\ v\big|_\Omega - v_D, & f \in \Gamma_{v,d}, \\ 0, & f \in \Gamma_{p,n}. \end{cases}$$

Definition 3.6 (DG Velocity Discretization)**.** For a given pressure $p \in \mathcal{V}_h$ and saturation $s \in \mathcal{V}_h$, let the total velocity $u_h \in \mathrm{RT}$ be the unique solution to

$$\begin{aligned} &\int_f (u_h \cdot \mathrm{n}_f)\,\mathrm{d}S \\ &\quad = \int_f -\mathrm{n}_f \cdot \{\!\{ \lambda(s)\kappa\nabla p - \kappa\big[\lambda_w(s)\varrho_w + \lambda_o(s)\varrho_o\big]G \}\!\} + \frac{\sigma_f}{h_f}\,[\![p]\!]_f^*\,\mathrm{d}S \end{aligned} \tag{3.2.5}$$

for all $f \in \Gamma_a$.

Saturation Equation

The saturation equation is discretized by a DG scheme in space using an upwind flux in the space of piecewise polynomial functions $\mathcal{V}_h$. For the temporal discretization, we use an explicit Euler scheme.

Definition 3.7 (DG Saturation Discretization)**.** For a given saturation $s_h^n \in \mathcal{V}_h$ and velocity $u \in [\mathcal{V}_h]^d$, the saturation $s_h^{n+1} \in \mathcal{V}_h$ is

given as the solution to

$$\begin{aligned}
&\sum_{e\in\mathcal{T}_h}\int_e \frac{\phi}{\Delta t}s_h^{n+1}v_h\,\mathrm{d}x \\
&\quad=\sum_{e\in\mathcal{T}_h}\int_e \frac{\phi}{\Delta t}s_h^{n}v_h\,\mathrm{d}x+\sum_{e\in\mathcal{T}_h}q_2v_h\,\mathrm{d}x \\
&\qquad+\sum_{e\in\mathcal{T}_h}\int_e \big[u+\lambda_o(s_h^n)(\varrho_w-\varrho_o)\kappa\cdot G\big]f_w(s_h^n)\cdot\nabla v_h\,\mathrm{d}x \\
&\qquad-\sum_{f\in\Gamma_a}\int_f \mathrm{n}_f\cdot\{\!\{u+\lambda_o(s_h^n)(\varrho_w-\varrho_o)\kappa\cdot G\}\!\}\ f_w(\chi(s_h^n))\,[\![v_h]\!]\,\mathrm{d}S \\
&\qquad-\sum_{f\in\Gamma_i\cup\Gamma_{s,d}}\frac{\sigma_f}{h_f}\int_f [\![s_h^n]\!]_f^*\,[\![v_h]\!]_f\,\mathrm{d}S,
\end{aligned}\tag{3.2.6}$$

for all $v_h\in\mathcal{V}_h$. Here the upwind function χ is given as

$$\chi(s_h^n)=\begin{cases}s_h^{n\uparrow}, & f\in\Gamma_i,\\ s_D, & f\in\Gamma_{s,d},\end{cases}\tag{3.2.7}$$

where $s_h^{n\uparrow}$ denotes the upwind value of s_h^n: If $f\in\Gamma_i$, there exist $e_1,e_2\in\mathcal{T}_h$ such that $f=\partial e_1\cap\partial e_2$. Then $s_h^{n\uparrow}=s_h^n\big|_{e_1}$ if $u_h\cdot\mathrm{n}_f\geq 0$ and $s_h^{n\uparrow}=s_h^n\big|_{e_2}$ otherwise. Remember that we assume n_f to point from e_1 to e_2. The penalty σ_f in Equation (3.2.6) is the same as for the pressure equation, see (3.2.3).

Slope Limiter

Discontinuous Galerkin methods require some kind of stabilization to avoid over- and undershoots if they are to be used in a two-phase flow context; see [73, 20], for example. Both artificial diffusion and slope limiters have been used as a means to this end. We will employ a slope limiter introduced by Dedner et al. [20] that is applicable to different element types (triangular, tetrahedral, hexahedral, Cartesian). We will shortly outline the most important features of the mass-conservative limiter that we use in our experiments for the

case of a piecewise linear saturation, where we use the "$\mathcal{DG}$ scheme" (in contrast to "$\mathcal{DG}+\mathcal{R}$", see [20, Section 6.1]). For more details on the method and a discussion of the general case of arbitrary polynomial order on s_h^n we refer to [20].

The first step is to compute a so-called *shock-detector* $\mathcal{S}: \mathcal{T}_h \to \mathbb{R}^+$, which for a given saturation $s_h \in \mathcal{V}_h$ and velocity $u_h \in [\mathcal{V}_h]^d$, reads:

$$\mathcal{S}(e) = \sum_{f \in \mathcal{I}_e^{\uparrow}} \left(0.08 \cdot d \cdot \sqrt{h_e}|f|\right)^{-1} \int_f [\![s_h]\!] \, \mathrm{d}S,$$

where $\mathcal{I}_e^{\uparrow} = \{ f \in \Gamma_a \,|\, f \subset \partial e, u_h \cdot \mathrm{n}_f < 0 \}$ denotes the upstream interfaces of e. Using $\mathcal{S}$ makes it possible to apply the limiter (i.e., reduce gradients of the saturation) only on cells in which (strong) discontinuities are present and therefore unwanted numerical oscillations may occur. In other regions the saturation is left unlimited. In this sense all cells $e \in \mathcal{T}_h$ with $\mathcal{S}(e) > 1$ or $s_h(x) \notin [0,1]$ for some $x \in e$ will be marked. Let $e^* \in \mathcal{T}_h$ be such a cell and $\{ e_i \,|\, i = 1, \dots, N_n \}$ its direct neighbors. We then compute

$$g_i = \nabla s_h \cdot (b_{e_i} - b_{e^*}) \quad \text{and} \quad d_i = \frac{1}{|e_i|} \int_{e_i} s_h \, \mathrm{d}x - \frac{1}{|e^*|} \int_{e^*} s_h \, \mathrm{d}x,$$

where b_e denotes the barycenter of the cell e. From g_i and d_i we compute the gradient scales m_i by

$$m_i = \begin{cases} 0, & \text{if } g_i d_i < 0, |g_i| > 10^{-8} \text{ and } |d_i| > 10^{-8}, \\ d_i/g_i, & \text{if } g_i d_i > 0, |g_i| > |d_i|, |g_i| > 10^{-8} \text{ and } |d_i| > 10^{-8}, \\ 1, & \text{otherwise.} \end{cases}$$

Finally, the stabilized saturation $\tilde{s}_h$ is computed as

$$\int_e \tilde{s}_h \varphi \, \mathrm{d}x = \int_e s_h \varphi \, \mathrm{d}x + \int_e \min_i(m_i) \nabla s_h \cdot (x - b_e) \varphi \, \mathrm{d}x \quad \forall \varphi \in \mathbb{P}_1(e) \tag{3.2.8}$$

in all marked cells $e \in \mathcal{T}_h$. In all other cells, we set $\tilde{s}_h = s_h$.

Algorithm 3.8 (High-Dimensional Two-Phase Flow Scheme)**.** *The overall high-dimensional simulation scheme for two-phase flow is as follows:*

1. *Project the initial data s_0 for the saturation onto the discrete space $\mathcal{V}_h$: Let $s_h^0 \in \mathcal{V}_h$ be given by*

$$\int_\Omega s_0 \psi \,\mathrm{d}x = \int_\Omega s_h^0 \psi \,\mathrm{d}x \quad \forall \psi \in \mathcal{V}_h.$$

2. *For all $n \in \{0, \dots, N_T - 1\}$,*
   a) *use the pressure scheme (Definition 3.3) for $\gamma_o = \lambda_o(s_h^n)$ and $\gamma_w = \lambda_w(s_h^n)$ to compute p_h^{n+1};*
   b) *use the velocity scheme (Definition 3.6) for $p = p_h^{n+1}$ to compute the total velocity u_h^{n+1};*
   c) *use the saturation scheme (Definition 3.7) for $u = u_h^{n+1}$ to compute the saturation $\overline{s}_h^{n+1}$;*
   d) *use the limiter (Equation (3.2.8)) for $s_h = \overline{s}_h^{n+1}$, $u_h = u_h^{n+1}$ to compute a stabilized version $\tilde{s}_h^{n+1}$ of the saturation and set $s_h^{n+1} = \tilde{s}_h^{n+1}$.*

Time-of-Flight Equation

The so-called time-of-flight $\tau(x)$ is defined as the time it takes for a passive particle to reach a given point $x \in \Omega$, starting from the closest point on the inflow boundary or a source. Here, $\tau(x)$ can be defined by integrating $\int |u|^{-1} \phi \,\mathrm{d}x$ along streamlines, or by solving $u \cdot \tau = \phi$. The time-of-flight inherits important information about the flow pattern and will be used in Section 5.3 to approximate the spatio-temporal behavior of the saturation without computing a whole temporal evolution the transport equation.

Consistent with our discretization of the saturation and pressure equation we use the DG discretization introduced in [66]: Find $\tau_h^s \in \mathcal{V}_h$ such that

$$-\int_e (\tau_h^s u_h) \cdot \nabla \psi_h \,\mathrm{d}x + \int_{\partial e} (\tau_h^{s\uparrow} u_h \cdot \mathrm{n}) \psi_h \,\mathrm{d}S = \int_e \phi \psi_h \,\mathrm{d}x$$

for all $\psi_h \in \mathcal{V}_h$ and all $e \in \mathcal{T}_h$. Here, $\tau_h^{s\uparrow}$ again denotes the upwind value of the two-valued function τ_h^s, see (3.2.7), and ϕ denotes the porosity of the porous medium.

In the absence of gravitational forces, the discretized time-of-flight equation can be permuted to a lower block-triangular form—if the computational mesh is reordered according to the direction of the flow—and hence solved very efficiently in a per-element fashion by a simple backsubstitution method; see [66] for details.

3.3 Parameterization

As described in Chapter 2 the RB method provides model order reduction for parameterized partial differential equations. It remains to clarify how the parameterization enters the model problem introduced in the last section. In this paragraph we will outline some possible parameterizations of the two-phase flow scheme as introduced in Algorithm 3.8.

For the pressure equation (3.1.1) possible parameterizations can be introduced via the boundary data: Assume a parameter set $\mathcal{P} \subset \mathbb{R}^\sigma$, a set of functions $v_N^i : \partial\Omega_{p,n} \to \mathbb{R}$ and $\theta_i^{\text{neu}} : \mathcal{P} \to \mathbb{R}$, $i \in \{1, \dots, Q_{\text{neu}}\}$ for the Neumann data and a set of functions $p_D^i : \partial\Omega_{p,d} \to \mathbb{R}$ and $\theta_i^{\text{dir}} : \mathcal{P} \to \mathbb{R}$, $i \in \{1, \dots, Q_{\text{dir}}\}$ for the Dirichlet data to be given. An affine decomposition of the linear form $l_h(\cdot;\cdot,\cdot)$ as in Assumption 2.6 could then be given by

$$l_h(w, \gamma_o, \gamma_w, \mu) = \sum_{i=1}^{Q_l} \theta_l^i(\mu)\, l_h^i(w, \gamma_o, \gamma_w),$$

where $Q_l = Q_{\text{neu}} + Q_{\text{dir}} + 1$ and the functions θ_l^i and l_h^i are given by

$$\theta_l^i(\mu) = \begin{cases} 1, & \text{if } i = 1, \\ \theta_i^{\text{dir}}, & \text{if } 1 < i \le Q_{\text{dir}} + 1, \\ \theta_i^{\text{neu}}, & \text{if } Q_{\text{dir}} + 1 < i \le Q_{\text{neu}} + Q_{\text{dir}} + 1, \end{cases}$$

and

$$\begin{aligned} l_h^1(w, \gamma_o, \gamma_w) = &\sum_{e \in \mathcal{T}_h} \int_e q_1 w + (\gamma_o \varrho_o + \gamma_w \varrho_w) \kappa\, G \cdot \nabla w \,\mathrm{d}x \\ &- \sum_{f \in \Gamma_i \cup \Gamma_{p,d}} \int_f \{\!\{ (\gamma_o \varrho_o + \gamma_w \varrho_w) \kappa\, G \cdot \mathrm{n}_f \}\!\} \, [\![w]\!] \,\mathrm{d}S, \end{aligned}$$

and for $1 < i \leq Q_{\mathrm{dir}} + 1$:

$$l_h^i(w, \gamma_o, \gamma_w) = \sum_{f \in \Gamma_{p,d}} \int_f \left(\frac{\sigma_f}{h_f} w - \gamma \kappa \nabla w \cdot \mathrm{n}_f \right) p_D^i \, \mathrm{d}S,$$

and finally for $Q_{\mathrm{dir}} + 1 < i \leq Q_{\mathrm{neu}} + Q_{\mathrm{dir}} + 1$:

$$l_h^i(w, \gamma_o, \gamma_w) = \sum_{f \in \Gamma_{p,n}} \int_f v_N^i w \, \mathrm{d}S.$$

The assembly of the right hand side vector for a low-dimensional (reduced) system can then be decomposed into offline- and online-parts as as suggested in Equations (2.3.1) and (2.3.2).

Another possible parameterization of the pressure equation that will be frequently used throughout the rest of this work concerns the wetting and non-wetting mobilities and the total mobility. More precisely: The functions γ_w, γ_o and γ in Definition 3.3 will be replaced by different types of parameterized functions in the following chapters. As a common feature, all parameterizations will take the affine form

$$\begin{aligned} \gamma_w(\delta, x) = \sum_{i=1}^{M} \theta_i(\delta) \Lambda_w^i(x), \quad \gamma_o(\delta, x) = \sum_{i=1}^{M} \theta_i(\delta) \Lambda_o^i(x), \\ \gamma(\delta, x) = \sum_{i=1}^{M} \theta_i(\delta) \Lambda^i(x), \end{aligned} \tag{3.3.1}$$

where $x \in \Omega$ and $\Lambda^i(x) = \Lambda_w^i(x) + \Lambda_o^i(x)$. The quantity δ will be used to model the parameter which can be either an element of $\mathbb{R}^m$ for some $m \in \mathbb{N}$ (Chapter 4 and Section 5.2) or a function (Section 5.3).

The saturation equation (3.1.5) can for example be parameterized via the boundary data, the right hand side or its initial values. In order to keep the notation comprehensible we will refrain from parameterizing the saturation equation.

Chapter 4

Online Greedy Basis Construction

Reduced basis methods as introduced in Chapter 2 usually gain huge speedups over their underlying high-dimensional discretization. Problems may, however, arise when the problem at hand shows high sensitivity with respect to the parameter or with growing dimension of the parameter. In both cases, the greedy basis construction algorithm from Section 2.1 may yield large bases. Considering the fact that the reduced systems are usually dense, growing reduced basis size can easily become a problem in terms of computational complexity and hence simulation time. Such a situation is shown in Figure 4.1 where the error convergence during the standard greedy algorithm is depicted for a problem with parameter dimension eight. Here already we see that the basis size is quite large for the desired tolerance of $1 \cdot 10^{-5}$.

In this chapter we study problems with high parameter dimension that yield unfeasibly large bases. Based on the notation introduced in Chapter 2, we introduce a new method for model order reduction that uses a large *dictionary* of basis vector candidates to build a small, parameter-adapted basis during the online phase. Our method holds some similarity with the *locally adaptive greedy method*

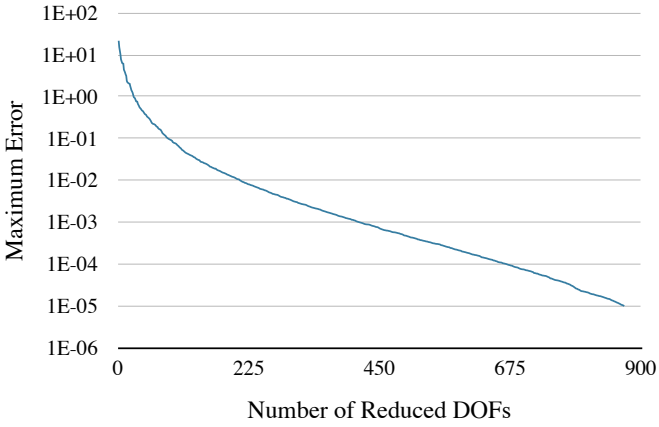


Figure 4.1 – Error convergence during greedy algorithm from Section 2.1 for a problem with $\dim(\mathcal{P}) = 8$.

introduced in [62]. As a main difference, our method does not use proximity in parameter space as an indicator for well-suited basis candidates in the basis construction but directly measures function similarity via error estimation. By using this approach we can always guarantee minimal basis sizes for a desired error tolerance. Further ideas about greedy methods and dictionary approaches can be found in [77].

In Section 4.1 we present our dictionary construction algorithm. Section 4.2 and Section 4.3 are dedicated to the online basis construction procedure. Finally we present numerical results in Section 4.4. The contents of this chapter were originally presented in [56].

4.1 Offline Data Computation

During the offline phase we construct a "dictionary" $\mathcal{D}$.

Definition 4.1 (Dictionary)**.** A *dictionary* $\mathcal{D}$ is a set

$$\mathcal{D} = \{\varphi_i \,|1 \leq i \leq D\}$$

of basis vector candidates.

A straightforward algorithm to construct the dictionary could be to choose a training set $\mathcal{P}_{\text{tr}} \subset \mathcal{P}$ (cf. Chapter 2) and compute

$$\mathcal{D} = \{u_h(\mu) \in \mathcal{V}_h \,|\mu \in \mathcal{P}_{\text{tr}}\}\,.$$

This idea restricts the size of the training set to a certain extent as its size is directly linked to the size of the dictionary. More elaborate dictionary construction algorithms are possible, of course. Since our focus in this chapter is on the online basis generation *from* the dictionary and not on the construction of the dictionary itself we will nevertheless use this simple algorithm for the experiments in Section 4.4. For possible approaches for improved dictionary construction see [11].

Together with the dictionary we compute the Gramian matrices $\mathbf{A}_D^q \in \mathbb{R}^{D\times D}$, $1 \leq q \leq Q_b$ and the vectors $\mathbf{b}_D^q \in \mathbb{R}^D$, $1 \leq q \leq Q_l$,

$$\begin{aligned}(\mathbf{A}_D^q)_{i.j} &= b^q(\varphi_j, \varphi_i), \quad \varphi_i, \varphi_j \in \mathcal{D},\\ (\mathbf{b}_D^q)_i &= l^q(\varphi_i), \quad \varphi_i \in \mathcal{D}\end{aligned}$$

that will be needed for reduced simulations during the online phase. For the Definitions of b, l, Q_b, Q_l, etc. see Chapter 2. Furthermore, we compute the matrix $\mathbf{G}$ from Section 2.5 for $\Phi = \mathcal{D}$.

4.2 Online Basis Construction

In this section we describe the algorithm that is used to construct a space $X_N(\mu)$ from the dictionary $\mathcal{D}$ for a given parameter $\mu \in \mathcal{P}$. As a means to this end we define a so-called *indicator function*:

Definition 4.2 (Indicator Function)**.** Given the high-dimensional space $\mathcal{V}_h$, a reduced-dimensional space X_N, a parameter set $\mathcal{P} \subset \mathbb{R}^d$ and an error estimator $\Delta : X_{\text{red}} \to \{\omega \,|\, \omega : \mathcal{P} \to [0,\infty)\}$ for a given linear subspace $X_{\text{red}} \subset \mathcal{V}_h$ of the discrete function space $\mathcal{V}_h$, we define the indicator function $\eta_\Delta : \mathcal{V}_h \times \mathcal{P} \to [0,\infty)$ as

$$\eta_\Delta(\varphi, \mu^*; X_N) = \Delta(X_N)(\mu^*) - \Delta(X_N \oplus \operatorname{span}(\{\varphi\}))(\mu^*).$$

The function η_Δ indicates the reduction of the estimated error (for example using the estimator $\Delta(X_{\text{red}})(\mu) = \Delta_{X_{\text{red}}}(\mu)$ from Theorem

2.10) for a given parameter $\mu^* \in \mathcal{P}$ in a given space X_N if X_N is enlarged with $\varphi \in \mathcal{V}_h$. By using the estimated error decrease in η_Δ we ensure that our indicator will always point out the ideal basis enrichment candidate $\varphi \in \mathcal{V}_h$ for a reduced space X_N and parameter μ^* in terms of (estimated) error decrease for the reduced solution $u_N(\mu^*) \in X_N$. In the following, we introduce an iterative greedy-type algorithm using this indicator that will therefore yield ideal (in the sense of minimal) basis sizes $|X_N|$ for a given tolerance for the error between the reduced- and high-dimensional solution.

Remark 4.3. *For the rest of this section, we will suppress the parameter μ in solutions $u(\mu)$ whenever no ambiguity arises. Further, without loss of generality, we henceforth assume the estimator $\Delta(X_N) : \mathcal{P} \to [0, \infty)$ to be given by the residual a posteriori estimator from Section 2.5.*

Algorithm 4.4. *Given a parameter $\mu^* \in \mathcal{P}$, an error tolerance $\varepsilon > 0$, a desired basis size $N \in \mathbb{N}$, $n = 0$ and $X_0 = \{0\}$ we repeat the following steps to construct a parameter-fit reduced basis space $X_N(\mu^*)$ from a precomputed dictionary $\mathcal{D}$:*

1. *Compute the reduced solution $u_n(\mu^*)$ in the space X_n, see Chapter 2, and evaluate the error estimator $\Delta_{X_n}(\mu^*)$. In case $\Delta_{X_n}(\mu^*) < \varepsilon$ or $n \geq N$ set $X_N(\mu^*) = X_n(\mu^*)$ and finish, else go on with Step 2.*

2. *Evaluate the indicator $\eta_\Delta(\varphi, \mu^*; X_n)$ for all dictionary elements $\varphi \in \mathcal{D}$.*

3. *Find the dictionary element that maximizes the indicator function:*

$$\varphi_{\max} = \arg\max_{\psi \in \mathcal{D}} \eta_\Delta(\psi, \mu^*; X_n).$$

4. *Set $n = n+1$ and enrich the reduced space with the linear span of the current dictionary element: $X_n = X_{n-1} \oplus \operatorname{span}(\{\varphi_{\max}\})$.*

For an illustration of Algorithm 4.4 see Figure 4.2.

Clearly, in a naive implementation, Step 2, which includes reduced simulation in the space $X_n \oplus \operatorname{span}(\{\varphi\})$ and evaluation of the error

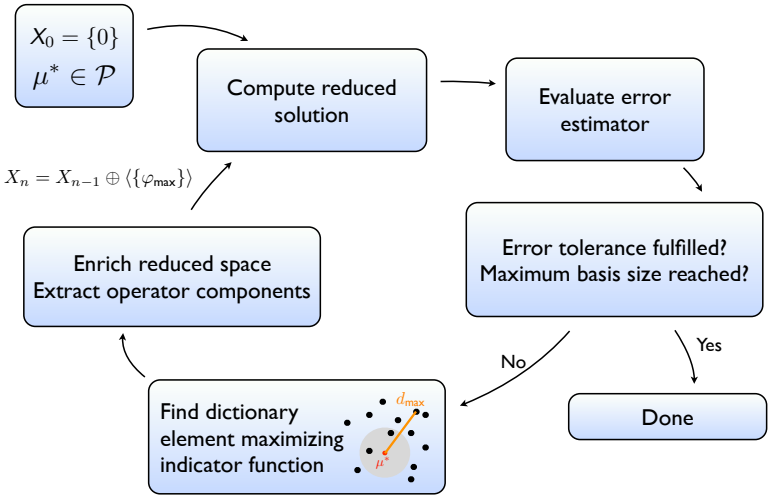


Figure 4.2 – Flowchart for the basis generation algorithm (Algorithm 4.4) used for online construction of parameter-fit reduced bases.

estimator for *all* dictionary elements $\varphi \in \mathcal{D}$, will be too costly to be applicable, especially for large dictionaries $\mathcal{D}$ as it has a complexity of $\mathcal{O}(DN^4)$ with $D = |\mathcal{D}|$. We will therefore now point out how Algorithm 4.4 can be performed with a complexity of $\mathcal{O}(DN^3)$. This will be favorable over the standard greedy RB approach, where the complexity is also cubic in the basis size but bases are usually a lot larger than with our method as they are precomputed during the offline phase to provide a best-fit for *all* parameters in the training set.

4.3 Online-Computation Vectorization

As a first step in the evaluation of the indicator η_Δ (see Definition 4.2) we need to compute all reduced solutions $u_{n,\varphi} \in X_n \oplus \operatorname{span}(\{\varphi\})$ for all dictionary elements $\varphi \in \mathcal{D}$ and a given space X_n. The following proposition allows us to compute these solutions with minimal effort simultaneously for the whole dictionary $\mathcal{D}$:

Proposition 4.5. *For a given parameter $\mu \in \mathcal{P}$ and $\varphi \in \mathcal{D}$, the solution $\mathbf{u}_{n,\varphi}(\mu)$ of Equation (2.2.2) in the space $X_n \oplus \operatorname{span}(\{\varphi\})$ is given by*

$$\mathbf{u}_{n,\varphi}(\mu) = \begin{pmatrix} \mathbf{u}_n \\ 0 \end{pmatrix} + \frac{\sigma(\varphi,\mu) - \boldsymbol{\beta}(\varphi,\mu)\mathbf{u}_n}{\gamma(\varphi,\mu) - \boldsymbol{\beta}(\varphi,\mu)\mathbf{A}_n^{-1}\boldsymbol{\alpha}(\varphi,\mu)} \cdot \begin{pmatrix} -\mathbf{A}_n^{-1}\boldsymbol{\alpha}(\varphi,\mu) \\ 1 \end{pmatrix}$$

with suitably chosen $\mathbf{A}_n \in \mathbb{R}^{n\times n}$, $\mathbf{u}_n \in \mathbb{R}^n$ and functions $\boldsymbol{\alpha} : \mathcal{D} \times \mathcal{P} \to \mathbb{R}^{n\times 1}$, $\boldsymbol{\beta} : \mathcal{D} \times \mathcal{P} \to \mathbb{R}^{1\times n}$, $\sigma, \gamma : \mathcal{D} \times \mathcal{P} \to \mathbb{R}$.

Proof. We define the matrix $\mathbf{A}_n \in \mathbb{R}^{n\times n}$ and the vectors $\mathbf{u}_n, \mathbf{b}_n \in \mathbb{R}^n$ for the space X_n as in (2.2.2). Let

$$\begin{aligned}
(\boldsymbol{\alpha}(\varphi))_i &= (\boldsymbol{\alpha}(\varphi,\mu))_i = b(\varphi_i, \varphi; \mu), \quad 1 \le i \le n, \\
(\boldsymbol{\beta}(\varphi))_i &= (\boldsymbol{\beta}(\varphi,\mu))_i = b(\varphi, \varphi_i; \mu), \quad 1 \le i \le n, \\
\gamma(\varphi) &= \gamma(\varphi,\mu) = b(\varphi,\varphi;\mu), \\
\sigma(\varphi) &= \sigma(\varphi,\mu) = l(\varphi;\mu).
\end{aligned}$$

for a given basis $\{\varphi_i | 1 \le i \le n\} \subset X_n$ of X_n. The projection of Equation (2.1.1) onto the space $X_n \oplus \operatorname{span}(\{\varphi\})$ for a given function $\varphi \in \mathcal{D}$ is then given by

$$\mathbf{A}_{n,\varphi} \cdot \mathbf{u}_{n,\varphi} = \mathbf{b}_{n,\varphi}, \tag{4.3.1}$$

where

$$\begin{aligned}
\mathbf{A}_{n,\varphi} &= \begin{pmatrix} \mathbf{A}_n & \boldsymbol{\alpha}(\varphi) \\ \boldsymbol{\beta}(\varphi) & \gamma(\varphi) \end{pmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}, \\
\mathbf{b}_{n,\varphi} &= \begin{pmatrix} \mathbf{b_n} \\ \sigma(\varphi) \end{pmatrix} \in \mathbb{R}^{n+1}.
\end{aligned}$$

Multiplication of Equation (4.3.1) with the invertible block diagonal matrix $\mathrm{diag}(\mathbf{A}_n^{-1}, 1)$ then yields:

$$\begin{aligned}
&\mathbf{A}_{n,\varphi} \cdot \mathbf{u}_{n,\varphi} && = \mathbf{b}_{n,\varphi},\\
\Leftrightarrow &\begin{pmatrix} \mathbf{A}_n^{-1} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{A}_n & \boldsymbol{\alpha}(\varphi) \\ \boldsymbol{\beta}(\varphi) & \gamma(\varphi) \end{pmatrix} \cdot \mathbf{u}_{n,\varphi} && = \begin{pmatrix} \mathbf{A}_n^{-1} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{b_n} \\ \sigma(\varphi) \end{pmatrix}\\
\Leftrightarrow &\begin{pmatrix} \mathbf{Id}_n & \mathbf{A}_n^{-1}\boldsymbol{\alpha}(\varphi) \\ \boldsymbol{\beta}(\varphi) & \gamma(\varphi) \end{pmatrix} \cdot \mathbf{u}_{n,\varphi} && = \begin{pmatrix} \mathbf{u}_n \\ \sigma(\varphi) \end{pmatrix}\\
\Leftrightarrow &\begin{pmatrix} \mathbf{Id}_n & \mathbf{A}_n^{-1}\boldsymbol{\alpha}(\varphi) \\ 0 & \gamma(\varphi) - \boldsymbol{\beta}(\varphi)\mathbf{A}_n^{-1}\boldsymbol{\alpha}(\varphi) \end{pmatrix} \cdot \mathbf{u}_{n,\varphi} && = \begin{pmatrix} \mathbf{u}_n \\ \sigma(\varphi) - \boldsymbol{\beta}(\varphi)\mathbf{u}_n \end{pmatrix},
\end{aligned}$$

where $\mathbf{Id}_n \in \mathbb{R}^{n\times n}$ denotes the n by n identity matrix.

