Westfälische
Wilhelms-Universität
Münster

A Localized Reduced Basis Approach for Heterogeneous Multiscale Problems

Diplomarbeit

wissen.leben
WWU Münster

Vorgelegt von Sven Kaulmann
Betreut von Prof. Dr. Mario Ohlberger
März 2011

Danksagung

> *All die Jahre, all die Stunden,*
> *all die Tage und Sekunden,*
> *auf dass die Zeit, die mir verbleibt,*
> *mich noch einmal zu dir treibt...*

Ich möchte diese Gelegenheit nutzen, mich bei Herrn Prof. Dr. Mario Ohlberger zu bedanken, der mir bereits während meines Studiums, speziell aber während der Erstellung dieser Arbeit, immer ein freundlicher und hilfsbereiter Ansprechpartner war. Mein großer Dank gilt ihm auch für die vielen neuen Impulse zur Thematik, die sehr zum Gelingen dieser Arbeit beigetragen haben.

Des Weiteren bedanke ich mich bei Herrn Dipl.-Math. Martin Drohmann und Herrn Dipl.-Math. Felix Albrecht für die vielen hilfreichen und aufmunternden Diskussionen und auch für das Korrekturlesen der Arbeit.

Besonders möchte ich meinen Eltern danken, die mir durch ihre vielfältige Unterstützung mein Studium ermöglicht haben.

Auch meiner guten Freundin Antonia Hombach gilt mein Dank. Sie hat mich in vielen Phasen meines Studiums getragen und motiviert und damit wesentlich zu dessen Gelingen beigetragen.

Contents

List of Figures

1 Introduction

Many real world problems give rise to mathematical models coined by data functions that behave differently on different length scales. On a small scale they show – possibly periodic – oscillations of different strength. On larger scales they show a totally different behaviour, a linear one for example. Discretization of these kinds of models can be a very cumbersome task, at least when a large physical domain has to be examined.

Typical challenges that arise in this context are memory shortage and unfeasible runtimes. Using standard discretization methods such as Finite Element (FE), Finite Volume (FV) or Discontinuous Galerkin (DG) methods, one usually runs into one or even both of them, even on up-to-date high-performance computers. In particular the large runtime becomes an issue when the application is embedded into a so-called many-query or real-time context, that is: the problem has to be solved multiple times, as for parameter studies, or for a given set of model data, simulation data is needed rapidly, as for flood prediction.

One typical example for this kind of application are two-phase flows in porous media and especially groundwater flow problems. Here, the permeability of the porous medium usually shows fluctuations on a scale of centimeters to meters, while the computational domain is spread over kilometers. Additionally, the quantities of interest, the pressure and the saturation, are characterized by a coupled system of partial differential equations that is solved iteratively. Thus, the equation comprising the troublesome coefficient has to be solved numerous times, such that we immediately find ourselves in the problem area described above.

Other applications that fit in this framework are, for example, questions that deal with composite materials and oil production.

In the work at hand, we will concentrate on two-phase flow problems as application. Nevertheless, our ansatz can be adopted to different applications as the main ideas and concepts are not restricted to this particular case.

Background

The interest in these kinds of multiscale problems and the amount of work in this field has been huge in the last decades. We would like to give a short – naturally incomplete – overview of the different approaches, especially those that motivated and influenced this work.

With the Multiscale Finite Element Method (MSFEM), Hou and co-workers [EH07, HWC99, HW97] introduced the idea of using local fine scale solutions as basis functions of a FE scheme on a coarse mesh. Here, the focus mainly lies in overcoming the extensive need for memory.

In a recent work, Aarnes and co-workers [AEJ08] continued this idea with their Mixed Multiscale Finite Element Method (MMFEM) using limited global information that incorporates information about the solution on both small and large scale into the base functions. As this approach necessitates the computation of global solutions in the construction of the base functions, it is only feasible in many-query or real-time contexts as those described above.

Further approaches for numerical multiscale problems are the Heterogenous Multiscale Method (HMM) [EEL$^+$07], the Variational Multiscale Method (VMM) [HS96,Hug95] and the Multiscale Finite Volume (MFV) method [JLT03]. The method to be introduced also bears a resemblance to the Reduced Basis Element method [MR02, MR04].

Apart from the multiscale field, the problems described above have another dimension: The many-query/real-time one. So far, we only gave a short impression of what has been done for the multiscale part. Still, the question of repeated or increased rapid solution for computationally expensive, so called Parametrized Partial Differential Equations (P^{2}DEs), has gained a lot of importance in the last decade, too. In this work, a P^{2}DE in its variational form denotes a partial differential equation of the form

$$a(u, v, \boldsymbol{\mu}) = b(v) \quad \forall v \in X \tag{1.1}$$

with a suitable Hilbert space X on a polygonally bounded convex domain $\Omega \subset \mathbb{R}^d$. The quantity $\boldsymbol{\mu} \in \mathcal{P}$ denotes a parameter vector stemming from a bounded parameter domain $\mathcal{P} \subset \mathbb{R}^{d_{\boldsymbol{\mu}}}$ and a and b are continuous bilinear and linear forms, respectively.

For these kinds of equations, the Reduced Basis (RB) method, originally introduced by Patera et al. [PR07], provides model order reduction by projection of the scheme of choice (FE,FV,...) onto a discrete space of a typically very small dimension, the *Reduced Basis Space*. The RB space is made up by the *reduced basis* which is a set of so-called *snapshots* which are – possibly orthonormalized – solutions to (1.1) using a discrete scheme with a high resolution (FE,FV, ...). This scheme will be denoted as the *high dimensional* one in the sequel. The scheme resulting from the projection step, in contrast, will be denoted as the *low dimensional* one.

Many different aspects of this method have been investigated. In particular worth mentioning are the questions of "optimal" choice of the reduced basis and a posteriori error estimation as this work will give new impulses on both ones.

The original *Greedy*-algorithm for basis construction (see [PR07], for example) selects parameter vectors $\boldsymbol{\mu}$ for the calculation of the snapshots using an a posteriori error estimator: It iteratively enlarges the RB space by evaluating the error estimator for all $\boldsymbol{\mu} \in \mathcal{P}_{\mathcal{D}}$ where $\mathcal{P}_{\mathcal{D}} \subset \mathcal{P}$ is a finite subset, and adding the snapshot for the parameter $\boldsymbol{\mu}$ that maximized the error bound to the reduced basis. This idea was expanded for time-dependent problems by Haasdonk and Ohlberger [HO08] using a Proper Orthogonal Decomposition (POD) to compress the information inherited in a whole trajectory into one single function. Most of the state-of-the-art RB methods rely on rapidly evaluable error estimators. Those are used in the construction of the RB space as well as later on in the low dimensional simulations. Most of these estimators compare the low dimensional solution to the matching high dimensional one and thus neglect the error introduced by

the high dimensional discretization. As they form crucial ingredients of the RB method, examples for such error estimators can be found in lots of publications dealing with the RB method, see the above-mentioned works by Patera et al. [PR07] or Haasdonk and Ohlberger [HO08] for example.

In this work we will use the DG discretization method. It provides large flexibility regarding the choice of the mesh and polynomial degrees which can be seen as a good advantage over the FE or FV method. For a good introduction to all different kinds of DG methods, we may redirect to the work of Arnold et al. [ABCM01] and to the book by M. Riviere [Riv08]. For this work, the so-called Symmetric Interior Penalty (SIP) method, stemming from the late 1970s and originally introduced by Wheeler [Whe78] and Arnold [Arn82], is of special interest. The SIP method yields a symmetric bilinear form – which is quite desirable in some cases – and enforces continuity of the solution by penalizing unsteadiness on inner intersections of the mesh. We will give more details on this method in Chapter 3.1.

Novelties in this Work

In this thesis, as mentioned already, we will deal with two-phase flow equations – as we neglect gravity and capillary effects – of the form

$$-\nabla \cdot (\lambda(S) k \nabla p) = f, \tag{1.2}$$

where p is the unknown pressure, $\lambda(S)$ denotes the total mobility, k is the – possibly oscillating – permeability and f a source term. The saturation S is given by

$$\partial_t S + \nabla \cdot (v f_w(S)) = 0, \tag{1.3}$$

where v and f_w are the total velocity and the fractional flow of one phase, respectively. As we will assume S, and thus λ to be dependent on a parameter $\boldsymbol{\mu}$, equation (1.2) describes a P^2DE. Details on the parameter dependence of this problem will be given in Section 2.2.

Using this problem as a hook, we will introduce a new discretization method, that brings together ideas from both worlds: We will use means from the multiscale community to handle scale separation in the coefficient k and a RB framework to handle the many-query nature of problem (1.2). In more detail, we will alter the MMFEM ansatz by constructing the basis functions from global fine scale solutions instead of local ones. These solutions will then be restricted to each entity of a coarser grid in turns, such that we end up with one base function per coarse grid cell. Doing this in turns, embedded into a Greedy algorithm, and compressing the information on each cell after each extension step using the POD algorithm, we obtain one set of base functions per coarse cell. The span of the collection of all these sets then forms the reduced basis.

As the compression step may produce different numbers of base functions on different coarse cells, we will need a scheme on the reduced basis that allows discontinuities over the edges of the coarse cells. We will therefore use an SIP scheme on the reduced space, denoted as the *reduced scheme*.

We will conclude the presentation of this new multiscale approach with some solvability analysis and an a posteriori error estimator, that we will deduce using duality techniques.

Finally, we will present some numerical results that demonstrate, that the new ansatz performs quite well for the stated problem (1.2)–(1.3).

Outline of the Thesis

The thesis is structured as follows: Chapter 2 is dedicated to the definition of the governing equation in analytical, weak and discrete forms. In Chapter 3, we introduce the new ansatz and provide solvability results for the resulting low dimensional scheme. Chapter 4 deals with error analysis: The above-named error estimator is introduced and proven. In Chapter 5, we give a short, incomplete overview of the implementation that was done. The numerical tests for all parts of the new scheme are to be found in Chapter 6.

2 The Model Problem

2.1 Strong Formulation

For a given Lipschitz-domain $\Omega \subset \mathbb{R}^d$ we are looking for $p \in \mathcal{C}^2(\overline{\Omega})$ such that for $f \in \mathcal{C}^0(\Omega)$ and sufficiently smooth mobility $\lambda \in L^\infty(\Omega)$ and permeability $k \in [L^\infty(\Omega)]^d$ it holds:

$$-\nabla \cdot (\lambda k \nabla p) = f \quad \text{in } \Omega, \tag{2.1}$$

$$p = g_D \quad \text{on } \partial\Omega, \tag{2.2}$$

where $g_D \in \mathcal{C}^0(\partial\Omega)$. We assume that there exist k_1, k_2 greater than zero, such that for all $\boldsymbol{a} \in \mathbb{R}^d$ we have

$$k_1 \boldsymbol{a} \cdot \boldsymbol{a} \leq \lambda(\boldsymbol{x}) k(\boldsymbol{x}) \boldsymbol{a} \cdot \boldsymbol{a} \leq k_2 \boldsymbol{a} \cdot \boldsymbol{a} \quad \forall \boldsymbol{x} \in \Omega, \tag{2.3}$$

where the last inequality is obvious due to the assumption on λ and k. Here and throughout the work we use bold faced letters for tensor quantities: lower case ones for vectors and upper case ones for matrices. We will denote the coordinates of a two-dimensional coordinate vector $\boldsymbol{x}$ by x and y.

2.2 Parameter Dependence

In the work at hand, as mentioned in the introduction, we assume the mobility λ to depend on a parameter vector $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_{N_\lambda})$ stemming from a bounded parameter domain $\mathcal{P} \subset \mathbb{R}^{N_\lambda}$. For the method to be introduced to be efficient, we require this dependency to take the form

$$\lambda(\boldsymbol{x}, \boldsymbol{\mu}) = \sum_{\nu=1}^{N_\lambda} \mu_\nu \lambda_\nu(\boldsymbol{x}), \tag{2.4}$$

with parameter independent functions λ_ν.

One example of a possible λ, which will also be used in the numerical experiments in Chapter 6, is the total mobility in a two-phase flow setting for oil and water phases. Here, for a given saturation S one usually sets

$$\lambda(\boldsymbol{x}) = \lambda_w(S(\boldsymbol{x})) + \lambda_o(S(\boldsymbol{x})) \tag{2.5}$$

$$= \frac{k_{rw}(S(\boldsymbol{x}))}{\eta_w} + \frac{k_{ro}(S(\boldsymbol{x}))}{\eta_o} \tag{2.6}$$

$$= \frac{S(\boldsymbol{x})^2}{\eta_w} + \frac{(1 - S(\boldsymbol{x}))^2}{\eta_o}, \tag{2.7}$$

where k_{rw} and k_{ro} are the relative permeabilities of water and oil phases correspondingly and η_w and η_o are the viscosities of the water and oil phase correspondingly.

Now, if we assume S to take the form (2.4):

$$S(\boldsymbol{x}, \boldsymbol{\mu}) = \sum_{\nu=1}^{N_S} \mu_\nu S_\nu(\boldsymbol{x}), \tag{2.8}$$

$\lambda(\boldsymbol{x}, \boldsymbol{\mu})$ can also be written as a linear combination of products of parameter dependent and parameter independent parts:

$$\begin{aligned}\lambda_w(\boldsymbol{x}, \boldsymbol{\mu}) &= \frac{1}{\eta_w} S(\boldsymbol{x}, \boldsymbol{\mu})^2 \\ &= \frac{1}{\eta_w} \sum_{m=1}^{N_S} \sum_{n=1}^{N_S} \mu_n \mu_m S_n(\boldsymbol{x}) S_m(\boldsymbol{x}),\end{aligned}$$

which obviously is of the form (2.4) and together with

$$\lambda_o(\boldsymbol{x}, \boldsymbol{\mu}) = \frac{1}{\eta_o} - \frac{2}{\eta_o} S(\boldsymbol{x}, \boldsymbol{\mu}) + \frac{\eta_w}{\eta_o} \lambda_w(\boldsymbol{x}, \boldsymbol{\mu})$$

provides the desired shape for λ:

$$\lambda(\boldsymbol{x}, \boldsymbol{\mu}) = \frac{1}{\eta_o} - \frac{2}{\eta_o} S(\boldsymbol{x}, \boldsymbol{\mu}) + \frac{\eta_o + \eta_w^2}{\eta_w \eta_o} \sum_{m=1}^{N_S} \sum_{n=1}^{N_S} \mu_n \mu_m S_n(\boldsymbol{x}) S_m(\boldsymbol{x}). \tag{2.9}$$

Remark 2.2.1. This choice of S could, for example, make sense to simulate the flooding of the domain by water. The functions S_ν would then be (regularized) indicator functions for different parts of the domain as indicated in Figure 2.1 and the parameter vector $\boldsymbol{\mu}$ would control which part of the domain is flooded.

In addition to the mobility λ, we will allow the boundary data to be dependent on the parameter, too. We assume g_D to take the form

$$g_D(\boldsymbol{x}, \boldsymbol{\mu}) = \sum_{\eta=1}^{N_\lambda} \mu_\eta g_\eta(\boldsymbol{x}) \tag{2.10}$$

with parameter independent functions g_ν.

Remark 2.2.2. For the rest of the work, in order to keep the notation clearer, we will suppress $\boldsymbol{\mu}$ and $\boldsymbol{x}$ in $\lambda(\boldsymbol{x}, \boldsymbol{\mu})$ whenever possible.

2.3 Weak Formulation

Definition 2.3.1 (Weak formulation)**.** Let $g_D \in H^{\frac{1}{2}}(\partial\Omega)$. Then, by the trace theorem, there exists $\overline{g_D} \in H^1(\Omega)$, $\overline{g_D} = g_D$ on $\partial\Omega$. Define

$$\begin{aligned}b(v, w) &= \int_\Omega \lambda k \nabla v \nabla w, \\ l(v) &= \int_\Omega f v - \int_\Omega \lambda k \nabla \overline{g_D} \nabla v.\end{aligned}$$

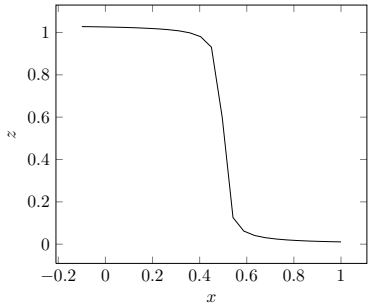

Figure 2.1: Regularized indicator function for the left part of the domain

We are then looking for $p_0 \in H_0^1(\Omega)$, such that

$$b(p_0, v) = l(v) \quad \forall v \in H_0^1(\Omega) \tag{2.11}$$

and get the weak solution of problem (2.1) as $p = \overline{g_D} + p_0$.

Remark 2.3.2. The weak solution p from Definition (2.11) fulfils

$$\int_\Omega \lambda k \nabla p \nabla v = \int_\Omega f v \quad \forall v \in H_0^1(\Omega).$$

2.4 Finite Element Formulation

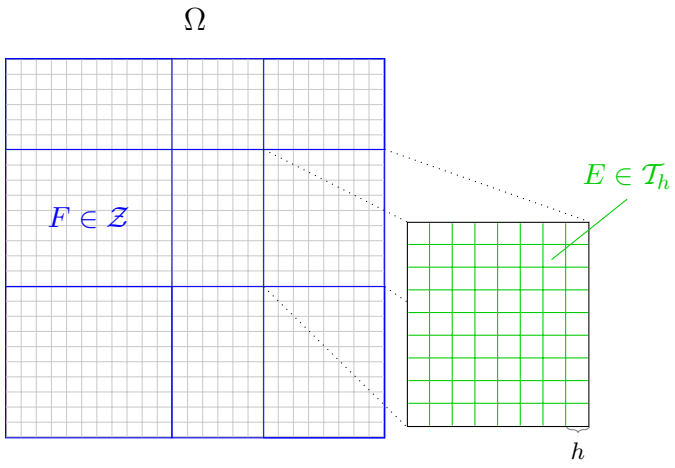

Figure 2.2: The two grids

The first step towards a discrete formulation of problem (2.11) is to define a grid. In addition to the grid $\mathcal{T}_h$ which we need for the FE method, we define a coarser grid $\mathcal{Z}$:

Definition 2.4.1 (Grid)**.** Let $\mathcal{T}_h$ be a conforming triangulation of the domain Ω. Denote the inner entities of codimension 1 of $\mathcal{T}_h$ by Γ_I, the boundary ones by Γ_B and the sum of all codim 1 entities by Γ. With each element $e \in \Gamma$ we associate a unique unit normal vector $\boldsymbol{n}_e$. Furthermore, let $h_E = \operatorname{diam}(E)$ be the diameter of element $E \in \mathcal{T}_h$ and $h = \max_{E \in \mathcal{T}_h} h_E$ be the mesh size of $\mathcal{T}_h$. We assume

$$\forall E \in \mathcal{T}_h : \forall e \in \partial E : |e| \leq h_E^{d-1} \leq h^{d-1}, \tag{2.12}$$

where d, as always in the sequel, denotes the world dimension.

In addition, we define a second grid $\mathcal{Z}$ whose codimension 1 entities match with certain intersections of $\mathcal{T}_h$ and whose cells enclose more than one cell of $\mathcal{T}_h$ as indicated in Figure 2.2. The sum of all intersections of cells in $\mathcal{Z}$ will be denoted by Ξ, the inner intersections by Ξ_I and the boundary ones by Ξ_B.

The first discrete formulation of problem (2.1) introduced here is a FE formulation. We introduce the discrete function spaces

$$\begin{aligned} S_{h,k} &= \left\{ v \in \mathcal{C}^0(\Omega) | v_{|E} \in \mathbb{P}_k(E) \; \forall E \in \mathcal{T}_h \right\}, \\ S_{h,k}^0 &= S_{h,k} \cap \{ v \in \mathcal{C}^0(\overline{\Omega}) |\; v|_{\partial\Omega} = 0 \}. \end{aligned}$$

Then the Finite Element formulation of (2.11) is:

Definition 2.4.2 (Finite Element Solution)**.** Let $p_{h,0} \in S_{h,1}^0$, be the solution of

$$b(p_{h,0}, v) = l(v) \quad \forall v \in S_{h,1}^0. \tag{2.13}$$

Then the Finite Element solution of (2.11) if given as $p_h = p_{h,0} + \overline{g_D}$.

3 The Localized Reduced Basis Scheme

3.1 Non-Discrete Discontinuous Galerkin Formulation

As described in the introduction (see Chapter 1), we may like to use a different number of base functions on each coarse cell, which may lead to discontinuities over intersections. Therefore the next important step is to find a variational formulation of the model problem (2.1) that allows discontinuities over intersections. As mentioned above, we choose to use a SIP method that provides the possibility to penalize the discontinuities and has the favorable quality of symmetry. For further examples and general analysis on this method we refer to the above-mentioned introductions to the DG method [ABCM01],[Riv08] and [Whe78].

We start with the definition of a new function space $H^k(\mathcal{T}_h)$ that allows discontinuities over element faces:

$$H^k(\mathcal{T}_h) = \left\{ v \in L^2(\Omega) \,\middle|\, v|_E \in H^k(E)\ \forall E \in \mathcal{T}_h \right\}.$$

A function $v \in H^k(\mathcal{T}_h)$ has a well defined trace along each intersection of a single element. Nevertheless, v may have two different traces on intersections e that are shared by two elements E_1, E_2. In such cases we define the mean value and jump of v on e:

$$\{v\} = \frac{1}{2}\, v|_{E_1} + \frac{1}{2}\, v|_{E_2}\,, \quad [v] = v|_{E_1} - v|_{E_2}\,, \tag{3.1}$$

if the unit normal vector $\boldsymbol{n}_e$ is oriented from E_1 to E_2.

For $e \in \Gamma_B$ we define

$$\{v\} = v, \quad [v] = v. \tag{3.2}$$

The jump and mean value fulfil the following equations that can be proven easily:

Lemma 3.1.1. *For $e = E_1 \cap E_2 \in \Gamma_I, v \in H^2(\mathcal{T}_h)$, the following hold:*

1. *If $v \in \mathcal{C}^0(E_1 \cup E_2)$: $[v] = 0$, $\{v\} = v$*
2. *If $w \in H^2(\mathcal{T}_h)$: $[vw] = [v]\{w\} + \{v\}[w]$.*

Definition 3.1.2 (Bilinear Form and Right Hand Side)**.** We now introduce the bilinear form

$$\begin{aligned} B_{\mathrm{DG}}(v,w) = \sum_{E\in\mathcal{T}_h} \int_E \lambda k \nabla v \cdot \nabla w - \sum_{e\in\Gamma} \int_e \{\lambda k \nabla v \cdot \boldsymbol{n}_e\}[w] \\ - \sum_{e\in\Gamma} \int_e \{\lambda k \nabla w \cdot \boldsymbol{n}_e\}[v] + J_{\sigma_J,\beta_J}(v,w), \end{aligned} \tag{3.3}$$

where the bilinear form

$$J_{\sigma_J,\beta_J}(v,w) = \sum_{e\in\Gamma} \frac{\sigma_J^e}{|e|^{\beta_J}} \int_e [v][w], \tag{3.4}$$

with $\sigma_J^e > 0$, $\beta_J > 0$ stated later, penalizes unsteadiness of the functions over intersections.

Additionally, let

$$L(v) = \sum_{E\in\mathcal{T}_h} \int_E fv + \sum_{e\in\Gamma_B} \int_e \left(\frac{\sigma_J^e}{|e|^{\beta_J}} v - \lambda k \nabla v \cdot \boldsymbol{n} \right) g_D. \tag{3.5}$$

We then obtain the non-discrete Discontinuous Galerkin formulation of problem (2.1): Find $p \in H^2(\mathcal{T}_h)$, such that

$$B_{\mathrm{DG}}(p,v) = L(v) \quad \forall v \in H^2(\mathcal{T}_h). \tag{3.6}$$

Remark 3.1.3. In the following theorem, we will show consistency of the DG formulation just introduced. Its proof is oriented on the standard proofs for Interior Penalty methods, see [Riv08] for example.

Theorem 3.1.4 (Consistency)**.** *Every weak solution of problem (2.1) with $p \in H^2(\Omega)$ is a solution of the non-discrete formulation (3.6).*

Proof. Let p be a weak solution of (2.1), that is:

$$\int_\Omega \lambda k \nabla p \cdot \nabla v = \int_\Omega fv \quad \forall v \in H_0^1(\Omega).$$

If we now integrate by parts and restrict the arising equation to one element $E \in \mathcal{T}_h$ we get

$$\int_E -\nabla \cdot (\lambda k \nabla p) v = \int_E fv.$$

This equation also holds for test functions v with $v \neq 0$ on $\partial\Omega$ (for $E \cap \partial\Omega = \emptyset$ this is obvious, for the other case, one should consider, that the boundary of E has zero measure). As the behaviour of v outside of E does not influence the equation and as $H^2(E) \subset H^1(E)$, we may hence switch to $v \in H^2(\mathcal{T}_h)$. Application of Green's theorem for weakly differentiable functions (see [Alt06], Theorem A6.8) yields

$$\int_E \lambda k \nabla p \cdot \nabla v - \int_{\partial E} \lambda k \nabla p \cdot \boldsymbol{n}_E v = \int_E fv.$$

By summation over all elements we get

$$\sum_{E\in\mathcal{T}_h} \int_E \lambda k \nabla p \cdot \nabla v - \sum_{E\in\mathcal{T}_h} \int_{\partial E} \lambda k \nabla p \cdot \boldsymbol{n}_E v = \int_\Omega fv. \tag{3.7}$$

With the definition (3.2) of the jump of the function v we can write

$$\sum_{E\in\mathcal{T}_h}\int_{\partial E}\lambda k\nabla p\cdot\boldsymbol{n}_E v=\sum_{e\in\Gamma_I}\int_e[\lambda k\nabla p\cdot\boldsymbol{n}_e v]+\sum_{e\in\Gamma_B}\int_e\lambda k\nabla p\cdot\boldsymbol{n}_e v,$$

where we switched from the normals $\boldsymbol{n}_E$ to $\boldsymbol{n}_e$ that coincide on each codimension 1 entity of each element E.
Following Lemma 3.1.1, we can write

$$[\lambda k\nabla p\cdot\boldsymbol{n}_e v]=[\lambda k\nabla p\cdot\boldsymbol{n}_e]\{v\}+\{\lambda k\nabla p\cdot\boldsymbol{n}_e\}[v].$$

As $p\in H^2(\Omega)$ we get with Green's theorem that $\lambda k\nabla p\cdot\boldsymbol{n}_e$ is uniquely defined on every $e\in\Gamma_I$ and thus

$$[\lambda k\nabla p\cdot\boldsymbol{n}_e]=0\quad\text{weak on every } e\in\Gamma_I. \tag{3.8}$$

Therefore

$$\sum_{E\in\mathcal{T}_h}\int_{\partial E}\lambda k\nabla p\cdot\boldsymbol{n}_E v=\sum_{e\in\Gamma_I}\int_e\{\lambda k\nabla p\cdot\boldsymbol{n}_e\}[v]+\sum_{e\in\Gamma_B}\int_e\lambda k\nabla p\cdot\boldsymbol{n}_e v.$$

Now, we get from (3.7)

$$\begin{aligned}&\sum_{E\in\mathcal{T}_h}\int_E\lambda k\nabla p\cdot\nabla v-\sum_{e\in\Gamma_I}\int_e\{\lambda k\nabla p\cdot\boldsymbol{n}_e\}[v]-\sum_{e\in\Gamma_B}\int_e\lambda k\nabla p\cdot\boldsymbol{n}_e v\\&=\int_\Omega fv.\end{aligned}$$

We now add $-\sum_{e\in\Gamma_B}\int_e\lambda k\nabla v\cdot\boldsymbol{n}_e p$ and $\sum_{e\in\Gamma_B}\frac{\sigma_j^e}{|e|^\beta}\int_e pv$ to both sides and get

$$\begin{aligned}&\sum_{E\in\mathcal{T}_h}\int_E\lambda k\nabla p\cdot\nabla v-\sum_{e\in\Gamma_I}\int_e\{\lambda k\nabla p\cdot\boldsymbol{n}_e\}[v]-\sum_{e\in\Gamma_B}\int_e\lambda k\nabla p\cdot\boldsymbol{n}_e v\\&\qquad-\sum_{e\in\Gamma_B}\int_e\lambda k\nabla v\cdot\boldsymbol{n}_e p+\sum_{e\in\Gamma_B}\frac{\sigma_j^e}{|e|^\beta}\int_e pv\\&=\int_\Omega fv-\sum_{e\in\Gamma_B}\int_e\lambda k\nabla v\cdot\boldsymbol{n}_e p+\sum_{e\in\Gamma_B}\frac{\sigma_j^e}{|e|^\beta}\int_e pv.\end{aligned}$$

Because of $[w]=w$ and $\{w\}=w$ on Γ_B (see (3.2)) and $[p]=0$ on Γ_I due to the regularity of p, the left side equals $B_{\mathrm{DG}}(p,v)$. As $p=g_D$ on Γ_B, the right side equals $L(v)$ and thus

$$B_{\mathrm{DG}}(p,v)=L(v)\quad\forall v\in H^2(\mathcal{T}_h),$$

that is: p is solution to the non-discrete Discontinuous Galerkin formulation. □

3.2 Discrete Discontinuous Galerkin Formulation

We get a discrete DG formulation by restricting the non-discrete DG formulation (3.6) to a finite dimensional function space. In our case, following the Reduced Basis idea, this function space shall be spanned by solutions of the main problem or a more simple problem with a similar structure. By this we achieve, that the ansatz space already comprises some information about the characteristics of the possible solutions to the problem.

The first step of the construction of the ansatz space is to choose M parameter vectors $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_M$ and solve the main equation with a finite element method, that is perform (2.13). The respective solutions shall be denoted by $p_1, \dots, p_M$.

Next, by restriction of $p_1, \dots, p_M$ to the cells $F \in \mathcal{Z}$ we define sets of functions

$$\widetilde{W}_M^F = \{\varphi_i = p_i|_F \,|i = 1, \dots, M\}.$$

From these local sets we now construct the local ansatz spaces using a Proper Orthogonal Decomposition (POD):

$$W_{M_F}^F = \left\langle \left\{\varphi_1^F, \dots, \varphi_{M_F}^F\right\}\right\rangle = \left\langle \mathrm{POD}\left(\left\{\varphi \;\middle|\; \varphi \in \widetilde{W}_M^F\right\}\right)\right\rangle,$$

where $\langle\cdot\rangle$ denotes the span of a given set.

Remark 3.2.1. The details of the POD are not subject to this work, the interested reader may be referred to [Jol02] for a good description of the method.

Now, the ansatz space for the discrete Discontinuous Galerkin scheme is finally given as

$$W_N = \bigcup_{F \in \mathcal{Z}} W_{M_F}^F, \tag{3.9}$$

where $N = \dim(W_N) = \sum_{F \in \mathcal{Z}} M_F$.

We may note here, that the construction of W_N will be explained in greater detail in Section 3.3.

A function $v \in W_N$ can now be written as

$$v = \sum_{F \in \mathcal{Z}} \sum_{i=1}^{M_F} a_i^F \varphi_i^F, \tag{3.10}$$

where $a_i^F \in \mathbb{R}$ for all $F \in \mathcal{Z}$, $i \in \{1, \dots, M_F\}$. Here φ_i^F is defined to be zero outside of F.

If $\{\varphi_1, \dots, \varphi_N\}$ is the base of W_N, given by merging the local bases $\{\varphi_1^F, \dots, \varphi_{M_F}^F\}$ of the spaces $W_{M_F}^F$, we can simplify this representation: Every $v \in W_N$ can then be written as

$$v = \sum_{i=1}^{N} a_i \varphi_i, \tag{3.11}$$

with $a_i \in \mathbb{R}\ \forall i \in \{1, \dots, N\}$.