Using back substitution we find the solution $\mathbf{u}_{n,\varphi}$:

$$\begin{aligned}
(\mathbf{u}_{n,\varphi})_{n+1} &= \frac{\sigma(\varphi) - \boldsymbol{\beta}(\varphi)\mathbf{u}_n}{\gamma(\varphi) - \boldsymbol{\beta}(\varphi)\mathbf{A}_n^{-1}\boldsymbol{\alpha}(\varphi)},\\
(\mathbf{u}_{n,\varphi})_k &= (\mathbf{u}_n)_k - (\mathbf{A}_n^{-1}\boldsymbol{\alpha}(\varphi))_k(\mathbf{u}_{n,\varphi})_{n+1}, \quad k \in \{1,\dots,n\}.
\end{aligned}$$

Which can be rewritten in the form

$$\mathbf{u}_{n,\varphi} = \begin{pmatrix} \mathbf{u}_n \\ 0 \end{pmatrix} + \frac{\sigma(\varphi) - \boldsymbol{\beta}(\varphi)\mathbf{u}_n}{\gamma(\varphi) - \boldsymbol{\beta}(\varphi)\mathbf{A}_n^{-1}\boldsymbol{\alpha}(\varphi)} \cdot \begin{pmatrix} -\mathbf{A}_n^{-1}\boldsymbol{\alpha}(\varphi) \\ 1 \end{pmatrix}.$$

□

Using Proposition 4.5, only one matrix-vector multiplication and two vector-vector multiplications are needed for the computation of one reduced solution in Step (2) in Algorithm 4.4. Only once per loop iteration in Algorithm 4.4 the matrix $\mathbf{A}_n$ needs to be inverted. The quantities $\boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma, \sigma$ and $\mathbf{A}_n$ can be extracted from $\mathbf{A}_D(\mu)$,$\mathbf{b}_D(\mu)$ where

$$\mathbf{A}_D(\mu) = \sum_{q=1}^{Q_b} \Theta_b^q(\mu)\mathbf{A}_D^q \in \mathbb{R}^{D\times D}, \quad \mathbf{b}_D(\mu) = \sum_{q=1}^{Q_l} \Theta_l^q(\mu)\mathbf{b}_D^q \in \mathbb{R}^D.$$

Here $\mathbf{A}_D^q$, $1 \le q \le Q_b$ and $\mathbf{b}_D^q$, $1 \le q \le Q_l$ are the precomputed quantities from Section 4.1.

In the sequel, we will outline how to efficiently evaluate the indicator function η_Δ for all possible extensions in Step (2) of Algorithm 4.4. When evaluating the indicator $\eta_\Delta(\varphi, \mu^*; X_n)$ for all dictionary elements φ, we need to evaluate $\Delta_{X_n \oplus \operatorname{span}(\{\varphi\})}(\mu^*)$ for all $\varphi \in \mathcal{D}$. The next proposition proves that these values can be computed simultaneously for the whole dictionary.

Proposition 4.6. *For suitable choice of matrices* $\mathbf{g}_1 \in \mathbb{R}^{(n+1)\times D}$, $\mathbf{g}_2 \in \mathbb{R}^{1\times D}$ *the vector* $\Delta \in \mathbb{R}^D$ *with*

$$\Delta = \begin{pmatrix} 1 & \cdots & 1 \end{pmatrix} \cdot \left(\begin{pmatrix} 1 & \cdots & 1 \\ -(\mathbf{u}_{n,\varphi_1})_1 & \cdots & -(\mathbf{u}_{n,\varphi_D})_1 \\ \vdots & \ddots & \vdots \\ -(\mathbf{u}_{n,\varphi_1})_{n+1} & \cdots & -(\mathbf{u}_{n,\varphi_D})_{n+1} \end{pmatrix} \circ \begin{pmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \end{pmatrix} \right)$$

contains the squared error estimators for all possible basis extensions:

$$\Delta = \left(\Delta_{X_n \oplus \operatorname{span}(\{\varphi_1\})}(\mu^*), \quad \cdots, \quad \Delta_{X_n \oplus \operatorname{span}(\{\varphi_D\})}(\mu^*) \right).$$

Here we used the Hadamard *product* $\mathbf{M} \circ \mathbf{N} \in \mathbb{R}^{m\times n}$ *of two matrices* $\mathbf{M}, \mathbf{N} \in \mathbb{R}^{m\times n}$, $(\mathbf{M} \circ \mathbf{N})_{i,j} = \mathbf{M}_{i,j} \cdot \mathbf{N}_{i,j}$.

Proof. Given a parameter $\mu^* \in \mathcal{P}$ we define

$$\mathbf{S}(\mu^*) = \begin{pmatrix} \Theta_l^1(\mu^*) & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \Theta_l^{Q_l}(\mu^*) & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & \mathbf{C}(\mu^*) & \\ 0 & & & \end{pmatrix} \in \mathbb{R}^{(Q_l + Q_b \cdot D)\times(D+1)},$$

where the coefficient matrix $\mathbf{C}(\mu^*) \in \mathbb{R}^{(Q_b \cdot D)\times D}$ is given as

$$\mathbf{C}(\mu^*) = \begin{pmatrix} \boldsymbol{\Theta}_B(\mu^*) & & 0 \\ & \ddots & \\ 0 & & \boldsymbol{\Theta}_B(\mu^*) \end{pmatrix},$$

with $\mathbf{\Theta}_B(\mu^*) \in \mathbb{R}^{Q_b}$, $(\mathbf{\Theta}_B(\mu^*))_k = \Theta_b^k(\mu^*)$. Using $\mathbf{S}(\mu^*)$ we define the matrix

$$\overline{\mathbf{G}} = \overline{\mathbf{G}}(\mu^*) = \mathbf{S}(\mu^*)^\intercal \cdot \mathbf{G} \cdot \mathbf{S}(\mu^*) \in \mathbb{R}^{(D+1)\times(D+1)},$$

with $\mathbf{G}$ as computed in Section 4.1. For the exposition of the rest of the simultaneous indicator evaluation we need some additional notation:

- Given a set of indices $I = [i_1, \ldots, i_m] \subset \mathbb{N}$ we define $I \dotplus l := [i_1 + l, \ldots, i_m + l]$ for $l \in \mathbb{N}$.
- Given a set of indices $I = [i_1, \ldots, i_m] \subset \mathbb{N}$ we define the set of indices $[I, l] \subset \mathbb{N}$: $[I, l] := [i_1, \ldots, i_m, l]$ for $l \in \mathbb{N}$.
- Given a matrix $\mathbf{M}$ and two sets of indices $I = [i_1, \ldots, i_{|I|}] \subset \mathbb{N}$, $J = [j_1, \ldots, j_{|J|}] \subset \mathbb{N}$ we define the matrix $\mathbf{M}_{I,J} \in \mathbb{R}^{|I|\times|J|}$

$$(\mathbf{M}_{I,J})_{k,l} := \mathbf{M}_{i_k, j_l}.$$

Assume a basis $\Phi \subset \mathcal{D}$ of the space X_n to be given. Let $I_\Phi \subset \mathbb{N}$ be an index set for Φ and $I_\mathcal{D} \subset \mathbb{N}$ be an index set for $\mathcal{D}$. Additionally we use the vectors $\mathbf{u}_{n,\varphi} = \mathbf{u}_{n,\varphi}(\mu^*)$ from Section 4.3.

Using the above notation we can define

$$\mathbf{g}_1 = \overline{\mathbf{G}}_{[1,I_\Phi \dotplus 1],[1,I_\Phi \dotplus 1]} \cdot \begin{pmatrix} 1 & \cdots & 1 \\ -(\mathbf{u}_{n,\varphi_1})_1 & \cdots & -(\mathbf{u}_{n,\varphi_D})_1 \\ \vdots & \ddots & \vdots \\ -(\mathbf{u}_{n,\varphi_1})_n & \cdots & -(\mathbf{u}_{n,\varphi_D})_n \end{pmatrix} + \overline{\mathbf{G}}_{[1,I_\Phi \dotplus 1],I_\mathcal{D} \dotplus 1} \cdot \begin{pmatrix} -(\mathbf{u}_{n,\varphi_1})_{n+1} & & 0 \\ & \ddots & \\ 0 & & -(\mathbf{u}_{n,\varphi_1})_{n+1} \end{pmatrix},$$

$$\mathbf{g}_2 = \begin{pmatrix} 1 & \cdots & 1 \end{pmatrix} \cdot \mathbf{g}_{2_1} + \mathbf{g}_{2_2},$$

where we used

$$\mathbf{g}_{2_1} = \left(\overline{\mathbf{G}}^{\intercal}_{I_{\mathcal{D}}+1,[1,I_{\Phi}+1]} \circ \begin{pmatrix} 1 & \cdots & 1 \\ -(\mathbf{u}_{n,\varphi_1})_1 & \cdots & -(\mathbf{u}_{n,\varphi_D})_1 \\ \vdots & \ddots & \vdots \\ -(\mathbf{u}_{n,\varphi_1})_n & \cdots & -(\mathbf{u}_{n,\varphi_D})_n \end{pmatrix} \right)$$

$$\mathbf{g}_{2_2} = (\overline{\mathbf{G}}_{2,2}, \cdots, \overline{\mathbf{G}}_{D+1,D+1}) \circ (-(\mathbf{u}_{n,\varphi_1})_{n+1}, \cdots, -(\mathbf{u}_{n,\varphi_D})_{n+1}).$$

Using this definition of $\mathbf{g}_1$ and $\mathbf{g}_2$, one can show by performing all remaining multiplications that the vector Δ as defined in the proposition indeed represents the desired error estimates. □

4.4 Experiments

In this section we present numerical results for the online greedy basis construction method introduced above. All tests were performed using our C++ library dune-rb, see Chapter 7.

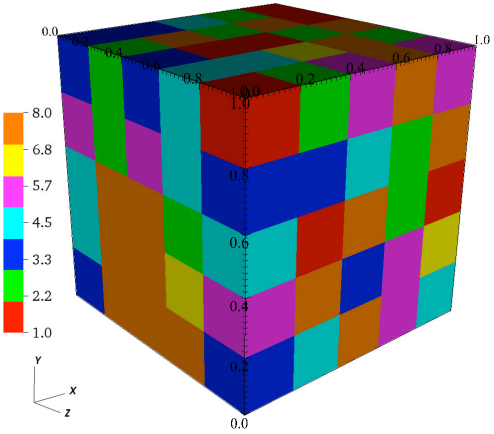
For our tests, we solve the pressure equation (Definition 3.3) ignoring gravity ($G = 0$) and using the total mobility γ given by

$$\gamma(\mu, x) = \sum_{i=1}^{8} (\mu)_i \cdot \chi_i(x),$$

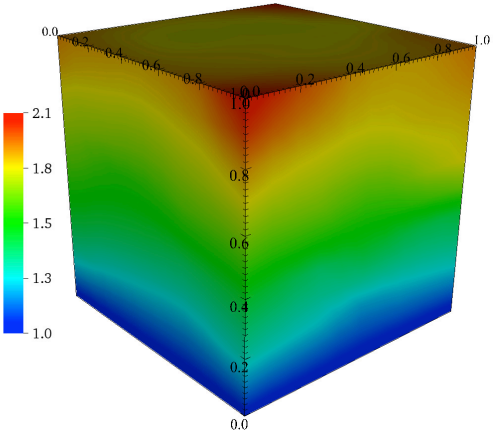
for $\mu \in \mathcal{P} = (0, 10]^8$. Here, as usual, $(\mu)_i \in \mathbb{R}$ denotes the i-th component of the vector μ and the functions $\chi_i : \Omega \to \{0, 1\}$ denote the characteristic functions for the eight subdomains of Ω sketched in Figure 4.3. Furthermore, Figure 4.3 shows a typical solution for a given parameter. For more details on the parameterization also see Section 3.3.

Offline Phase

We discretize Ω using 1000 cubes. The training set $\mathcal{P}_{\text{tr}} \subset \mathcal{P}$ is given by a lognormal distribution centered at 2 in each component. We generate a traditional greedy-basis Φ_G using Algorithm 2.7 (see Section 2.4) with a training set $\mathcal{P}^G_{\text{tr}}$ with $|\mathcal{P}^G_{\text{tr}}| = 1000$ and a tolerance of 10^{-5} and a dictionary $\mathcal{D}$ using the approach described in Section 4.1 using a training set $\mathcal{P}^{\mathcal{D}}_{\text{tr}} \supset \mathcal{P}^G_{\text{tr}}$ with $|\mathcal{P}^{\mathcal{D}}_{\text{tr}}| = 2000$. While the



(a)



(b)

Figure 4.3 – Heat diffusion coefficient $\lambda(\mu) : \Omega \to \mathbb{R}$ (a) and solution $u_h(\mu)$ (b) for $\mu = (1, 2, 3, 4, 5, 6, 7, 8)$.

generation of the traditional greedy-basis takes more than 9 hours and produces about 600 megabytes of data, the generation of the dictionary takes only one hour but produces about 1.1 gigabytes of data.

Online Phase

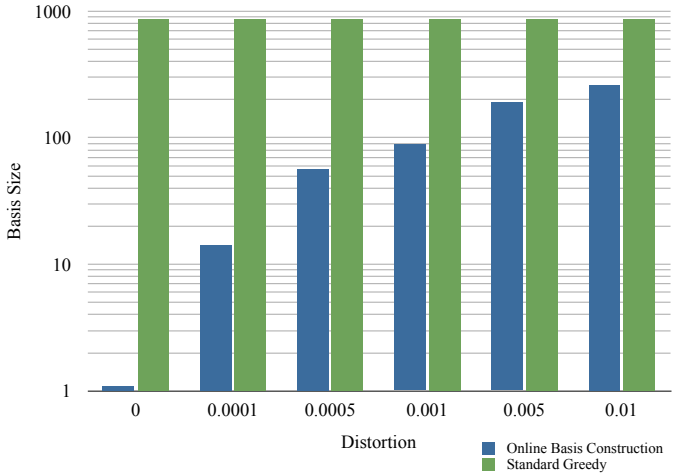
Using both the basis Φ_G and the dictionary $\mathcal{D}$ we run online simulations on the test set

$$\mathcal{T} = \{\mu + \rho r \,|\, \mu \in \mathcal{P}^G_{\mathrm{tr}}, r \in \mathcal{R}\}$$

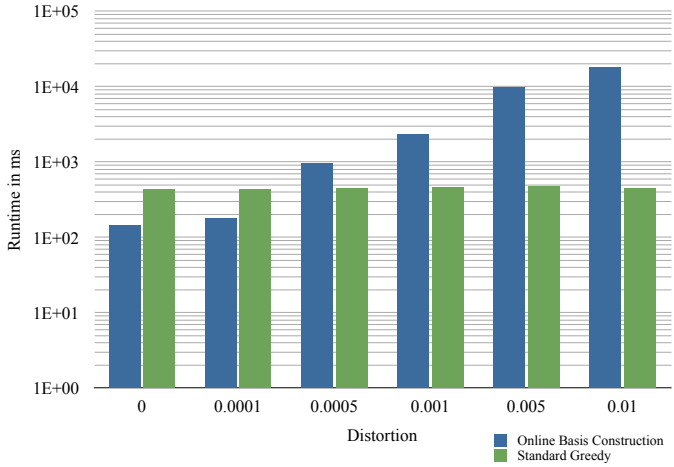
where $\mathcal{R}$ contains random numbers in $(0,1]^8$ and $\rho \in \mathbb{R}$ denotes a distortion scale. For our online basis construction algorithm we use the same error tolerance as for the standard greedy algorithm: $\varepsilon = 10^{-5}$.

Figure 4.4(a) shows the resulting basis sizes N for the standard greedy method (which is fixed by the basis construction during the offline phase, here: $N = 871$) and the basis size N resulting from Algorithm 4.4 for different values of the distortion scale ρ. We see that, especially for small disturbances of the training parameters, our online basis generation algorithm yields substantially smaller bases.

For small disturbances ρ this pays out in terms of runtime: Figure 4.4(b) shows mean online runtimes for the two algorithms and different values of ρ. This runtime includes reduced simulation, error estimation and, for our algorithm, the time needed for basis construction. Beginning with distortions in the range of $5 \cdot 10^{-5}$ our algorithm is slower than the standard approach as we then need lots of basis enrichment iterations in Algorithm 4.4. Still, it pays out to use our algorithm even in these cases as it consistently fulfills the error bound while the standard greedy method violates the error tolerance for $\rho = 5 \cdot 10^{-3}$ (error: $\max_{\mu\in\mathcal{T}} \Delta_{X_N}(\mu) = 1.64 \cdot 10^{-5}$) and $\rho = 1 \cdot 10^{-2}$ (error: $\max_{\mu\in\mathcal{T}} \Delta_{X_N}(\mu) = 3.28 \cdot 10^{-5}$) as we see in Table 4.1.



(a)



(b)

Figure 4.4 – Mean basis size N (a) and total runtime (b) during the online phase for the standard greedy method and our online basis construction algorithm for different values of ρ.

Distortion	Estimator Indicator	Standard Greedy
0	$4.09 \cdot 10^{-6}$	$8.69 \cdot 10^{-6}$
$1 \cdot 10^{-4}$	$9.87 \cdot 10^{-6}$	$8.64 \cdot 10^{-6}$
$5 \cdot 10^{-4}$	$9.99 \cdot 10^{-6}$	$8.63 \cdot 10^{-6}$
$1 \cdot 10^{-3}$	$1 \cdot 10^{-5}$	$8.45 \cdot 10^{-6}$
$5 \cdot 10^{-3}$	$9.99 \cdot 10^{-6}$	$1.64 \cdot 10^{-5}$
$1 \cdot 10^{-2}$	$9.99 \cdot 10^{-6}$	$3.28 \cdot 10^{-5}$

Table 4.1 – Estimated errors $\max_{\mu \in \mathcal{T}} \Delta_{X_N}(\mu)$ on the test set $\mathcal{T}$ during the online phase.

4.5 Summary and Outlook

In this chapter we presented a novel model order reduction approach that constructs an extensive dictionary of basis vector candidates during a preparatory *offline* step. During the *online* phase, a custom-made reduced basis space is built up for a given parameter and simulations are carried out in that space.

We are able to demonstrate the applicability of our approach in theory and in numerical experiments. Our approach yields smaller bases and, at least for small distortions from the training parameters, comparable runtimes as the standard greedy basis construction procedure.

One possible extension to our approach would be a smarter dictionary construction algorithm: by using a greedy-type algorithm to construct the dictionary during the offline phase—that is by iteratively extending the dictionary with solutions for the parameter worst approximated in the current dictionary—we hope to establish dictionaries with good approximation quality while enabling low online runtimes at the same time.

In [23] we applied the idea of online basis construction to parameterized evolution systems.

Chapter 5

The Localized Reduced Basis Multiscale Method

In this chapter, we combine the idea of problem-specific local basis functions on a coarse mesh that is used in multiscale methods (see Chapter 6) with the idea of model order reduction for parameterized partial differential equations that depend affinely on the parameter, see Chapter 2.

The motivating application for the derivation of this method is two-phase flow in a porous medium. As described in the introduction in Chapter 1, these applications usually yield large equation systems due to small-scale heterogeneities in the physical parameters and large extent of the computational domain. At the same time, possible parameterizations for these kinds of problems come to mind easily: Parameterized boundary data, well locations, injection and production rates and others are examples that establish the idea to use model order reduction techniques like the reduced basis method in this setting to allow real-time or many-query simulations (see Chapter 2). Nevertheless, RB methods easily become unfeasible for these applications as there is no control over the number of snapshots that is necessary to guarantee a given error tolerance during greedy basis construction (Algorithm 2.7, Section 2.4) in the

offline phase. More precisely: One can only control the approximation quality of the reduced basis *or* the number of snapshots (and hence basis size), not both at the same time. For that reason, with increasing size of the detailed problem, the basis construction becomes unreasonably expensive and a different approach may be needed.

In [2, 57], we introduced the so-called localized reduced basis multiscale (LRBMS) method. It connects ideas from numerical multiscale methods with the RB approach and aims at reducing the offline time of the RB method while ensuring at least identical approximation quality during the online time, possibly at a slightly increased cost in terms of online runtime.

The rest of this chapter is dedicated to the exposition of the details of this method and its application to stationary elliptic and instationary parabolic (two-phase flow) problems. Most parts of this chapter were originally published in [2, 57].

5.1 Derivation of the Method

In this section, we describe the key features of our novel model order reduction technique. The main idea is a spatial localization of the approximating function space based on a *coarse* mesh on the physical domain. This *localized* function space will be spanned by a set of local bases, where each may for example consist of localized global snapshots on a fine mesh.

5.1.1 Discretization

In addition to the *fine* mesh $\mathcal{T}_h$ from Section 3.2, we introduce a second, coarser mesh on the domain Ω: Let $\mathcal{T}_H$ be an admissible mesh with mesh size $H \gg h > 0$,

$$H = \max_{E \in \mathcal{T}_H} H_E = \max_{E \in \mathcal{T}_H} \operatorname{diam}(E).$$

Cells in $\mathcal{T}_H$ will be denoted by E, intersections of two elements or of one element and the boundary by F, and the set of all coarse intersections by Γ_A. Furthermore, we assume the two meshes to be

matching in the sense that for each $F \in \Gamma_A$ there exist $m \in \mathbb{N}$ and $f_1, \ldots, f_m \in \Gamma_a$ such that $F = \bigcup_{i=1}^{m} f_i$, see Figure 5.1.

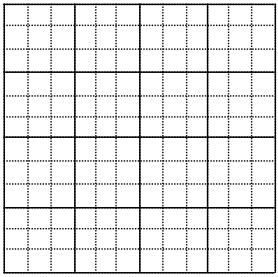


Figure 5.1 – Coarse (solid lines) and fine (dotted lines) grids with matching interfaces.

Based on the coarse mesh $\mathcal{T}_H$ we introduce the localized reduced-dimensional function space $\mathcal{W}_N$.

Definition 5.1 (Reduced Broken Space)**.** For every coarse grid cell $E \in \mathcal{T}_H$, we assume a set of linearly independent functions $\varphi_1^E, \ldots, \varphi_{N^E}^E \in \mathcal{V}_h$ with $\operatorname{supp}(\varphi_i^E) \subset E$ to be given. We then define *local reduced bases*

$$\Phi^E = \{\varphi_1^E, \ldots, \varphi_{N^E}^E\}$$

and call the set Φ_N of size $N = \sum_{E \in \mathcal{T}_H} N^E$,

$$\Phi_N = \bigcup_{E \in \mathcal{T}_H} \Phi^E$$

the *global reduced basis*. Further, we define the *reduced broken space*, spanned by the global reduced basis Φ_N:

$$\mathcal{W}_N = \operatorname{span}(\Phi_N).$$

We point out that, because of our discontinuous Galerkin ansatz, see Definition 3.1, the reduced broken space is nested in the fine-scale space: $\mathcal{W}_N \subset \mathcal{V}_h$. Using the reduced basis Φ_N, we compute the coarse-scale pressure approximation $p_H \in \mathcal{W}_N$.

Definition 5.2 (LRBMS Pressure Discretization)**.** For a given reduced broken space $\mathcal{W}_N$ and mobilities γ_o, $\gamma_w, \gamma : \Omega \to \mathbb{R}^{\geq 0}$ where $\gamma = \gamma_o + \gamma_w$, the reduced solution $p_H \in \mathcal{W}_N$ to Equation (3.1.1) is given by

$$b_h(p_H, w; \gamma) = l_h(w; \gamma_o, \gamma_w), \quad \forall w \in \mathcal{W}_N,$$

where $b_h(\cdot, \cdot; \cdot)$ and $l_h(\cdot; \cdot, \cdot)$ were given in Equations (3.2.2) and (3.2.4).

Next we would like to demonstrate that problem 5.2 has a unique solution. As a means to this end, we will introduce an energy norm and prove that $b_h(\cdot, \cdot; \cdot)$ is coercive with respect to that norm. All analysis is performed on $\mathcal{V}_h \supset \mathcal{W}_N$ which yields the desired results in $\mathcal{W}_N$.

The proof of coercivity of $b_h(\cdot, \cdot; \cdot)$ is standard and was originally given in [54] for a slightly different formulation of the bilinear form. We repeat it here in detail for the sake of completeness. The proof is written along the lines of [73], Section 2.7.1. We start by introducing the energy norm $\|\cdot\|_{\mathcal{E}_\gamma}$.

Definition 5.3. Given a function $v \in \mathcal{V}_h$ and $\gamma : \Omega \to \mathbb{R}^{\geq 0}$ we define the energy norm

$$\|v\|_{\mathcal{E}_\gamma} = \left(\sum_{e \in \mathcal{T}_h} \int_e \gamma \kappa \nabla v \cdot \nabla v \, \mathrm{d}x + \sum_{f \in \Gamma_j} \frac{\sigma_f}{h_f} \int_f [\![v]\!] \, [\![v]\!] \, \mathrm{d}S \right)^{1/2}.$$

Remark 5.4. *It was shown in [54] that $\|\cdot\|_{\mathcal{E}_\gamma}$ indeed defines a norm on $\mathcal{V}_h$ if $\gamma\kappa$ fulfills* (3.1.3)*.*

Theorem 5.5. *Let the constant c_f in Equation* (3.2.3) *be bounded from below by a sufficiently large constant and let $\gamma : \Omega \to \mathbb{R}^{\geq 0}$ be given such that $\gamma\kappa$ fulfills* (3.1.3)*. The bilinear form $b_h(\cdot, \cdot; \cdot)$ is then coercive on $\mathcal{V}_h$:*

$$b_h(v, v; \gamma) \geq \alpha \|v\|^2_{\mathcal{E}_\gamma}, \quad \forall v \in \mathcal{V}_h.$$

Proof. We recall the definition of b_h:

$$b_h(v,w;\gamma) = \sum_{e\in\mathcal{T}_h}\int_e \gamma\kappa\nabla v\cdot\nabla w\,\mathrm{d}x + \sum_{f\in\Gamma_j}\frac{\sigma_f}{h_f}\int_f [\![v]\!]\,[\![w]\!]\,\mathrm{d}S$$
$$- \sum_{f\in\Gamma_j}\int_f \left(\{\!\{\gamma\kappa\nabla v\cdot \mathrm{n}_f\}\!\}\,[\![w]\!] + \{\!\{\gamma\kappa\nabla w\cdot \mathrm{n}_f\}\!\}\,[\![v]\!]\right)\mathrm{d}S.$$

We will now compute an upper bound for the last sum. Given an intersection $f\in\Gamma_a$ of two elements $e_1,e_2\in\mathcal{T}_h$ we derive for the L^2-norm

$$\begin{aligned}
&\|\,\{\!\{\sqrt{\gamma\kappa}\nabla v\cdot \mathrm{n}_f\}\!\}\,\|_{L^2(f)}\\
&\quad\le \tau_{e_1,f}\|\,(\sqrt{\gamma\kappa}\nabla v\cdot \mathrm{n}_f)|_{e_1}\|_{L^2(f)} + \tau_{e_2,f}\|\,(\sqrt{\gamma\kappa}\nabla v\cdot \mathrm{n}_f)|_{e_2}\|_{L^2(f)}\\
&\quad\le k_2\left(\tau_{e_1,f}\|\,(\nabla v\cdot \mathrm{n}_f)|_{e_1}\|_{L^2(f)} + \tau_{e_2,f}\|\,(\nabla v\cdot \mathrm{n}_f)|_{e_2}\|_{L^2(f)}\right)\\
&\quad\le C\cdot k_2\cdot {h_f}^{-1/2}\left(\tau_{e_1,f}\|\nabla v\|_{L^2(e_1)} + \tau_{e_2,f}\|\nabla v\|_{L^2(e_2)}\right),
\end{aligned}$$

where we used the definition (3.2.1) of the mean, the boundedness of $\gamma\kappa$ (3.1.3) and the trace theorem (see [6], Equation (2.5)).