The discrete DG scheme now reads as follows: Find $p_N \in W_N$, such that

$$B_{\mathrm{DG}}(p_N, v) = L(v) \quad \forall v \in W_N, \tag{3.12}$$

or, equivalently, find $p_N \in W_N$, such that

$$B_{\mathrm{DG}}(p_N, \varphi_i) = L(\varphi_i) \quad \forall i \in \{1, \dots, N\}. \tag{3.13}$$

Using the representation (3.11) of p_N, $\lambda = \sum_{\nu=1}^{N_\lambda} \mu_\nu \lambda_\nu$ and $g_D = \sum_{\eta=1}^{N_\lambda} \mu_\eta g_\eta$, the coefficient vector $\boldsymbol{a} = (a_1, \dots, a_N)^\top$ fulfils

$$\left(\sum_{\nu=1}^{N_\lambda} \mu_\nu \left(\boldsymbol{A}^\nu - \boldsymbol{B}^\nu - (\boldsymbol{B}^\nu)^\top \right) + \boldsymbol{C} \right) \cdot \boldsymbol{a} = \boldsymbol{b} + \sum_{\nu=1}^{N_\lambda} \mu_\nu c^\nu + \sum_{\nu=1}^{N_\lambda} \sum_{\eta=1}^{N_\lambda} \mu_\nu \mu_\eta \boldsymbol{d}^{\nu,\eta} \tag{3.14}$$

with

$$\begin{aligned}
(\boldsymbol{A}^\nu)_{ij} &= \sum_{E \in \mathcal{T}_h} \int_E \lambda_\nu(x) k \nabla \varphi_i \cdot \nabla \varphi_j, \quad (i,j) \in \{1, \dots, N\}^2 \\
(\boldsymbol{B}^\nu)_{ij} &= \sum_{e \in \Gamma} \int_e \{\lambda_\nu(x) k \nabla \varphi_i \cdot \boldsymbol{n}\} [\varphi_j], \quad (i,j) \in \{1, \dots, N\}^2 \\
(\boldsymbol{C})_{ij} &= \sum_{e \in \Gamma} \frac{\sigma^e}{|e|^\beta} \int_e [\varphi_i][\varphi_j], \quad (i,j) \in \{1, \dots, N\}^2 \\
(\boldsymbol{b})_i &= \int_\Omega f \varphi_i \\
(\boldsymbol{c}^\eta)_i &= \sum_{e \in \Gamma_B} \int_e \frac{\sigma^e}{|e|^\beta} \varphi_i g_\eta, \quad i \in \{1, \dots, N\} \\
(\boldsymbol{d}^{\nu,\eta})_i &= - \sum_{e \in \Gamma_B} \int_e \lambda_\nu k \nabla \varphi_i \cdot \boldsymbol{n}\, g_\eta \quad i \in \{1, \dots, N\}.
\end{aligned} \tag{3.15}$$

Remark 3.2.2. In the sequel we will also use coefficient vectors $\boldsymbol{a}^F \in \mathbb{R}^{M_F}$, that shall be made up of the coefficients to all base functions on the coarse cell F. This will prove to be advantageous when we use the description (3.10) of a function $v \in W_N$.

3.2.1 Properties

In the following, we will prove some important properties of the discrete DG scheme.

For the proof of existence and uniqueness of a solution to the discrete DG scheme we will use the following energy norm on W_N:

$$\|v\|_{\mathcal{E}} = \left(\sum_{E \in \mathcal{T}_h} \int_E \lambda k \nabla v \cdot \nabla v + J_{\sigma_J, \beta_J}(v, v) \right)^{\frac{1}{2}}. \tag{3.16}$$

Remark 3.2.3. As it will turn out to be convenient in the development of error estimators, we will prove the following results for $v \in H^2(\mathcal{T}_h)$. This, together with $W_N \subset H^2(\mathcal{T}_h)$, yields the corresponding results for $v \in W_N$.

Lemma 3.2.4. *$\|\cdot\|_{\mathcal{E}}$ defines a norm on $H^2(\mathcal{T}_h)$.*

Proof. Positive homogeneity: This is obvious.
Subadditivity: We have

$$\|v+w\|_{\mathcal{E}}^2 = \|v\|_{\mathcal{E}}^2 + \|w\|_{\mathcal{E}}^2 + 2\left(\sum_{E\in\mathcal{T}_h}\int_E \lambda k\nabla v\cdot\nabla w + \sum_{e\in\Gamma}\frac{\sigma_J^e}{|e|^{\beta_J}}\int_e [v][w]\right).$$

Using the Cauchy-Schwarz and l^1-l^2–Hölder inequality we have

$$\begin{aligned}
&\sum_{E\in\mathcal{T}_h}\int_E \lambda k\nabla v\cdot\nabla w + \sum_{e\in\Gamma}\frac{\sigma_J^e}{|e|^{\beta_J}}\int_e [v][w] \\
&\le \sum_{E\in\mathcal{T}_h}\|\sqrt{\lambda k}\nabla v\|_{0,E}\|\sqrt{\lambda k}\nabla w\|_{0,E} + \sum_{e\in\Gamma}\frac{\sigma_J^e}{|e|^{\beta_J}}\|[v]\|_{0,e}\|[w]\|_{0,e} \\
&\le \left(\sum_{E\in\mathcal{T}_h}\|\sqrt{\lambda k}\nabla v\|_{0,E}^2\right)^{1/2}\left(\sum_{E\in\mathcal{T}_h}\|\sqrt{\lambda k}\nabla w\|_{0,E}^2\right)^{1/2} \\
&\quad + \left(\sum_{e\in\Gamma}\frac{\sigma_J^e}{|e|^{\beta_J}}\|[v]\|_{0,e}^2\right)^{1/2}\left(\sum_{e\in\Gamma}\frac{\sigma_J^e}{|e|^{\beta_J}}\|[w]\|_{0,e}^2\right)^{1/2}.
\end{aligned}$$

Now use $\sqrt{ab}+\sqrt{cd} \le \sqrt{a+c}\cdot\sqrt{b+d}$ for positive real values a,b,c,d. We then have

$$\sum_{E\in\mathcal{T}_h}\int_E \lambda k\nabla v\cdot\nabla w + \sum_{e\in\Gamma}\frac{\sigma_J^e}{|e|^{\beta_J}}\int_e [v][w] \le \|v\|_{\mathcal{E}}\|w\|_{\mathcal{E}}, \tag{3.17}$$

which yields

$$\begin{aligned}
\|v+w\|_{\mathcal{E}}^2 &\le \|v\|_{\mathcal{E}}^2 + \|w\|_{\mathcal{E}}^2 + 2\|v\|_{\mathcal{E}}\|w\|_{\mathcal{E}} \\
&= \left(\|v\|_{\mathcal{E}} + \|w\|_{\mathcal{E}}\right)^2,
\end{aligned}$$

as desired.
Positive definiteness: Let $v \in H^2(\mathcal{T}_h)$ be an arbitrary vector with $\|v\|_{\mathcal{E}} = 0$. As both summands on the right side of (3.16) are non-negative we can conclude:

$$0 = \int_E \lambda k\nabla v\cdot\nabla v \ge k_1\int_E \nabla v\cdot\nabla v = \underbrace{k_1}_{>0}\|\nabla v\|_{0,E}^2 \quad \forall E\in\mathcal{T}_h$$

and therefore $v = \text{const}$ a.e. in E. The second term now yields

$$\sum_{e\in\Gamma_I}\frac{\sigma_J^e}{|e|^{\beta_J}}\int_e [v]^2 = 0$$

and

$$\sum_{e\in\Gamma_B} \frac{\sigma_J^e}{|e|^{\beta_J}} \int_e v^2 = 0.$$

The first equation, together with $\sigma_J^e > 0$, implies that the jumps over all intersections are zero, that is: v is constant in Ω. Finally, the second equation implies $\|v\|_{0,e} = 0$ which, together with the Sobolev embedding theorem for Lipschitz-domains and $v \in H^2(E)$ means $v = 0$ in Ω. □

Next, we would like to prove coercivity for the bilinear form B_{DG} in the energy norm. This will lead us directly to an existence and uniqueness result. As it will turn out to be convenient in the sequel, we prove the coercivity on a more general space than W_N. Therefore we introduce the spaces $V_{k,h}$:

$$V_{k,h} = \left\{ v \in L^2(\Omega) \,|\, v|_E \in \mathbb{P}_k(E) \forall E \in \mathcal{T}_h \right\}. \tag{3.18}$$

Remark 3.2.5. As for the consistency of the DG formulation, 3.1.4, the following three results and their proofs are oriented on the standard results as given in most of the literature dealing with SIP methods, see [Riv08, ABCM01, BMM$^+$99] for example.

Theorem 3.2.6 (Coercivity)**.** *Let σ_J^e be bounded from below by a sufficiently large constant and $\beta_J(d-1) \geq 1$. The bilinear form B_{DG} then is coercive on the space of piecewise linear functions $V_{1,h}$, that is: There exists a positive constant κ, such that*

$$B_{DG}(v,v) \geq \kappa \|v\|_{\mathcal{E}}^2 \quad \forall v \in V_{1,h}.$$

Proof. Using the Cauchy-Schwarz inequality we can deduce:

$$\sum_{e\in\Gamma} \int_e \{\lambda k \nabla v \cdot \boldsymbol{n}_e\} [v] \leq \sum_{e\in\Gamma} \left(\frac{1}{|e|^{\beta_J}} \right)^{\frac{1}{2}-\frac{1}{2}} \| \{\lambda k \nabla v \cdot \boldsymbol{n}_e\} \|_{0,e} \cdot \|[v]\|_{0,e}.$$

Now, let e be the intersection of two elements E_1^e, E_2^e, then we have

$$\|\{\lambda k \nabla v \cdot \boldsymbol{n}_e\}\|_{0,e} \leq \frac{1}{2} \| \left(\lambda k \nabla v \cdot \boldsymbol{n}_e\right)|_{E_1^e} \|_{0,e} + \frac{1}{2} \| \left(\lambda k \nabla v \cdot \boldsymbol{n}_e\right)|_{E_2^e} \|_{0,e},$$

and, using (2.3),

$$\|\{\lambda k \nabla v \cdot \boldsymbol{n}_e\}\|_{0,e} \leq \frac{1}{2} k_2 \left(\left\| \left(\nabla v \cdot \boldsymbol{n}_e\right)|_{E_1^e} \right\|_{0,e} + \left\| \left(\nabla v \cdot \boldsymbol{n}_e\right)|_{E_2^e} \right\|_{0,e} \right).$$

We apply the trace theorem ([Arn82], Equation 2.5) and get

$$\|\{\lambda k \nabla v \cdot \boldsymbol{n}_e\}\|_{0,e} \leq \frac{Ck_2}{2} \left(h_{E_1^e}^{-\frac{1}{2}} \|\nabla v\|_{0,E_1^e} + h_{E_2^e}^{-\frac{1}{2}} \|\nabla v\|_{0,E_2^e} \right),$$

the second order derivatives in the trace theorem vanish because $v \in V_{1,h}$. Therefore, with use of (2.12)

$$\begin{aligned}\int_e \{\lambda k \nabla v \cdot \boldsymbol{n}_e\}\,[v] &\leq \frac{Ck_2}{2}|e|^{\beta_J/2}\left(h_{E_1^e}^{-\frac{1}{2}}\|\nabla v\|_{0,E_1^e} + h_{E_2^e}^{-\frac{1}{2}}\|\nabla v\|_{0,E_2^e}\right)\\ &\quad\cdot\left(\frac{1}{|e|^{\beta_J}}\right)^{1/2}\|[v]\|_{0,e}\\ &\leq \frac{Ck_2}{2}\left(h_{E_1^e}^{\frac{\beta_J}{2}(d-1)-\frac{1}{2}} + h_{E_2^e}^{\frac{\beta_J}{2}(d-1)-\frac{1}{2}}\right)\\ &\quad\cdot\left(\|\nabla v\|_{0,E_1^e}^2 + \|\nabla v\|_{0,E_2^e}^2\right)^{\frac{1}{2}}\left(\frac{1}{|e|^{\beta_J}}\right)^{\frac{1}{2}}\|[v]\|_{0,e}\\ &\leq Ck_2\left(\|\nabla v\|_{0,E_1^e}^2 + \|\nabla v\|_{0,E_2^e}^2\right)^{\frac{1}{2}}\left(\frac{1}{|e|^{\beta_J}}\right)^{\frac{1}{2}}\|[v]\|_{0,e},\end{aligned}$$

where we used

$$(h_1 a + h_2 b) \leq (h_1 + h_2)\cdot(a^2+b^2)^{1/2} \quad \forall h_1, h_2, a, b \geq 0$$

in the second step and assumed $\beta_J(d-1) \geq 1$ and $h < 1$ in the last step. An analogous bound can be derived if e is a boundary intersection.

Now, let $n_{\max}$ be the maximum number of neighbors, that one element can have. We can then conclude

$$\begin{aligned}\sum_{e\in\Gamma}\int_e \{\lambda k \nabla v \cdot n\}\,[v] &\leq Ck_2\left(\sum_{e\in\Gamma_I}\frac{1}{|e|^{\beta_J}}\|[v]\|_{0,e}^2\right)^{1/2}\left(\sum_{e\in\Gamma_I}\|\nabla v\|_{0,E_1^e}^2 + \|\nabla v\|_{0,E_2^e}^2\right)^{1/2}\\ &\quad + Ck_2\left(\sum_{e\in\Gamma_B}\frac{1}{|e|^{\beta_J}}\|[v]\|_{0,e}^2\right)^{1/2}\left(\sum_{e\in\Gamma_B}\|\nabla v\|_{0,E^e}^2\right)^{1/2}\\ &\leq Ck_2\sqrt{n_{\max}}\left(\sum_{e\in\Gamma}\frac{1}{|e|^{\beta_J}}\|[v]\|_{0,e}^2\right)^{1/2}\left(\sum_{E\in\mathcal{T}_h}\|\nabla v\|_{0,E}^2\right)^{1/2}.\end{aligned}$$

Using quality (2.3) of λk, we get using Young's inequality with $\delta > 0$:

$$\begin{aligned}\sum_{e\in\Gamma}\int_e \{\lambda k \nabla v \cdot n\}\,[v] &\leq \frac{C^2k_2^2 n_{\max}}{2\delta k_1}\sum_{e\in\Gamma}\frac{\|[v]\|_{0,e}^2}{|e|^{\beta_J}} + \frac{\delta}{2}\sum_{E\in\mathcal{T}_h}\int_E k_1\nabla v^2\\ &\leq \frac{C^2k_2^2 n_{\max}}{2\delta k_1}\sum_{e\in\Gamma}\frac{\|[v]\|_{0,e}^2}{|e|^{\beta_J}} + \frac{\delta}{2}\sum_{E\in\mathcal{T}_h}\int_E \lambda k\nabla v^2.\end{aligned}$$

Using the definition 3.1.2 of B_{DG} we have

$$B_{\mathrm{DG}}(v,v) \geq (1-\delta)\sum_{E\in\mathcal{T}_h}\|\sqrt{\lambda k}\nabla v\|_{0,E}^2 + \sum_{e\in\Gamma}\frac{\sigma_J^e - \frac{C^2k_2^2 n_{\max}}{\delta k_1}}{|e|^{\beta_J}}\|[v]\|_{L^2(e)}^2,$$

which yields

$$B_{\mathrm{DG}}(v,v) \geq \kappa \|v\|_{\mathcal{E}} \quad \forall v \in H^2(\mathcal{T}_h)$$

with $\delta = \frac{1}{2}, \sigma_J^e \geq \frac{2C^2 k_2^2 n_{\max}}{k_1}$. Summing up, we have shown that B_{DG} is coercive if $\beta_J(d-1) \geq 1$ and σ_J^e is bounded from below by a sufficiently large constant. □

Theorem 3.2.7 (Continuity)**.** *For $\sigma_J^e \geq 1$ the bilinear form B_{DG} is continuous on the space of piecewise linear functions, that is: There exists $\kappa_s \geq 0$, such that*

$$B_{DG}(v,w) \leq \kappa_s \|v\|_{\mathcal{E}} \|w\|_{\mathcal{E}} \quad \forall v, w \in V_{1,h}.$$

Proof. Let $v, w \in V_{1,h}$. Using the definition of B_{DG}, Equation (3.17) and the Cauchy-Schwarz inequality, we directly have

$$B_{\mathrm{DG}}(v,w) \leq \|v\|_{\mathcal{E}} \|w\|_{\mathcal{E}} + \sum_{e \in \Gamma} \int_e |\{\lambda k \nabla v \cdot n\}\,[w]| + \sum_{e \in \Gamma} \int_e |\{\lambda k \nabla w \cdot n\}[v]|\,.$$

It thus only remains to bound the two last terms. Analog to the proof of Theorem 3.2.6 (just replace the second v by a w), we get

$$\sum_{e \in \Gamma} \int_e |\{\lambda k \nabla v \cdot n\}\,[w]| \leq C k_2 \sqrt{n_{\max}} \left(\sum_{E \in \mathcal{T}_h} \|\nabla v\|_{0,E}^2 \right)^{1/2} \left(\sum_{e \in \Gamma} \frac{1}{|e|^{\beta_J}} \|[w]\|_{0,e}^2 \right)^{1/2},$$

and the similar term where v and w are exchanged. Together with (2.3) and $\sqrt{ab} + \sqrt{cd} \leq \sqrt{a+c} \cdot \sqrt{b+d}$ (for $a, b, c, d \geq 0$) we conclude

$$\begin{aligned}
&\sum_{e \in \Gamma} \int_e |\{\lambda k \nabla v \cdot n\}\,[w]| + \sum_{e \in \Gamma} \int_e |\{\lambda k \nabla w \cdot n\}[v]| \\
&\qquad \leq \frac{C k_2 \sqrt{n_{\max}}}{\sqrt{k_1}} \left(\sum_{e \in \Gamma} \frac{1}{|e|^{\beta_J}} \|[w]\|_{0,e}^2 + \sum_{E \in \mathcal{T}_h} \|\sqrt{k_1} \nabla w\|_{0,E}^2 \right)^{1/2} \\
&\qquad\qquad \cdot \left(\sum_{e \in \Gamma} \frac{1}{|e|^{\beta_J}} \|[v]\|_{0,e}^2 + \sum_{E \in \mathcal{T}_h} \|\sqrt{k_1} \nabla v\|_{0,E}^2 \right)^{1/2} \\
&\qquad \leq \frac{C k_2 \sqrt{n_{\max}}}{\sqrt{k_1}} \|w\|_{\mathcal{E}} \|v\|_{\mathcal{E}},
\end{aligned}$$

if $\sigma^e \geq 1$. Summing up, we have

$$B_{\mathrm{DG}}(v,w) \leq \kappa_s \|v\|_{\mathcal{E}} \|w\|_{\mathcal{E}}, \quad \kappa_s = 1 + \frac{C k_2 \sqrt{n_{\max}}}{\sqrt{k_1}}.$$

□

Theorem 3.2.8 (Existence and Uniqueness)**.** *Let*

$$\sigma_J^e \geq \max\left\{1, \frac{2C^2k_2^2n_{\max}}{k_1}\right\} \quad \textit{and} \quad \beta_J(d-1) \geq 1.$$

Then there exists a unique solution to the discrete DG scheme (3.12).