Now, for given real numbers $\alpha,\beta,x,y\ge 0$ we have:

$$\begin{aligned}
&(\alpha x+\beta y)^2\\
&\quad=\alpha^2x^2+\beta^2y^2+2\alpha\beta xy\\
&\quad\le\alpha^2x^2+\beta^2y^2+\alpha\beta(x^2+y^2)\\
&\quad\le\alpha^2x^2+\beta^2y^2+\alpha\beta(x^2+y^2)+\alpha^2y^2+\beta^2x^2+\alpha\beta(x^2+y^2)\\
&\quad=(\alpha^2+\beta^2+2\alpha\beta)(x^2+y^2)\\
&\quad=(\alpha+\beta)^2(x^2+y^2),
\end{aligned}$$

that is: $(\alpha x+\beta y)\le(\alpha+\beta)(x^2+y^2)^{1/2}$. Using this and the Cauchy-Schwarz inequality we get

$$\begin{aligned}
&\int_f \{\!\{\gamma\kappa\nabla v\cdot \mathrm{n}_f\}\!\}\,[\![v]\!]\,\mathrm{d}S\\
&\quad\le C\cdot k_2 {h_f}^{-1/2}\left(\tau_{e_1,f}\|\nabla v\|_{L^2(e_1)} + \tau_{e_2,f}\|\nabla v\|_{L^2(e_2)}\right)\|\,[\![v]\!]\,\|_{L^2(f)}\\
&\quad\le C\cdot k_2 {h_f}^{-1/2}\left(\|\nabla v\|^2_{L^2(e_1)} + \|\nabla v\|^2_{L^2(e_2)}\right)^{1/2}\|\,[\![v]\!]\,\|_{L^2(f)},
\end{aligned}$$

where we used $\tau_{e_1,f} + \tau_{e_2,f} = 1$.

For boundary intersections f with $f \in \partial\Omega \cap \partial e$ for $e \in \mathcal{T}_h$ we can derive in a similar fashion:

$$\int_f \{\!\{\gamma\kappa\nabla v \cdot \mathrm{n}_f\}\!\} \, [\![v]\!] \, \mathrm{d}S \leq Ck_2 {h_f}^{-1/2} \|\nabla v\|_{L^2(e)} \| \, [\![v]\!] \, \|_{L^2(f)}.$$

Using the number $n_{\max}$ of maximum element neighbors in the mesh we can proceed

$$\begin{aligned}
&\sum_{f\in\Gamma_j} \int_f \{\!\{\gamma\kappa\nabla v \cdot \mathrm{n}_f\}\!\} \, [\![v]\!] \, \mathrm{d}S \\
&\leq Ck_2 \sum_{f\in\Gamma_i} {h_f}^{-1/2} \left(\|\nabla v\|^2_{L^2(e_1)} + \|\nabla v\|^2_{L^2(e_2)} \right)^{1/2} \| \, [\![v]\!] \, \|_{L^2(f)} \\
&\quad + Ck_2 \sum_{f\in\Gamma_{p,d}} {h_f}^{-1/2} \|\nabla v\|_{L^2(e)} \| \, [\![v]\!] \, \|_{L^2(f)} \\
&\leq Ck_2 \Big(\sum_{f\in\Gamma_i} \| \, [\![v]\!] \, \|^2_{L^2(f)} {h_f}^{-1} \Big)^{1/2} \Big(\sum_{f\in\Gamma_i} \|\nabla v\|^2_{L^2(e_1)} + \|\nabla v\|^2_{L^2(e_2)} \Big)^{1/2} \\
&\quad + Ck_2 \Big(\sum_{f\in\Gamma_{p,d}} \| \, [\![v]\!] \, \|^2_{L^2(f)} {h_f}^{-1} \Big)^{1/2} \Big(\sum_{f\in\Gamma_{p,d}} \|\nabla v\|^2_{L^2(e)} \Big)^{1/2} \\
&\leq Ck_2 \sqrt{n_{\max}} \Big(\sum_{f\in\Gamma_j} {h_f}^{-1} \| \, [\![v]\!] \, \|^2_{L^2(f)} \Big)^{1/2} \Big(\sum_{e\in\mathcal{T}_h} \|\nabla v\|^2_{L^2(e)} \Big)^{1/2}.
\end{aligned}$$

For $\delta > 0$ we get using Young's inequality and (3.1.3)

$$\begin{aligned}
&\sum_{f\in\Gamma_j} \int_f \{\!\{\gamma\kappa\nabla v \cdot \mathrm{n}_f\}\!\} \, [\![v]\!] \, \mathrm{d}S \\
&\qquad \leq \frac{C^2 k_2^2 n_{\max}}{2\delta k_1} \sum_{f\in\Gamma_j} \frac{\| \, [\![v]\!] \, \|^2_{L^2(f)}}{\operatorname{diam}(f)} + \frac{\delta}{2} \sum_{e\in\mathcal{T}_h} \int_e k_1 \nabla v^2 \, \mathrm{d}x \\
&\qquad \leq \frac{C^2 k_2^2 n_{\max}}{2\delta k_1} \sum_{f\in\Gamma_j} \frac{\| \, [\![v]\!] \, \|^2_{L^2(f)}}{\operatorname{diam}(f)} + \frac{\delta}{2} \sum_{e\in\mathcal{T}_h} \int_e \gamma\kappa \nabla v^2 \, \mathrm{d}x,
\end{aligned}$$

such that we get using $\operatorname{diam}(f) = h_f$

$$
\begin{aligned}
&b_h(v, v; \gamma) \\
&= \sum_{e \in \mathcal{T}_h} \int_e \gamma \kappa \nabla v \cdot \nabla v \, \mathrm{d}x + \sum_{f \in \Gamma_j} \frac{\sigma_f}{h_f} \int_f [\![v]\!] \, [\![v]\!] \, \mathrm{d}S \\
&\quad - 2 \sum_{f \in \Gamma_j} \int_f \{\!\{\gamma \kappa \nabla v \cdot \mathrm{n}_f\}\!\} \, [\![v]\!] \, \mathrm{d}S \\
&\geq \sum_{e \in \mathcal{T}_h} \int_e \gamma \kappa \nabla v \cdot \nabla v \, \mathrm{d}x + \sum_{f \in \Gamma_j} \frac{\sigma_f}{h_f} \int_f [\![v]\!] \, [\![v]\!] \, \mathrm{d}S \\
&\quad - \frac{C^2 k_2^2 n_{\max}}{\delta k_1} \sum_{f \in \Gamma_j} \frac{1}{h_f} \int_f [\![v]\!]^2 \, \mathrm{d}S - \delta \sum_{e \in \mathcal{T}_h} \int_e \gamma \kappa \nabla v^2 \, \mathrm{d}x \\
&= (1 - \delta) \sum_{e \in \mathcal{T}_h} \int_e \gamma \kappa \nabla v^2 \, \mathrm{d}x + \sum_{f \in \Gamma_j} \frac{\sigma_f - \frac{C^2 k_2^2 n_{\max}}{\delta k_1}}{h_f} \int_f [\![v]\!]^2 \, \mathrm{d}S.
\end{aligned}
$$

Now choose $\delta = \frac{1}{2}$ to get

$$
\begin{aligned}
b_h(v, v; \gamma) &\geq \frac{1}{2} \sum_{e \in \mathcal{T}_h} \int_e \gamma \kappa \nabla v^2 \, \mathrm{d}x + \sum_{f \in \Gamma_j} \frac{\sigma_f - \frac{2C^2 k_2^2 n_{\max}}{k_1}}{h_f} \int_f [\![v]\!]^2 \, \mathrm{d}S \\
&\geq \frac{1}{2} \sum_{e \in \mathcal{T}_h} \int_e \gamma \kappa \nabla v^2 \, \mathrm{d}x + \frac{1}{2} \sum_{f \in \Gamma_j} \frac{\sigma_f}{h_f} \int_f [\![v]\!]^2 \, \mathrm{d}S \\
&= \alpha \, \|v\|_{\mathcal{E}_\gamma}^2
\end{aligned}
$$

with $\alpha = \frac{1}{2}$ if c_f is chosen such that $\sigma_f \geq \frac{4C^2 k_2^2 n_{\max}}{k_1}$. Here the constants k_1, k_2 obviously depend on the total mobility γ. □

Theorem 5.6 (Existence and Uniqueness of Solutions)**.** *Given $c_f > 0$ such that $\sigma_f \geq \frac{4C^2 k_2^2 n_{\max}}{k_1}$ the LRBMS pressure discretization 5.2 has a unique solution.*

Proof. The LRBMS problem 5.2 is a linear equation for the unknown pressure. Due to the finite dimension of $\mathcal{W}_N$, uniqueness and existence of solutions are equivalent. It therefore suffices to show that, given two solutions $p_1, p_2 \in \mathcal{W}_N$ to 5.2, we have $p_1 = p_2$.

If p_1 and p_2 are solutions to 5.2, we have

$$\begin{aligned} b_h(p_1 - p_2, p_1 - p_2; \gamma) &= l_h(p_1 - p_2; \gamma_o, \gamma_w) - l_h(p_1 - p_2; \gamma_o, \gamma_w) \\ &= 0. \end{aligned}$$

Using the coercivity result 5.5 we get

$$0 = b_h(p_1 - p_2, p_1 - p_2; \gamma) \geq \alpha \left\| p_1 - p_2 \right\|_{\mathcal{E}_\gamma}^2,$$

for $\alpha > 0$. This yields $p_1 = p_2$ since $\|\cdot\|_{\mathcal{E}_\gamma}$ is a norm. □

5.1.2 Offline–Online Decomposition

The computations for the LRBMS method can be split into computationally demanding parts that will be executed during the offline phase and computationally inexpensive parts to be performed during the online phase. More exactly, all computations depending on the fine mesh size h can be performed during the offline phase rendering the online phase totally independent of h. The computational effort during the online phase is usually polynomial in the size of the reduced broken space. As indicated in Section 3.3 we will replace the wetting, non-wetting and total mobilities (γ_w, γ_o and γ) in Definition 5.2 by the affine sum (3.3.1).

During the offline phase, we compute a reduced basis Φ of size $N \in \mathbb{N}$, $\Phi = \{\varphi_1, \ldots, \varphi_N\}$ and the projection of all parameter-independent parts of Definition 5.2, that is: all parts that depend on the location in space but not on the parameter-dependent functions θ_q. We can identify the terms $\underline{b}_q, \underline{c} \in \mathbb{R}^{N \times N}$, $q = 1, \ldots, M$ with

$$\begin{aligned} (\underline{b}_q)_{ij} = &\sum_{e \in \mathcal{T}_h} \int_e \Lambda^q \kappa \nabla \varphi_i \nabla \varphi_j \, \mathrm{d}x \\ &- \sum_{f \in \Gamma_j} \int_f \left[\{\!\{ \Lambda^q \kappa \nabla \varphi_i \cdot \mathrm{n}_f \}\!\} \left[\!\left[\varphi_j \right]\!\right] + \{\!\{ \Lambda^q \kappa \nabla \varphi_j \cdot \mathrm{n}_f \}\!\} \left[\!\left[\varphi_i \right]\!\right] \right] \mathrm{d}S, \\ (\underline{c})_{ij} = &\sum_{f \in \Gamma_j} \frac{\sigma_f}{h_f} \int_f \left[\!\left[\varphi_i \right]\!\right] \left[\!\left[\varphi_j \right]\!\right] \mathrm{d}S, \end{aligned}$$

and the terms $\underline{e}$, $\underline{d}_q \in \mathbb{R}^N$, $q = 1, \dots, M$ for the right hand side:

$$(\underline{e})_i = \sum_{e \in \mathcal{T}_h} \int_e q_1 \varphi_i \,\mathrm{d}x + \sum_{f \in \Gamma_{p,d}} \int_f \frac{\sigma_f}{h_f} \varphi_i p_D \,\mathrm{d}S + \sum_{f \in \Gamma_{p,n}} \int_f v_N \varphi_i \,\mathrm{d}S,$$

$$\begin{aligned}(\underline{d}_q)_i = &\sum_{e \in \mathcal{T}_h} \int_e (\Lambda_o^q \varrho_o + \Lambda_w^q \varrho_w) \kappa \, G \cdot \nabla \varphi_i \,\mathrm{d}x \\ &- \sum_{f \in \Gamma_i \cup \Gamma_{p,d}} \int_f \{\!\{ (\Lambda_o^q \varrho_o + \Lambda_w^q \varrho_w) \kappa \, G \cdot \mathrm{n}_f \}\!\} \, [\![\varphi_i]\!] \,\mathrm{d}S \\ &- \sum_{f \in \Gamma_{p,d}} \int_f (\Lambda^q \kappa \nabla \varphi_i \cdot \mathrm{n}_f) \, p_D \,\mathrm{d}S.\end{aligned}$$

where the profiles Λ_w^q, Λ_o^q and Λ^q will be defined in the subsequent sections.

Let $\theta_q = \theta_q(\mu)$ for a given parameter $\mu \in \mathcal{P}$ in the case that only the pressure equation is solved (see Section 5.2) or $\theta_q = \theta_q(s)$ for a given saturation $s \in \mathcal{V}_h$ in the case that full two-phase flow is solved (see Section 5.3). During the online phase, the solution $\underline{p}_H \in \mathbb{R}^N$ of the discrete equivalent of the reduced equation 5.2 is then given by

$$\left(\underline{c} + \sum_{q=1}^{M} \theta_q \underline{b}_q \right) \underline{p}_H = \underline{e} + \sum_{q=1}^{M} \theta_q \underline{d}_q. \tag{5.1.1}$$

The exact definition of the coefficients θ_i, $i = 1, \dots, M$ will be given in the respective sections below.

Now, while the computation of the quantities $\underline{b}_q, \underline{c}, \underline{d}_q$ and $\underline{e}$ has a complexity polynomial in $|\mathcal{T}_h|$, computing the sums in the reduced system (5.1.1) has a complexity polynomial in N where $N \ll |\mathcal{T}_h|$.

Remark 5.7. *As mentioned in Section 3.3, other parameterizations like boundary value parameterizations would be possible for Problem (3.1.1-3.1.7). As long as these parameterizations are parameter-separable, the offline–online decomposition works analogously. For different kinds of parameterizations, see Section 3.3 and e.g., [69, 57, 37].*

5.1.3 Basis Construction

The basis construction for the reduced broken space that yields the local reduced bases will be described separately for the two application scenarios as a general notation fitting both would make the exposition unnecessarily complicated.

5.2 Application to Stationary Elliptic Problems

After explaining the main ideas of the LRBMS method (except for the basis construction algorithm), we will now demonstrate its application to a stationary elliptic equation, more precisely: The pressure equation (Definition 5.2). Applying the method to the stationary equation without coupling it to the saturation equation allows us to take a deeper look at its features without studying the coupling between the two equations. The results of this section were published in [2].

5.2.1 Parameterization

In this section, the parameterization of the wetting, non-wetting and total mobility γ_w, γ_o and γ in Definition 3.3 is for a given parameter $\mu \in \mathcal{P} \subset \mathbb{R}^k$ ($k \in \mathbb{N}$) defined as

$$\gamma_w(\mu, x) = \sum_{q=1}^{M} \theta_q(\mu)\Lambda_w^q(x), \quad \gamma_o(\mu, x) = \sum_{q=1}^{M} \theta_q(\mu)\Lambda_o^q(x), \quad x \in \Omega,$$

and $\gamma(\mu, x) = \gamma_w(\mu, x) + \gamma_o(\mu, x)$, where Λ_w^q, $\Lambda_o^q : \Omega \to \mathbb{R}$, $q = 1, \ldots, M$ are analytical functions. For an exact definition of the profiles and coefficients see Section 5.2.3.

5.2.2 Basis Construction

We obtain the reduced broken space $\mathcal{W}_N$ by the following algorithm combining a localized variant of the greedy algorithm [79] with a final compression step by a PCA.

Algorithm 5.8 (Localized Greedy for Stationary Heat Diffusion)**.** *Given a finite set of training parameters $\mathcal{P}_{tr} \subset \mathcal{P}$, a maximum basis size $N_{max} \in \mathbb{N}$, $N_{max} > 0$, an error tolerance $\epsilon_{tol} \in \mathbb{R}^{>0} = \{x \in \mathbb{R} | x > 0\}$ and a PCA tolerance $\epsilon_{PCA} \in \mathbb{R}^{>0}$, the following algorithm produces a reduced broken space $\mathcal{W}_N$:*

(i) Pick a parameter $\mu_0 \in \mathcal{P}_{tr}$ and initialize the local bases by $\Phi^E_{(0)} := \emptyset$. Set $k = 0$ and $N^E_{(0)} = 0$ for all $E \in \mathcal{T}_H$.

(ii) Given local bases $\Phi^E_{(k-1)}$ of size $N^E_{(k-1)} \in \mathbb{N}$ for all $E \in \mathcal{T}_H$ and a parameter $\mu_{k-1} \in \mathcal{P}_{tr}$, compute a global fine DG snapshot $p_h(\mu_{k-1}) \in \mathcal{V}_h$ and set the extended local bases as $\Phi^E_{(k)} := \Phi^E_{(k-1)} \cup \{p_h(\mu_{k-1})|_E\}$ with size $N^E_{(k)} := N^E_{(k-1)} + 1$ for all $E \in \mathcal{T}_H$. Set the global basis as $\Phi_{(k)} := \bigcup_{E \in \mathcal{T}_H} \Phi^E_{(k)}$ of size $N_{(k)} := \sum_{E \in \mathcal{T}_H} N^E_{(k)}$ and $\mathcal{W}_{N_{(k)}} = \operatorname{span}(\Phi_{(k)})$. Compute all offline quantities for this basis (that is: the quantities $\underline{b}_q, \underline{c}, \underline{d}_q, \underline{e}$ from Section 5.1.2 and the necessary quantities for the error estimator, see Section 2.5).

(iii) Compute LRBMS approximations $p_N(\mu) \in \mathcal{W}_{N_{(k)}}$ for all training parameters $\mu \in \mathcal{P}_{tr}$ using the current basis and evaluate the error estimator to find the parameter $\mu_{(k)} \in \mathcal{P}_{tr}$ which maximizes the error estimator.

(iv) IF $N_{(k)} < N_{max}$ and if the estimated error for $p_N(\mu_{(k)})$ is larger than ϵ_{tol}: Set $k := k + 1$ and repeat from (ii).
ELSE: Apply the PCA to $\Phi^E_{(k)}$ with tolerance ϵ_{PCA} on each $E \in \mathcal{T}_H$ to obtain the localized orthogonalized reduced bases Φ^E of size $N^E \leq N^E_{(k)}$ on each $E \in \mathcal{T}_H$.

(v) Define the global reduced basis Φ of size $N_: = \sum_{E \in \mathcal{T}_H} N^E$ as $\Phi := \bigcup_{E \in \mathcal{T}_H} \Phi^E$ and compute all offline quantities for this basis.

5.2.3 Numerical Experiments

In the following we will demonstrate the performance of our approach using two different examples which fit in the framework of our

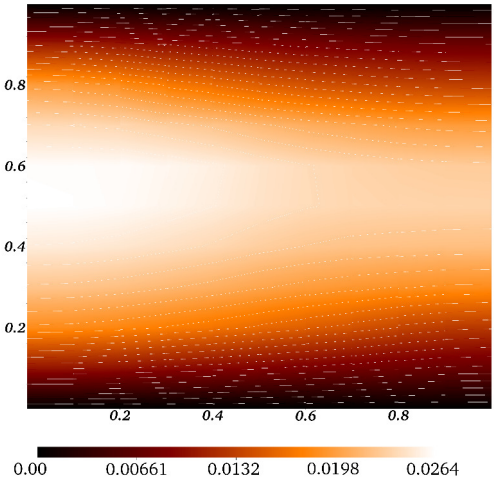


Figure 5.2 – Solution of the thermalblock problem with isolines for $\mu = (3, 6, 9, 2, 5, 8, 1, 4, 7, 10, 3, 6, 9, 2, 5, 8)^\intercal$.

elliptic problem. The first example, the 2D thermal block example, is easily scalable concerning parameter and spatial complexity and is therefore well-suited to highlight the basic principle of our method: The scaling between offline and online cost. The second example is much closer to real-world applications: Here we use 3D real-world data taken from the SPE10 benchmark problem [18], putting our method in the context of two-phase flow equations and multiscale problems. For details on implementation aspects see Chapter 7.

The Thermal Block Example

In this example, we solve the pressure equation (Definition 3.3) on $\Omega = [0, 1]^2$ ignoring gravity ($G = (0, 0)^\intercal$), with a constant permeability $\kappa \equiv 1$ and a parameter-dependent total mobility given by

$$\gamma(\mu, x) = \sum_{q=1}^{M} \theta_q(\mu) \Lambda^q(x),$$

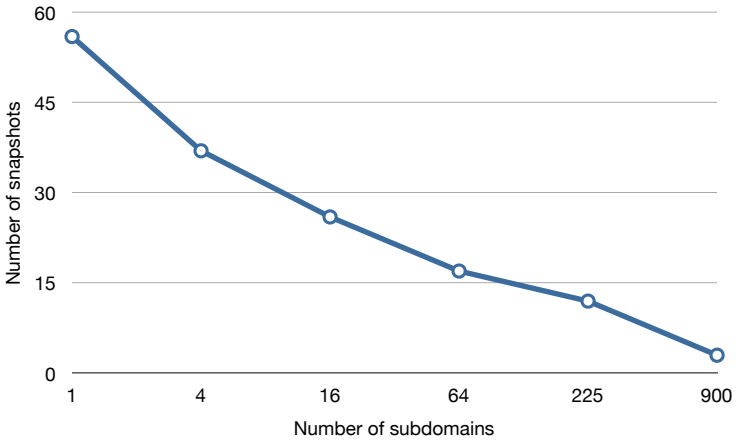


Figure 5.3 – Number of snapshots needed during the generation of the reduced basis with the localized greedy algorithm against number of coarse mesh elements.

with $\Lambda^q(x) := \chi_{\Omega_q}(x)$ where

$$\chi_{\Omega_q} : \Omega \to \{0,1\}, \quad \chi_{\Omega_q}(x) = \begin{cases} 1, & x \in \Omega_q, \\ 0, & \text{else.} \end{cases} \tag{5.2.1}$$

Here the $M \in \mathbb{N}$, $M > 0$ blocks $\Omega_q \subset \Omega$ with $Q_q \cap Q_s = \emptyset$ for $q \neq s$ are given by an equidistant mesh on Ω consisting of $M = 4 \cdot 4 = 16$ square elements. The coefficients $\theta_q(\mu)$ for $1 \leq q \leq M$ are given by $\theta_q(\mu) = \mu_q$ for $\mu_q \in [0.1, 10]$. We complete the description of the problem setting by choosing, cf. (3.1.7),

$$q_1 \equiv 1, \quad p_D \equiv 0, \quad v_N \equiv 0,$$
$$\Gamma_{p,d} = [0,1] \times \{0\} \cup [0,1] \times \{1\}, \quad \Gamma_{p,n} = \partial\Omega \setminus \Gamma_{p,d}.$$

We discretized this problem using a rectangular grid with $30 \cdot 30 = 900$ fine grid elements and applied our method using rectangular coarse meshes with $1, 2 \cdot 2, 4 \cdot 4, 8 \cdot 8, 15 \cdot 15$ and $30 \cdot 30$ elements (*subdomains*). We ran the localized greedy algorithm 5.8 with a

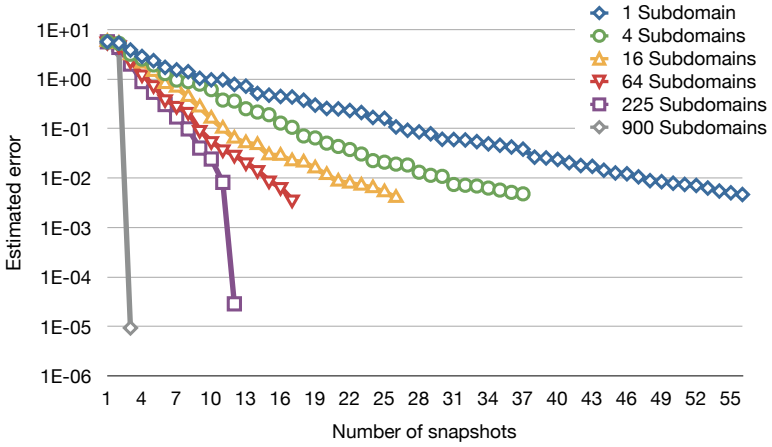


Figure 5.4 – Maximum estimated absolute error over training set against extension step during greedy algorithm for different sizes of the coarse mesh.

tolerance of $\epsilon_{\text{tol}} = 0.05$ for the estimated absolute error over the training set which consisted of 100 randomized parameters in the above-mentioned parameter domain. The training set consisted of the same random parameters for all basis generation procedures. A typical fine DG solution is plotted in Figure 5.2.

Figure 5.3 shows the number of total snapshots needed during the greedy algorithm to fulfill the error tolerance for different sizes of the coarse mesh. We see that we are able to scale the offline cost in terms of needed snapshots by choosing different coarse meshes. The number of snapshots ranges from 56 for one coarse grid element (which corresponds to a standard RB method) to only three snapshots for 900 coarse grid elements. The latter is the expected behavior since three linearly independent functions on each element are sufficient to represent a linear function in two spatial dimensions. This scaling quality of our approach can also be seen in Figure 5.4 where we demonstrate the evolution of the maximum estimated absolute error over the training set during the offline procedure. We see how with an increasing number of coarse elements the error decent gets

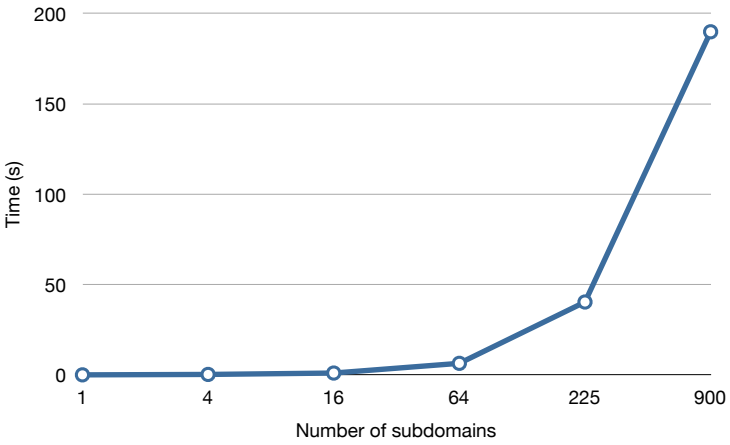


Figure 5.5 – Time needed to update the error estimator during the greedy algorithm: Mean over all steps in seconds against number of coarse mesh elements.

steeper.

Clearly these qualities come with additional costs: With increasing size of the coarse grid, the update of the error estimator becomes more and more costly as we have to solve the high-dimensional equation once for every new basis function on every coarse element. This increase in the offline cost can be seen in Figure 5.5, where we compare the mean estimator update time for the different coarse meshes. As the necessary computations in each step are independent of each other, this step can be easily parallelized. Thus it should be possible to compensate for the additional costs by increasing the number of used CPUs with increasing coarse grid size. For the experiments at hand we used a shared memory parallelization which needs to be further improved by more sophisticated parallelization techniques as our approach is obviously not able to fully compensate for the additional costs.

An estimator not relying on global computations would be desirable in this context. Such an estimator was recently introduced by Schindler et al. [75].

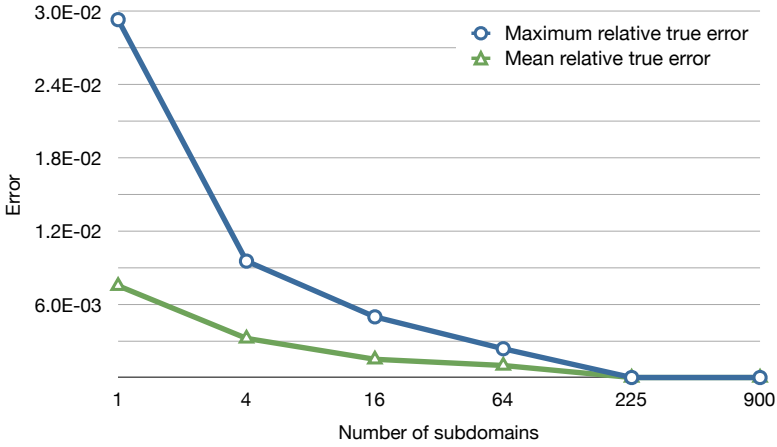


Figure 5.6 – Maximum and mean relative true error in the energy norm over the test set for different coarse mesh sizes.

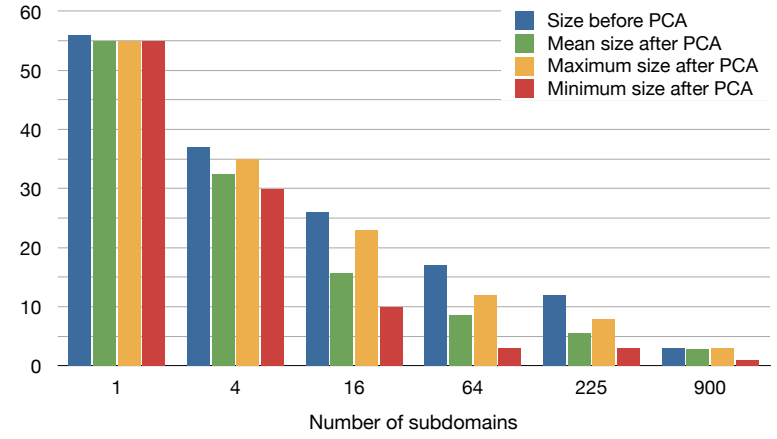


Figure 5.7 – Mean, maximal and minimal number of local basis functions on the different coarse meshes before and after the PCA.