Proof. In 3.2.4 we demonstrated that $\|\cdot\|_{\mathcal{E}}$ defines a norm on W_N. In Theorem 3.2.7 we demonstrated continuity and in Theorem 3.2.6 coercivity of the bilinear form. The Lax-Milgram theorem [Eva10] thus yields existence and uniqueness of a solution to the discrete DG scheme if the linear form L is bounded. Now, the boundedness of

$$L(v) = \sum_{E\in\mathcal{T}_h}\int_E fv + \sum_{e\in\Gamma_B}\int_e \left(\frac{\sigma_J^e}{|e|^{\beta_J}}v - \lambda k\nabla v\cdot\boldsymbol{n}\right)g_D \tag{3.19}$$

over W_N in the energy norm is easy to see:

The first term can be bounded in the energy norm using the Cauchy-Schwarz inequality, $v \in W_N \subset V_{1,h}$, $\lambda \in L^\infty(\Omega)$ and $k \in [L^\infty(\Omega)]^d$.

The second term is bounded directly by the energy norm and for the third term we can apply the same technique as in the proof of the continuity result 3.2.7. □

3.3 Reduced Basis Generation

In this section we describe in detail how the basis functions of W_N are generated. As mentioned earlier, the basis should basically consist of Finite Element solutions of (2.1). This brings the advantage that a lot of information about the behavior of new solutions to the main equation is already contained in the basis, which can then be chosen to be rather small in comparison to a Finite Element basis.

We will now go a step further: By defining a coarser grid $\mathcal{Z}$ and applying the POD in each large cell whenever a new basis function is added, we achieve a basis that is adapted even better to the problem. Regions of the domain where different solutions (that is: solutions to different parameters $\boldsymbol{\mu}$) differ only little will be covered by very few basis functions while regions where the parameter has a huge effect on the solution will call for a larger basis.

As the error estimator of Section 4.1 depends on the parameter $\boldsymbol{\mu}$ we will not be able to bound the error over all $\boldsymbol{\mu} \in \mathcal{P}$. Instead we will measure the quality of our Galerkin approximation using the error over all $\boldsymbol{\mu} \in \mathcal{M}_{\text{train}}$ where $\mathcal{M}_{\text{train}} \subset \mathcal{P}$ is a finite subset of the parameter domain $\mathcal{P}$.

Remark 3.3.1. Using the splitting (3.9), the following algorithm will build one basis of size M_F per subdomain $F \in \mathcal{Z}$. All those combined then build a basis of size $N = \sum_{F \in \mathcal{Z}} M_F$ on the whole domain.

Given an error tolerance Δ, a maximum basis size $N_{\max}$ and a POD-tolerance Δ_{POD}, the basis Φ is generated in 7 steps:

1. Choose a parameter $\boldsymbol{\mu}_0$ for the first basis function. One may accelerate the rest of the algorithm by judiciously choosing this parameter. If no deeper insight into the dependence of the solution on the parameter $\boldsymbol{\mu}$ is available, one may just choose a random initial parameter $\boldsymbol{\mu}_0 \in \mathcal{P}$.
2. Perform a detailed simulation, that is a FE simulation for $\boldsymbol{\mu}_0$ and construct the first basis functions $\varphi_1, \dots, \varphi_{|\mathcal{Z}|}$ from the result p_1 by restricting it to the elements $F \in \mathcal{Z}$:

$$\varphi_i(\boldsymbol{x}) = \begin{cases} p_1(\boldsymbol{x}) & \text{if } \boldsymbol{x} \in F_i, \\ 0 & \text{else.} \end{cases} \qquad \forall i \in \{1, \dots, |\mathcal{Z}|\}.$$

   Add $\varphi_1, \dots, \varphi_{|\mathcal{Z}|}$ to the basis Φ.
3. Compute the offline-parts of the DG scheme (3.15) and of the error estimator (4.19)-(4.22) for the current basis.
4. Compute the reduced solutions, that is solve (3.14) for all $\boldsymbol{\mu} \in \mathcal{M}_{\text{train}}$ using the current basis. Then evaluate the error estimator (4.8) for all solutions and find the parameter $\boldsymbol{\mu}_{\max} \in \mathcal{M}_{\text{train}}$ that maximizes the error bound.

   If the error bound for the reduced solution to $\boldsymbol{\mu}_{\max}$ is already smaller than Δ, abort.
5. Compute the detailed solution $\bar{p}$ for $\boldsymbol{\mu}_{\max}$.

6. On every coarse cell $F \in \mathcal{Z}$: Let $\bar{p}_r$ be the restriction of $\bar{p}$ to F. Apply the POD(Δ_{POD}) to the degrees of freedom of the existing basis functions and $\bar{p}_r$. Set the arising functions as basis for F.
7. If $N = \sum_{F \in \mathcal{Z}} M_F < N_{\max}$, continue with step 3.

Remark 3.3.2. In step six it is possible to measure how much information about the original functions the result of the POD contains. There are different ways of doing so. We will fix the percentage of total variation that the selected principal components shall contribute as suggested in [Jol02], Chapter 6. This may lead to different numbers of base functions on each coarse cell.

4 Error Analysis

4.1 A Posteriori Error Estimator

In this section we will derive an a posteriori error estimator that will help us to construct the reduced basis.

For space dimensions $d = 2$ and $d = 3$ we will demonstrate adjoint consistency for the bilinear form B_{DG} and use a duality technique in the derivation of the estimator. Both steps are well-known in the derivation of error estimators. Still, using the standard procedure, we will run into problems as to gain additional orders in h, one usually applies an interpolation estimate between the space of the dual solution and the ansatzspace (W_N in our case). This will not work here, due to the insufficient information that we have about the shape of the functions in W_N. We will stress this problem and our solution below.

While duality techniques are broadly used in the literature, we add different non-standard techniques and thus present a brand new approach for error estimation in situations where no interpolation estimates are available.

For the sake of simplicity and readability in the exposure of the error estimator, we will restrict the mobility λ and the permeability k to $\lambda k|_E \in \mathbb{P}_1(E)$ for all $E \in \mathcal{T}_h$ and $(\lambda k) \in \mathcal{C}^0(\Omega)$. Without this assumption we would just have to introduce additional terms to the error estimator or replace all occurrences of λk by their measure in L^∞. For the finite element scheme to resolve the behaviour of the analytical solution, the grid size has to be smaller than the oscillation scale of λk. Therefore, this assumption does not even represent a severe restriction.

Lemma 4.1.1. *The bilinear form B_{DG} is adjoint consistent, that is: For $v \in H^2(\Omega)$ with*

$$-\nabla \cdot (\lambda k \nabla v) = h \quad \text{in } \Omega, \tag{4.1}$$

$$v = 0 \quad \text{on } \partial\Omega, \tag{4.2}$$

where $h \in L^2(\Omega)$, holds:

$$B_{DG}(\varphi, v) = \sum_{E \in \mathcal{T}_h} \int_E h\varphi \quad \forall \varphi \in H^2(\mathcal{T}_h).$$

Proof. Let φ, v be as suggested in the lemma. We then have by the definition of B_{DG}:

$$\begin{aligned} B_{\mathrm{DG}}(\varphi, v) = &\sum_{E \in \mathcal{T}_h} \int_E \lambda k \nabla\varphi \cdot \nabla v + \sum_{e \in \Gamma} \frac{\sigma^e}{|e|^\beta} \int_e [\varphi][v] \\ &- \sum_{e \in \Gamma} \left(\{\lambda k \nabla\varphi \cdot n\}[v] + \{\lambda k \nabla v \cdot n\}[\varphi] \right). \end{aligned}$$

As $v \in H^2(\Omega) \subset \mathcal{C}^0(\Omega)$ and $v = 0$ on $\partial\Omega$, all jumps of v over $e \in \Gamma_I$ and all integrals of v over Γ_B vanish. It thus suffices to show

$$\sum_{E\in\mathcal{T}_h} \int_E \lambda k \nabla\varphi \cdot \nabla v - \sum_{e\in\Gamma} \{\lambda k \nabla v \cdot n\}[\varphi] = \sum_{E\in\mathcal{T}_h} \int_E h\varphi. \tag{4.3}$$

We have by Greens formula ([Alt06], Theorem A6.8):

$$\sum_{E\in\mathcal{T}_h} \int_E \lambda k \nabla v \cdot \nabla\varphi = -\sum_{E\in\mathcal{T}_h} \int_e \nabla \cdot (\lambda k \nabla v)\varphi + \sum_{E\in\mathcal{T}_h} \int_{\partial E} \lambda k \nabla v \cdot n\varphi.$$

By the definition of the jump we get

$$\begin{aligned}\sum_{E\in\mathcal{T}_h} \int_{\partial E} \lambda k \nabla v \cdot n\varphi &= \sum_{e\in\Gamma_I} \int_e [\lambda k \nabla v \cdot n\varphi] + \sum_{e\in\Gamma_B} \int_e \lambda k \nabla v \cdot n\varphi \\ &= \sum_{e\in\Gamma_I} \int_e ([\lambda k \nabla v \cdot n]\{\varphi\} + \{\lambda k \nabla v \cdot n\}[\varphi]) \\ &\quad + \sum_{e\in\Gamma_B} \int_e \lambda k \nabla v \cdot n\varphi.\end{aligned} \tag{4.4}$$

Now, as $v \in H^2(\Omega)$, we have $[\lambda k \nabla v \cdot n] = 0$ (see (3.8)). Inserting (4.4) into (4.3) we need to show

$$-\sum_{E\in\mathcal{T}_h} \int_e \nabla \cdot (\lambda k \nabla v)\varphi = \sum_{E\in\mathcal{T}_h} h\varphi,$$

which is obvious due to the requirements on v. □

As described at the beginning of the section, at this point we could demonstrate an a posteriori error identity of the form

$$\|p - p_N\|_{0,\Omega}^2 = L(v) - B_{\mathrm{DG}}(p_N, v),$$

where v would be the solution to (4.1) for $h = p - p_N$. We would then use the consistency of B_{DG} to insert an additional v_h, stemming from some discrete space, into the equation. Using an interpolation estimate of the form $\|v - v_h\|_a \leq h^k \|v\|_b$ with $k \in \mathbb{N}$ and matching norms $\|\cdot\|_a$ and $\|\cdot\|_b$, we would gain an additional order in h. As this is not possible here for the reasons described above, we will now introduce a linear post processing operator $\mathcal{R}_{1,2} : V_{1,h} \to V_{2,h}$. This operator will not provide a theoretical convergence order, it only serves to guarantee any convergence of the error.

We will not discuss the concrete choice of this reconstruction operator in this work. For the theory it suffices to know, that for any piecewise linear function v_h, $\mathcal{R}_{1,2}(v_h)$ is a piecewise polynomial of order two, that $\mathcal{R}_{1,2}$ is linear and we assume that for any $v_h \in V_{1,h}$ with $v_h \in C^0(E_1 \cap E_2)$ for two entities $E_1, E_2 \in \mathcal{T}_h$ we have $\mathcal{R}_{1,2}(v_h) \in C^0(E_1 \cap E_2)$.

Using this post processing operator, we now give an a posteriori error identity that will lead directly to the desired error estimator.

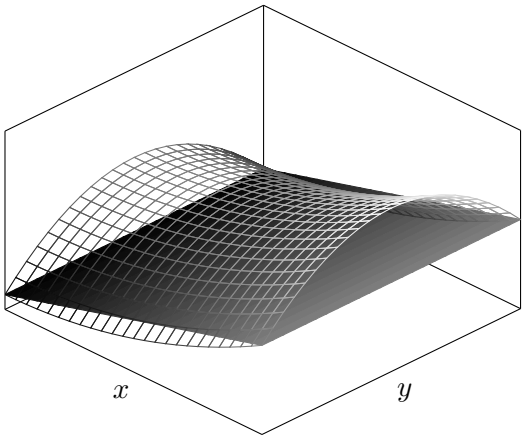

Figure 4.1: Reconstruction of a polynomial of order 1 to a polynomial of order 2 in 2D.

Lemma 4.1.2 (A posteriori error identity)**.** *Let λ, k be sufficiently smooth and $\Omega, \partial\Omega$ such that for each $h \in L^2(\Omega)$, there exists a solution $v \in H^2(\Omega)$ to the adjoint problem*

$$\begin{aligned} -\nabla \cdot (\lambda k \nabla v) &= h \quad \text{in } \Omega \\ v &= 0 \quad \text{on } \partial\Omega. \end{aligned} \tag{4.5}$$

Then, if $p \in H^2(\Omega)$, $\widetilde{p_N} = \mathcal{R}_{1,2}(p_N)$ are weak and post processed Discontinuous Galerkin solutions to problem (2.1), respectively, the following holds:

$$\|p - \widetilde{p_N}\|_{0,\Omega}^2 = L(v) - B_{DG}(\widetilde{p_N}, v), \tag{4.6}$$

where v is the solution of (4.5) for $h = p - \widetilde{p_N}$.

Proof. In Lemma 4.1.1 choose $h = p - \widetilde{p_N}$ and $\varphi = p - \widetilde{p_N}$, then we have

$$\begin{aligned} \|p - \widetilde{p_N}\|_{0,\Omega}^2 &= B_{\mathrm{DG}}(p - \widetilde{p_N}, v) \\ &= L(v) - B_{\mathrm{DG}}(\widetilde{p_N}, v), \end{aligned}$$

where we used Theorem 3.1.4. □

The same problem that we described for the residual term also applies for the jump terms in B_{DG}. We will need an additional ingredient to achieve convergence in these terms without an interpolation estimate. We thus, following a concept proposed in [BMM$^+$99], introduce an affine lifting operator $\boldsymbol{r}_e : H^2(\mathcal{T}_h) \to \boldsymbol{V}_{2,h}$. For any $w \in H^2(\mathcal{T}_h)$ and any intersection $e \in \Gamma$, $\boldsymbol{r}_e(w)$ is defined as the solution of

$$\int_\Omega \boldsymbol{r}_e(w) \cdot \boldsymbol{\tau}_h = -\int_e [w]\{\boldsymbol{\tau}_h \cdot \boldsymbol{n}\} \quad \forall \boldsymbol{\tau}_h \in \boldsymbol{V}_{2,h}. \tag{4.7}$$

Here, we made use of the space

$$\boldsymbol{V}_{k,h} = \left\{ \boldsymbol{\tau}_h \in \left[L^2(\Omega)\right]^d \,\middle|\, \boldsymbol{\tau}_h|_E \in [\mathbb{P}_k(E)]^d \ \forall E \in \mathcal{T}_h \right\}.$$

Given this operator, the following two results where shown in [BMM$^+$99].

Lemma 4.1.3. *There exists a positive constant C_r, independent of h, such that*

$$\|[v_h]\|_{0,e} \leq C_r h_e^{1/2} \|\boldsymbol{r}_e(v_h)\|_{0,\Omega}$$

for each $v_h \in V_{2,h}$ and for each $e \in \Gamma$.

Lemma 4.1.4. *Let $\varphi \in H^2(\mathcal{T}_h)$ and $v_h \in V_{2,h}$. Then*

$$\sum_{e\in\Gamma} \int_e [v_h]\{\nabla\varphi \cdot \boldsymbol{n}\} \leq C_r \sum_{E\in\mathcal{T}_h} \sum_{e\subset\partial E} \|\boldsymbol{r}_e(v_h)\|_{0,\Omega} \left(|\varphi|_{1,E} + h_e |\varphi|_{2,E} \right).$$

Here the positive constant C_r, given by Lemma 4.1.3, depends on the minimal-angle-bound.

Remark 4.1.5. From Lemma 4.1.4 and its proof one can easily derive the bounds

$$\begin{aligned}
\sum_{e\in\Gamma_I} \int_e [v_h]\{\nabla\varphi \cdot \boldsymbol{n}\} &\leq C_r \sum_{e\in\Gamma_I} \|\boldsymbol{r}_e(v_h)\|_{0,\Omega} \left(|\varphi|_{1,E_e} + h_e |\varphi|_{2,E_e} \right), \\
\sum_{e\in\Gamma_B} \int_e [v_h]\{\nabla\varphi \cdot \boldsymbol{n}\} &\leq C_r \sum_{e\in\Gamma_B} \|\boldsymbol{r}_e(v_h)\|_{0,\Omega} \left(|\varphi|_{1,E_e} + h_e |\varphi|_{2,E_e} \right)
\end{aligned}$$

for $\varphi \in H^2(\mathcal{T}_h)$ and $v_h \in V_{2,h}$. Here E_e denotes the inside cell E of an intersection e.

At this point, we have everything at hand to formulate the central result of this section.

Theorem 4.1.6 (A posteriori error estimator)**.** *Let p and p_N be the weak (2.11) and Discontinuous Galerkin solution (3.12), respectively. Then we have*

$$\begin{aligned}
\|p - p_N\|_{0,\Omega} \leq \|\widetilde{p_N} - p_N\|_{0,\Omega} &+ \sum_{E\in\mathcal{T}_h} \eta_1^E(p_N) \\
&+ \sum_{e\in\Gamma_I} \eta_2^e(p_N) + \sum_{e\in\Gamma_B} \eta_3^e(p_N),
\end{aligned} \tag{4.8}$$

where

$$\begin{aligned}
\widetilde{p_N} &= \mathcal{R}_{1,2}(p_N) \\
\eta_1^E(p_N) &= \frac{C_o^2}{k_1} \|f + \nabla\cdot(\lambda k \nabla\widetilde{p_N})\|_{0,E}, \\
\eta_2^e(p_N) &= (C_o + h_e)\frac{C_r C_o}{k_1} \|\boldsymbol{r}_e(\lambda k \nabla\widetilde{p_N}\cdot\boldsymbol{n})\|_{0,\Omega} \\
&\quad + C_r \left(\frac{C_o k_2}{k_1} + h_e \right) \|\boldsymbol{r}_e(\widetilde{p_N})\|_{0,\Omega}, \\
\eta_3^e(p_N) &= C_r \cdot \left(\frac{C_o k_2}{k_1} + h_e \right) \|\boldsymbol{r}_e(\widetilde{p_N} - g_D)\|_{0,\Omega}.
\end{aligned} \tag{4.9}$$

Proof. We start by stating

$$\|p - p_N\|_{0,\Omega} \leq \|p - \widetilde{p_N}\|_{0,\Omega} + \|\widetilde{p_N} - p_N\|_{0,\Omega}.$$

As the term $\|\widetilde{p_N} - p_N\|_{0,\Omega}$ is already in a posteriori form, it suffices to bound $\|p - \widetilde{p_N}\|_{0,\Omega}$. Inserting the definitions of B_{DG} and L in (4.6) yields:

$$\begin{aligned}
\|p - \widetilde{p_N}\|_{0,\Omega}^2 = &\sum_{e\in\Gamma_B}\int_e \left(\frac{\sigma^e}{|e|^\beta}v - \lambda k\nabla v\cdot\boldsymbol{n}\right) g_D \\
&+ \sum_{E\in\mathcal{T}_h}\int_E fv - \sum_{E\in\mathcal{T}_h}\int_E \lambda k\nabla\widetilde{p_N}\cdot\nabla v \\
&+ \sum_{e\in\Gamma}\int_e \{\lambda k\nabla\widetilde{p_N}\cdot\boldsymbol{n}\}[v] \\
&+ \sum_{e\in\Gamma}\int_e \{\lambda k\nabla v\cdot\boldsymbol{n}\}[\widetilde{p_N}] - \sum_{e\in\Gamma}\int_e \frac{\sigma^e}{|e|^\beta}[v][\widetilde{p_N}].
\end{aligned}$$

As $v \in H^2(\Omega) \subset \mathcal{C}^0(\Omega)$ for space dimensions $d = 2, 3$ and $v = 0$ on $\partial\Omega$, this reduces to

$$\begin{aligned}
\|p - \widetilde{p_N}\|_{0,\Omega}^2 = &\sum_{E\in\mathcal{T}_h}\int_E fv - \sum_{E\in\mathcal{T}_h}\int_E \lambda k\nabla\widetilde{p_N}\nabla v \\
&+ \sum_{e\in\Gamma_B}\int_e \lambda k\nabla v\cdot\boldsymbol{n}\,(\widetilde{p_N} - g_D) \\
&+ \sum_{e\in\Gamma_I}\int_e \{\lambda k\nabla v\cdot\boldsymbol{n}\}[\widetilde{p_N}].
\end{aligned}$$

Integration by parts in the second term yields

$$\begin{aligned}
\|p - \widetilde{p_N}\|_{0,\Omega}^2 = &\sum_{E\in\mathcal{T}_h}\int_E (f + \nabla\cdot(\lambda k\nabla\widetilde{p_N}))v \\
&- \sum_{e\in\Gamma_I}\int_e [\lambda k\nabla\widetilde{p_N}\cdot\boldsymbol{n}]v + \sum_{e\in\Gamma_I}\int_e \{\lambda k\nabla v\cdot\boldsymbol{n}\}[\widetilde{p_N}] \\
&+ \sum_{e\in\Gamma_B}\int_e \lambda k\nabla v\cdot\boldsymbol{n}\,(\widetilde{p_N} - g_D) \\
\leq &\sum_{E\in\mathcal{T}_h}\|f + \nabla\cdot(\lambda k\nabla\widetilde{p_N})\|_{0,E}\,\|v\|_{0,E} \qquad (4.10)\\
&+ \sum_{e\in\Gamma_I} C_r\|\boldsymbol{r}_e(\lambda k\nabla\widetilde{p_N}\cdot\boldsymbol{n})\|_{0,\Omega}(\|v\|_{0,E_e} + h_e\,|v|_{1,E_e}) \\
&+ \sum_{e\in\Gamma_I} C_r\|\boldsymbol{r}_e(\widetilde{p_N})\|_{0,\Omega}\,(\|\lambda k\nabla v\|_{0,E_e} + h_e\|\nabla\cdot(\lambda k\nabla v)\|_{0,E_e}) \\
&+ \sum_{e\in\Gamma_B} C_r\|\boldsymbol{r}_e(\widetilde{p_N} - g_D)\|_{0,\Omega}\,(\|\lambda k\nabla v\|_{0,E_e} + h_e\|\nabla\cdot(\lambda k\nabla v)\|_{0,E_e})\,,
\end{aligned}$$

where we used the Cauchy-Schwarz inequality, Lemma 4.1.4 and Remark 4.1.5.

It thus suffices to show that each factor on the right hand side containing v is bounded from above by $\|p - \widetilde{p_N}\|_{0,\Omega}$.

First of all, as v is the solution to equation (4.1) with $h = p - \widetilde{p_N}$, it is obvious that

$$\|\nabla \cdot (\lambda k \nabla v)\|_{0,E} = \|p - \widetilde{p_N}\|_{0,E}.$$

Furthermore, we have $\|\lambda k \nabla v\|_{0,E} \leq k_2 \|\nabla v\|_{0,E}$ by (2.3) and

$$\begin{aligned}
\|\nabla v\|_{0,\Omega}^2 &= \int_\Omega \nabla v \cdot \nabla v \leq \frac{1}{k_1} \int_\Omega \lambda k \nabla v \nabla v = -\frac{1}{k_1} \int_\Omega \nabla \cdot (\lambda k \nabla v) v \\
&= \frac{1}{k_1} \int_\Omega (p - \widetilde{p_N}) v \leq \frac{1}{k_1} \|p - \widetilde{p_N}\|_{0,\Omega} \|v\|_{0,\Omega} \\
&\leq \frac{C_o}{k_1} \|p - \widetilde{p_N}\|_{0,\Omega} \|\nabla v\|_{0,\Omega},
\end{aligned}$$

that is:

$$\|\nabla v\|_{0,\Omega} \leq \frac{C_o}{k_1} \|p - \widetilde{p_N}\|_{0,\Omega}, \quad \|\lambda k \nabla v\|_{0,\Omega} \leq \frac{C_o k_2}{k_1} \|p - \widetilde{p_N}\|_{0,\Omega},$$

where C_o is the optimal Poincaré constant. Altogether, we have

$$\begin{aligned}
\|p - \widetilde{p_N}\|_{0,\Omega}^2 \leq & \sum_{E \in \mathcal{T}_h} \|f + \nabla \cdot (\lambda k \nabla \widetilde{p_N})\|_{0,E} \frac{C_o^2}{k_1} \|p - \widetilde{p_N}\|_{0,\Omega} \\
& + \sum_{e \in \Gamma_I} C_r \|\boldsymbol{r}_e(\lambda k \nabla \widetilde{p_N} \cdot \boldsymbol{n})\|_{0,\Omega} (C_o + h_e) \frac{C_o}{k_1} \|p - \widetilde{p_N}\|_{0,\Omega} \\
& + \sum_{e \in \Gamma_I} C_r \|\boldsymbol{r}_e(\widetilde{p_N})\|_{0,\Omega} \left(\frac{C_o k_2}{k_1} + h_e \right) \|p - \widetilde{p_N}\|_{0,\Omega} \\
& + \sum_{e \in \Gamma_B} C_r \|\boldsymbol{r}_e(\widetilde{p_N} - g_D)\|_{0,\Omega} \left(\frac{C_o k_2}{k_1} + h_e \right) \|p - \widetilde{p_N}\|_{0,\Omega}.
\end{aligned}$$

The desired result now follows after division by $\|p - \widetilde{p_N}\|_{0,\Omega}$. □

Remark 4.1.7. As $\widetilde{p_N}$ is continuous inside the large cells, the three quantities $\boldsymbol{r}_e(\widetilde{p_N})$, $\boldsymbol{r}_e(\lambda k \nabla \widetilde{p_N} \cdot \boldsymbol{n})$ and $\boldsymbol{r}_e(\widetilde{p_N} - g_D)$ have to be computed only for intersections e shared by two large cells. Furthermore, we will be able to partly precompute those quantities in a offline phase. The evaluations of the error estimator will thus not be too costly.

4.2 Offline-Online Decomposition

We will have to evaluate the error estimator quite often during the construction of a reduced basis and we would like to use it online to control the quality of the reduced solutions. For these reasons it is very desirable to split the computation of (4.8) into an offline and online phase. In this section, we will show that this is possible for the given error estimator.

For this section, let

$$\widetilde{p_N} = \sum_{F\in\mathcal{Z}} \sum_{i=1}^{M_F} a_i^F \widetilde{\varphi}_i^F,$$

where $\widetilde{\varphi}_i^F = \mathcal{R}_{1,2}(\varphi_i^F)$. Then a straightforward computation gives

$$\begin{aligned}\|f + \nabla\cdot(\lambda k \nabla\widetilde{p_N})\|_{0,F}^2 &= \|f\|_{0,F}^2 + 2\sum_{j=1}^{N_\lambda}\sum_{i=1}^{M_F} \mu_j a_i^F \int_F f\nabla(\lambda_j(x) k \nabla\widetilde{\varphi}_i^F(x)) \\ &+ \sum_{j=1}^{N_\lambda}\sum_{i=1}^{M_F}\sum_{\sigma=1}^{N_\lambda}\sum_{\nu=1}^{M_F} \mu_j a_i^F \mu_\sigma a_\nu^F \int_F \nabla\cdot(\lambda_j(x) k\nabla\widetilde{\varphi}_i^F(x))\nabla\cdot(\lambda_\sigma(x) k\nabla\widetilde{\varphi}_\nu^F(x)),\end{aligned}$$

for all coarse cells $F \in \mathcal{Z}$.

For the rest of the paragraph we assume e to be the intersection of the fine elements E_1 and E_2, that belong to coarse cells F_1 and F_2, respectively.
Solving Equation (4.7) can also be done in two steps: an offline and online step: For $k \in \{1,2\}, i \in \{1,\dots,M_{F_k}\}$ let $\delta_i^k \in \boldsymbol{V}_{2,h}$ be the solution to

$$\int_\Omega \delta_i^k \cdot \boldsymbol{\tau}_h = -\int_e \widetilde{\varphi}_i^{F_k}\{\boldsymbol{\tau}_h\cdot\boldsymbol{n}\} \quad \forall \boldsymbol{\tau}_h \in \boldsymbol{V}_{2,h}, \tag{4.11}$$

and we directly get for all $\boldsymbol{\tau}_h \in \boldsymbol{V}_{2,h}$:

$$\begin{aligned}\int_\Omega \Big(\sum_{i=1}^{M_{F_1}} a_i^{F_1}\delta_i^1 - \sum_{j=1}^{M_{F_2}} a_j^{F_2}\delta_j^2\Big)\cdot\boldsymbol{\tau}_h &= -\int_e \Big(\sum_{i=1}^{M_{F_1}} a_i^{F_1}\widetilde{\varphi}_i^{F_1} - \sum_{j=1}^{M_{F_2}} a_j^{F_2}\widetilde{\varphi}_j^{F_2}\Big)\{\boldsymbol{\tau}_h\cdot\boldsymbol{n}\} \\ &= -\int_e [\widetilde{p_N}]\{\boldsymbol{\tau}_h\cdot\boldsymbol{n}\},\end{aligned}$$

which means

$$\boldsymbol{r}_e(\widetilde{p_N}) = \sum_{i=1}^{M_{F_1}} a_i^{F_1}\delta_i^1 - \sum_{i=1}^{M_{F_2}} a_i^{F_2}\delta_i^2.$$

Accordingly, the norm of $\boldsymbol{r}_e(\widetilde{p_N})$ can be expressed as

$$\begin{aligned}\|\boldsymbol{r}_e(\widetilde{p_N})\|_{0,\Omega}^2 = &\sum_{i,j=1}^{M_{F_1}} a_i^{F_1} a_j^{F_1} \int_\Omega \delta_i^1 \cdot \delta_j^1 + \sum_{i,j=1}^{M_{F_2}} a_i^{F_2} a_j^{F_2} \int_\Omega \delta_i^2 \cdot \delta_j^2 \\ &- 2 \sum_{i=1}^{M_{F_1}} \sum_{j=1}^{M_{F_2}} a_i^{F_1} a_j^{F_2} \int_\Omega \delta_i^1 \cdot \delta_j^2.\end{aligned} \tag{4.12}$$

Likewise, we treat the expression $\|\boldsymbol{r}_e(\lambda k \nabla \widetilde{p_N} \cdot \boldsymbol{n})\|_{0,\Omega}$: Let $\psi_i^{m,\sigma} \in \boldsymbol{V}_{2,h}$, where $m \in \{1,2\}$, $i \in \{1,\dots,M_{F_m}\}$ and $\sigma \in \{1,\dots,N_\lambda\}$, be the solution to

$$\int_\Omega \psi_i^{m,\sigma} \cdot \boldsymbol{\tau}_h = -\int_e k(\boldsymbol{x}) \lambda_\sigma(\boldsymbol{x}) \nabla \widetilde{\varphi}_i^{F_m} \cdot \boldsymbol{n} \{\boldsymbol{\tau}_h \cdot \boldsymbol{n}\} \quad \forall \boldsymbol{\tau}_h \in \boldsymbol{V}_{2,h}. \tag{4.13}$$