Similar to the update time for the error estimator, the total training time (time for reduced simulations and estimator evaluations for all training parameters during the offline phase) increases with increasing coarse grid sizes as the reduced basis Φ grows a lot faster since $|\mathcal{T}_H|$ new basis vectors are added in each extension step of Algorithm 5.8. Nevertheless, since we aim at applications with extremely costly detailed simulations, this increase will usually be negligible.

Furthermore, our localized method has another nice quality: The online error over a given fixed test set decreases with increasing numbers of coarse elements. This is due to the greater flexibility in the reduced scheme with our method. This phenomenon can be seen in Figure 5.6 where the maximum relative true error (norm of difference between high- and reduced-dimensional solution) in the energy norm over the test set drops from 0.03 for 1 subdomain to 0.002 for 64 subdomains and finally to $7 \cdot 10^{-12}$ for 900 subdomains.

Finally, in Figure 5.7 we see the effect of the PCA. As described in Algorithm 5.8 we applied the PCA on each coarse element, using a tolerance of $\epsilon_{\text{PCA}} = 1 \cdot 10^{-7}$. We see that, up to a certain point, refinements of the coarse grid lead to bigger differences between maximum and minimum basis sizes. This behavior is expected as finer coarse meshes lead to a greater resemblance of the snapshots on each coarse element (in particular in regions where the snapshots do not differ too much, for example close to the Dirichlet boundaries). Beginning with a coarse grid size of about 225 this effect starts to reduce as the bases on the coarse elements get more and more compact and therefore the possibility for reduction with the PCA diminish. Also remarkable is the fact that the PCA reduces the standard greedy basis (which corresponds to 1 subdomain in Figure 5.7) only by one function. On one coarse element the greedy algorithm seems to work as good as the PCA: The basis is already very compact.

The SPE10 Example

The second example uses real-world data taken from the SPE10 benchmark problem [18]: we use the permeability field given in the SPE10 benchmark and a mobility function that models the flooding of the domain by one of the phases to solve the pressure equation on

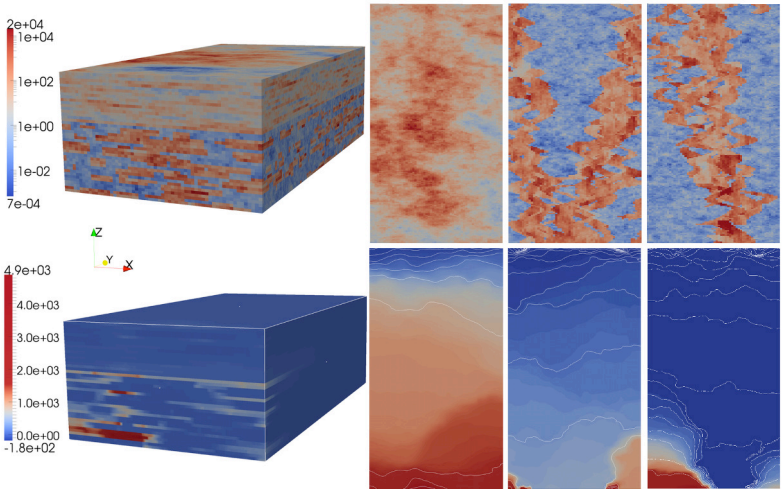


Figure 5.8 – Permeability (top) and solution (bottom) of the SPE10 problem for $\mu = (0.01, 0.01, 0.95, 0.01, 0.01, 0.01)^T$. Left: Whole field, z-axis scaled by 4. Right: Cuts along the x-y-plane at different values of z, demonstrating the channel structures in the permeability and matching solution with isolines (bottom).

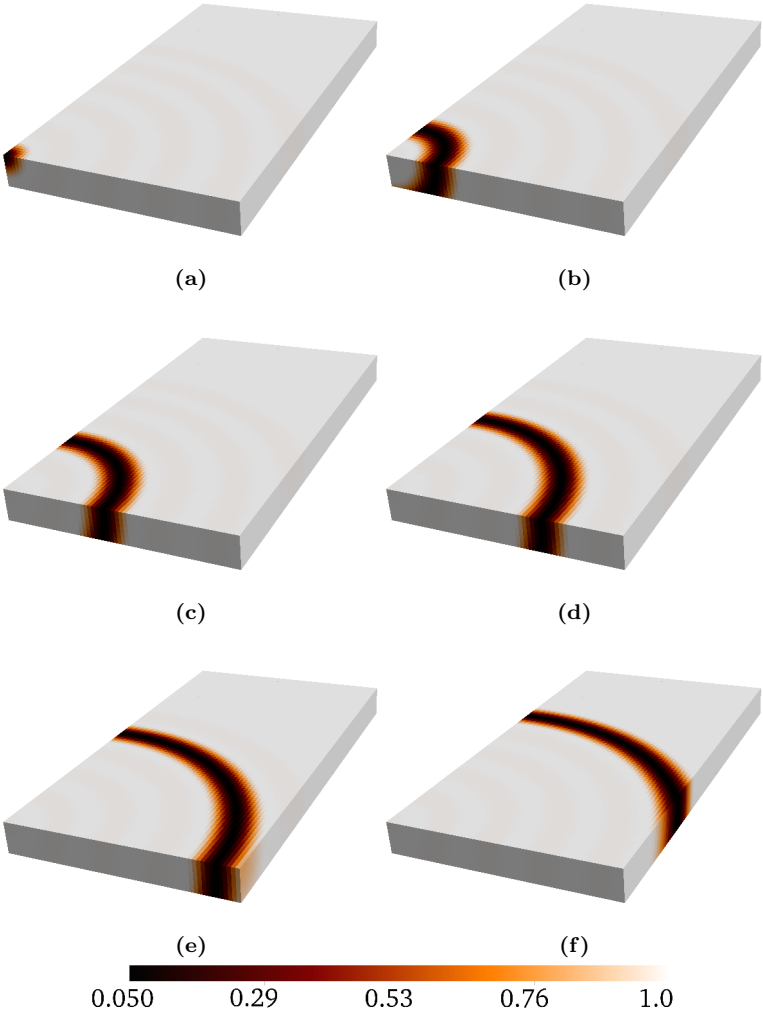


Figure 5.9 – Mobility for $\mu = \mu_i \in \mathcal{P}$ for $i = 1$ (a) to $i = 6$ (f) where $(\mu_i)_k = 0.95$ if $k = i$ and $(\mu_i)_k = 0.01$ else.

$\Omega = [0, 365.76] \times [0, 670.56] \times [0, 51.816]$. The fine mesh on Ω consists of $60 \cdot 220 \cdot 42 = 554400$ rectangular cuboids. The permeability field is displayed in Figure 5.8 (top).

For a given center $x_0 \in \mathbb{R}^3$, a given transition width $\varepsilon \in \mathbb{R}^{>0}$, $M \in \mathbb{N}^{\geq 1}$ and radii $\alpha_q \in \mathbb{R}^{>0}$ for $1 \leq q \leq M$ the total mobility is given by

$$\gamma(\mu, x) = \sum_{q=1}^{M} \theta_q(\mu) \Lambda^q(x),$$
$$\Lambda^q(x) = 1 - \chi_{B^{x_0}_{\alpha_q+\varepsilon}}(x) \cdot (1 - \chi_{B^{x_0}_{\alpha_q-\varepsilon}}(x)) \cdot \sin^2\left(\pi \tfrac{|x-x_0|+\varepsilon-\alpha_q}{2\varepsilon}\right)$$

where we used the coefficients $\theta_q(\mu) = \mu_q$ for $1 \leq q \leq M$. We assume $(\mu)_q \in (0, 1]$ and $\sum(\mu)_q \leq 1$. Furthermore B^c_r denotes the ball with radius r and center c. The functions $\chi_{B^c_r}$ again denote the characteristic functions (5.2.1).

We used $x_0 = (0, 0, 365.76)'$, $\varepsilon = 83.82$, $M = 6$ and $\alpha_q \approx 128 \cdot (q - 1)$ in our example. In Figure 5.9 we display the mobility function for six choices of μ. Finally, we set

$$\begin{aligned} q_1 &\equiv 0, \\ p_D &= 0 \text{ on } \Gamma_{p,d} = \{(x, y, z) \in \partial\Omega \,|\, y = 670.56\}, \\ v_N &= 1 \text{ on } \Gamma^1_{p,n} = \{(x, y, z) \in \partial\Omega \,|\, y = 0\}, \\ v_N &= 0 \text{ on } \Gamma^2_{p,n} = \partial\Omega \setminus \Gamma_{p,d} \cup \Gamma^1_{p,n} \end{aligned}$$

to complete the problem definition. A typical fine DG solution of the SPE10 example is plotted in Figure 5.8 (bottom).

We compute our LRBMS approximation on different coarse mesh configurations consisting of $1, 2 \cdot 2 \cdot 2 = 8$, $2 \cdot 4 \cdot 2 = 16$ and $4 \cdot 4 \cdot 2 = 32$ equally sized rectangular cuboids.

Figure 5.10 shows the evolution of the maximum estimated absolute error over the training set during the greedy algorithm, the tolerance of which was set to $\epsilon_{\text{tol}} = 0.1$. Again the number of snapshots needed to fulfill the error tolerance decreases with increasing numbers of coarse elements. In this example the decrease is not as big as in the thermal block example which is most certainly due to the larger spatial dimensions of the fine grid and smaller amount

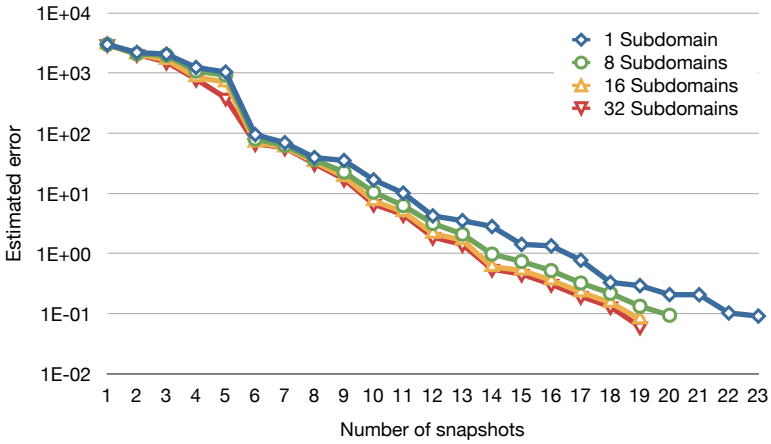


Figure 5.10 – Maximum estimated absolute error over the training set against number of snapshots needed during greedy algorithm for the different coarse meshes.

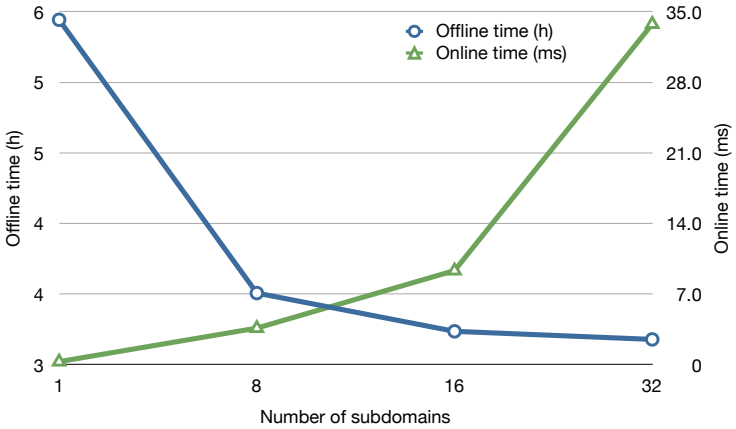


Figure 5.11 – Offline time (snapshot computation, computation of reduced quantities and training time) and online time (time for one online simulation) against number of subdomains (different units on the ordinate axes).

of coarse elements. Similar to the previous example, the estimator update times increase with the number of coarse grid elements: The mean time for one update rises from approximately 0.36 hours for 1 coarse grid element to 0.59 hours, 1 hour and 1.69 hours for 8, 16 and 32 coarse grid elements, respectively. Again, this increase can be compensated by an efficient parallelization. Apart from the error estimator, our new scheme pays out for this test case: For 16 and 32 coarse elements we need 4 snapshots less than a standard RB method needs, which saves about 37 minutes of computation time during the offline phase. This effect can be seen nicely in Figure 5.11: With increasing numbers of coarse grid elements, the offline time (excluding the time needed to update the error estimator) drops from 5.4 hours to 3.46 hours while the online time rises from 0.3 milliseconds to 33 milliseconds. Note that here the offline time comprises the runtime for snapshot computations as well as update times of the reduced operator and the training time, which is the time needed to compute the reduced simulations for all training parameters and to evaluate the error estimator for those reduced simulations.

In this case, the reduction of the offline time is not only due to the smaller amount of snapshots that need to be computed but also due to the faster projection of the high-dimensional quantities onto the respective reduced spaces.

In Figure 5.12 we see an effect that we know from the previous example already: The maximum relative true error over a given test set decreases with increasing numbers of coarse elements. Here, the mean error stays at the same level. Although for this example the decrease in the error is not as decisive, it is worth noting that the approximation qualities of our scheme are better than those of a standard RB scheme.

Finally, similar to the previous example, we compare the effect of the PCA for the different coarse meshes, this time using $\epsilon_{\text{PCA}} = 1 \cdot 10^{-4}$. In Figure 5.13 we observe the same behavior as for the thermal block example: With a rising number of subdomains, the difference between maximum and minimum numbers of basis functions on the different subdomains increases. We further see that, although the greedy algorithm needed 19 snapshots for both 16 and 32 subdomains to fulfill the error tolerance, the basis sizes after the PCA decreases.

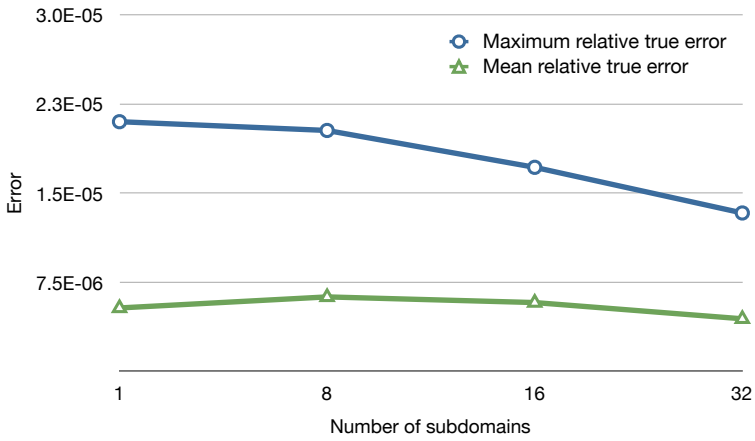


Figure 5.12 – Mean and maximum relative true error (error between reconstructed reduced and detailed simulation) in the energy norm over the test set for different coarse meshes.

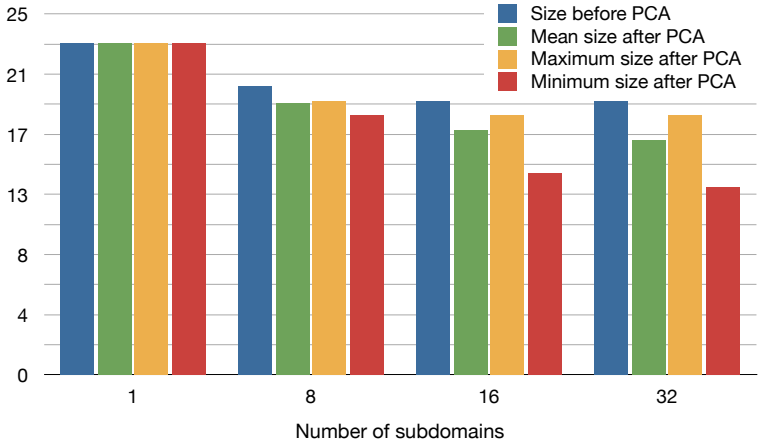


Figure 5.13 – Mean, maximal and minimal local basis sizes for the different coarse meshes before and after the PCA.

5.2.4 Summary and Outlook

In this section we applied the localized reduced basis multiscale method in the context of stationary heat diffusion with heterogeneous coefficients. We were able to demonstrate the ability of our approach to drastically reduce the offline cost in reduced basis methods while slightly increasing the online cost. We also demonstrated that our approach has at least equal approximation quality when compared to standard RB methods.

The error estimator shows pleasant performance for small coarse grid sizes but for coarse grids with more elements, a localized estimator that does not rely on global computations would be preferable.

In the next chapter we will apply the LRBMS in a two-phase flow context where it will replace the high-dimensional scheme for the unknown pressure.

5.3 Application to Two-Phase Flow

In this chapter, we will use the implicit pressure, explicit saturation (IMPES) formulation of two-phase flow introduced in Chapter 3 to prove the applicability of the LRBMS in multiphase flow regimes. As the (implicit) pressure equation usually accounts for the vast majority of the overall computation time in IMPES formulations, we will concentrate on reducing the computational effort here by using the LRBMS method as a low-dimensional surrogate for the high-dimensional pressure solver.

Application of the LRBMS method yields a potentially costly offline phase which computes localized reduced-dimensional bases for the pressure. These bases then operate as low-dimensional surrogates of the high-dimensional DG discretization during the actual two-phase flow simulation (here referred to as the online phase). The online phase is carried out using a sequential splitting scheme to decouple the high-dimensional saturation equation and reduced-dimensional pressure equation.

A key point in this section is the use of the so-called time-of-flight τ to parameterize our two-phase model. As described in Section 3.2, the time-of-flight is the time it takes for an inert particle to travel

along a pathline from the closest point on the inflow boundary and to a given point inside the domain, or alternatively, the time it takes for a particle to travel from a given interior point and to the closest point on the outflow boundary. The time-of-flight carries valuable information about the characteristics of the flow, and our idea is to use $\tau^s(x)$, defined for a given spatial point x and a given saturation snapshot s, as a means to efficiently compute a limited number of mobility profiles $\Lambda^1, \dots, \Lambda^M$, which are assumed to approximate the actual mobility $\lambda(s)$ for a given saturation s via $\lambda(s) \approx \sum \mu_i \Lambda^i$ for a given parameter $\mu \in \mathbb{R}^M$, see Section 3.3. The coupling between the saturation and pressure equation via the mobility is now replaced by a coupling via the parameter μ and we apply model order reduction techniques for parameterized problems to the pressure equation, now being parametrized by μ.

The remainder of this section is structured as follows: After introducing the details on the parameterization in the next section, we describe the usage of the time-of-flight in this context in Section 5.3.2. Section 5.3.3 is dedicated to the derivation of the LRBMS basis construction algorithm in the two-phase flow setting. Section 5.3.4 gives details on the coupling between the reduced pressure equation and the high-dimensional saturation equation and in Section 5.3.5 we give some remarks on the offline–online decomposition in the two-phase flow setting. Finally, in Section 5.3.6, we demonstrate the applicability of the method in numerical experiments.

5.3.1 Parameterization

At this point we provide details on the parameterization indicated above. As stated earlier, the coupling between the saturation and pressure equation will be realized via a parameterization of the pressure equation: We introduce wetting and non-wetting phase mobilities $\Lambda_w, \Lambda_o : \mathcal{V}_h \times \Omega \to \mathbb{R}^+$ that depend on the saturation s via

$$\Lambda_w(s, x) = \sum_{q=1}^{M} \theta_q(s) \Lambda_w^q(x), \quad \Lambda_o(s, x) = \sum_{q=1}^{M} \theta_q(s) \Lambda_o^q(x), \quad x \in \Omega, \tag{5.3.1}$$

where $\Lambda_w^q, \Lambda_o^q \in \mathcal{V}_h,\ q = 1, \dots, M$ denote saturation-independent mobility profiles that are precomputed during the offline phase and the coefficients θ_q are computed during the online phase via a least-squares fitting.

In the sequel we describe the least-squares fitting used for the computation of the coefficients θ_q, the construction of the profiles Λ_w^q, Λ_o^q is described in the next section.

For a given saturation $s \in \mathcal{V}_h$, let $\rho(s) \in \mathcal{V}_h$ be given by

$$\int_\Omega \rho(s)\varphi \,\mathrm{d}x = \int_\Omega \lambda(s)\varphi \,\mathrm{d}x, \quad \forall \varphi \in \mathcal{V}_h,$$

where λ denotes the total mobility function (3.1.2). Then choose a basis in the vector space $\mathcal{V}_h$ and let $r(s) \in \mathbb{R}^{\mathcal{N}}$ denote the coefficient vector of $\rho(s)$ in that basis. Further, let $a^q \in \mathbb{R}^{\mathcal{N}}$ denote the coefficient vector for the vector space element $\Lambda^q \in \mathcal{V}_h$, $\Lambda^q = \Lambda_w^q + \Lambda_o^q$ for $q = 1, \dots, M$.

The least-squares approximation problem then reads as follows: Find $\Theta(s) \in \mathbb{R}^M$ as solution to

$$\Theta(s) = \arg\min_{\vartheta \in \mathbb{R}^M} \|r(s) - \sum_{q=1}^{M} \vartheta_q a^q\|_2^2. \tag{5.3.2}$$

Here we used the Euclidean norm $\|\cdot\|_2 : \mathbb{R}^d \to [0, \infty)$. As the observations $\{a^q, q = 1, \dots, M\}$ were always linearly independent in our experiments, we obtain the unique solution $\Theta(s)$ to the least-squares problem via a QR-decomposition using Householder transformations. If linear independence of the observations can not be guaranteed, using a singular value decomposition provides a unique minimal-norm solution.

Finally we set $\theta_q(s) = (\Theta(s))_q$ for $q = 1, \dots, M$.

5.3.2 Profile Computation

As the coupling between the saturation and the pressure is realized via the mobility profiles, we expect these profiles to have large impact on the overall approximation quality of our scheme. In the sequel we introduce two algorithms for the construction of $\Lambda_w^q, \Lambda_o^q \in \mathcal{V}_h,\ q = 1, \dots, M$.

The first algorithm is rather straightforward: it assumes that a saturation front of height one moves trough the domain, following the isolines of the time-of-flight. Given a fixed number of desired profiles, we define a time step size and set an artificial saturation profile to one whenever the time-of-flight at the current point in space is smaller than the current time step and to zero otherwise. Applying the mobility function to the resulting saturation profiles for all time steps yields the desired mobility profiles:

Definition 5.9 (One-Zero-Profiles)**.** For a given time-of-flight $\tau_h^{s_0}$, an end time T and $M \in \mathbb{N}$, $M \geq 1$, define the mobility profiles $\Lambda_o^1, \dots, \Lambda_o^M \in \mathcal{V}_h$ and $\Lambda_w^1, \dots, \Lambda_w^M \in \mathcal{V}_h$ as

$$\Lambda_\alpha^1(x) = \lambda_\alpha(0)$$
$$\Lambda_\alpha^M(x) = \lambda_\alpha(1)$$
$$\Lambda_\alpha^q(x) = \begin{cases} \lambda_\alpha(0), & \text{if } \tau_h^{s_0}(x) > (q-1) \cdot \frac{T}{M-2}, \\ \lambda_\alpha(1), & \text{otherwise,} \end{cases} \quad \forall q \in \{2, \dots, M-1\}$$

where λ_α denotes the mobility (3.1.2) of phase α. Set $\Lambda^q = \Lambda_w^q + \Lambda_o^q$ for $q = 1, \dots, M$.

The second profile construction algorithm is a bit more involved as numerical experiments testing the first algorithm demonstrated that using one-zero profiles gives rise to large errors especially with nonlinear relative permeability functions where the saturation shows a much more complicated behavior than for linear relative permeabilities. Therefore the second algorithm uses data from a full one-dimensional two-phase flow simulation to approximate the form of the saturation in one spatial direction: We compute a full trajectory using the high-dimensional discretization in one spatial dimension with a simplified model with unit permeability and porosity. Here we assume the spatial direction chosen is the x-direction. We choose one saturation profile to be representative from this trajectory.

In the next step we again define a time step size from the desired number of profiles and front positions from the isolines of the time-of-flight just like in the previous algorithm. The important difference here is that we do not assume a drop of the saturation from one to zero at the position of the front but we stretch the one-dimensional

saturation profile such that its shock is located at the x-coordinate of the front position and then use this stretched profile to define the saturation values for all positions (x, y) with the same y-coordinate. Repeating this for all cells in the mesh and all time steps we define M saturation profiles. Applying the mobility to these saturations gives rise to the following mobility profiles. For the sake of simplicity we assume $\Omega \subset \mathbb{R}^2$, that is $d = 2$ for the following definition. An extension to $d = 3$ is straightforward.

Definition 5.10 (Extended One-Dimensional Profiles)**.** For a given time-of-flight-function $\tau_h^{s_0}$, end time T and $M \in \mathbb{N}$, $M \geq 1$, define the mobility profiles $\Lambda_o^1, \dots, \Lambda_o^M \in \mathcal{V}_h$ and $\Lambda_w^1, \dots, \Lambda_w^M \in \mathcal{V}_h$ via the following algorithm:

1. Using Algorithm 3.8 in one spatial dimension ($d = 1$) with unit permeability ($\kappa \equiv 1$) and unit porosity ($\phi \equiv 1$) compute the saturation trajectory $\{s_{h,\text{one}}^1, \dots, s_{h,\text{one}}^{N_T}\}$ and select one saturation profile $s_{h,\text{one}}^{\bar{t}}$ for $\bar{t} \in [0, T]$, $\bar{t} = i \cdot \Delta_t$ for $i \in \mathbb{N}$.

2. Let the point $x^{\max} \in \Omega$ be given as $x^{\max} = \arg\max_{x \in \Omega} \|x\|_\infty$ where $\|\cdot\|_\infty$ denotes the infinity norm.

3. Set $\Lambda_\alpha^1(x) = \lambda_\alpha(0)$ and $\Lambda_\alpha^M(x) = \lambda_\alpha(1)$ for all $x \in \Omega$.

4. For $q \in \{2, \dots, M-1\}$ set

$$\Lambda_\alpha^q((x_1, x_2)^\intercal) = \begin{cases} \lambda_\alpha\left(s_{h,\text{one}}^{\bar{t}}\left(\frac{x_1 \cdot \bar{\xi}}{\xi^*(q,x_2)}\right)\right), & \text{if } \frac{x_1 \cdot \bar{\xi}}{\xi^*(q,x_2)} \leq (x^{\max})_1 \\ \lambda_\alpha(0), & \text{else,} \end{cases}$$

where $\xi^*(q, x_2) \in \mathbb{R}$ is chosen such that

$$\tau_h^{s_0}(\xi^*(q, x_2), x_2) \leq (q-1) \cdot \frac{T}{M-2}$$

and

$$\tau_h^{s_0}(\xi^*(q, x_2) + \varepsilon, x_2) > (q-1) \cdot \frac{T}{M-2} \quad \forall \varepsilon > 0.$$

Here, $\bar{\xi}$ denotes the position of the front in the one-dimensional saturation profile at time $\bar{t}$, see Figure 5.14. This position

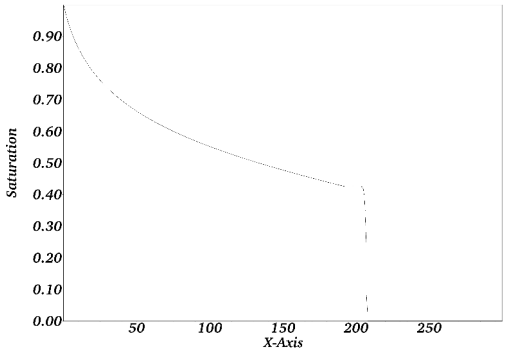


Figure 5.14 – One-dimensional saturation profile using quadratic relative permeabilities and a simplified model after 55 hours with saturation front approximately at position $x = 210$.

can either be set by hand or computed analytically from the velocity and height of the front which can both be computed from the fractional flow function (3.1.6).

Remark 5.11. *For the numerical experiments in Section 5.3.6 we set the mobility to be constant in y-direction in every mesh element when using the profile construction from Definition 5.10.*

5.3.3 Basis Construction

As reasoned above, one of the key ingredients in the LRBMS method is the novel basis construction algorithm. In the sequel we apply our localized greedy algorithm 5.8 in the two-phase flow context.