Then, for all $\boldsymbol{\tau}_h \in \boldsymbol{V}_{2,h}$ we clearly have

$$\begin{aligned}&\int_\Omega \Big(\sum_{\sigma=1}^{N_\lambda} \sum_{i=1}^{M_{F_1}} \mu_j a_i^{F_1} \psi_i^{1,\sigma} - \sum_{\nu=1}^{N_\lambda} \sum_{j=1}^{M_{F_2}} \mu_\nu a_j^{F_2} \psi_j^{2,\nu} \Big) \cdot \boldsymbol{\tau}_h \\ &\qquad = -\sum_{\sigma=1}^{N_\lambda} \sum_{i=1}^{M_{F_1}} \mu_\sigma a_i^{F_1} \int_e k(\boldsymbol{x}) \lambda_\sigma(\boldsymbol{x}) \nabla \widetilde{\varphi}_i^{F_1} \cdot \boldsymbol{n} \{\boldsymbol{\tau}_h \cdot \boldsymbol{n}\} \\ &\qquad\quad + \sum_{\nu=1}^{N_\lambda} \sum_{j=1}^{M_{F_2}} \mu_\nu a_j^{F_2} \int_e k(\boldsymbol{x}) \lambda_\nu(\boldsymbol{x}) \nabla \widetilde{\varphi}_j^{F_2} \cdot \boldsymbol{n} \{\boldsymbol{\tau}_h \cdot \boldsymbol{n}\} \\ &\qquad = -\int_e \Big((\lambda k \nabla \widetilde{p_N} \cdot \boldsymbol{n})|_{E_1} - (\lambda k \nabla \widetilde{p_N} \cdot \boldsymbol{n})|_{E_2} \Big) \{\boldsymbol{\tau}_h \cdot \boldsymbol{n}\} \\ &\qquad = -\int_e [\lambda k \nabla \widetilde{p_N} \cdot \boldsymbol{n}] \{\boldsymbol{\tau}_h \cdot \boldsymbol{n}\},\end{aligned}$$

which indicates

$$\boldsymbol{r}_e(\lambda k \nabla \widetilde{p_N} \cdot \boldsymbol{n}) = \sum_{\sigma=1}^{N_\lambda} \sum_{i=1}^{M_{F_1}} \mu_\sigma a_i^{F_1} \psi_i^{1,\sigma} - \sum_{\nu=1}^{N_\lambda} \sum_{j=1}^{M_{F_2}} \mu_\nu a_j^{F_2} \psi_j^{2,\nu}.$$

And finally, the desired quantity is

$$\begin{aligned}\|\boldsymbol{r}_e(\lambda k \nabla \widetilde{p_N} \cdot \boldsymbol{n})\|_{0,\Omega}^2 = &\sum_{\sigma,\nu=1}^{N_\lambda} \sum_{i,j=1}^{M_{F_1}} \mu_\sigma a_i^{F_1} \mu_\nu a_j^{F_1} \int_\Omega \psi_i^{1,\sigma} \psi_j^{1,\nu} \\ &+ \sum_{\sigma,\nu=1}^{N_\lambda} \sum_{i,j=1}^{M_{F_2}} \mu_\sigma a_i^{F_2} \mu_\nu a_j^{F_2} \int_\Omega \psi_i^{2,\sigma} \psi_j^{2,\nu} \\ &- 2 \sum_{\sigma,\nu=1}^{N_\lambda} \sum_{i=1}^{M_{F_1}} \sum_{j=1}^{M_{F_2}} \mu_\sigma a_i^{F_1} \mu_\nu a_j^{F_2} \int_\Omega \psi_i^{1,\sigma} \psi_j^{2,\nu}.\end{aligned} \tag{4.14}$$

The last expression in the error estimator needs to be computed for all intersections $e \in \Gamma_B$ belonging to the boundary of the domain. If e is such an intersection and F_1 the associated fine cell, the reconstruction of $\widetilde{p_N}$ can, analogous to the matching term on inner intersections, be written as

$$\boldsymbol{r}_e(\widetilde{p_N}) = \sum_{i=1}^{M_{F_1}} a_i^{F_1} \delta_i^1, \tag{4.15}$$

where $\delta_i^1 \in \boldsymbol{V}_{2,h}$ is the solution to

$$\int_\Omega \delta_i^1 \cdot \boldsymbol{\tau}_h = -\int_e \widetilde{\varphi}_i^{F_1} \{\boldsymbol{\tau}_h \cdot \boldsymbol{n}\} \quad \forall \boldsymbol{\tau}_h \in \boldsymbol{V}_{2,h}. \tag{4.16}$$

Together with

$$\boldsymbol{r}_e(\widetilde{p_N} - g_D) = \boldsymbol{r}_e(\widetilde{p_N}) - \boldsymbol{r}_e(g_D) \tag{4.17}$$

the decomposition of the last line in (4.9) can be done easily:

$$\begin{aligned} \|\boldsymbol{r}_e(\widetilde{p_N} - g_D)\|_{0,\Omega}^2 = \|\boldsymbol{r}_e(\widetilde{p_N})\|_{0,\Omega}^2 + \sum_{\nu,\eta=1}^{N_\lambda} \mu_\nu \mu_\eta \int_\Omega \boldsymbol{r}_e(g_\nu) \boldsymbol{r}_e(g_\eta) \\ - 2 \sum_{j=1}^{N_\lambda} \sum_{i=1}^{M_{F_1}} a_i^{F_1} \mu_j \int_\Omega \delta_i^1 \boldsymbol{r}_e(g_j). \end{aligned} \tag{4.18}$$

4.2.1 Summary: Offline-Online Decomposition of Error Estimator

At this point, a short summary is in order.

For the first error estimator term, we have to compute

$$\begin{aligned} \|f\|_{0,F}^2, \\ \boldsymbol{k}^{F,\sigma} \in \mathbb{R}^N, \quad \boldsymbol{k}_i^{F,\sigma} = 2 \int_F f \nabla \cdot (\lambda_\sigma(x) k \nabla \widetilde{\varphi}_i^F(x)), \\ \boldsymbol{K}^{F,\sigma,\nu} \in \mathbb{R}^{N \times N}, \quad \boldsymbol{K}_{i,j}^{F,\sigma,\nu} = \int_F \nabla \cdot (\lambda_\sigma(x) k \nabla \widetilde{\varphi}_i^F) \nabla \cdot (\lambda_\nu(x) k \nabla \widetilde{\varphi}_j^F) \end{aligned} \tag{4.19}$$

for all $F \in \mathcal{Z}, \sigma, \nu \in \{1, \dots, N_\lambda\}$.

The second, third and fourth expression of (4.9) need to be computed for all entities of codimension 1 being the intersection of two fine cells E_1 and E_2 with different enclosing coarse cells F_1 and F_2 as all those terms vanish on intersections inside a coarse cell due to the regularity of $\widetilde{p_N}$.

For the second part of the error estimator we need

$$\begin{aligned} \boldsymbol{M}^{E_1,E_1} \in \mathbb{R}^{M_{F_1} \times M_{F_1}}, \quad \boldsymbol{M}^{E_1,E_2} \in \mathbb{R}^{M_{F_1} \times M_{F_2}}, \quad \boldsymbol{M}^{E_2,E_2} \in \mathbb{R}^{M_{F_2} \times M_{F_2}}, \\ \boldsymbol{M}_{i,j}^{E_m,E_n} = \int_\Omega \delta_i^m \cdot \delta_j^n \end{aligned} \tag{4.20}$$

with δ_i^m defined in (4.11) and for the third

$$\begin{gathered}\boldsymbol{N}^{E_1,E_1,\sigma,\nu} \in \mathbb{R}^{M_{F_1}\times M_{F_1}}, \boldsymbol{N}^{E_1,E_2,\sigma,\nu} \in \mathbb{R}^{M_{F_1}\times M_{F_2}}, \boldsymbol{N}^{E_2,E_2,\sigma,\nu} \in \mathbb{R}^{M_{F_2}\times M_{F_2}}, \\ \boldsymbol{N}_{i,j}^{E_m,E_n,\sigma,\nu} = \int_\Omega \psi_i^{m,\sigma} \cdot \psi_j^{n,\nu}\end{gathered} \tag{4.21}$$

for all $\sigma, \nu \in \{1, \dots, N_\lambda\}$. For the definition of $\psi_i^{m,\sigma}$ see (4.13).

Finally, the fourth part of the error estimator that needs to be computed for all $e \in \Gamma_B$ with associated $E_1 \in \mathcal{T}_h$ and all $\sigma \in \{1, \dots, N_\lambda\}$, requires

$$\begin{aligned}\boldsymbol{P}^{E_1} \in \mathbb{R}^{N_\lambda \times N_\lambda}, \quad \boldsymbol{P}_{i,j}^{E_1} &= \int_\Omega \boldsymbol{r}_e(g_i) \cdot \boldsymbol{r}_e(g_j), \\ \boldsymbol{m}^{E_1,\sigma} \in \mathbb{R}^{M_{F_1}}, \quad \boldsymbol{m}_i^{E_1,\sigma} &= \int_\Omega \delta_i^1 \cdot \boldsymbol{r}_e(g_\sigma), \\ \boldsymbol{Q}^{E_1} \in \mathbb{R}^{M_{F_1}\times M_{F_1}}, \quad \boldsymbol{Q}_{i,j}^{E_1} &= \int_\Omega \delta_i^1 \cdot \delta_j^1\end{aligned} \tag{4.22}$$

where δ_i^1 was defined in (4.16).

Once all these matrices are computed, the evaluation of the error estimator can be done very rapidly, as the numerical results in Chapter 6 will demonstrate. For one evaluation of the error estimator, the quantities (4.19)–(4.22) have to be combined in the following manner.

Let Λ be the set of intersections defined by $\Lambda = \{e \in \Gamma_I | \exists f \in \Xi_I : e \subset f\}$ and for each $e \in \Gamma_I$ let E_1, E_2 be the neighboring fine cells with enclosing coarse cells F_1, F_2 respectively. For boundary intersections $e \in \Gamma_B$ let E_1, F_1 denote the neighboring fine and coarse cell, respectively. Then, the error estimator takes the form

$$\begin{aligned}&\|p - p_N\|_{0,\Omega} \le \|\widetilde{p_N} - p_N\|_{0,\Omega} \\ &+ C_1 \sum_{F\in\mathcal{Z}} \left(\|f\|_{0,F}^2 + \sum_{\sigma=1}^{N_\lambda} \mu_\sigma \boldsymbol{k}^{F,\sigma} \cdot \boldsymbol{a}^F + \sum_{\sigma=1}^{N_\lambda}\sum_{\nu=1}^{N_\lambda} \mu_\sigma\mu_\nu \boldsymbol{K}^{F,\sigma,\nu}\boldsymbol{a}^F \cdot \boldsymbol{a}^F \right)^{1/2} \\ &+ C_2 \sum_{e\in\Lambda} \left(\boldsymbol{M}^{E_1,E_1}\boldsymbol{a}^{F_1} \cdot \boldsymbol{a}^{F_1} + \boldsymbol{M}^{E_2,E_2}\boldsymbol{a}^{F_2} \cdot \boldsymbol{a}^{F_2} - 2\boldsymbol{M}^{E_1,E_2}\boldsymbol{a}^{F_1} \cdot \boldsymbol{a}^{F_2} \right)^{1/2} \\ &+ C_3 \sum_{e\in\Lambda} \sum_{\sigma,\nu=1}^{N_\lambda} \mu_\sigma\mu_\nu \Big(\boldsymbol{N}^{E_1,E_1,\sigma,\nu}\boldsymbol{a}^{F_1} \cdot \boldsymbol{a}^{F_1} + \boldsymbol{N}^{E_2,E_2,\sigma,\nu}\boldsymbol{a}^{F_2} \cdot \boldsymbol{a}^{F_2} \\ &\qquad - 2\boldsymbol{N}^{E_1,E_2,\sigma,\nu}\boldsymbol{a}^{F_1} \cdot \boldsymbol{a}^{F_2} \Big)^{1/2} \\ &+ C_2 \sum_{e\in\Gamma_B} \left(\boldsymbol{P}^{E_1}\boldsymbol{\mu} \cdot \boldsymbol{\mu} + \boldsymbol{Q}^{E_1}\boldsymbol{a}^{F_1} \cdot \boldsymbol{a}^{F_1} - 2\sum_{\sigma=1}^{N_\lambda} \mu_\sigma \boldsymbol{m}^{E_1,\sigma} \cdot \boldsymbol{a}^{F_1} \right)^{1/2}\end{aligned} \tag{4.23}$$

where we used $\widetilde{p_N} = \sum_{F\in\mathcal{Z}} \sum_{i=1}^{M_F} a_i^F \widetilde{\varphi}_i^F$, $\boldsymbol{a}^F = (a_0^F, \dots, a_{M_F}^F)$ and the constants

$$C_1 = \frac{C_o^2}{k_1}, \quad C_2 = C_r\left(\frac{C_o k_2}{k_1} + h\right), \quad C_3 = (C_o + h)\frac{C_r C_o}{k_1}.$$

5 Implementation

In this chapter we will shortly describe the implementation of the presented method that was done for this work. We will provide a basic insight into the main classes and explain their usage.

All implementation was done using the Distributed and Unified Numerics Environment (DUNE)[1]. DUNE is a modular set of tools for solving Partial Differential Equations (PDEs), written in `C++`. The module DUNE-GRID provides interfaces for structured and unstructured rectangular and simplicial grids in different dimensions. An important and largely used extension to the so-called core modules is DUNE-FEM[2]. It provides tools for Finite Element methods such as function spaces, discrete function spaces, equation system solvers and others. Nearly all of our methods will, at some point, make use of the DUNE-FEM `Parameter`-singleton. It reads program parameters from a given file that can then be accessed using methods like

```
Parameter::getValue<T>("parameter_name", defaultValue).
```

The module DUNE-RB[3], implements, besides other things, reduced basis spaces and so-called discrete function lists. DUNE-RB provides most of the means needed for the reduced framework. The above-mentioned modules build the basis for the new implementation that is now to be presented. It is split up in seven major classes:

- model problem class
- high dimensional scheme
- multi-domain grid part
- constrained reduced basis space
- reduced operator
- error estimation
  - lifting
  - equation storage
  - online matrix storage
  - error estimator
- reduced basis generation.

We will now give a short description of each of them. All the new classes are implemented in a namespace "RB", we will use the notation `RB::ClassName` occasionally to avoid mix-ups with classes defined elsewhere. When helpful, we will give short, mostly

[1] http://www.dune-project.org
[2] http://dune.mathematik.uni-freiburg.de
[3] http://www.morepas.org

symbolic, code snippets that demonstrate the usage of the respective classes. Those snippets, however, will not be self-contained in that sense that used types will not be defined, for example.

5.1 Model Problem Class

This class that was heavily inspired by the matching class in the DUNE-FEMHOWTO module[2], provides all the necessary data functions like the right hand side f, the permeability k and the exact solution (if applicable). It also incorporates the parameter independent parts λ_ν and g_ν of the mobility and boundary data as well as the parameter $\boldsymbol{\mu}$. The parameter can be replaced using the function

```
ProblemInterface::replaceParameter(const MuType& mu).
```

5.2 High Dimensional Scheme

The high dimensional, FE scheme as described in (2.13) is implemented in the `Algorithm` class. This class, that was taken from the DUNE-FEMHOWTO module, implements a bracket operator that will automatically create the system matrix and right hand side vector of Equation (2.13) and solve the arising equation system. The high dimensional solution is then returned. The system matrix operator as well as the implementation of the right hand side were taken from the DUNE-FEMHOWTO module, too.

5.3 Multi-Domain Grid Part

The `RB::MultiDomainGridPart` class extends the usual DUNE-FEM grid part to situations with different subdomains consisting of a set of normal grid entities. Through the method `addToSubDomain(subDomain, entity)` entities can be added to a certain subdomain. The subdomains here are identified through integers. After entities have been added to the multi-domain grid part, different methods provide means to check for affiliation to a certain subdomain, to get the total number of subdomains and to check if an intersection of the fine grid is part of a subdomain-subdomain boundary.