Algorithm 5.12 (Localized Greedy for Two-Phase Flow)**.** *Based on the coarse mesh $\mathcal{T}_H$, an error measure Δ_N and a profile construction algorithm $\mathfrak{P}$, we compute the global reduced basis Φ_N using the following algorithm:*

1. *Initialization*
   a) *Given the initial saturation s_0, compute the initial pressure $p_h^0 \in \mathcal{V}_h$ and velocity $u_h^0 \in RT$, see Algorithm 3.8.*

b) From u_h^0 compute the time-of-flight $\tau_h^{s_0}$.

c) From the time-of-flight and $M \in \mathbb{N}$, $M \geq 1$, compute mobility profiles $\Lambda_o^1, \ldots, \Lambda_o^M \in \mathcal{V}_h$ and $\Lambda_w^1, \ldots, \Lambda_w^M \in \mathcal{V}_h$ using the profile construction algorithm $\mathfrak{P}$.

d) Choose a desired maximum basis size $N_{\max} \in \mathbb{N}$, an approximation tolerance ϵ_{tol}, $\mathcal{P} = \left\{\mu \in [\mathbb{R}^+]^M \middle| \sum_i \mu_i \leq 1\right\}$ and a discrete subset $\mathcal{P}_{tr} \subset \mathcal{P}$, the so-called training parameter set. Furthermore let $\Phi^E = \emptyset$ for all $E \in \mathcal{T}_H$, $\Xi = \bigcup_{E \in \mathcal{T}_H} \Phi^E$, $W = \operatorname{span} \Xi$ and $\Psi = \emptyset$.

2. Basis Extension

a) For each $\mu \in \mathcal{P}_{tr}$ compute $p_H(\gamma) \in W$ as the solution to the coarse-scale (LRBMS) pressure discretization 5.2, that is

$$b_h(p_H(\gamma), w; \gamma) = l_h(w; \gamma_o, \gamma_w) \quad \forall w \in W$$

with $\gamma_\alpha = \sum_{q=1}^M \mu_q \Lambda_\alpha^q$ and $\gamma = \gamma_w + \gamma_o$.

b) Evaluate the error estimator for each parameter in the training set: $\epsilon_\mu := \Delta_N(\mu, W)$ and find the parameter with the worst approximation in the current basis: $\mu_{\max} = \arg\max_{\mu \in \mathcal{P}_{tr}} \epsilon_\mu$.

c) If $\epsilon_{\mu_{\max}} > \epsilon_{tol}$ compute $p_h(\gamma^{\max})$ as solution to the high-dimensional equation (see Definition 3.3) for the wetting and non-wetting mobilities mobilities

$$\gamma_w^{\max} = \sum (\mu_{\max})_q \, \Lambda_w^q, \quad \gamma_o^{\max} = \sum (\mu_{\max})_q \, \Lambda_o^q.$$

Otherwise go to step 3a.

d) Extend the set of snapshots: $\Psi = \Psi \cup \{p_h(\gamma^{\max})\}$.

e) For each $E \in \mathcal{T}_H$:

i. Use the Gram-Schmidt algorithm to orthonormalize the restriction $p_h(\gamma^{\max})\big|_E$ with respect to the current local basis Φ^E:

$$\widetilde{p} = ortho(p_h(\gamma^{\max})\big|_E, \Phi^E).$$

ii. Extend the local reduced basis: $\Phi^E = \Phi^E \cup \{\widetilde{p}\}$.

f) Set $\Xi = \bigcup_{E \in \mathcal{T}_H} \Phi^E$ *and* $W = \operatorname{span} \Xi$.

g) If $|\Xi| < N_{\max}$, *go back to step 2a.*

3. Data Compression (optional)

a) Set $\widetilde{\Phi}^E = \Psi\big|_E$, *that is: Define local bases* $\widetilde{\Phi}^E$ *per coarse element* $E \in \mathcal{T}_H$ *as restrictions of the snapshots* Ψ *to the coarse elements.*

b) Apply the principal component analysis using the tolerance ϵ_{PCA} *on each element:*

$$\Phi^E = PCA(\widetilde{\Phi}^E, \epsilon_{PCA}).$$

The details of this step can be found in [52], for example.

4. Finalization

a) Define the global reduced basis Φ_N *for* $N = \sum_{E \in \mathcal{T}_H} |\Phi^E|$ *as*

$$\Phi_N = \bigcup_{E \in \mathcal{T}_H} \Phi^E.$$

b) Define the global reduced basis space $\mathcal{W}_N$ *as*

$$\mathcal{W}_N = \operatorname{span} \Phi_N.$$

In summary, the basis construction performs the following: In an "initialization" step, we compute the pressure and velocity from the initial saturation data. From the velocity and the time-of-flight we compute approximate mobility profiles using one of the algorithms introduced in Section 5.3.2. This will incorporate important features of the problem, like low-permeability-lenses, for example, into the mobility profiles and therefore also into the reduced basis for the pressure. After fixing some input data like the desired basis size, we proceed to the basis extension step.

In the basis extension step, we add localized orthonormalizations of high-dimensional snapshots to initially empty local bases until

either the maximum total basis size is reached or a prescribed error tolerance is fulfilled in the current overall basis, which is the union of all local bases. Additionally, we save the original snapshots.

An optional "compression" step defines local per-coarse-element bases by restricting the global untouched snapshots from the last step to each coarse element. Next, we apply a data compression algorithm, the principal component analysis, to each local basis to reduce the basis size in regions where redundant information may be present, like in regions far from sinks and sources, for example. At this point, it is important that we do not use the orthonormalized local bases from the extension step for the local compressions. Redundancies were canceled out in those bases and therefore the data compression would not give meaningful results.

We conclude the algorithm with the "finalization" step by defining one global reduced basis as the joint of all local bases and the reduced broken space as its linear span.

The main ideas of this approach are the restriction of the basis to elements of a coarser grid and subsequent per-element data compression. By the restriction to a coarse grid we reduce the number of snapshots needed to fulfill a desired error tolerance on a prescribed training set of parameters during the offline phase. This is easy to see for the limit case in which $\mathcal{T}_H = \mathcal{T}_h$, as in this case the reduced space coincides with the high-dimensional discrete function space after a finite number of basis extensions. For $|\mathcal{T}_H| \leq |\mathcal{T}_h|$, this effect was demonstrated in Section 5.2 and can be seen again in the numerical experiments in Section 5.3.6.

The downside of this approach is that the size of the global reduced basis increases with the number of coarse elements. This is where the idea of per-element data compression comes into play and allows us to keep the total basis size N, which is the main factor in online computation complexity, in an agreeable range by reducing the local basis size in regions where little or no variation is inherent in the local bases, as may be the case in the absence of sinks or sources in the neighboring elements. The effectiveness of the PCA-step will be backed up by the experiments in Section 5.3.6.

5.3.4 Reduced Two-Phase Flow Scheme

We now have all parts together to introduce our overall reduced approximation scheme for two-phase flow in porous media using an IMPES-type coupling.

Algorithm 5.13 (LRBMS Two-Phase Flow Scheme)**.** *Compute the approximations $p_H^1, \dots, p_H^{N_T}$ for the pressure and $s_H^1, \dots, s_H^{N_T}$ for the saturation using the LRBMS method as follows.*

1. *Choose a maximum basis size $N_{\max} \in \mathbb{N}$, the number of mobility profiles $M \in \mathbb{N}$ and the number of time steps $N_T \in \mathbb{N}$.*
2. *Compute a reduced basis $\Phi = \{\varphi_1, \dots, \varphi_N\} \subset \mathcal{V}_h$ of size $N \in \mathbb{N}$ using Algorithm 5.12 for mobility profiles $\Lambda_\alpha^1, \dots, \Lambda_\alpha^M \in \mathcal{V}_h$ where $\alpha \in \{o, w\}$.*
3. *Compute the quantities $\underline{b}_q, \underline{c}, \underline{d}_q$ and $\underline{e}$ from Section 5.1.2.*
4. *Project the initial data s_0 to the fine grid:*

$$\int_\Omega s_H^0 v \,\mathrm{d}x = \int_\Omega s_0 v \,\mathrm{d}x, \quad \forall v \in \mathcal{V}_h.$$

5. *For $n = 0, \dots, N_T - 1$,*
   a) *compute the reduced-dimensional pressure $\underline{p}_H^{n+1} \in \mathbb{R}^N$ as solution to Equation (5.1.1) for $s = s_H^n$. This means that we need to compute the least-squares fit (5.3.2) to the given saturation $s_H^n \in \mathcal{V}_h$ and then solve the reduced dimensional system (5.1.1).*
   b) *given the reduced-dimensional pressure solution $\underline{p}_H^{n+1}$, reconstruct a function $p_h^r \in \mathcal{V}_h$ by setting*

$$p_h^r = \sum_{i=1}^{N} \left(\underline{p}_H^{n+1}\right)_i \varphi_i.$$

   c) *compute the velocity $u_h^{n+1} \in RT$ from the reconstructed pressure using (3.2.5) for $s = s_H^n$ and $p = p_h^r$.*
   d) *compute the fine-scale saturation $s_H^{n+1} \in \mathcal{V}_h$ using (3.2.6).*

Notice the difference to the quantities computed using the high-dimensional two-phase flow scheme: While in Algorithm 3.8 we used the linear mobilities (3.1.2) in the DG-bilinear-form and in the right hand side for the pressure, we now use the parameterized mobilities (5.3.1). We use the notation s_H^n for saturations computed with the LRBMS scheme—although the saturation itself is *not* computed in a reduced space—to point out the dependency on the coarse-scale pressure.

The novelties in this scheme are the coupling between pressure and saturation via a parameterized mobility and the replacement of the pressure equation by a reduced-dimensional substitute. While both the least-squares approximation in the basis construction step (2) and the reconstruction step (5b) still need to prove their efficiency, we expect this scheme to allow largely accelerated computations with acceptable additional error. The validity of this assumption will be investigated in the numerical experiments.

5.3.5 Offline–Online Decomposition

The only critical part in (5.1.1) is the computation of the coefficients $\theta_i(s)$, $i = 1, \dots, M$ that depends on the grid size h because of the least-squares approximation (5.3.2). Nevertheless, as the complexity of the least-squares fit is only $\mathcal{O}(1/h)$, we still expect largely accelerated computations compared to a high-dimensional pressure solve, which has a complexity of $\mathcal{O}(1/h^2)$ or even $\mathcal{O}(1/h^3)$.

5.3.6 Numerical Experiments

In this section, we demonstrate the advantages of our approach introduced in Sections 5.3.3 and 5.3.4 by means of different benchmark problems. All implementation was done using the Distributed and Unified Numerics Environment (DUNE), see [10, 9, 21, 19] and Chapter 7. For the results shown in this section we will use the true error in an energy norm as error measure:

$$\Delta_N(\mu, W) = \left[b_h(p_H(\gamma) - p_h(\gamma), p_H(\gamma) - p_h(\gamma); \bar{\lambda})\right]^{1/2}$$

where $\gamma = \sum \mu_q \Lambda^q$ and $\bar{\lambda} = \sum_{q=1}^{M} \bar{\mu}_q \Lambda^q$ for a fixed parameter $\bar{\mu}$, $p_H(\gamma)$ denotes the LRBMS-pressure-solution in the space W, see

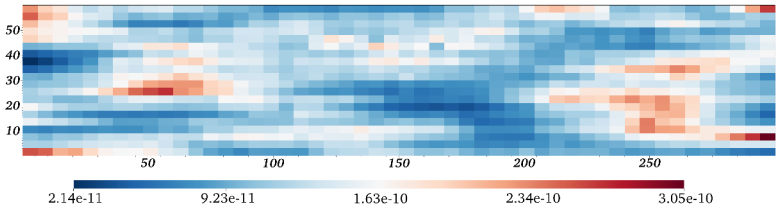


Figure 5.15 – The permeability κ [m^2] used in the 2D benchmark problem.

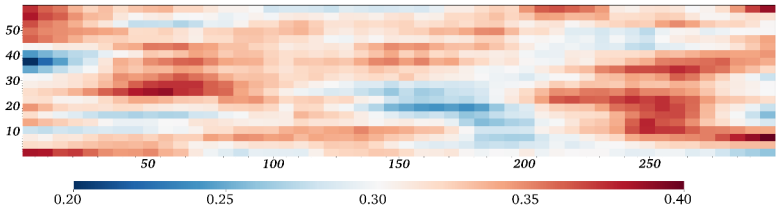


Figure 5.16 – The porosity ϕ used in the 2D benchmark problem.

Definition 5.2, and $p_h(\gamma)$ denotes the high-dimensional pressure solution, see Definition 3.3. For the exposition of well-applicable *a-posteriori* estimators for our setting we refer to [75, 76].

All tests in this section were run using the tolerance $\epsilon_{\text{PCA}} = 10^{-15}$ for the PCA.

First Test: Linear Relative Permeabilities

Our first benchmark models the replacement of the non-wetting phase by the wetting phase in $\Omega = [0, 300] \times [0, 60]$ for $T = t^{N_T} = 3 \cdot 10^5$. The fine mesh $\mathcal{T}_h$ consists of $400 \cdot 160 = 64000$ rectangles and we use $N_T = 6000$ time steps for the temporal discretization such that $\Delta_t = 50$. The permeability and porosity fields are shown in Figures 5.15 and 5.16, respectively.

The domain is initially fully saturated with the non-wetting phase ($s_0 \equiv 0$), which is then displaced by the wetting phase entering from

the left boundary, modelled via the boundary conditions (see (3.1.7))

$$\begin{aligned} s_D &= 1.0 \quad \text{in } \Gamma_{s,d} \times [0,T], & v_N &= 3 \cdot 10^{-4} \quad \text{in } \Gamma^1_{p,n} \times [0,T], \\ p_D &= 10 \quad \text{in } \Gamma_{p,d} \times [0,T], & v_N &= 0 \quad \text{in } \Gamma^2_{p,n} \times [0,T], \end{aligned}$$

on $\Gamma_{s,d} = \Gamma_{p,d} = \{0\} \times [0,60]$ and $\Gamma_{p,n} = \Gamma^1_{p,n} \cup \Gamma^2_{p,n}$, where $\Gamma^1_{p,n} = \{300\} \times [0,60]$, $\Gamma^2_{p,n} = [0,300] \times \{0\} \cup [0,300] \times \{60\}$. In this benchmark no sources are used ($q_1 \equiv q_2 \equiv 0.0$) and we neglect gravity so that $G = (0.0, 0.0)^\intercal$.

For this test the relative permeabilities in Equation (3.1.2) for simplicity are given via the linear relations

$$k_{\mathrm{rw}}(s) = s, \quad k_{\mathrm{ro}}(s) = 1 - s$$

and we choose the densities $\varrho_w = 999.749$, $\varrho_o = 890$ and viscosities $\eta_w = 0.00130581$ and $\eta_o = 0.008$. Figure 5.17 shows the saturation s_h computed with the full scheme (Algorithm 3.8) after approximately 3.5, 10, 15, and 48 hours.

From the time-of-flight, which is depicted in Figure 5.18, we compute the reduced basis Φ via Algorithm 5.12 for coarse meshes with sizes $|\mathcal{T}_H| = 1, 4, 8, 16, 32$ and using One-Zero-Profiles (Definition 5.9) with $M = 8$. The resulting wetting mobility profiles are depicted in Figure 5.19. We use the tolerance $\epsilon_{\mathrm{tol}} = 10^{-4}$, a training set $\mathcal{P}_{\mathrm{tr}}$ consisting of 300 randomly distributed parameters μ in $[0.0001, 1]^8$ with $\sum(\mu)_i = 1$, and the maximum size $N_{\max} = \infty$ for Algorithm 5.12. The resulting basis sizes can be seen in Table 5.1.

We observe that with increasing size of the coarse mesh (first column), the number of snapshots computed in Step 2c of Algorithm 5.12 (fourth column) decreases significantly from 134 to 69. At the same time, the overall basis size (the sum of all local basis sizes, second column) increases. Notice that the basis size is not necessarily equal the product between the number of snapshots and the coarse grid size. This is because we orthonormalize each new snapshot with respect to the existing basis in each extension during the basis generation, reject local extensions with norms below a certain threshold to avoid linear dependencies in the resulting basis and hence reduce the number of local basis functions.

In Table 5.1 we also see the impact of the local data compression using PCA (Step 3b in Algorithm 5.12): With increasing coarse

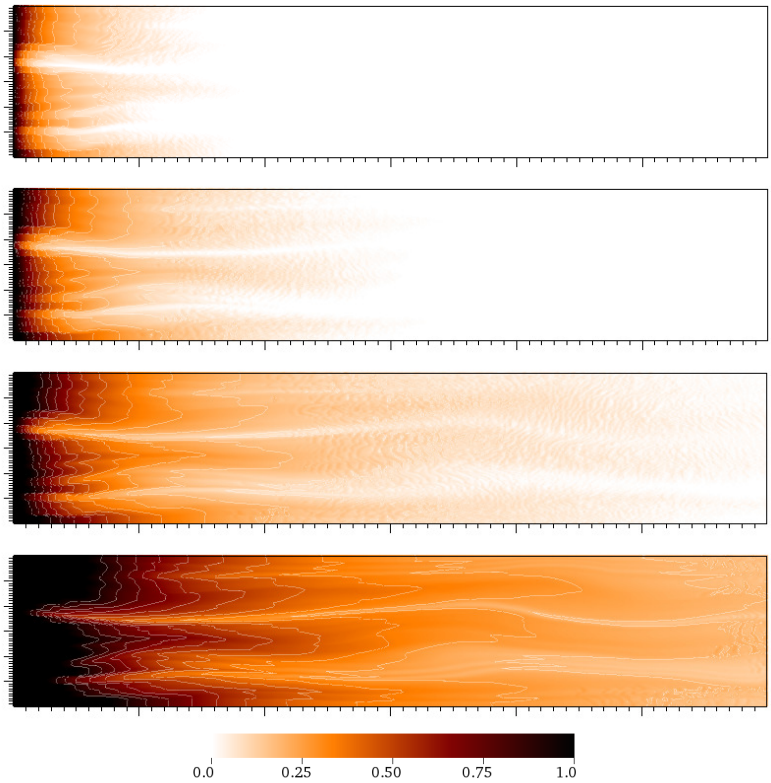


Figure 5.17 – Wetting phase saturation s_h computed using Algorithm 3.8 after 3.5, 10, 15 and 48 hours including contour lines.

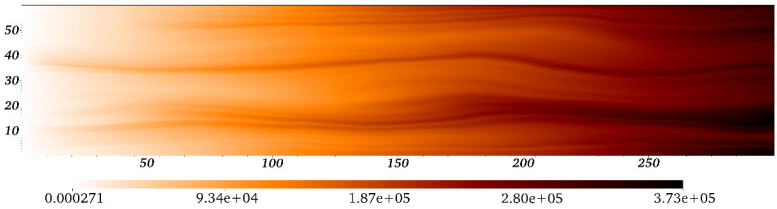


Figure 5.18 – The time-of-flight τ_h^s for the first test case for $s = 0$.

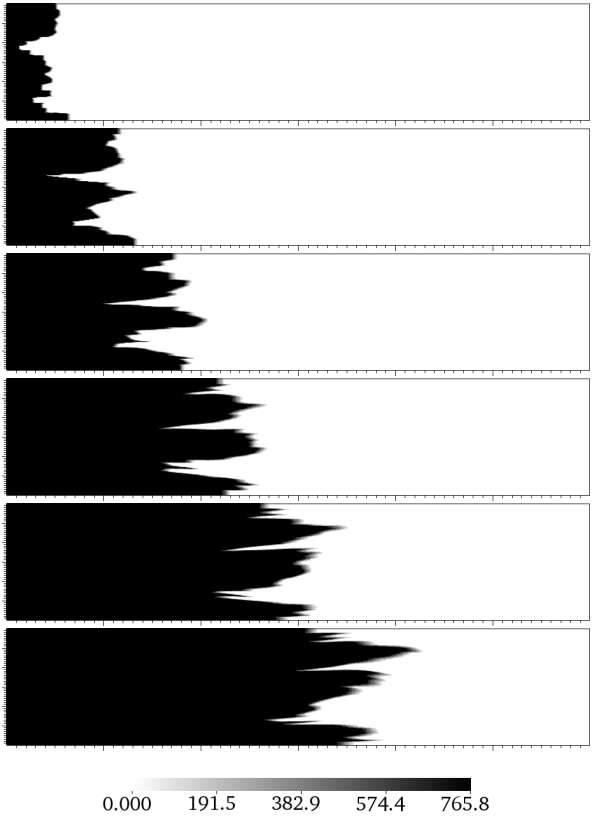


Figure 5.19 – Wetting mobility profiles computed using Algorithm 5.12 for the time-of-flight depicted in Figure 5.18. Not shown are the profiles for $q = 1$ and $q = M$ which have constant values $\Lambda_w^1 \equiv 0$ and $\Lambda_w^M \equiv 765.808$, respectively.

Coarse Grid Size	Basis Size w/o PCA	Basis Size w/ PCA	Number of Snapshots
1x1	134		134
4x1	353	155	105
8x1	565	264	84
8x2	1030	443	78
16x2	1771	751	69

Table 5.1 – Basis sizes $|\Phi|$ resulting from Algorithm 5.12 for the 2D benchmark problem and a fine mesh with 64000 elements: Size of the coarse mesh, sum of all local basis sizes before and after application of the PCA, number of snapshots computed during the basis generation.

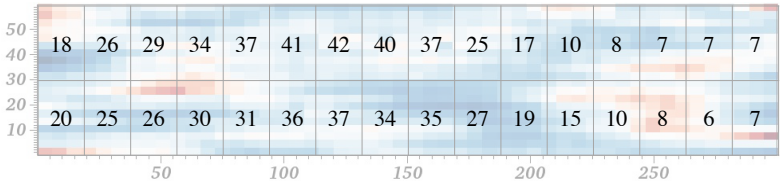


Figure 5.20 – Foreground: local basis sizes after the basis generation performed for Table 5.1, last line, including the PCA; background: permeability field.

mesh size, the PCA is able to reduce the local basis sizes significantly on the different coarse elements.

This effect can be seen again in Figure 5.20 where we plot the local basis sizes after the PCA step for the test run presented in Table 5.1, last line. We observe that the local basis size varies strongly from left to right, due to the fact that the peak saturation does not reach the right half of the computational domain before the end time T. Also, the basis size differs from bottom to top, especially in regions where the permeability, which is plotted in the background of Figure 5.20, shows strong variation from bottom to top.

In Table 5.2 we see the resulting discrepancies between the saturation s_h^n computed by the SWIP-DG method on the fine mesh

(Algorithm 3.8) and the saturation s_H^n computed with the reduced scheme 5.13. In Table 5.3 we present the respective discrepancies between the SWIP-DG pressure and the reduced pressure during the two-phase flow simulation. We present both the relative L^2 and H^1 discrepancies for the saturation

$$\Delta_{L^2}^{n,s} = \frac{\|s_h^n - s_H^n\|_{L^2(\mathcal{T}_h)}}{\|s_h^n\|_{L^2(\mathcal{T}_h)}}, \quad \Delta_{H^1}^{n,s} = \frac{\|s_h^n - s_H^n\|_{H^1(\mathcal{T}_h)}}{\|s_h^n\|_{H^1(\mathcal{T}_h)}}, \tag{5.3.3}$$

the discrepancy in the energy norm

$$\Delta_{\mathcal{E}}^{n,s} = \frac{\left[b_h(s_H^n - s_h^n, s_H^n - s_h^n; \bar{\lambda})\right]^{1/2}}{\left[b_h(s_h^n, s_h^n; \bar{\lambda})\right]^{1/2}}, \quad \bar{\lambda} = \sum_{q=1}^{M} \bar{\mu}_q \Lambda^q \tag{5.3.4}$$

as well as the respective quantities for the pressure:

$$\Delta_{L^2}^{n,p} = \frac{\|p_h^n - p_H^n\|_{L^2(\mathcal{T}_h)}}{\|p_h^n\|_{L^2(\mathcal{T}_h)}}, \quad \Delta_{H^1}^{n,p} = \frac{\|p_h^n - p_H^n\|_{H^1(\mathcal{T}_h)}}{\|p_h^n\|_{H^1(\mathcal{T}_h)}}, \tag{5.3.5}$$

$$\Delta_{\mathcal{E}}^{n,p} = \frac{\left[b_h(p_H^n - p_h^n, p_H^n - p_h^n; \bar{\lambda})\right]^{1/2}}{\left[b_h(p_h^n, p_h^n; \bar{\lambda})\right]^{1/2}}, \quad \bar{\lambda} = \sum_{q=1}^{M} \bar{\mu}_q \Lambda^q. \tag{5.3.6}$$

Here we used $\bar{\mu} = (0.125, \ldots, 0.125)$, $\sum_i \bar{\mu}_i = 1$ and $\|\cdot\|_{L^2(\mathcal{T}_h)}$ and $\|\cdot\|_{H^1(\mathcal{T}_h)}$ denote the element-wise L^2- and H^1-norm, respectively.

For different coarse grids, the second column in each table displays the number of snapshots computed during the extension step 2c in Algorithm 5.12 as a measure for the amount of work needed in that step. Then, in the sections for the L^2, H^1 and energy relative discrepancies, we present mean discrepancies over all time steps and the discrepancy at the last time step t^{N_T} for basis generations without and with usage of the PCA.

Over different coarse mesh configurations, we consistently observe a mean L^2-discrepancy for the saturation of approximately 5.6% (standard deviation: 1.6%), both with and without the PCA, which is reduced to 4.3% at end time. The respective H^1-discrepancies are slightly bigger with a mean of 7.9% (standard deviation: 2.9%) and 5.5% at end time. The discrepancies in the energy norm are still slightly bigger with a mean of 13.2% (standard deviation 5.7%)

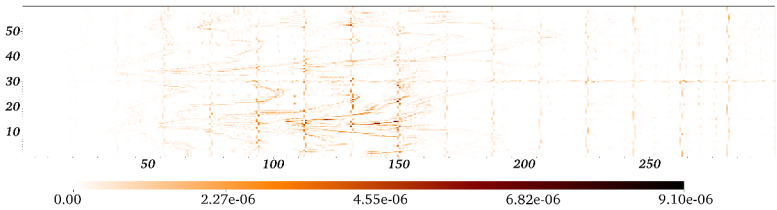


Figure 5.21 – Relative mass loss ζ^n for the final time step $t^n = 3 \cdot 10^5$.

which largely reduces towards the end time to reach a final value of about 7.9%. The mean L^2-discrepancy for the pressure is approximately 1.4% (standard deviation: 0.3%) for all coarse grid configurations, and reduces to 1.3% at end time, the respective H^1-discrepancies are in the same ranges. The discrepancies in the energy norm are slightly bigger with a mean of 5.1% (standard deviation 0.87%) and reduce to a final value of 4.85%. In conclusion, we can say that the data compression step using the PCA increases neither the error for the pressure nor the error for the saturation noticeably.

While we can ensure mass conservation on the coarse grid by adding per-coarse-cell unit basis functions to the reduced basis, the method is not mass-conservative on the fine grid. In Figure 5.21 we plot the relative mass loss

$$\zeta^n : \mathcal{T}_h \to \mathbb{R}^+, \quad \zeta^n(e) = \frac{1}{\max_{x \in \partial e} \|u_h^n(x)\|_{l^2}} \left| \int_{\partial e} u_h^n \cdot \mathrm{n} \, \mathrm{d}S \right|$$

as a measure of the lack of mass conservation at end time $t^n = 3 \cdot 10^5$ for the velocity computed with the LRBMS scheme on a 16×2 coarse grid. We see that the velocity lacks mass conservation mainly on coarse cell intersections and in regions with large gradients in the mobility profiles Λ^i. Overall, the relative lack of mass conservation is well below $2 \cdot 10^{-5}$. Although we therefore found this to be only a minor problem in the tests—the saturation usually did not grow above a value of 1.05 and hardly ever fell below zero—the question how to ensure mass-conservation on the fine grid seems like an interesting and important issue.

As the discrepancies are increased neither by adding more coarse cells, nor by application of the PCA, we find the localized reduced basis multiscale method to be well applicable in this context: Introduction of more coarse cells reduces the number of costly snapshots to be computed during the basis construction, while application of the PCA yields smaller, more compact bases, leading to faster computations during the two-phase flow simulation.

Test runs with a different tolerance $\epsilon_{\text{tol}} = 10^{-5}$ for the greedy basis construction (Algorithm 5.12) exhibit approximately the same discrepancies in both the L^2 and H^1 norms. Further, computations with higher numbers of mobility profiles ($M = 10, 12, 20$) give roughly the same discrepancies. This gives rise to the assumption that the error is dominated by different phenomena, which we consider to be twofold: First, the assumption that the time-of-flight is invariant for the whole simulation is not valid in some regions. Therefore, the position of the saturation front and its impact on the mobility cannot be represented correctly. Second, the profile of the mobility along streamlines is not approximated well enough due to the simple one-zero mobility profiles.

To support the first statement, we present in Figure 5.22 the absolute value of the difference between the SWIP-DG approximation $s_h^{N_T}$ and our approximation of the saturation $s_H^{N_T}$ at end time t^{N_T}. We see that huge discrepancies arise in three distinct positions: around the point $\mathbf{x} = (30, 35)$, the point $\mathbf{x} = (40, 10)$, and the point $\mathbf{x} = (80, 1)$. These positions are located directly downwind along the streamlines from points where the time-of-flight changes significantly during the test run, see Figure 5.23. The error produced in those regions is then transported through the domain along the streamlines, hence the error distribution to be seen in Figure 5.22 is established.

We can establish the second statement using M high-dimensional saturation approximations $s_h^{n_1}, \dots, s_h^{n_M}$ at times $t^1, \dots, t^M$ to form the profiles in Step 1c of Algorithm 5.12:

$$\Lambda_w^q = \lambda_w(s_h^q), \quad \Lambda_o^q = \lambda_o(s_h^q).$$

In doing so, we ensure that the shape of the mobility profiles along streamlines is correct and the error of the LRBMS two-phase flow

$\lvert\mathcal{T}_H\rvert$	$\lvert\Psi\rvert$	$\Delta_{L^2}^{\varnothing,s}$ w/o PCA (%)	$\Delta_{L^2}^{t^{N_T},s}$ w/o PCA (%)	$\Delta_{L^2}^{\varnothing,s}$ w/ PCA (%)	$\Delta_{L^2}^{t^{N_T},s}$ w/ PCA (%)
1x1	134	5.59	4.26		
4x1	105	5.59	4.25	5.61	4.26
8x1	84	5.58	4.25	5.59	4.26
8x2	78	5.58	4.25	5.6	4.26
16x2	69	5.59	4.25	5.59	4.26

(a) L^2-errors

$\lvert\mathcal{T}_H\rvert$	$\lvert\Psi\rvert$	$\Delta_{H^1}^{\varnothing,s}$ w/o PCA (%)	$\Delta_{H^1}^{t^{N_T},s}$ w/o PCA (%)	$\Delta_{H^1}^{\varnothing,s}$ w/ PCA (%)	$\Delta_{H^1}^{t^{N_T},s}$ w/ PCA (%)
1x1	134	7.88	5.49		
4x1	105	7.88	5.45	7.92	5.53
8x1	84	7.89	5.53	7.89	5.47
8x2	78	7.9	5.51	7.92	5.55
16x2	69	7.89	5.5	7.91	5.56

(b) H^1-errors

$\lvert\mathcal{T}_H\rvert$	$\lvert\Psi\rvert$	$\Delta_{\mathcal{E}}^{\varnothing,s}$ w/o PCA (%)	$\Delta_{\mathcal{E}}^{t^{N_T},s}$ w/o PCA (%)	$\Delta_{\mathcal{E}}^{\varnothing,s}$ w/ PCA (%)	$\Delta_{\mathcal{E}}^{t^{N_T},s}$ w/ PCA (%)
1x1	134	13.14	7.3		
4x1	105	13.07	7.1	13.28	7.92
8x1	84	13.28	7.93	13	7.24
8x2	78	13.26	7.82	13.28	7.96
16x2	69	13.21	7.69	13.31	8.01

(c) Energy-norm-errors

Table 5.2 – Relative L^2 (a) and H^1 (b) discrepancies and discrepancies (5.3.4) in the energy norm (c) between the saturation computed with Algorithm 5.13 and the saturation computed with the fine-scale scheme from Algorithm 3.8 for different coarse mesh configurations: Number of snapshots $|\Psi|$ needed during the basis construction algorithm 5.12, mean relative discrepancy and relative discrepancy at end time t^{N_T} without usage of the PCA, respective quantities after usage of the PCA.

$\lvert\mathcal{T}_H\rvert$	$\lvert\Psi\rvert$	$\Delta_{L^2}^{\varnothing,p}$ w/o PCA (%)	$\Delta_{L^2}^{t^{N_T},p}$ w/o PCA (%)	$\Delta_{L^2}^{\varnothing,p}$ w/ PCA (%)	$\Delta_{L^2}^{t^{N_T},p}$ w/ PCA (%)
1x1	134	1.36	1.33		
4x1	105	1.36	1.33	1.36	1.34
8x1	84	1.35	1.32	1.36	1.33
8x2	78	1.36	1.32	1.36	1.33
16x2	69	1.36	1.32	1.36	1.33

(a) L^2-errors

$\lvert\mathcal{T}_H\rvert$	$\lvert\Psi\rvert$	$\Delta_{H^1}^{\varnothing,p}$ w/o PCA (%)	$\Delta_{H^1}^{t^{N_T},p}$ w/o PCA (%)	$\Delta_{H^1}^{\varnothing,p}$ w/ PCA (%)	$\Delta_{H^1}^{t^{N_T},p}$ w/ PCA (%)
1x1	134	7.88	5.49		
4x1	105	7.88	5.45	7.92	5.53
8x1	84	7.89	5.53	7.89	5.47
8x2	78	7.9	5.51	7.92	5.55
16x2	69	7.89	5.5	7.91	5.56

(b) H^1-errors

$\lvert\mathcal{T}_H\rvert$	$\lvert\Psi\rvert$	$\Delta_{\mathcal{E}}^{\varnothing,p}$ w/o PCA (%)	$\Delta_{\mathcal{E}}^{t^{N_T},p}$ w/o PCA (%)	$\Delta_{\mathcal{E}}^{\varnothing,p}$ w/ PCA (%)	$\Delta_{\mathcal{E}}^{t^{N_T},p}$ w/ PCA (%)
1x1	134	5.12	4.85		
4x1	105	5.13	4.85	5.12	4.85
8x1	84	5.13	4.85	5.12	4.85
8x2	78	5.12	4.85	5.12	4.85
16x2	69	5.12	4.85	5.12	4.85

(c) Energy-norm-errors

Table 5.3 – Relative L^2 (a) and H^1 (b) errors and errors (5.3.6) in the energy norm (c) between the pressure computed with Algorithm 5.13 and the pressure computed with the fine-scale scheme from Algorithm 3.8 for different coarse mesh configurations: Number of snapshots $|\Psi|$ needed during the basis construction algorithm 5.12, mean relative discrepancy and relative discrepancy at end time t^{N_T} without usage of the PCA, respective quantities after usage of the PCA.

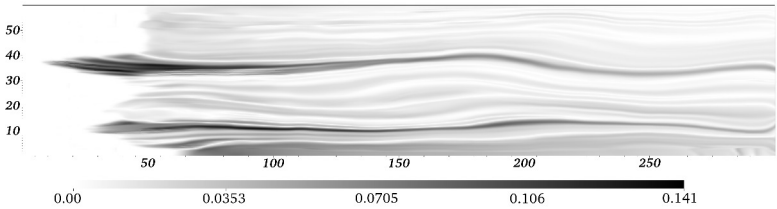


Figure 5.22 – Absolute value of the difference in the saturation $s_h^{N_T}$ computed with the full high-dimensional scheme and the saturation $s_H^{N_T}$ computed with the LRBMS scheme at end time T.

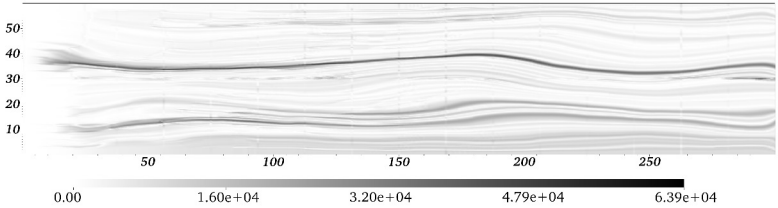


Figure 5.23 – Absolute difference between the time-of-flight for the initial saturation s_0 and the time-of-flight for the saturation computed with the LRBMS scheme for a coarse grid with 16×2 cells at time $t = 10000s$.

scheme should decrease drastically. Indeed, for $M = 8$, this procedure decreases the mean relative L^2-discrepancy to 2.3% with a standard deviation of 2.0%. Even more: The L^2-discrepancy at $t = T$ is reduced by a factor of ten to 0.45%. Obviously, using multiple high-dimensional saturation profiles is not possible in general as the model order reduction would become superfluous, but two other remedies could be implemented: One could be to recompute the mobility profiles and the reduced basis after a certain number of time steps t^n using the time-of-flight for s_h^n. Another approach is to use more realistic mobility profiles by applying the mobility to full one-dimensional saturation trajectories as is realized by the second mobility profile computation algorithm (Definition 5.10).

In Table 5.4 we present runtimes for the basis construction part of

the LRBMS two-phase flow scheme. Again, we give the coarse grid configuration (column one) and the number of snapshots needed (column two). The third column then shows the time needed for the basis construction (Algorithm 5.12). The timings presented here include the time for the computation of the mobility profiles (about 13 seconds), the so-called "training"-step (computation of all reduced solutions, evaluation of the error measure, selection of the parameter for basis extension), and the extension-step including the computation of a high-dimensional snapshot and application of the Gram-Schmidt procedure. It does not contain the application of the PCA which is consistently below one second for all coarse grid configurations and hence can be considered negligible. The time needed to compute the reduced basis Φ is 50 minutes on one coarse cell (which corresponds to a standard reduced basis method), goes up to 1 hour 7 minutes for a coarse grid of size 4×1 and is then reduced to 45 minutes for all other coarse grids. The increase in total runtime from line one to line two can be explained by the relatively high number of iterations that is still needed to reach the error tolerance with an increased per-step cost due to the larger reduced bases. This effect would vanish if the snapshot computation (column four) was more expensive, for larger fine grids, for example. As mentioned earlier, we use the true error as error measure. The time for error estimation in column five includes the time needed to reconstruct a high-dimensional snapshot from each reduced solution and compute the difference in the energy norm. The time for computation of the high-dimensional snapshots itself is not included.

In conclusion we can say that for the test case at hand, introduction of more coarse cells yields slight reductions in terms of runtime. As the computation of the snapshots shows a speedup by a factor of two for the computation on 32 coarse cells compared to the computation on one coarse cell, we can expect the runtime-gain to increase drastically as the size of the fine mesh—and hence the time for the computation of one high-dimensional snapshot—is increased.

Finally, in Table 5.5 we present runtimes for the two-phase flow simulation (Step 5 in Algorithm 5.13) for uncompressed bases (that is: bases that were computed without usage of the PCA) and compressed bases on different coarse grids and, for comparison, the same runtimes for a full high-dimensional computation. We see that for

Coarse Grid Size	Num. of Snapshots	Basisgen. Total	Snapshot Computation	Error Estimation
1x1	134	50m 6s	15m 39s	19m 52s
4x1	105	67m 19s	12m 19s	12m 15s
8x1	84	45m 34s	9m 52s	9m 3s
8x2	78	46m 36s	9m 9s	9m 37s
16x2	69	45m 48s	8m 6s	10m 25s

Table 5.4 – Runtimes for the basis construction (see Algorithm 5.12): Coarse grid size, number of snapshots computed, total runtime, total time for snapshot computation, total time for error estimation.

the reduced simulations, more than 50% of the overall runtime of about 2 hours 40 minutes is spend in the application of the ODE solver and slope limiter for the saturation equation. About one hour is spend for the elliptic equation with about 45 minutes for the reconstruction of the flux (see Definition 3.6). For uncompressed bases, computing all pressure solutions takes five to 15 minutes, depending on the coarse grid and respective basis size, for compressed bases those times drop to two to five minutes.

The time to solution for one reduced pressure computation therefore ranges from 20 milliseconds on the 1×1, 4×1 and 8×1 coarse grids to 50 milliseconds on the 8×2 and 16×2 coarse grids using the PCA. Comparing these runtimes to the time needed for a high-dimensional simulation we see the advantage of our method: The high-dimensional test run takes approximately 16 hours with nearly 90% of the time spent in the treatment of the elliptic equation: more than two hours are spent assembling the pressure system, solving it takes approximately 11 hours in total (approximately seven seconds per solve). The speed-up for the solution of the pressure equation is approximately a factor 140. As nothing was done to speed up the transport solve, the overall speed-up for the two-phase flow simulation (high-dimensional vs. basis construction and reduced simulation) is five.

Coarse Grid Size	Total	Elliptic Part	Pressure Solver	Velocity Recons.	Saturation Part
Without PCA					
1x1	2h 37m	1h 0m	5m 28s	46m	1h 37m
4x1	2h 41m	1h 1m	5m 5s	47m	1h 40m
8x1	2h 38m	1h 0m	5m 25s	47m	1h 37m
8x2	2h 40m	1h 4m	10m 40s	45m	1h 36m
16x2	2h 45m	1h 9m	15m 3s	45m	1h 36m
With PCA					
4x1	2h 34m	0h 56m	1m 59s	45m	1h 38m
8x1	2h 31m	0h 55m	2m 0s	44m	1h 36m
8x2	2h 41m	0h 55m	3m 0s	44m	1h 46m
16x2	2h 37m	0h 57m	4m 42s	44m	1h 39m
High-Dimensional					
	15h 58m	14h 23m	11h 13m	43m	1h 35m

Table 5.5 – Runtimes for Step 5 of Algorithm 5.13 for uncompressed and compressed bases: Coarse grid size, total runtime, time spent for treatment of the elliptic equation, thereof time spent in pressure solve and velocity reconstruction and time spent in saturation part. Last line: same numbers for a full high-dimensional simulation.

Second Test: Quadratic Relative Permeabilities, Unit Total Permeability

For the second test case we use quadratic relative permeability functions

$$k_{\mathrm{rw}}(s) = s^2, \quad k_{\mathrm{ro}}(s) = (1-s)^2$$

in Equation (3.1.2) and the computational domain is given as $\Omega = (0,1) \times (0,1)$. Again we simulate the replacement of the non-wetting phase by the wetting phase. We use $s_0 \equiv 0$, $T = t^{N_T} = 0.008$, $N_T = 8000$, unit permeability and porosity ($\kappa \equiv 1$, $\phi \equiv 1$), $\eta_w = 0.0004$,

$\eta_o = 0.002$ and a fine mesh $\mathcal{T}_h$ consisting of $50 \cdot 50 = 2500$ rectangles. We define the boundary conditions

$$\begin{aligned} s_D &= 1.0 \quad \text{in } \Gamma_{s,d} \times [0,T], & v_N &= 3 \cdot 10^{-4} \quad \text{in } \Gamma^1_{p,n} \times [0,T], \\ p_D &= 10 \quad \text{in } \Gamma_{p,d} \times [0,T], & v_N &= 0 \quad \text{in } \Gamma^2_{p,n} \times [0,T], \end{aligned}$$

on $\Gamma_{s,d} = \Gamma_{p,d} = \{0\} \times [0,1]$ and $\Gamma_{p,n} = \Gamma^1_{p,n} \cup \Gamma^2_{p,n}$, where $\Gamma^1_{p,n} = \{1\} \times [0,1]$, $\Gamma^2_{p,n} = [0,1] \times \{0\} \cup [0,1] \times \{1\}$. Finally, no sources are used ($q_1 \equiv q_2 \equiv 0.0$) and we neglect gravity ($G = (0.0, 0.0)^\intercal$).

From the time-of-flight we construct reduced bases for coarse meshes with one and $4 \cdot 1 = 4$ coarse cells using Algorithm 5.12. Here we use extended one-dimensional profiles for $M = 16$ to compute the non-wetting and wetting mobility profiles. The resulting profiles are depicted in Figure 5.24. As our problem is constant in the vertical direction we only plot the mobilities (and all other quantities for this test case) over the horizontal axis.

Further we used $\epsilon_{\text{tol}} = 10^{-7}$ and $N_{\max} = \infty$ for Algorithm 5.12. The training set $\mathcal{P}_{\text{tr}}$ consisted of 1000 randomly distributed parameters μ in $[0.0001, 1]^{16}$ with $\sum(\mu)_i = 1$. The basis construction yielded a basis of size $|\Phi| = 45$ for $|\mathcal{T}_H| = 1$ (number of snapshots: 45) and of $|\Phi| = 68$ for $|\mathcal{T}_H| = 4$ (number of snapshots: 18).

In this benchmark we are only interested in the capability of our two-phase flow scheme of capturing the shape of the saturation during the simulation and whether we are able to use quadratic relative permeabilities when using the extended one-dimensional profiles for the mobilities. Towards this end we plot the saturations resulting from the high-dimensional two-phase flow scheme and the reduced scheme for the two reduced bases on $|\mathcal{T}_H| = 1$ and $|\mathcal{T}_H| = 4$ in Figure 5.25.

We see that we get a very good matching of the whole saturation profile on both coarse grid configurations. Especially the saturation front is captured very accurately. This can also be seen in the error values: For the coarse grid with $|\mathcal{T}_H| = 1$ the mean relative L^2 error is 0.41% (standard deviation (sd) 0.12%) and the relative L^2 error at end time is 0.45% for the saturation. The respective values for the energy norm (see Equation 5.3.4, $\bar{\mu}_1 = \ldots = \bar{\mu}_{16} = 0.0625$) are 1.32% (sd 0.36%) and 2.34%. The errors for the coarse grid configuration with $|\mathcal{T}_H| = 4$ are roughly the same. For the pressure,

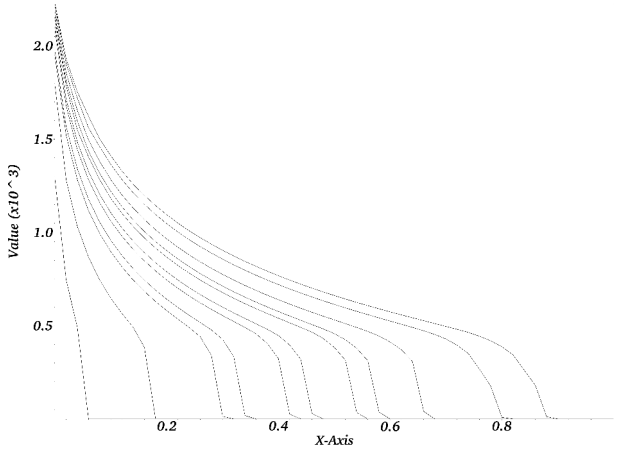


Figure 5.24 – Wetting mobility profiles for $q = 2, \ldots, 15$ (left to right) computed with Algorithm 5.12 (plot along the horizontal axis). Not shown are the profiles for $q = 1$ and $q = 16$ which have constant values $\Lambda_w^1 \equiv 0$ and $\Lambda_w^{16} \equiv 2500$.

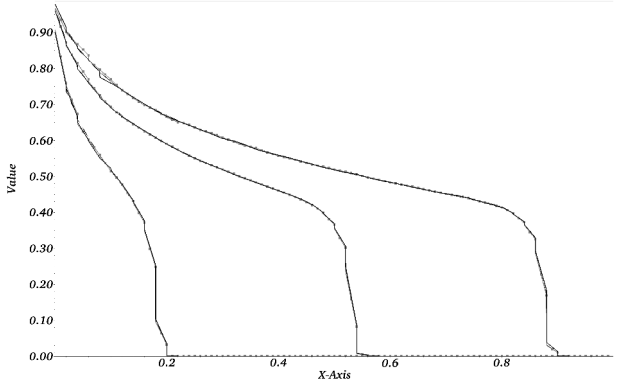


Figure 5.25 – Saturation computed with high-dimensional scheme (dashed), reduced scheme with $|\mathcal{T}_H| = 1$ (gray with points) and reduced scheme with $|\mathcal{T}_H| = 4$ (black, solid) at timesteps 500, 1500 and 2500 (left to right).

all above-mentioned error quantities are well below 0.5%.

In summary we can say that the LRBMS two-phase flow scheme is well-applicable for this benchmark which is admittedly rather simple regarding the permeability and porosity but more complex regarding the relative permeability function when compared to the last test case. We conclude that the extended one-dimensional profiles show better performance for quadratic relative permeability functions than the one-zero profiles which we found to be unusable in conjunction with non-linear relative permeabilities.

Third Test: Quadratic Relative Permeabilities, Non-Trivial Total Permeability

In our third test case, the relative permeabilities in Equation (3.1.2) are again given via the quadratic relations

$$k_{\mathrm{rw}}(s) = s^2, \quad k_{\mathrm{ro}}(s) = (1-s)^2$$

and we define the densities $\varrho_w = 999.749$, $\varrho_o = 890$ and viscosities $\eta_w = 0.0004$ and $\eta_o = 0.002$. We use the same domain as in the first test case: $\Omega = [0, 300] \times [0, 60]$, the same boundary and initial data and the same permeability and porosity fields (see Figures 5.15 and 5.16, respectively). Gravity is again neglected. The computational mesh is given by $200 \cdot 80 = 16000$ rectangles. The end time is again given as $T = t^{N_T} = 3 \cdot 10^5$ and we use $N_T = 6000$ time steps. Figure 5.26 shows the saturation s_h computed with the full scheme (Algorithm 3.8) after approximately 5.5, 14, 25 and 44 hours.

From the time-of-flight we compute the reduced basis Φ via Algorithm 5.12 for coarse meshes with sizes $|\mathcal{T}_H| = 1, 4, 8, 16, 32$ and using the extended one-dimensional profiles (Definition 5.10) with $M = 16$. The resulting wetting mobility profiles are depicted in Figure 5.27.

We use the tolerance $\epsilon_{\mathrm{tol}} = 10^{-6}$, a training set $\mathcal{P}_{\mathrm{tr}}$ consisting of 1000 randomly distributed parameters μ in $[0.0001, 1]^{16}$ with $\sum(\mu)_i = 1$, and the maximum size $N_{\max} = \infty$ for Algorithm 5.12. The resulting basis sizes can be seen in Table 5.6.

Again we observe that with increasing size of the coarse grid, the number of snapshots necessary to fulfill the error tolerance in our greedy basis construction algorithm decreases. Clearly, at the same

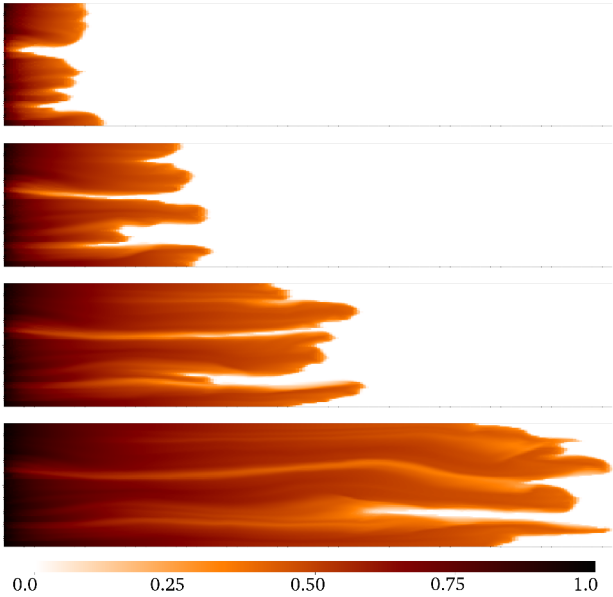


Figure 5.26 – Wetting phase saturation s_h computed using Algorithm 3.8 after approximately 5.5, 14, 25 and 44 hours (top to bottom) for the third test case.

Coarse Grid Size	Basis Size w/o PCA	Basis Size w/ PCA	Number of Snapshots
1x1	138		138
4x1	327	186	102
8x1	539	315	86
8x2	936	556	75
16x2	1598	935	65

Table 5.6 – Basis sizes resulting from Algorithm 5.12 for the third test case: Size of the coarse mesh, sum of all local basis sizes before and after application of the PCA, number of snapshots computed during the basis generation.

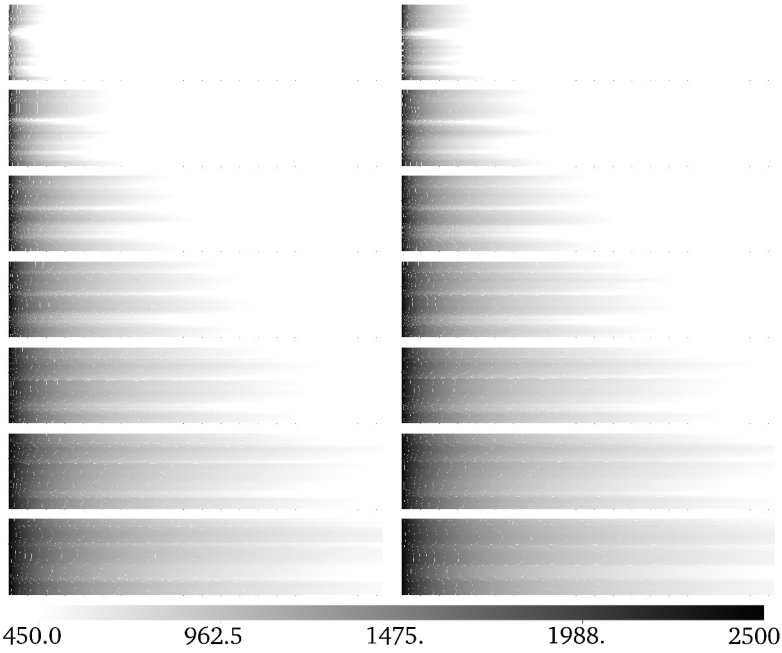


Figure 5.27 – Wetting mobility profiles computed with Algorithm 5.12 for the second test case. Not shown are the profiles for $q = 1$ and $q = M$ which have constant values $\Lambda_w^1 \equiv 0$ and $\Lambda_w^M \equiv 2500$, respectively.

time the size of the global reduced basis increases as we now introduce roughly $|\mathcal{T}_H|$ new basis functions in each extension step. At the same time we note that the compression step (Step 3 of Algorithm 5.12) reduces the basis sizes drastically: For the two rightmost cells in the coarse grid configuration with 16×2 cells, the *local* basis size is reduced from 65 to six, for example.

In Table 5.7 we present the error measures $\Delta_{L^2}^{n,s}$, $\Delta_{H^1}^{n,s}$ and $\Delta_{\mathcal{E}}^{n,s}$ for the saturation and the respective quantities for the pressure in Table 5.8 (see Equations (5.3.3)–(5.3.6)).

We observe that the mean error increased largely when compared to the first test case, in most cases it nearly doubled. We believe the reason for this to be the more complex behavior of the saturation when using quadratic relative permeabilities. As seen in Figure 5.26, the saturation shows viscous fingering which is much harder to capture using our scheme than the more linear behavior of the saturation in the first test case.

Furthermore, the extended one-dimensional profiles now capture the shape of the mobility in the horizontal direction more accurately but flows in vertical direction are not captured which leads to wrong fluxes in that direction during the reduced two-phase flow simulation. Nevertheless we consider the ability to use the more complex quadratic relative permeability function to be an improvement over the first test case and over the one-zero profiles.

5.3.7 Summary and Outlook

We applied the localized reduced basis multiscale method to two-phase flows in porous media. We introduced a new formulation of two-phase flow using a parameterized mobility function that realizes the coupling between the equations for the unknown pressure and saturation. The mobility uses a certain number of precomputed profiles to match a given mobility via a least-squares fitting during the main simulation and the parameters arising from the least-squares fitting are used as parameters to the pressure equation which is discretized using the LRBMS method.

We were able to demonstrate significant reduction of computation time at moderate error when compared to a full high-dimensional computation for linear relative permeability functions. As the prop-

$\lvert\mathcal{T}_H\rvert$	$\lvert\Psi\rvert$	$\Delta_{L^2}^{\varnothing,s}$ w/o PCA (%)	$\Delta_{L^2}^{t^{N_T},s}$ w/o PCA (%)	$\Delta_{L^2}^{\varnothing,s}$ w/ PCA (%)	$\Delta_{L^2}^{t^{N_T},s}$ w/ PCA (%)
1x1	138	8.57	2.48		
4x1	102	8.58	2.45	9.27	4.93
8x1	86	8.57	2.41	8.7	2.91
8x2	75	8.57	2.44	8.68	2.83
16x2	65	8.54	2.43	8.58	2.59

(a) L^2-errors

$\lvert\mathcal{T}_H\rvert$	$\lvert\Psi\rvert$	$\Delta_{H^1}^{\varnothing,s}$ w/o PCA (%)	$\Delta_{H^1}^{t^{N_T},s}$ w/o PCA (%)	$\Delta_{H^1}^{\varnothing,s}$ w/ PCA (%)	$\Delta_{H^1}^{t^{N_T},s}$ w/ PCA (%)
1x1	138	9.12	2.8		
4x1	102	9.12	2.75	9.98	5.56
8x1	86	9.11	2.7	9.35	3.52
8x2	75	9.11	2.73	9.37	3.52
16x2	65	9.09	2.74	9.21	3.12

(b) H^1-errors

$\lvert\mathcal{T}_H\rvert$	$\lvert\Psi\rvert$	$\Delta_{\mathcal{E}}^{\varnothing,s}$ w/o PCA (%)	$\Delta_{\mathcal{E}}^{t^{N_T},s}$ w/o PCA (%)	$\Delta_{\mathcal{E}}^{\varnothing,s}$ w/ PCA (%)	$\Delta_{\mathcal{E}}^{t^{N_T},s}$ w/ PCA (%)
1x1	138	27.9	9.59		
4x1	102	27.51	7.69	37.42	38.84
8x1	86	27.54	7.69	30.43	17.63
8x2	75	27.4	7.25	29.95	16.41
16x2	65	27.4	7.62	28.71	12.71

(c) Energy-norm-errors

Table 5.7 – Relative L^2 (a) and H^1 (b) discrepancies and discrepancies (5.3.4) in the energy norm (c) between the saturation computed with Algorithm 5.13 and the saturation computed with the fine-scale scheme from Algorithm 3.8 for different coarse mesh configurations: Number of snapshots $|\Psi|$ needed during the basis construction algorithm 5.12, mean relative discrepancy and relative discrepancy at end time t^{N_T} without usage of the PCA, respective quantities after usage of the PCA.