Constrained Reduced Basis Space

The Constrained Reduced Basis Space (CRB Space), derived from the DUNE-FEM `DiscreteFunctionSpaceDefault`, is a discrete function space that provides the possibility to add custom base functions. These base functions are assumed to stem from the `RB::ConstrainedDiscreteFunction` class that limits a given function to a given subdomain of a multidomain grid part, such that the arising function will evaluate to zero outside of the given subdomain.

When a new base function is added to the CRB Space, the implementation will automatically determine to which subdomain this function belongs and add it to the basis of

Listing 5.1: Construction of a reduced basis space

```
// instantiate a multi domain grid part
MultiDomainGridPartType multiDomainGridPart(grid);
// mark the desired subdomains
markSubDomains(multiDomainGridPart);
// get a high dimensional discrete function space
DiscreteFunctionSpaceType feSpace(multiDomainGridPart);
// get a constrained reduced basis space
ConstrainedRBSpaceType rbSpace(feSpace);
// compute high dimensional solutions to the problem
ConstrainedDiscreteFunctionType
  baseFunction1("Base One", feSpace);
highdimensionalSolution(baseFunction1);
ConstrainedDiscreteFunctionType
baseFunction2("Base Two", feSpace);
highdimensionalSolution(baseFunction2);
// restrict the functions to matching subdomains
restrictFunction(baseFunction1, 1);
restrictFunction(baseFunction2, 2);
// add constrained discrete functions as basis
rbSpace.addBaseFunction(baseFunction1);
rbSpace.addBaseFunction(baseFunction2);
// get function in the constrained rb space
AdaptiveDiscreteFunction<ConstrainedRBSpaceType>
  reducedFunction("Reduced", rbSpace);
... // do something with the reduced function
// project the reduced function back to the fe space
rbSpace.project(reducedFunction, reconstruction);
```

that subdomain. Using this space, we can define reduced functions as demonstrated in Listing (5.1).

With the function

```
project(ReducedFunctionType& reduced,FEFunctionType& ret),
```

the CRB Space also provides the possibility to project the reduced functions back into the high dimensional function space, denoted by `feSpace` in Listing 5.1.

5.4 Reduced Operator

The class `ReducedOperator` realizes the matrix and right hand side of (3.14). The reduced operator is instantiated using a reduced basis space `rbSpace`, a model problem

Listing 5.2: Construction and usage of the reduced operator

```
// get a reduced operator
ReducedOperatorType  reducedOperator(rbSpace, problem, h);
// assemble parameter independent parts of the equation
reducedOperator.assemble();
// set the parameters for lambda and the boundary function
problem.replaceParameter(mu);
// get the system matrix
MatrixType& systemMatrix =  reducedOperator.systemMatrix();
// get the right hand side
VectorType& rhs = reducedOperator.rightHandSide();
```

`problem` and the grid width `h`. A call to `assemble()` will assemble the parameter independent parts of the equation system 3.14. When the method `systemMatrix()` is called, the parameter independent parts $\mathbf{A}^\nu$ and $\mathbf{B}^\nu$ are scaled by the parameter components μ_ν and added to the matrix $\mathbf{C}$. The same happens for the right hand side when the method `rightHandSide()` is called. Additionally, the matrix and right hand side vector can be saved to disk for later usage by calling the method `saveData()`. Both can be read from disk by giving a fourth, optional constructor argument `readData` that will read the matrices and right hand side vectors from a location specified in the parameter file.

An exemplary usage of this operator is presented in Listing (5.2)

5.5 Error Estimation

The implementation of the error estimator is split into 4 classes: The first one holds the computation of the liftings of $\widetilde{\varphi}_i^F$ and $\lambda_\sigma k \nabla \widetilde{\varphi}_i^F \cdot \boldsymbol{n}$, that is, the computation of the solutions to (4.11) and (4.13). The second one is nothing more than a simple storage and mapping solution for all liftings that need to be computed. It will not be described in more detail. The third part of the error estimator realizes the assemblation and storage of the online quantities $\|f\|_{0,F}, \boldsymbol{k}^{F,\sigma}, \boldsymbol{K}^{F,\sigma,\nu}, \boldsymbol{M}^{E_i,E_j}, \boldsymbol{N}^{E_i,E_j,\sigma,\nu}, \boldsymbol{P}^{E_i}, \boldsymbol{m}^{E_i}$ and $\boldsymbol{Q}^{E_i}$, see (4.19)–(4.22). Finally, the last part of the error estimator is the estimator itself, mainly performing the sum (4.23).

5.5.1 Liftings

There are two classes providing liftings: One for inner and for boundary intersections. As they do not differ too much, we will concentrate on the description of the lifting class for inner intersections $e \in \Gamma_I$. When constructed, the class `InnerLifting` first assembles the system matrix for the equations (4.11) and (4.13), that is the matrix

$$\boldsymbol{S} \in \mathbb{R}^{n \times n}, \quad \boldsymbol{S}_{i,j} = \int_\Omega \boldsymbol{\tau}_i \cdot \boldsymbol{\tau}_j.$$

Here, $n = |\boldsymbol{V}_{2,h}|$ denotes the size of $\boldsymbol{V}_{2,h}$ and $\{\boldsymbol{\tau}_i | i = 1, \ldots, n\}$ denotes a basis of $\boldsymbol{V}_{2,h}$. In our case we actually only need those entries of $\boldsymbol{S}$ that belong to basis functions $\boldsymbol{\tau}_i$ with support in e. For all other base functions, the matching entries in the right hand side vector will be zero. After the necessary entries of the matrix $\boldsymbol{S}$ were assembled, the right hand side of the two equations is evaluated for each base function $\boldsymbol{\tau}_i$ with support in e. The results are written into a vector. Now, the inverse of the matrix is multiplied to all vectors, giving rise to the solution vectors. After this is done, the lifting is ready for evaluation. The evaluation of the liftings is done via the methods

```
void evaluateDeltaLocal(const unsigned int i,
                        const EntityPointerType& entityPtr,
                        const DomainType& xLocal,
                        RangeType& ret) const;
```

and

```
void evaluatePsiLocal(const unsigned int i,
                      const EntityPointerType& entityPtr,
                      const unsigned int sigma,
                      const DomainType& xLocal,
                      RangeType& ret) const;
```

that evaluate the solutions to (4.11) and (4.13), respectively. For performance reasons, there are methods `replaceMap(entityPtr)` and `computeLocalIndices()` resetting static members of the lifting class for each new entity where the liftings shall be evaluated. These two methods *need* to be called each time a new fine cell is reached.

5.5.2 Online Matrix Storage

This class, as mentioned above, assembles and stores the matrices (4.19)–(4.22). New matrices are added via the method `add(...)`. This will automatically add all needed matrices for the given intersection and register them for assemblation. The assemblation itself is then performed on the call of `void assemble(reconstructedBaseFuncs)`. This method expects the reconstructed base functions as argument. These are needed for the integrals in (4.19).

5.5.3 Error Estimator

The class `RB::ErrorEstimator` mainly serves three purposes: The first one is to find and mark all intersections of the fine grid that also belong to coarse cell intersections, that is: all $e \in \Gamma_I \cap \Xi_I$. This is accomplished by the private method `init()` that is run on construction time automatically. For all such intersections found by `init()`, a new lifting is added to the `EquationStorage` class, all online matrices for the error estimator are added to the `OnlineMatrixStorage` class and the intersection is remembered in order to be able to perform the sum (4.23) later on.

The second task of the `ErrorEstimator` class is to provide an interface to the online matrix storage. After construction of the error estimator, the user needs to call the

Listing 5.3: Usage of the Error Estimator

```
// get a new error estimator, this calls the init() method
ErrorEstimator<ReducedFunctionType, ProblemType>
  errorEstimator(rbSpace, problem);
// assemble the online matrices
errorEstimator.assembleOnlineMatrices();
... // get a reduced solution
// evaluate the error estimator
double error = errorEstimator.evaluate(reducedSolution);
```

```
void assembleOnlineMatrices()
```

method that will call the `assemble()` method in the online matrix storage.

The third purpose is the actual evaluation of the error estimator, (4.23). This is done by calling the method

```
double evaluate(const ReducedFunctionType& func);
```

that returns the absolute error as defined by equation (4.8).

In Listing (5.3) we demonstrate the usage of the error estimator.

5.6 Reduced Basis Generation

In this paragraph we describe the different possibilities provided to construct a reduced basis. The class that inherits all basis construction methods is the `RBGeneration` class. It is initialized with a – possibly empty – reduced basis space and the problem. One may then use the `init()` method to initialize the reduced space with the high dimensional solution using the current parameter $\boldsymbol{\mu}$ and or the `init(DiscreteFunction& p)` method to use a custom function as initial base function. The methods

```
void detailedSimulation(DiscreteFunction& p);
```

and

```
void reducedSimulation(ReducedFunctionType& p);
```

compute the detailed solution (2.13) and reduced solution (3.13) to the main problem. The method `extend(DiscreteFunction& p)` will perform step six of the basis generation algorithm described in Section 3.3.

The most important functionality of the `RBGeneration` class, however, is the construction of the reduced basis with one of two possible algorithms: The POD-Greedy algorithm described in Section 3.3 and an algorithm using a uniform parameter grid.

5.6.1 Uniform

Although this sort of basis construction is not desirable in real applications, it may help for debugging and testing purposes. This algorithm constructs a uniform parameter space spread over a given range, computes the high dimensional solution for each parameter and adds it to the current basis, compressing the information using the POD as described beforehand. The algorithm is called via `uniformExtension()`.

5.6.2 POD-Greedy

The POD-Greedy algorithm described in Section 3.3 can be called via

```
greedyExtension(const boost::uuids::uuid& uid),
```

where uid is a unique identifier that is needed to distinguish the profile data of different POD-Greedy runs that will be written to a file "`rbgeneration_uid.log`" and "`rbgeneration_uid.paramlog`". The `greedyExtension` method expects several parameters in the DUNE-FEM Parameter singleton, those can be seen in Table 5.1.

podgreedy.paramspace.begin	The beginning of the desired parameter training set, "$0, 1, 2.5, 0$" for example
podgreedy.paramspace.end	The ending of the training set, "$1, 2, 3.5, 1$" for example
podgreedy.paramspace.size	The number of training parameters in each direction of the manifold with the corners `podgreedy.paramspace.begin` and `podgreedy.paramspace.end`
podgreedy.desiredError	The desired maximum error
podgreedy.maxRuntime	The desired maximum runtime for the algorithm
podgreedy.desiredSize	The desired maximum basis size

Table 5.1: The mandatory parameters for the POD-Greedy implementation

5.7 Post Processing Operator

As mentioned already, we did not deal with concrete choice of the post processing operator $\mathcal{R}_{1,2}$ (see Section 4.1) for this work. In the implementation we used a Finite Element scheme of order two as high dimensional scheme and thus were able to leave out the reconstruction step.

6 Numerical Experiments

In this section we will test the implementation described in Chapter 5. For this purpose we now introduce four different test problems.

6.1 Test Problems

In all of the given test scenarios, if not indicated different, we assume the domain to be $\Omega = [0,1]^2$ and the main equation (2.1) will always be the prototype for the problems considered.

6.1.1 Polynomial, Parameter Independent

This test case will help us in verifying the high dimensional (FE) scheme and to make some observations on the error estimator.

To get a facile parameter independent problem, we neglect the mobility λ and the permeability k. Furthermore we choose polynomial boundary data of degree $q \in \mathbb{N}$, such that we end up with

$$\begin{aligned} -\Delta p &= -q \cdot (q-1) x^{q-2} \qquad \text{in } \Omega, \\ p(\boldsymbol{x}) &= x^q \qquad \text{on } \partial\Omega. \end{aligned} \tag{6.1}$$

In this case, we have the exact solution

$$p(\boldsymbol{x}) = x^q.$$

6.1.2 Oscillating, Parameter Independent

The next problem, that was originally stated in [Ohl05], introduces the oscillating permeability field

$$A_{\varepsilon,\alpha}(x) = \frac{2}{3}(1+\alpha x) \cdot \left(1 + \cos\left(2\pi \frac{x}{\varepsilon}\right)^2\right),$$

where $\alpha \in [0,1]$.

In Figure 6.1 we show $A_{\varepsilon,1}$ for different values of ε. The highly oscillatory nature of $A_{\varepsilon,\alpha}$ will certainly pose a severe challenge to the high dimensional scheme.

Because of this rapid oscillation and the non-constant behaviour of $A_{\varepsilon,\alpha}$ on the large scale for $\alpha \neq 0$, this test case is already close to a real life application such as two-phase flow in a porous medium where one can observe rapid oscillations like this one.

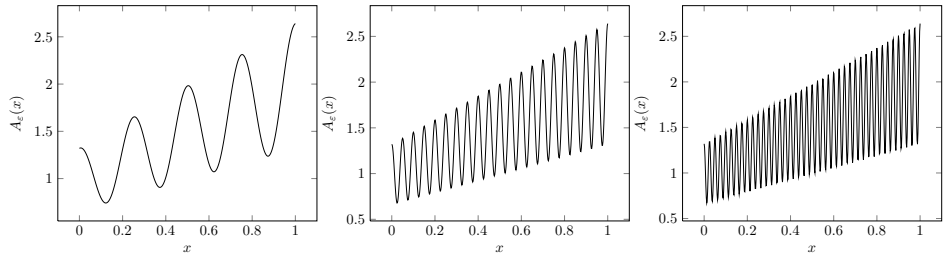

Figure 6.1: $A_{\varepsilon,1}$ for $\varepsilon = 0.5, \varepsilon = 0.1$ and $\varepsilon = 0.05$

For the permeability to fit into the framework of our theory, we set the matrix-valued function $k_{\varepsilon,\alpha} : \Omega \to \mathbb{R}^{2\times 2}$ to

$$k_{\varepsilon,\alpha}(\boldsymbol{x}) = \operatorname{diag}(A_{\varepsilon,\alpha}(x), A_{\varepsilon,\alpha}(x)). \tag{6.2}$$

Setting the right hand side to

$$f \equiv -1$$

and $\alpha = 1$ we get the problem

$$\begin{aligned} -\nabla \cdot (k_{\varepsilon,1}(\boldsymbol{x})\nabla p) &= -1 \qquad \text{in } \Omega \\ p &= 0 \qquad \text{on } \Gamma_D := 0 \times (0,1) \cup 1 \times (0,1), \\ k_{\varepsilon,1}(\boldsymbol{x})\nabla p &= 0 \qquad \text{on } \Gamma_N := \partial\Omega \backslash \Gamma_D. \end{aligned} \tag{6.3}$$

This problem has the exact solution

$$p_\varepsilon(\boldsymbol{x}) = \int_0^x \frac{t}{A_{\varepsilon,1}(t)}\mathrm{d}t + C_\varepsilon \int_0^x \frac{1}{A_{\varepsilon,1}(t)}\mathrm{d}t,$$

where

$$C_\varepsilon = -\int_0^1 \frac{t}{A_{\varepsilon,1}(t)}\mathrm{d}t \left(\int_0^1 \frac{1}{A_{\varepsilon,1}(t)}\mathrm{d}t\right)^{-1}.$$

6.1.3 Parameter Dependent

This test case will serve to give a criterion to check whether the basis generation and the reduced scheme (3.13) are working as expected. Here we forgo the permeability k and use the mobility λ as introduced in Section 2.2, that is:

$$\lambda(\boldsymbol{x}, \boldsymbol{\mu}) = \frac{1}{\eta_o} - \frac{2}{\eta_o} S(\boldsymbol{x}, \boldsymbol{\mu}) + \frac{\eta_o + \eta_w^2}{\eta_w \eta_o} \sum_{m=1}^{N_S} \sum_{n=1}^{N_S} \mu_n \mu_m S_n(\boldsymbol{x}) S_m(\boldsymbol{x}). \tag{6.4}$$

For a description of the different quantities see the above-mentioned section.