$\lvert\mathcal{T}_H\rvert$	$\lvert\Psi\rvert$	$\Delta_{L^2}^{\varnothing,p}$ w/o PCA (%)	$\Delta_{L^2}^{t^{N_T},p}$ w/o PCA (%)	$\Delta_{L^2}^{\varnothing,p}$ w/ PCA (%)	$\Delta_{L^2}^{t^{N_T},p}$ w/ PCA (%)
1x1	138	0.81	1.12		
4x1	102	0.8	1.08	0.84	1.27
8x1	86	0.8	1.08	0.84	1.15
8x2	75	0.79	1.07	0.93	1.25
16x2	65	0.8	1.09	0.81	1.1

(a) L^2-errors

$\lvert\mathcal{T}_H\rvert$	$\lvert\Psi\rvert$	$\Delta_{H^1}^{\varnothing,p}$ w/o PCA (%)	$\Delta_{H^1}^{t^{N_T},p}$ w/o PCA (%)	$\Delta_{H^1}^{\varnothing,p}$ w/ PCA (%)	$\Delta_{H^1}^{t^{N_T},p}$ w/ PCA (%)
1x1	138	9.12	2.8		
4x1	102	9.12	2.75	9.98	5.56
8x1	86	9.11	2.7	9.35	3.52
8x2	75	9.11	2.73	9.37	3.52
16x2	65	9.09	2.74	9.21	3.12

(b) H^1-errors

$\lvert\mathcal{T}_H\rvert$	$\lvert\Psi\rvert$	$\Delta_{\mathcal{E}}^{\varnothing,p}$ w/o PCA (%)	$\Delta_{\mathcal{E}}^{t^{N_T},p}$ w/o PCA (%)	$\Delta_{\mathcal{E}}^{\varnothing,p}$ w/ PCA (%)	$\Delta_{\mathcal{E}}^{t^{N_T},p}$ w/ PCA (%)
1x1	138	4.94	5.17		
4x1	102	4.95	5.19	4.92	5.19
8x1	86	4.94	5.19	4.95	5.22
8x2	75	4.94	5.19	4.97	5.26
16x2	65	4.95	5.2	4.97	5.23

(c) Energy-norm-errors

Table 5.8 – Relative L^2 (a) and H^1 (b) errors and errors (5.3.6) in the energy norm (c) between the pressure computed with Algorithm 5.13 and the pressure computed with the fine-scale scheme from Algorithm 3.8 for different coarse mesh configurations: Number of snapshots $|\Psi|$ needed during the basis construction algorithm 5.12, mean relative discrepancy and relative discrepancy at end time t^{N_T} without usage of the PCA, respective quantities after usage of the PCA.

erties of the flow get more complicated when using non-linear relative permeabilities we introduced an algorithm computing mobility profiles reusing the shape of full one-dimensional computations. This made our approach applicable even in these non-linear settings.

Mass conservation can only be guaranteed on the coarse mesh of our approach. Ensuring mass conservation on the fine mesh as well seems like an interesting and challenging field for future work. Furthermore using the efficient PDE solver introduced in Chapter 6 for snapshot computation seems like a promising approach as it would it allow for problems with even larger numbers of degrees of freedom to be treated efficiently.

Chapter 6

Multiscale Methods

In Chapter 5 we introduced a localized version of the RB method that is able to reduce the workload during the offline phase significantly by decreasing the necessary number of high-dimensional snapshot computations. In this chapter we introduce the multiscale finite element method (MsFEM). The MsFEM allows very memory-efficient computations of solutions to multiscale problems and—as we will demonstrate in this chapter—has a high potential for good scaling performance on modern high-performance computing architecture. The MsFEM can therefore be seen as a tool for making the offline phase of the LRBMS even more efficient and hence more applicable to real world problems.

The MsFEM is one example of so-called *multiscale* methods. Multiscale methods incorporate information about physical properties (such as permeabilities, porosities, etc) on a fine scale into coarse scale equations [46, 28]. Different methods have emerged in the last decade: The variational multiscale method offers a general framework for the construction of hierarchical approximation spaces [47]. The multiscale finite volume method [51] constructs coarse-scale transmissibilities that capture fine-scale effects. This leads to a multi-point approximation for the finite volume discretization on the coarse scale. Recently, a more robust two-point finite volume

method was proposed [65]. In [40], an adaptive version of the heterogeneous multiscale method [26], which aims at capturing the macroscopic scale of the problem by estimating necessary information from a microscopic model, was applied to immiscible two-phase flows in porous media. Finally, Henning et al. [70] recently applied the partition of unity method in the multiscale context as a means for reliable numerical homogenization for elliptic equations with rough coefficients. The multiscale finite element method [44, 28] uses solutions to local fine-scale problems to incorporate fine-scale details into coarse-scale basis functions; a recent development in this direction is the generalized multiscale finite element method [27]. The mixed multiscale finite element method [17, 1] is based on the same principles but in addition allows for mass-conserving reconstruction of the fine-scale velocities. Related methods include the numerical subgrid upscaling method [4] and the multiscale mortar method [5].

The rest of this chapter is structured as follows: In the next section we introduce the MsFEM in its most basic form and give hints to analysis results and extensions. In Section 6.2 we introduce a concept for computational parallelization of the MsFEM and in Section 6.3 we present preliminary numerical results for our implementation of this concept. Details about this implementation can be found in Chapter 7.

6.1 The Multiscale Finite Element Method

The governing equation for the introduction of the MsFEM is the pressure equation (3.1.1) without the consideration of gravitational effects which we shortly recall here: We compute the unknown pressure p such that

$$-\nabla \cdot (\lambda \kappa \nabla p) = q_1 \quad \text{in } \Omega \times (0,T), \tag{6.1.1}$$

in the space-time domain $\Omega \times (0,T)$. For the sake of simplicity we assume that $\lambda(s,x,t) = \lambda(x)$ everywhere in the space-time domain $\Omega \times (0,T)$, that is to say: we neglect the dependency of the mobility on the saturation and time in this section. Finally we equip Equation

(6.1.1) with homogeneous Dirichlet boundary values:

$$p = 0 \quad \text{on } \partial\Omega.$$

As mentioned before, the central idea of the MsFEM is to incorporate fine-scale details of the physical properties into basis functions on a coarse mesh. We will therefore reuse the fine mesh $\mathcal{T}_h$ from Section 3.2 and the coarse mesh $\mathcal{T}_H$ from Section 5.1.

Based on the two computational meshes we introduce two discrete function spaces $\mathcal{W}_H$ and $\mathcal{W}_h$:

$$\mathcal{W}_H = \{v \in \mathcal{C}^0(\Omega) | \, v|_E \in \mathbb{P}_1(E) \, \forall E \in \mathcal{T}_H\},$$
$$\mathcal{W}_h = \{v \in \mathcal{C}^0(\Omega) | \, v|_e \in \mathbb{P}_1(e) \, \forall e \in \mathcal{T}_h\},$$

where $\mathbb{P}_1(\omega)$ denotes the space of polynomials of maximum order one on the set $\omega \subset \Omega$.

Finally, following the introduction of the MsFEM in [43] and [41] we introduce multiscale basis functions:

Definition 6.1 (Multiscale Basis Functions and Local Problems)**.** Given an index $i \in \{1, \ldots, \dim(\mathcal{W}_H)\}$ and denoting by $\{\varphi_i\}$ the usual nodal finite element basis of $\mathcal{W}_H$, we call the solution $\Phi^i \in \mathcal{W}_h$ of

$$\begin{aligned} \int_E \lambda(x)\kappa(x)\nabla\Phi^i(x) \cdot \nabla\psi_h(x) \, \mathrm{d}x = 0 \quad \forall \psi_h \in \mathcal{W}_h \cap H_0^1(E), \\ \Phi^i(x) = \varphi_i(x) \quad \text{on } \partial E \end{aligned} \tag{6.1.2}$$

for all $E \in \mathcal{T}_H$ *multiscale basis function*. The problem (6.1.2) will also be referred to as *local problem* in the sequel.

Given the multiscale basis Φ with

$$\Phi = \{\Phi^i | 1 \leq i \leq |\mathcal{W}_H|\}$$

we define the MsFEM space

$$\mathcal{W}_H^{\text{ms}} = \text{span}(\Phi) \subset \mathcal{W}_h.$$

Now we are ready to define the MsFEM solution to the pressure equation.

Definition 6.2 (MsFEM Solution)**.** The MsFEM solution $p_H^{\text{ms}} \in \mathcal{W}_H^{\text{ms}}$ to Equation (6.1.1) is given by

$$\int_\Omega \lambda(x)\kappa(x)\nabla p_H^{\text{ms}}(x) \cdot \nabla\Psi(x)\,\mathrm{d}x = \int_\Omega q_1(x)\Psi(x)\,\mathrm{d}x \quad \forall\Psi \in \mathcal{W}_H^{\text{ms}}.$$

In different works multiple features of the MsFEM method were investigated. Hou et al. demonstrated that the order of convergence of the MsFEM coincides with that of the linear finite element method if the heterogeneities in $\lambda \cdot \kappa$ are well-resolved by the fine mesh $\mathcal{T}_h$ (see [43] and references therein).

Furthermore, different authors have shown a negative impact of so-called *resonance errors* that occur when the fine mesh scale and the physical scale of the problem are close. For those cases *oversampling* methods were suggested that reduce the influence of the boundary conditions on the multiscale basis functions, see [28, 41], for example.

Finally, we point out that an extension of the MsFEM to nonzero Dirichlet- and general Neumann-boundary-values can be done. In this case, additional local problems (6.1.2) for the boundary values need to solved.

6.2 Parallelization Concept

The key idea of our parallelization concept is to reflect the different layers—coarse and fine—of the MsFEM in the code and to make use of the independence of the different local problems.

The coarse scale of the problem, represented by a coarse-scale mesh and a coarse-scale discrete function space, is handed out to different distributed memory compute nodes via the Message Passing Interface (MPI), more specifically its abstraction in the dune-grid interface, see Chapter 7. We do not base the coarse-scale parallelization directly on the coarse mesh but on coarse *patches*.

Definition 6.3 (Coarse Patch)**.** Let $\mathcal{T}_H$ (see Section 6.1) denote the coarse mesh of a MsFEM method. We then call any subset of $\mathcal{T}_H$ consisting of coarse mesh entities $\{E|E \in \mathcal{T}_H\}$ a *coarse patch*.

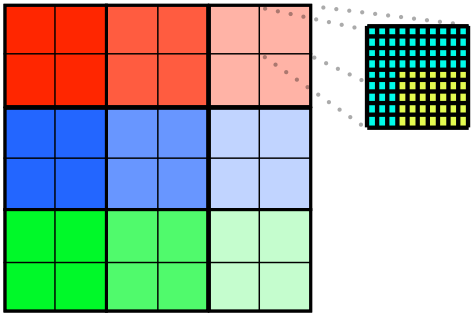


Figure 6.1 – Illustration of parallelization concept: Coarse patches for distributed memory parallelization, fine scale mesh for basis function computation using shared memory parallelization.

For an illustration of possible coarse patches see Figure 6.1 where each of the nine coarse patches consists of 4 coarse mesh elements and is colorized in a distinct color shade. The construction of these coarse patches can be handled by the load-balancing in the DUNE grid-interface: Calling the `loadBalance()` function on a grid will result in multiple coarse mesh elements being grouped together and distributed to a MPI process if the number of MPI processes is smaller than the number of coarse mesh elements. At the moment we rely on the automatic load-balancing of the respective coarse grid implementation as all local problems are assumed to have approximately the same computational demands. For cases where this assumption is not valid, in the case of complex coarse meshes, for example, more evolved algorithms for the generation of the coarse patches would need to be implemented.

Now, after distributing the computations on the coarse mesh as described above, the computation of all multiscale basis functions (see Definition 6.1) on the coarse mesh elements of each coarse patch needs to be performed. Firstly, the computation of several multiscale basis functions can be done in parallel on one compute node. Secondly, each single computation of a basis function can be parallelized. For both parallelizations a shared-memory approach is advisable as different local problems on one coarse element share

the same system matrix. Furthermore, in this case, an ideal shared-memory parallelization is faster than a distributed memory parallelization as the latter suffers from communication overhead and we assume the equation system for one basis function to fit in the memory of one compute node. Clearly, the word *ideal* in this case includes that locking does not introduce noticeable slowdown.

Another important concept is the so-called *virtual grid refinement layer*. Although our formulation of the MsFEM in Section 6.1 assumes the multiscale basis functions to be globally continuous (as they belong to $\mathcal{W}_h$), error estimates for non-conforming formulations were derived in [29, 45] under certain conditions on the properties of the coefficients.

Now, if we assume that the multiscale basis functions do not need to be conforming, there is no need to use restrictions of a function space on the global fine mesh $\mathcal{T}_h$ in Equation (6.1.2) but local meshes and according local discrete function spaces can be used. This frees us from any limitations introduced by the coarse scale mesh and we only assume the local meshes used for the computation of the multiscale basis functions on the coarse cell $E \in \mathcal{T}_H$ to cover the whole cell E. In other terms: The local problems for a given coarse mesh element E are now solved on a computational mesh $\mathcal{T}_h^E$ with $\{x \in E\} \subset \mathcal{T}_h^E$.

This way it becomes possible, for example, to use a simple structured mesh for the local problems in conjunction with an unstructured mesh on the coarse scale. Apart from efficiency considerations this also has the advantage of smaller memory consumption for the computation of the multiscale basis functions which can be paramount on large-scale clusters with very limited per-core memory.

Remark 6.4. *The parallelization concept presented in this section is not restricted to the MsFEM context. It was demonstrated in [68] that multiple multiscale methods, such as the VMM, the HMM or the MsFEM can be cast into the same mathematical abstraction allowing a unified implementation based on the principles introduced above.*

6.3 Numerical Experiments

In this section we present results for a simple 3D-benchmark for our multiscale-library dune-multiscale, see Chapter 7. Our model problem is the stationary heat diffusion equation in $\Omega = (0,1)^3$: Find $p^\epsilon \in H^1(\Omega)$ such that

$$\begin{aligned} -\nabla \cdot (\kappa^\epsilon(x)\nabla p^\epsilon(x)) &= f^\epsilon(x) \quad \text{in } \Omega, \\ p^\epsilon(x) &= 0 \quad \text{on } \Gamma_p = \{x \in \partial\Omega \,|\, x_3 \in \{0,1\}\}, \\ -\kappa^\epsilon(x)\nabla p(x) \cdot \mathrm{n} &= 0 \quad \text{on } \partial\Omega \setminus \Gamma_p, \end{aligned} \tag{6.3.1}$$

where k^ϵ, f^ϵ are for $x = (x_1, x_2, x_3)^\intercal \in \mathbb{R}^3$ given by

$$\begin{aligned} k^\epsilon(x) &= \frac{1}{8\pi^2} \begin{pmatrix} 2(2+\cos(\frac{2\pi x_1}{\epsilon}))^{-1} & 0 & 0 \\ 0 & 1+\frac{1}{2}\cos(\frac{2\pi x_1}{\epsilon}) & 0 \\ 0 & 0 & 1 \end{pmatrix}, \\ f^\epsilon(x) &= -\nabla \cdot (k^\epsilon(x)\nabla \bar{p}^\epsilon(x)). \end{aligned}$$

Here, the exact solution $\bar{p}^\epsilon$ is given by

$$\bar{p}^\epsilon(x) = \sin(2\pi x_1)\sin(2\pi x_2) + \frac{\epsilon}{2}\cos(2\pi x_1)\sin(2\pi x_2)\sin(2\pi\frac{x_1}{\epsilon}).$$

The heat diffusion coefficient κ^ϵ shows oscillations on a small scale defined by $\epsilon > 0$. We will choose $\epsilon = 0.05$. In Figure 6.2 we show the exact solution $\bar{p}^\epsilon$ for $\epsilon = 0.05$. In Figure 6.3 the two scales in the heat-diffusion model problem can be observed: On the coarse scale, a sinusoidal wave with wavelength one can be seen while on the fine scale (in the close-up) an oscillation with much shorter wavelength ($\epsilon = 0.05$) is noticable. For the MsFEM formulation of Equation (6.3.1) see Section 6.1.

In the following experiments, our focus is on the strong and weak scaling properties of our implementation. Here the term *strong scaling* refers to runtime tests where the computational effort for solving the problem is kept constant while the number of MPI-processes is increased. The term *weak scaling* refers to tests where the computational effort is increased by the same factor as the number of MPI-processes.

In Figure 6.4 we present strong scaling results for our MsFEM implementation using hexahedral meshes on both the fine and coarse

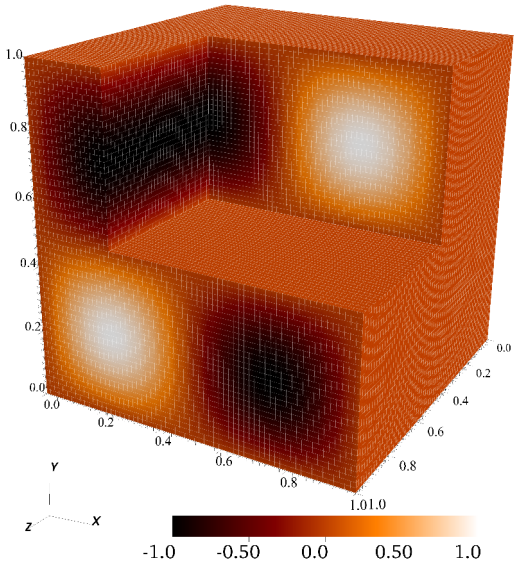


Figure 6.2 – Computational mesh and exact solution $\bar{p}^{\epsilon}$ to Equation 6.3.1 for $\epsilon = 0.05$: block $[0.25, 1] \times [0.5, 1] \times [0.5, 1]$ cut out.

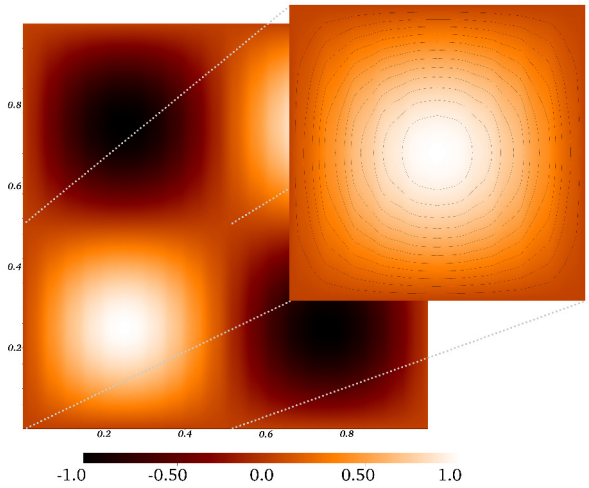


Figure 6.3 – Exact solution to Equation 6.3.1 for $\epsilon = 0.05$: slice (using z-normal) and close-up with contour lines.

scale. The coarse-scale mesh $\mathcal{T}_H$ consists of $32 \cdot 32 \cdot 32 = 32768$ cubes. Each coarse mesh element is subdivided into 4096 cubes leading to $134,217,728$ fine elements in total. Remember though that for the MsFEM, we never need to set up a fine mesh on the whole domain. For this test we do not use oversampling techniques like those mentioned in Section 6.1. We use a biconjugate gradient stabilized (BiCGStab) solver for both the computation of the MsFEM basis functions and for the solution of the coarse-scale problem and we run the test on the *Cheops* cluster in Cologne (Intel Xeon Westmere X5650 CPUs, 2.66 GHz, Hexacore, QDR Infiniband) using 16 to 1024 cores.

We see that most parts of our implementation scale nearly perfectly as expected: Assembling and solving the local problems for the multiscale basis functions does not necessitate any kind of communication between the different MPI processes and therefore this part of the code scales perfectly. The coarse right hand side assembly necessitates only one communication which is negligible compared to the overall runtime of the assembly. The communication of shared

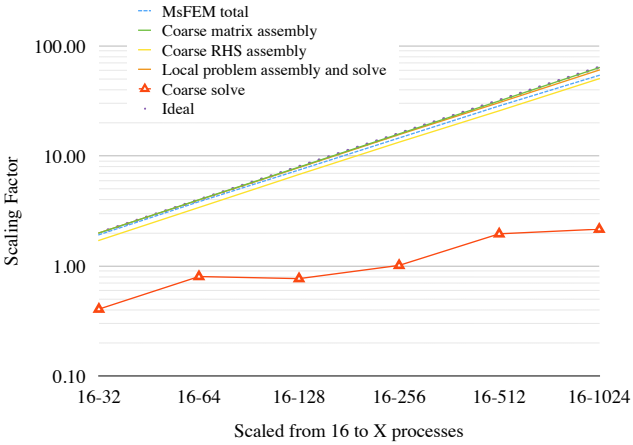


Figure 6.4 – Strong scaling for model problem (6.3.1) from 16 to 1024 cores: Scaling factor in wall runtime for different parts of the MsFEM solver.

degrees of freedom of the global coarse matrix is done during the coarse solve step. Therefore the coarse matrix assembly also scales near to perfect. Only the coarse solve step does not show a desirable strong scaling. This is to be explained by the fact that the coarse problem size is still fairly small and that the communication overhead does not pay out at this problem size: With roughly 32000 coarse cells in total, each process solves a subproblem on only 32 cells when computing with all available compute nodes and therefore the communication dominates the overall runtime. We expect this problem to vanish when the size of the coarse grid is increased.

Next, Figure 6.5 shows results from a weak scaling test using the same setup as above: Again we scale from 16 to 1024 cores on the Cheops cluster and we do not use oversampling strategies. The local meshes again consist of 4096 elements but the coarse mesh size is now adapted to the number of MPI processes: on 16 processes we use 512 coarse elements and this size is doubled in each step such that we use 32768 elements on 1024 processes.

Again we see that most parts of our MsFEM solver show good scaling, only the coarse solver shows a tendency to higher runtimes with

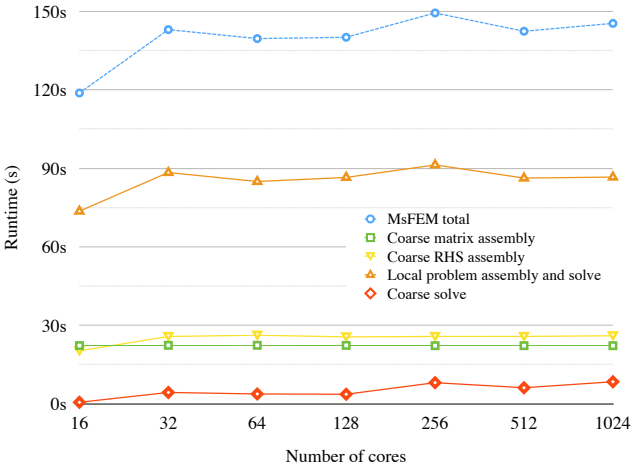


Figure 6.5 – Weak scaling for model problem (6.3.1) from 16 to 1024 cores: Wall runtime in seconds for different parts of the MsFEM solver.

increasing numbers of processes for the reasons explained above. Remember that, as this is a weak scaling test, the runtimes for all MPI configurations should ideally be the same.

Most importantly, the assembly and solve of the local problems, which obviously accounts for most of the overall runtime, shows near-ideal behavior.

Finally, we plot the distribution of the overall runtime among the different parts of our implementation using the mesh configuration of the strong scaling test on 16, 128 and 1024 cores in Figure 6.6.

As reasoned above, the percentage of runtime spent in the coarse solver increases with increasing number of MPI processes due to increased communication overhead while the time needed for assembling the coarse system stays roughly the same. Nevertheless, the most important result drawn from Figure 6.6 is that the assembly and solution of the local problems accounts for the vast majority of the overall runtime. As this part of the code was shown to be scaling (nearly) perfectly above, we believe that the good overall strong and weak scaling for the MsFEM shown up to 1024 cores will continue

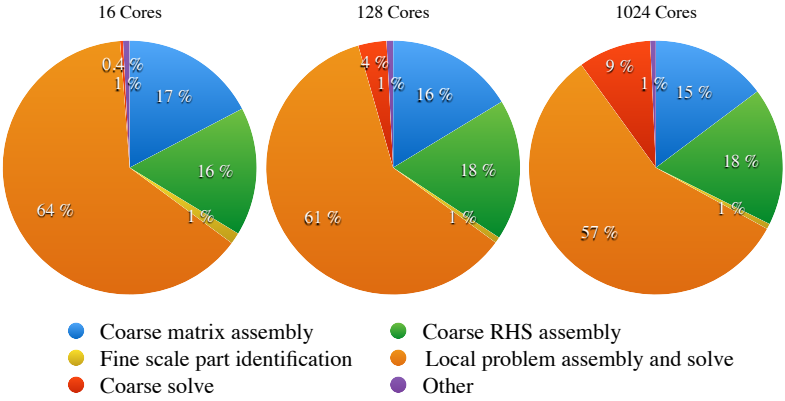


Figure 6.6 – Wall time distribution among different parts of the MsFEM solver on 16, 128 and 1024 cores.

on bigger clusters.

6.4 Summary and Outlook

We introduced the multiscale finite element method which can be used to render the offline phase of the localized reduced basis multiscale method even more efficient by replacing the high-dimensional DG discretization. We introduced a general parallelization concept that does not only apply to the MsFEM but to a general abstraction of multiscale methods.

In numerical tests we were able to demonstrate good scaling of our implementation of this parallelization concept on a mid-sized cluster.

In the next step, the coupling between LRBMS and MsFEM could be addressed and scaling tests on higher numbers of cores should be done.

Chapter 7

Software Concepts

In this chapter we shortly outline some of the principal design concepts of our implementations of the reduced basis method (dune-rb) and the multiscale finite element method (dune-multiscale). Both implementations are based on the *Distributed and Unified Numerics Environment* (DUNE) [9, 21, 19]. DUNE is a modular C++ framework providing interfaces to and implementations of computational meshes (dune-grid), iterative solvers (dune-istl) and two implementations of commonly used discretizations for partial differential equations (dune-fem and dune-pdelab).

In Section 7.1 we give some details on the implementation of reduced basis methods that was used to produce all RB-related results in this work. In Section 7.2 we give some pointers to our implementation of the parallelization concept for the multiscale finite element method as introduced in Chapter 6. We conclude this chapter with a short summary and outlook in Section 7.3.

7.1 The dune-rb Module

The dune-rb module[1] is a joint effort by M. Drohmann, F. Schindler and the author of this document of implementing reduced basis methods within the DUNE framework. It provides small and clean interfaces for all parts of RB methods as well as implementations for the most commonly used methods.

The module was designed with a focus on modularity, enabling the user to implement only parts being particular to his individual approach and reusing others. One example could be a situation where a user wants to implement his own basis generation algorithm but reuse the shipped reduced basis space implementation.

Our interfaces and the implementation are directly based on linear algebra quantities as opposed to the usage of special discrete function spaces as in the dune-fem module. This way we try to introduce as little overhead as possible.

Furthermore, our software strictly separates the two computational phases of RB methods (offline- and online-phase) in the code.

In the following paragraphs we will shortly outline the most important parts of our implementation.

7.1.1 Linear Algebra

One important concept of reduced basis methods that is not represented in the standard linear algebra in DUNE is the separable parametric tensors from Equation (2.3.2). In Listing 7.1 we present our interface for the so-called separable parametric container. It provides the method `setSymbolicCoefficients` to set the coefficients $\Theta^q_{B/L}(\mu)$ via a vector of strings which contain the analytical coefficient functions depending on the argument `variable` which defaults to `mu`. Now, after the components $\mathbf{A}^q_N$ / $\mathbf{b}^q_N$ were set via the `component` method, the complete tensors can be retrieved via the `complete` method, which, for a given parameter `param`, performs the sum (2.3.2).

In addition to the interface from Listing 7.1, implementations for dense matrices and vectors and sparse matrices exist using the

[1] http://users.dune-project.org/repositories/projects/dune-rb-dev.git

Eigen[2] linear algebra package.

7.1.2 Reduced Basis Space

The reduced basis space from Definition 2.4 is represented via the interface shown in Listing 7.2. We provide means to set and get the basis vectors of the RB space via the methods `setDofVector` and `getDofVector` and methods to set and get the whole basis via `setDofMatrix` and `getDofMatrix`. Furthermore, new single basis vectors can be added via `addDofVector` and a set of new basis vectors can be added via `addDofMatrix`. For the methods operating on whole sets of basis vectors, the single vectors are assumed to be collected as the columns of a matrix. Finally, the reduced basis space can be saved to and loaded from disk via the `save` and `load` methods.

At the moment, dune-rb ships an implementation of the RB space suitable for the LRBMS method with arbitrary numbers of coarse grid elements.

7.1.3 Generators

One of the key concepts of dune-rb is the separation of *generators* and *connectors*. Generators are used to produce and process high-dimensional data during the offline phase while connectors are a result of the offline phase, process only reduced-dimensional data and are used during the online phase.

Examples for generators are the reduced solver generator which performs the Galerkin projection (2.3.1) and yields a reduced equation system used during the online phase or the reduced basis generator which performs a basis construction algorithm and yields the reduced basis.

Listing 7.3 shows the interface for the reduced basis generator. The reduced basis can be initialized via the `init` method and the main basis extension loop is started using the `generate` method which takes the training set `paramSample` as argument. Additional arguments `saveStep` and `savePath` can be passed to the algorithm in order to back up intermediate data in regular intervals such that the computation can be resumed after a premature exit.

[2] `http://eigen.tuxfamily.org`

Listing 7.1 – Interface for separable parametric matrix and vector from Equation 2.3.2 (excerpt)

```
#include <dune/rb/parameter/function.hh>

namespace Dune { namespace RB {
namespace LA { namespace SeparableParametric {
namespace Container {
template< typename ComponentImp,
          typename ComponentsVectorImp,
          typename CoefficientsVectorImp,
          typename ParameterImp >
class Interface {
public:
  typedef ComponentImp ComponentType;
  typedef ComponentsVectorImp ComponentsVectorType;
  typedef ComponentType CompleteType;
  typedef Parameter::Function<double, 50, double, 50>
    ParameterFunctionType;
  typedef ParameterImp ParameterType;

  virtual ComponentType& component(const int q) = 0;
  virtual int numComponents() const = 0;
  virtual bool
  hasParameterindependentPart() const = 0;
  virtual ComponentType&
  parameterindependentPart() = 0;

  virtual const CompleteType&
  complete(const ParameterType& param) const = 0;

  virtual void setSymbolicCoefficients(
    std::vector<std::string> symbolicCoefficients,
    const std::string variable = "mu") = 0;

  virtual const std::string getVariable() const = 0;

  virtual bool
  save(const std::string& path) const = 0;
  virtual bool load(const std::string& path) = 0;
}; }}}}}
```

Listing 7.2 – Interface for the reduced basis space X_N from (2.4) (excerpt)

```
namespace Dune {
namespace RB {
namespace Offline {
namespace Space {

template< class Traits >
class Interface
{
public:
  typedef typename Traits::DofMatrixType
  DofMatrixType;
  typedef typename Traits::DofVectorType
  DofVectorType;

  virtual bool
  save (const std::string& path) const = 0;
  virtual bool
  load (const std::string& path) = 0;

  virtual const int size () const = 0;

  virtual const DofMatrixType&
  getDofMatrix() const = 0;
  virtual const DofVectorType&
  getDofVector(const int i) const = 0;

  virtual void
  setDofMatrix(const DofMatrixType& dofMatrix) = 0;
  virtual void
  setDofVector(const int i,
               const DofVectorType& dofVector) = 0;

  virtual void
  addDofMatrix(const DofMatrixType& dofMatrix) = 0;
  virtual void
  addDofVector(const DofVectorType& dofVector) = 0;
}; }}}}
```

Listing 7.3 – Interface for the reduced basis generator (excerpt)

```
namespace Dune { namespace RB { namespace Offline {
namespace Generator { namespace ReducedBasis {
class Interface
{
  typedef RBTraits::ParameterType ParameterType;
  typedef RBTraits::ParameterSampleType
    ParameterSampleType;

  virtual void init(const ParameterType& param) = 0;

  virtual void
  generate(ParameterSampleType& paramSample,
           const int saveStep = -1,
           const std::string savePath = "") = 0;

  virtual bool
  save(const std::string& savePath) const = 0;
}; }}}}}
```

7.1.4 Connectors

As described above, connectors are classes only processing reduced-dimensional data (with the exception of the high-dimensional solver connector, see below) Instances of these classes are used to carry out the reduced-dimensional simulation and error estimation during the online phase.

One example is the estimator connector which is shown in Listing 7.4. The estimator connector provides methods for the initialization and update of its data. These methods could, for example, be used to pass and update the matrix G from Equation (2.5.2) to the connector. The method `estimate` could then perform the evaluation (2.5.3) for the given parameter `mu` and reduced solution `vector`. Finally, `save` and `load` methods can be used to save and load the respective data.

dune-rb ships an implementation of the residual estimator described in Section 2.5 and an estimator computing the exact error

Listing 7.4 – Interface for the estimator connector (excerpt)

```
namespace Dune { namespace RB {
namespace Reduced { namespace Estimator {
namespace Connector {
template< class DofVectorImp, class ParameterImp >
class Interface
{
public:
  typedef DofVectorImp DofVectorType;
  typedef ParameterImp ParameterType;

  virtual void init() = 0;
  virtual void update() = 0;

  virtual double
  estimate(const ParameterType& mu,
           const DofVectorType& vector) = 0;

  virtual bool
  save(const std::string& path) const = 0;
  virtual bool
  load(const std::string& path) = 0;

}; }}}}}
```

Listing 7.5 – Interface for the detailed solver connector (excerpt)

```
namespace Dune { namespace RB { namespace Detailed {
namespace Solver { namespace Connector {
class Interface
{
public:
  virtual const ModelType& model() const = 0;

  virtual void init() = 0;

  virtual const MatrixType&
  getSystemMatrix() const = 0;
  virtual const VectorType&
  getRightHandSide() const = 0;

  virtual void
  visualize(const DofVectorType& vector) const = 0;

  virtual bool
  solve(const ParameterType& param,
        DofVectorType& solution) = 0;

  virtual const MapperType& mapper() const = 0;
}; }}}}}
```

to the respective high-dimensional solution.

7.1.5 Link to High-Dimensional Solver

The algorithm for the computation of high-dimensional snapshots is supposed to be provided by the user. As a means to this end, dune-rb provides an interface which allows for easy, non-intrusive coupling of user code. Listing 7.5 shows the interface for the so-called detailed solver connector. The detailed solver connector stores information about the underlying model class. A call to the `init` method could, for example assemble the underlying equation system using the model. Calls to the `getSystemMatrix` and `getRightHandSide` methods return the affinely decomposed system matrix and right

hand side for the high-dimensional equation (see Definition 2.3), both using the separable parametric container interface introduced above. The `visualize` method can be used to visualize a high-dimensional solution computed using the `solve` method for a given parameter `param`. Finally, the `mapper` method returns a class providing mappings from a grid-cell-local numbering of the degrees of freedom of the discrete function space to a global numbering. Currently we assume this mapping to be provided by the dune-fem module [21].

7.1.6 Miscellaneous

A couple of different, smaller concepts and helper classes are needed to perform the simulations presented in this thesis. For the sake of completeness we list them here without providing details on their implementation.

For the implementation of the LRBMS, a pseudo grid implementation for the definition of the coarse grid together with an appropriate parser for grid configuration files was developed. Also, the localization of high-dimensional solutions to the elements of this coarse grid was implemented. Finally, the principal component analysis was implemented for general data sets given via of matrix of column-vectors.

For the realization of the online greedy basis construction method from Chapter 4, a dictionary class was introduced and the above-named generators and connectors (dictionary generator, estimator generator, reduced solver generator and respective connectors) were specialized for the dictionary context.

7.2 The dune-multiscale Module

The dune-multiscale software package[3] was initiated by P. Henning as a serial implementation of the MsFEM on triangular meshes. Recently, in the context of the EXA-DUNE project, founded by the german science foundation under the contract number OH 98/5-

[3] https://github.com/wwu-numerik/DUNE-Multiscale

1, the code base was nearly completely renewed to implement the parallelization concept from Chapter 6.

Here, we shortly summarize the state of the art in dune-multiscale as of June 2014 [64]. So far we implemented the virtual grid refinement layer and the distributed-memory parallelization part of the above-named concept, the shared-memory parallelization is under current development. The course of action is as follows:

1. Generate a coarse grid using the dune-grid module
2. load-balance the grid to the given number of MPI processes using the abstraction of the MPI routines in the dune-grid interface
3. On every MPI-rank, given a coarse patch $S \subset \mathcal{T}_H$
   a) For every $E \in S$
      i. Generate a local structured mesh $\mathcal{T}_h^E$ covering E: $\{x \in E\} \subset \mathcal{T}_h^E$
      ii. Solve the local problems (Definition 6.1) on E for all nodal finite element basis functions φ_i with nonzero support in E: $\text{supp}(\varphi_i) \cap E \neq \emptyset$
      iii. Assemble the contribution of entity E to the coarse-scale system (see Definition 6.2)
   b) Solve the coarse scale system

Currently, our implementation can handle rectangular meshes in two and three spatial dimensions and arbitrary Neumann- and Dirichlet-boundary values. Further, a simple oversampling technique is provided where the computational domain for the local problems is enlarged by a given number of fine cells and the solutions (i.e. the multiscale basis functions) are then restricted to the original domain, see Figure 7.1 for an illustration.

7.3 Summary and Outlook

We introduced our implementation of the reduced basis method dune-rb. We presented some of the key classes and concepts which

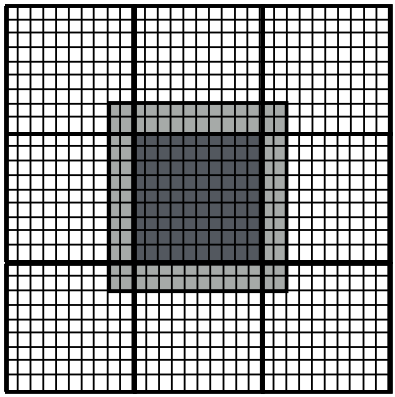


Figure 7.1 – Illustration of a simple oversampling technique: The MsFEM basis functions for the center coarse mesh element are computed in the enlarged area marked in light gray and then are restricted to the dark area again.

aim at separating the computations of the offline and online phase of the method in our code and at providing easy exchangeability of certain parts like the basis construction algorithm, estimators or others.

Furthermore we introduced our software package dune-multiscale which is based on the general parallelization concept introduced in Chapter 6. Currently, dune-multiscale implements the distributed-memory part of this concept. Shared-memory parallelization is under active development and will be available in the near future.

Chapter 8

Conclusion and Outlook

In this thesis we introduced and investigated several schemes that allow for efficient simulations of multiscale problems. As a means to this end we first introduced a novel basis construction algorithm for reduced basis methods: Based on a dictionary of basis vector candidates that is constructed during the offline phase, we perform a full greedy algorithm during the online phase to construct a parameter-fit reduced basis. We demonstrated that the necessary computations during the online phase can be vectorized and therefore performed efficiently. We applied this approach to a stationary heat diffusion problem in three spatial dimensions and observed good performance of our algorithm in terms of runtimes for small deviations from the training parameters. Furthermore our method provides at least identical approximation quality as a comparable reduced basis computed with a conventional basis construction algorithm. In future contributions to this topic, the question of sensible dictionary construction should be investigated.

In a second step we introduced a novel basis construction algorithm for reduced basis methods that is based on ideas from multiscale methods: In addition to the (fine) mesh which is used for the detailed computations, a second (coarser) computational mesh is introduced. Based on this coarse mesh, localized reduced bases

are constructed and a global reduced formulation, the localized reduced basis multiscale method (LRBMS), based on a discontinuous Galerkin discretization, is defined. We first applied this technique to stationary heat-diffusion problems with heterogeneous diffusion coefficients and demonstrated that we are able to displace workload between the offline and online phase by changing the size of the coarse mesh. Next, we introduced a parameterized formulation of two-phase flow in porous media in which the parameter couples the equations for the saturation and the pressure. We used the LRBMS to introduce model order reduction for the pressure equation and were able to demonstrate good runtime reduction of the overall two-phase flow simulation at notable but acceptable error increase for saturation and pressure when compared to a full high-dimensional two-phase flow simulation. Our scheme is not mass-conservative on the fine mesh. As this may limit the applicability of the approach, future work could investigate possible strategies to ensure mass conservation.

Next we introduced the multiscale finite element method (MsFEM). The MsFEM is a technique for efficient simulation of multiscale problems and could be used to render the offline phase of the LRBMS more cost-efficient for large-scale problems. We presented a parallelization concept that aims at massively parallel simulations for multiscale methods such as the MsFEM. It was motivated that multiscale methods promise good scalability even to exa-scale architectures since the biggest parts of the overall workload can be done in parallel without any communication. This was supported by our scaling tests run on a mid-sized cluster. In these tests we saw nearly perfect scaling for all costly parts of the multiscale finite element method. Future work in this direction should include the implementation of missing parts of the parallelization concept and tests with higher numbers of cores.

Finally we presented our software framework dune-rb which provides interfaces for all parts needed for a reduced basis simulation and ships implementations for the most commonly used ones and the current state of our implementation of the aforementioned parallelization concept for multiscale methods.

Bibliography

[1] J. E. Aarnes, S. Krogstad, and K.-A. Lie. Multiscale mixed/mimetic methods on corner-point grids. *Computational Geosciences*, 12(3):297–315, Jan. 2008.

[2] F. Albrecht, B. Haasdonk, S. Kaulmann, and M. Ohlberger. The localized reduced basis multiscale method. In A. Handlovičová, Z. Minarechová, and D. Ševčovič, editors, *Algoritmy 2012*, pages 393–403, Bratislava, Apr. 2012. Slovak University of Technology in Bratislava, Publishing House of STU.

[3] A. C. Antoulas. *Approximation of large-scale dynamical systems*, volume 6 of *Advances in Design and Control*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 2005.

[4] T. Arbogast. Implementation of a locally conservative numerical subgrid upscaling scheme for two-phase Darcy flow. *Computational Geosciences*, 6(3–4):453–481, 2002.

[5] T. Arbogast, G. Pencheva, M. F. Wheeler, and I. Yotov. A multiscale mortar mixed finite element method. *Multiscale Modeling & Simulation. A SIAM Interdisciplinary Journal*, 6(1):319–346, 2007.

[6] D. N. Arnold. An interior penalty finite element method with discontinuous elements. *SIAM Journal on Numerical Analysis*, 19(4):742–760, Aug. 1982.

[7] D. N. Arnold, F. Brezzi, B. Cockburn, and L. D. Marini. Unified Analysis of Discontinuous Galerkin Methods for Elliptic Problems. *SIAM Journal on Numerical Analysis*, 39(5), May 2001.

[8] 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. *Comptes Rendus Mathematique*, 339(9):667–672, Oct. 2004.

[9] 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, June 2008.

[10] 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, July 2008.

[11] J. Bernlöhr. Online Reduzierte Basis Generierung für parameterabhängige elliptische partielle Differentialgleichungen. *Diploma Thesis from University of Stuttgart*, June 2012.

[12] P. Binev, A. Cohen, W. Dahmen, R. A. DeVore, G. Petrova, and P. Wojtaszczyk. Convergence Rates for Greedy Algorithms in Reduced Basis Methods. *SIAM Journal on Mathematical Analysis*, 43(3):1457–1472, 2011.

[13] A. Buffa, Y. Maday, A. T. Patera, C. Prud'homme, and G. Turinici. A priori convergence of the greedy algorithm for the parametrized reduced basis method. *ESAIM: Mathematical Modelling and Numerical Analysis*, 46(3):595–603, 2012.

[14] S. Chaturantabut and D. C. Sorensen. Discrete Empirical Interpolation for nonlinear model reduction. *Proceedings of the 48h*

IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference, pages 4316–4321, Dec. 2009.

[15] S. Chaturantabut and D. C. Sorensen. Application of POD and DEIM on dimension reduction of non-linear miscible viscous fingering in porous media. *Mathematical and Computer Modelling of Dynamical Systems*, 17(4):337–353, Aug. 2011.

[16] Y. Chen, J. S. Hesthaven, and Y. Maday. A seamless reduced basis element method for 2d Maxwell's problem: An introduction. In J. S. Hesthaven and E. M. Rønquist, editors, *Spectral and High Order Methods for Partial Differential Equations*, volume 76 of *Lecture Notes in Computational Science and Engineering*, pages 141–152. Springer, Berlin, Heidelberg, Sept. 2011.

[17] Z. Chen and T. Y. Hou. A mixed multiscale finite element method for elliptic problems with oscillating coefficients. *Mathematics of Computation*, 72(242):541–576, 2003.

[18] M. A. Christie and M. J. Blunt. Tenth SPE Comparative Solution Project: A Comparison of Upscaling Techniques. *SPE Reservoir Evaluation & Engineering*, 4(4), 2001.

[19] A. Dedner, B. Flemisch, and R. Klöfkorn, editors. *Advances in DUNE: Proceedings of the DUNE User Meeting, Held in October 6th-8th 2010 in Stuttgart, Germany.* Springer, Berlin Heidelberg, 2012 edition, Apr. 2012.

[20] A. Dedner and R. Klöfkorn. A generic stabilization approach for higher order discontinuous Galerkin methods for convection dominated problems. *Journal of Scientific Computing*, 47(3):365–388, 2011.

[21] A. Dedner, R. Klöfkorn, M. Nolte, and M. Ohlberger. A generic interface for parallel and adaptive discretization schemes: abstraction principles and the Dune-Fem module. *Computing*, 90(3–4):165–196, Aug. 2010.

[22] M. Dihlmann, M. Drohmann, and B. Haasdonk. Model reduction of parametrized evolution problems using the reduced basis method with adaptive time-partitioning. In *Proceedings of ADMOS*, 2011.

[23] M. Dihlmann, S. Kaulmann, and B. Haasdonk. Online reduced basis construction procedure for model reduction of parametrized evolution systems. In I. Troch and F. Breitenecker, editors, *7th Vienna International Conference on Mathematical Modelling (MATHMOD 2012)*, pages 112–117, Vienna, Austria, Feb. 2012. Vienna University of Technology.

[24] M. Drohmann, B. Haasdonk, and M. Ohlberger. Reduced basis approximation for nonlinear parametrized evolution equations based on empirical operator interpolation. *SIAM Journal on Scientific Computing*, 34(2):937–969, 2012.

[25] M. Drohmann, B. Haasdonk, and M. Ohlberger. Reduced basis model reduction of parametrized two-phase flow in porous media. In *MATHMOD 2012 - 7th Vienna International Conference on Mathematical Modelling, Vienna, February 15–17, 2012*, Feb. 2012.

[26] W. E and B. Engquist. The heterogeneous multiscale methods. *Communications in Mathematical Sciences*, 1(1):87–132, 2003.

[27] Y. Efendiev, J. Galvis, and T. Y. Hou. Generalized multiscale finite element methods (GMsFEM). arxiv:1301.2866v2 [math.na], arXiv.org, Jan. 2013.

[28] Y. Efendiev and T. Y. Hou. *Multiscale Finite Element Methods*, volume 4 of *Surveys and Tutorials in the Applied Mathematical Sciences*. Springer Science+Business Media, New York, 2009.

[29] Y. R. Efendiev, T. Y. Hou, and X.-H. Wu. Convergence of a nonconforming multiscale finite element method. *SIAM Journal on Numerical Analysis*, 37(3):888–910, Jan. 2000.

[30] J. Eftang, A. Patera, and E. Rønquist. An "*hp*" certified reduced basis method for parametrized elliptic partial differential

equations. *SIAM Journal on Scientific Computing*, 32(6):3170–3200, 2010.

[31] J. L. Eftang, A. T. Patera, and E. M. Rønquist. An hp certified reduced basis method for parametrized parabolic partial differential equations. In J. S. Hesthaven and E. M. Rønquist, editors, *Spectral and High Order Methods for Partial Differential Equations*, volume 76 of *Lecture Notes in Computational Science and Engineering*, pages 179–187. Springer Berlin Heidelberg, 2011.

[32] A. Ern, I. Mozolevski, and L. Schuh. Discontinuous Galerkin approximation of two-phase flows in heterogeneous porous media with discontinuous capillary pressures. *Computer Methods in Applied Mechanics and Engineering*, 199(23-24):1491–1501, Apr. 2010.

[33] M. A. Grepl, Y. Maday, N.-C. Nguyen, and A. T. Patera. Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations. *ESAIM: Mathematical Modelling and Numerical Analysis*, 41(3):575–605, 2007.

[34] M. A. Grepl and A. T. Patera. A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations. *ESAIM: Mathematical Modelling and Numerical Analysis*, 39(1):157–181, 2005.

[35] B. Haasdonk. Convergence rates of the POD–Greedy method. *ESAIM: Mathematical Modelling and Numerical Analysis*, 47(3):859–873, Sept. 2012.

[36] B. Haasdonk, M. Dihlmann, and M. Ohlberger. A training set and multiple bases generation approach for parameterized model reduction based on adaptive grids in parameter space. *Mathematical and Computer Modelling of Dynamical Systems*, 17(4):423–442, 2011.

[37] B. Haasdonk and M. Ohlberger. Reduced basis method for finite volume approximations of parametrized linear evolution equations. *ESAIM: Mathematical Modelling and Numerical Analysis*, 42(2):277–302, 2008.

[38] B. Haasdonk and M. Ohlberger. Reduced basis method for explicit finite volume approximations of nonlinear conservation laws. In *Hyperbolic problems: theory, numerics and applications*, pages 605–614. Amer. Math. Soc., Providence, RI, 2009.

[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] P. Henning, M. Ohlberger, and B. Schweizer. Adaptive heterogeneous multiscale methods for immiscible two-phase flow in porous media. arxiv:1307.2123 [math.na], arXiv.org, July 2013.

[41] P. Henning and D. Peterseim. Oversampling for the Multiscale Finite Element Method. *Multiscale Modeling & Simulation*, 11(4):1149–1175, 2013.

[42] J. S. Hesthaven, B. Stamm, and S. Zhang. Efficient greedy algorithms for high-dimensional parameter spaces with applications to empirical interpolation and reduced basis methods. *ESAIM: Mathematical Modelling and Numerical Analysis*, 48(1):259–283, Jan. 2014.

[43] T. Y. Hou. A Multiscale Finite Element Method for Elliptic Problems in Composite Materials and Porous Media. *Journal of Computational Physics*, 134(1):21–21, May 1997.

[44] T. Y. Hou and X.-H. Wu. A multiscale finite element method for elliptic problems in composite materials and porous media. *Journal of Computational Physics*, 134(1):169–189, 1997.

[45] T. Y. Hou, X.-H. Wu, and Y. Zhang. Removing the cell resonance error in the multiscale finite element method via a Petrov-Galerkin formulation. *Communications in Mathematical Sciences*, 2(2):185–205, June 2004.

[46] T. J. R. Hughes. Multiscale phenomena: Green's functions, the Dirichlet-to-Neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods. *Computer Methods in Applied Mechanics and Engineering*, 127(1-4):387–401, 1995.

[47] T. J. R. Hughes, G. R. Feijóo, L. Mazzei, and J.-B. Quincy. The variational multiscale method—a paradigm for computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, 166(1-2):3–24, 1998.

[48] D. 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. *Comptes Rendus Mathematique*, 345(8):473–478, Dec. 2006.

[49] L. Iapichino. *Reduced Basis Methods for the Solution of Parametrized PDEs in Repetitive and Complex Networks with Application to CFD.* PhD thesis, École Polytechnique Fédérale de Lausanne, Lausanne, Oct. 2012.

[50] L. Iapichino, A. Quarteroni, and G. Rozza. A reduced basis hybrid method for the coupling of parametrized domains represented by fluidic networks. *Computer Methods in Applied Mechanics and Engineering*, 221:63–82, 2012.

[51] P. Jenny, S. H. Lee, and H. Tchelepi. Multi-scale finite-volume method for elliptic problems in subsurface flow simulations. *Journal of Computational Physics*, 187:47–67, 2003.

[52] I. T. Jolliffe. *Principal Component Analysis.* Springer, New York, Oct. 2002.

[53] M. Kahlbacher and S. Volkwein. Galerkin proper orthogonal decomposition methods for parameter dependent elliptic systems. *Discussiones Mathematicae. Differential Inclusions, Control and Optimization*, 27(1):95–117, 2007.

[54] S. Kaulmann. A Localized Reduced Basis Approach for Heterogeneous Multiscale Problems. Master's thesis, Westfälische Wilhelms Universität Münster, Einsteinstrasse 62, 48149 Münster, Mar. 2011.

[55] S. Kaulmann, B. Flemisch, B. Haasdonk, K. A. Lie, and M. Ohlberger. The localized reduced basis multiscale method for two-phase flows in porous media. *International Journal for Numerical Methods in Engineering*, 2014. to appear.

[56] S. Kaulmann and B. Haasdonk. Online Greedy Reduced Basis Construction Using Dictionaries. In J. P. B. Moitinho de Almeida, P. Díez, C. Tiago, and N. Parés, editors, *VI International Conference on Adaptive Modeling and Simulation (ADMOS 2013)*, pages 365–376, Lisbon, Portugal, May 2013.

[57] S. Kaulmann, M. Ohlberger, and B. Haasdonk. A new local reduced basis discontinuous Galerkin approach for heterogeneous multiscale problems. *Comptes Rendus Mathematique*, 349(23-24):1233–1238, Dec. 2011.

[58] D. Knezevic, D. B. P. Huynh, and A. T. Patera. A static condensation reduced basis element method: approximation and a posteriori error estimation. *ESAIM: Mathematical Modelling and Numerical Analysis*, 47:213–251, Jan. 2013.

[59] L. Machiels, Y. Maday, I. B. Oliveira, A. T. Patera, and D. V. Rovas. Output bounds for reduced-basis approximations of symmetric positive definite eigenvalue problems. *Comptes Rendus Mathematique*, 331(2):153–158, 2000.

[60] L. Machiels, Y. Maday, and A. T. Patera. Output bounds for reduced-order approximations of elliptic partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 190(26–27):3413–3426, 2001.

[61] Y. Maday and E. M. Rønquist. The reduced basis element method: Application to a thermal fin problem. *SIAM Journal on Scientific Computing*, 26(1):240–258, 2004.

[62] Y. Maday and B. Stamm. Locally adaptive greedy approximations for anisotropic parameter reduced basis spaces. *SIAM Journal on Scientific Computing*, 35(6):A2417–A2441, 2013.

[63] I. Martini, G. Rozza, and B. Haasdonk. Reduced basis approximation and a-posteriori error estimation for the coupled Stokes-Darcy system. *Advances in Computational Mathematics*, 2014. to appear.

[64] R. Milk, F. Schindler, A. Buhr, S. Kaulmann, and P. Henning. DUNE-Multiscale-super, June 2014. doi: 10.5281/zenodo.12281, Oct. 2014.

[65] O. Møyner and K.-A. Lie. A multiscale two-point flux-approximation method. *Journal of Computational Physics*, 275(0):273–293, 2014.

[66] J. R. Natvig, K.-A. Lie, B. Eikemo, and I. Berre. A discontinuous Galerkin method for computing single-phase flow in porous media. *Advances in Water Resources*, 30(12):2424–2438, 2007.

[67] N.-C. Nguyen, K. Veroy, and A. T. Patera. Certified real-time solution of parametrized partial differential equations. In *Handbook of Materials Modeling*, pages 1529–1564. Springer Netherlands, Dordrecht, 2005.

[68] M. Ohlberger. Error control based model reduction for multiscale problems. In A. Handlovičová, Z. Minarechová, and D. Ševčovič, editors, *Algoritmy 2012*, pages 1–10. Slovak University of Technology in Bratislava, Apr. 2012.

[69] A. T. Patera and G. Rozza. Reduced basis approximation and a posteriori error estimation for parametrized partial differential equations. *Version 1.0, Copyright MIT 2006, to appear in (tentative rubric) MIT Pappalardo Graduate Monographs in Mechanical Engineering*, 2006.

[70] D. Peterseim, P. Henning, and P. Morgenstern. Multiscale partition of unity. arxiv:1312.5922 [math.na], arXiv.org, Dec. 2013.

[71] C. Prud'homme, D. V. Rovas, K. Veroy, L. Machiels, Y. Maday, A. T. Patera, and G. Turinici. Reliable real-time solution of parametrized partial differential equations: Reduced-basis output bound methods. *Journal of Fluids Engineering-Transactions of the Asme*, 124(1):70–80, Mar. 2002.

[72] C. Prud'homme, D. V. Rovas, K. Veroy, and A. T. Patera. A mathematical and computational framework for reliable real-time solution of parametrized partial differential equations. *ESAIM: Mathematical Modelling and Numerical Analysis*, 36(5):747–771, 2002.

[73] B. Rivière. *Discontinuous Galerkin Methods for Solving Elliptic and Parabolic Equations: Theory and Implementation*. SIAM, Philadelphia, 2008.

[74] D. V. Rovas, L. Machiels, and Y. Maday. Reduced-basis output bound methods for parabolic problems. *IMA Journal of Numerical Analysis*, 26(3):423–445, July 2006.

[75] F. Schindler and M. Ohlberger. A-posteriori error estimates for the localized reduced basis multi-scale method. arxiv:1401.7173 [math.na], arXiv.org, Jan. 2014.

[76] K. Smetana. A new certification framework for the port reduced static condensation reduced basis element method. *Technical Report 64, MIT (USA), MechE*, 2014. Submitted to Computer Methods in Applied Mechanics and Engineering.

[77] V. Temlyakov. *Greedy approximation*, volume 20 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 2011.

[78] K. Urban, S. Volkwein, and O. Zeeb. Greedy sampling using nonlinear optimization. In A. Quarteroni and G. Rozza, editors, *Reduced Order Methods for Modeling and Computational Reduction*, pages 137–157. Springer International Publishing, 2014.

[79] K. Veroy, C. Prud'homme, D. V. Rovas, and A. T. Patera. A posteriori error bounds for reduced-basis approximation of parametrized noncoercive and nonlinear elliptic partial differential equations. In *16th AIAA Computational Fluid Dynamics Conference*, Orlando, Florida, June 2003. American Institute of Aeronautics and Astronautics.

Selbstständigkeitserklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die aus anderen Quellen entnommenen Stellen als solche gekennzeichnet habe.

Freiburg, den 02.02.2015

Sven Alebrand (geb. Kaulmann)