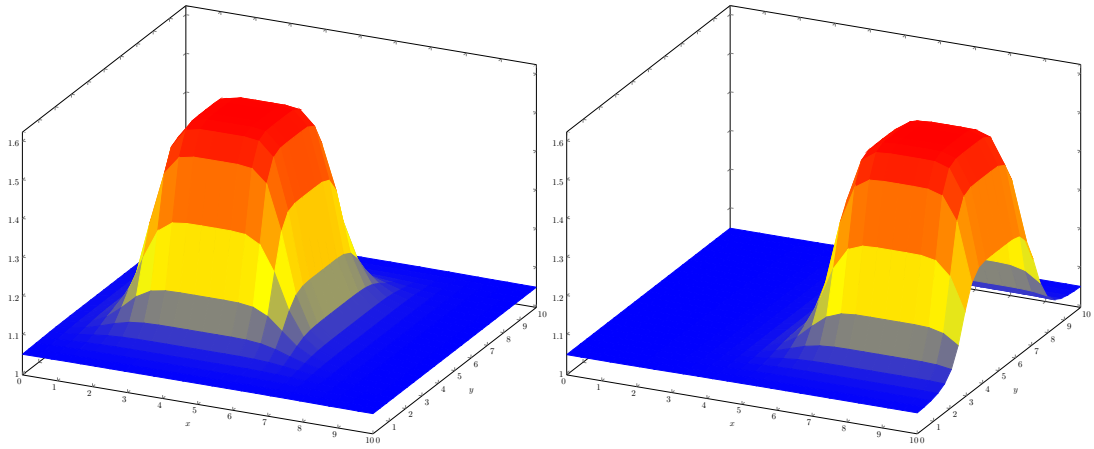

Figure 6.2: $S_1(\boldsymbol{x})$ and $S_2(\boldsymbol{x})$ (see Equation (6.5)) for $N_S = 2$

For the problem to be characterized completely we choose the right hand side

$$f(\boldsymbol{x}) = -2,$$

the boundary values

$$g_D(\boldsymbol{x}) = 0$$

and the saturation

$$S(\boldsymbol{x}, \boldsymbol{\mu}) = \sum_{\nu=1}^{N_S} \mu_\nu S_\nu(\boldsymbol{x}), \tag{6.5}$$

$$S_\nu(\boldsymbol{x}) = \frac{1}{3} \arctan\left(-\left(2\nu - \frac{N_S x}{4}\right)^6 - \frac{1}{100}(y-5)^6 \right) + \frac{\pi}{2}, \tag{6.6}$$

with a parameter $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_{N_S})$. We will use the domain $\Omega = [0, 10]^2$ in this example.

Note. For $N_S = 2$, $S_1(\boldsymbol{x})$ and $S_2(\boldsymbol{x})$ are shown in Figure 6.2.

The full problem now is: Find p such that

$$\begin{aligned} -\nabla \cdot (\lambda(\boldsymbol{x}, \boldsymbol{\mu}) \nabla p) &= -2 \qquad \text{in } \Omega, \\ p(\boldsymbol{x}) &= 0 \qquad \text{on } \partial\Omega. \end{aligned} \tag{6.7}$$

6.1.4 Oscillating, Parameter Dependent

This test case uses the parameter dependent mobility λ introduced in test case 6.1.3 and the oscillatory permeability $k_{\varepsilon,\alpha}$ with $\alpha = 0$ introduced in test case 6.1.2. With this

Level	$\|p_{\text{FE}} - p_{\text{exact}}\|_{0,\Omega}$	EOC (L^2)	$\|p_{\text{FE}} - p_{\text{exact}}\|_{1,\Omega}$	EOC (H^1)
0	$2.28 \cdot 10^{-2}$		$1.46 \cdot 10^{-1}$	
1	$5.71 \cdot 10^{-3}$	2.00	$7.24 \cdot 10^{-2}$	1.01
2	$1.43 \cdot 10^{-3}$	2.00	$3.61 \cdot 10^{-2}$	1.00
3	$3.57 \cdot 10^{-4}$	2.00	$1.80 \cdot 10^{-2}$	1.00
3	$8.91 \cdot 10^{-5}$	2.00	$9.02 \cdot 10^{-3}$	1.00
4	$2.23 \cdot 10^{-5}$	2.00	$4.51 \cdot 10^{-3}$	1.00
5	$5.57 \cdot 10^{-6}$	2.00	$2.26 \cdot 10^{-3}$	1.00
6	$1.39 \cdot 10^{-6}$	2.00	$1.13 \cdot 10^{-3}$	1.00
7	$3.48 \cdot 10^{-7}$	2.00	$5.64 \cdot 10^{-4}$	1.00
8	$8.72 \cdot 10^{-8}$	2.00	$2.82 \cdot 10^{-4}$	1.00

Table 6.1: Error convergence for the quadratic problem 6.1.1 for $q = 2$. All numbers rounded to two digits.

model problem we will demonstrate the multiscale capabilities of our method. Here we use $\Omega = [0, 10]^2$, the right hand side

$$f \equiv -2$$

and the boundary value function

$$g_D \equiv 0.$$

We get the full problem

$$\begin{aligned} -\nabla \cdot (\lambda(\boldsymbol{x}, \boldsymbol{\mu}) k_{\varepsilon,0}(\boldsymbol{x}) \nabla p) &= -2 \qquad \text{in } \Omega, \\ p(\boldsymbol{x}) &= 0 \qquad \text{on } \partial\Omega. \end{aligned} \tag{6.8}$$

6.2 Finite Element Scheme

This test deals with the high dimensional, the Finite Element scheme. We use test problem 6.1.1 for $q = 2$ and test problem 6.1.2 to verify orders of convergence one expects from theory. We use the high dimensional algorithm described in Section 5.2 with an initial triangulation consisting of two triangles. After each simulation, we apply a red-refinement to all cells, enlarging the grid by a factor of four and halving the grid width h. As we use a linear Finite Element method, we expect the order of convergence to be two, thus the error should be decreasing by a factor of four and the Experimental Order of Convergence (EOC) should be two.

For the first test case, the polynomial, parameter independent one with $q = 2$, the behaviour is exactly as expected. The numeric results for this test can be seen in Table 6.1.

h	$\epsilon_{L^2}^{0.5}$	$\epsilon_{L^2}^{0.1}$	$\epsilon_{L^2}^{0.01}$	$\epsilon_{H^1}^{0.5}$	$\epsilon_{H^1}^{0.1}$	$\epsilon_{H^1}^{0.01}$
0.71	$1.70e{-2}$	$2.29e{-2}$	$2.29e{-2}$	0.11	0.11	0.12
0.35	$6.52e{-3}$	$1.43e{-2}$	$1.43e{-2}$	$7.92e{-2}$	0.12	0.11
0.18	$2.97e{-3}$	$2.70e{-3}$	$1.29e{-2}$	$5.40e{-2}$	$5.38e{-2}$	$7.21e{-2}$
$8.84e{-2}$	$7.27e{-4}$	$1.93e{-3}$	$1.95e{-3}$	$2.62e{-2}$	$5.78e{-2}$	$5.58e{-2}$
$4.42e{-2}$	$1.77e{-4}$	$1.50e{-3}$	$1.72e{-3}$	$1.29e{-2}$	$4.70e{-2}$	$5.02e{-2}$
$2.21e{-2}$	$4.47e{-5}$	$5.40e{-4}$	$1.99e{-3}$	$6.52e{-3}$	$2.87e{-2}$	$7.70e{-2}$
$1.10e{-2}$	$1.12e{-5}$	$1.49e{-4}$	$9.19e{-4}$	$3.27e{-3}$	$1.53e{-2}$	$6.22e{-2}$
$5.52e{-3}$	$2.80e{-6}$	$3.82e{-5}$	$1.67e{-3}$	$1.63e{-3}$	$7.78e{-3}$	$5.17e{-2}$
$2.76e{-3}$	$7.41e{-7}$	$9.60e{-6}$	$7.48e{-4}$	$8.17e{-4}$	$3.90e{-3}$	$3.47e{-2}$
$1.38e{-3}$	$1.57e{-7}$	$2.39e{-6}$	$2.17e{-4}$	$4.09e{-4}$	$1.95e{-3}$	$1.89e{-2}$
$6.91e{-4}$			$5.64e{-5}$			$9.68e{-3}$
$3.45e{-4}$			$1.53e{-5}$			$4.87e{-3}$

Table 6.2: L^2-error $\epsilon_{L^2}^{\varepsilon} = \|p_{\mathrm{FE},\varepsilon} - p_{\mathrm{exact},\varepsilon}\|_{0,\Omega}$ and H^1-error $\epsilon_{H^1}^{\varepsilon} = \|p_{\mathrm{FE},\varepsilon} - p_{\mathrm{exact},\varepsilon}\|_{1,\Omega}$ between FE and exact solution for $\varepsilon = 0.5$,$\varepsilon = 0.1$ and $\varepsilon = 0.01$ for test problem 6.1.2. All numbers rounded to two digits.

We see that the EOC for the L^2 norm is approximately two (all numbers are rounded to two digits). The EOC for the H^1 norm is about one. Thus, we can say, that the scheme is working quite well for this setting.

The errors and EOC values for the second, the oscillating, parameter independent test case are to be found in Tables 6.2, 6.3. For $\varepsilon = 0.5$ (see second and fifth column in both tables), the L^2 and H^1 errors are converging nicely beginning with the third grid refinement. From then on, the EOC for the L^2 error is approximately two and the EOC for the H^1 error is approximately one, both as expected.

The low EOC values after the first and second refinements are to be explained by the large grid width ($h \approx 0.71$ and $h \approx 0.35$) in these steps: The FE scheme is not able to resolve the oscillations in the solution and thus the error shows a wrong behavior. Beginning with the third refinement, the high dimensional algorithm resolves the oscillations and the EOC reaches the desired values of two and one for the L^2 and H^1 error, respectively.

Exactly the same behavior can be observed for $\varepsilon = 0.1$ and $\varepsilon = 0.01$, only that in these cases, the stabilization of the EOC takes longer due to the finer oscillations.

In conclusion, we may say that the high dimensional scheme works fine in both (oscillating and non-oscillating) cases and produces the convergence we expect from theory.

6.3 Error Estimator

In this section, we use test cases 6.1.2, 6.1.1 to benchmark the sharpness and convergence behavior of our new error estimator. For the tests in this section, we build a basis on the unit square, divided into four equally large subdomains vertically. This basis is made up

h	$\mathrm{EOC}_{0.5}^{L^2}$	$\mathrm{EOC}_{0.1}^{L^2}$	$\mathrm{EOC}_{0.01}^{L^2}$	$\mathrm{EOC}_{0.5}^{H^1}$	$\mathrm{EOC}_{0.1}^{H^1}$	$\mathrm{EOC}_{0.01}^{H^1}$
$7.1 \cdot 10^{-1}$						
$3.5 \cdot 10^{-1}$	1.38	0.68	0.68	0.52	−0.07	0.07
$1.8 \cdot 10^{-1}$	1.14	2.4	0.15	0.55	1.13	0.63
$8.8 \cdot 10^{-2}$	2.03	0.49	2.73	1.05	−0.1	0.37
$4.4 \cdot 10^{-2}$	2.04	0.36	0.18	1.02	0.3	0.15
$2.2 \cdot 10^{-2}$	1.98	1.47	−0.21	0.99	0.71	−0.62
$1.1 \cdot 10^{-2}$	2	1.86	1.11	1	0.9	0.31
$5.5 \cdot 10^{-3}$	2	1.97	−0.86	1	0.98	0.27
$2.8 \cdot 10^{-3}$	1.92	1.99	1.16	1	0.99	0.57
$1.4 \cdot 10^{-3}$	2.24	2.01	1.78	1	1	0.88
$6.9 \cdot 10^{-4}$			1.94			0.97
$3.5 \cdot 10^{-4}$			1.88			0.99

Table 6.3: Experimental Orders of Convergence $\mathrm{EOC}_{\varepsilon}^{L^2}$ for L^2-error and $\mathrm{EOC}_{\varepsilon}^{H^1}$ for H^1-error as given by Table 6.2 for $\varepsilon = 0.5$,$\varepsilon = 0.1$ and $\varepsilon = 0.01$ for test problem 6.1.2. All numbers rounded to two digits.

by the exact solution to the respective problem, such that we have a basis of size four. We then generate a reduced solution by directly setting all of its degrees of freedom to one, such that the reconstruction of this reduced solution would be the high dimensional solution again. Proceeding this way, we do not introduce any additional errors stemming from a reduced simulation and we can be sure to get a good impression of the sharpness and also the convergence behavior of our error estimator.

In Table 6.4 we compare the exact L^2 and H^1 errors and matching EOC values to the estimated error and EOC for the estimated error for test problem 6.1.1 with $q = 4$. The exact errors were computed using a finite element method with polynomial degree one.

While we see a nearly perfect convergence of order two (in L^2) for the exact error, the estimated error shows an EOC of about one. Our experiments showed that in this scenario, the estimated error is governed by the term $\sum_{E \in \mathcal{T}_h} \eta_1^E(p_N)$ in (4.8). As we did not apply any interpolation estimates to the residual but measure it directly, this term behaves like the error in the energy norm, which converges with order one. Therefore this behavior of the error estimator is not too surprising.

We also see that our error estimator overestimates the error largely, by a factor of about 118 for the coarsest grid already and even with an increasing factor further on, due to the poor convergence order of one.

In Table 6.5 we present the same quantities as above, now using the test case 6.1.2 with $\varepsilon = 0.5$. Again, like in Section 6.2 we observe a low value for all EOCs up to the point where the grid width falls under a certain fraction of the characteristic size of the oscillation, in this case 0.5. Furthermore, this test confirms our statements concerning the sharpness and convergence order of our error estimator.

h	$\epsilon^{\text{ex}}_{L^2}$	$\text{EOC}^{\text{ex}}_{L^2}$	$\epsilon^{\text{ex}}_{H^1}$	$\text{EOC}^{\text{ex}}_{H^1}$	ϵ^{est}	EOC^{est}
0.35	$3.00e{-2}$		0.38		3.54	
0.18	$7.62e{-3}$	1.98	0.19	0.98	1.36	1.38
$8.84e{-2}$	$1.91e{-3}$	1.99	$9.67e{-2}$	1	0.56	1.27
$4.42e{-2}$	$4.78e{-4}$	2	$4.84e{-2}$	1	0.25	1.16
$2.21e{-2}$	$1.20e{-4}$	2	$2.42e{-2}$	1	0.12	1.09
$1.10e{-2}$	$2.97e{-5}$	2.01	$1.21e{-2}$	1	$5.68e{-2}$	1.05
$5.52e{-3}$	$7.28e{-6}$	2.03	$6.05e{-3}$	1	$2.79e{-2}$	1.03

Table 6.4: Exact L^2 and H^1 errors and estimated error with respective EOC values for test problem 6.1.1

h	$\epsilon^{\text{ex}}_{L^2}$	$\text{EOC}^{\text{ex}}_{L^2}$	$\epsilon^{\text{ex}}_{H^1}$	$\text{EOC}^{\text{ex}}_{H^1}$	ϵ^{est}	EOC^{est}
0.35	$6.52e{-3}$		$7.92e{-2}$		7.08	
0.18	$2.97e{-3}$	1.14	$5.40e{-2}$	0.55	9.28	-0.39
$8.84e{-2}$	$7.27e{-4}$	2.03	$2.62e{-2}$	1.05	3.73	1.31
$4.42e{-2}$	$1.77e{-4}$	2.04	$1.29e{-2}$	1.02	1.41	1.41
$2.21e{-2}$	$4.46e{-5}$	1.98	$6.52e{-3}$	0.99	0.56	1.34
$1.10e{-2}$	$1.12e{-5}$	2	$3.27e{-3}$	1	0.24	1.21
$5.52e{-3}$	$2.78e{-6}$	2.01	$1.63e{-3}$	1	0.11	1.12

Table 6.5: Exact L^2 and H^1 errors and estimated error with respective EOC values for test problem 6.1.2

6.4 Basis Generation

In this section we have a look at the POD-Greedy basis generation process. We will use test case 6.1.3 with four equally large vertical subdomains to get an impression of the construction process and the quality of the generated basis. Our experiments will reveal two major weaknesses of our theory.

The first shortcoming is the pessimistic estimation of the error between the reduced and the weak solution. The consequences can be observed in Tables 6.6, 6.7 where we present results stemming from a POD-Greedy basis generation loop *without* usage of the POD. We use test case 6.1.3 with $N_S = 2$ and $\boldsymbol{\mu} \in [0, 0.5]^2$ and a triangulation of $\Omega = [0, 10]^2$ with width $h = 0.44$ in Table 6.6 and with width $h = 0.22$ in Table 6.7.

In Table 6.6 we see that in extension step five the maximum, minimum and mean error in the current basis begin to stagnate. At this point the basis is not enlarged anymore because the function worst approximated already belongs to the basis. The huge error we see is not due to a huge distance between the respective high dimensional solution and the (reconstructed) reduced solution. In other words: it is not due to a bad representation of the high dimensional solution in the current basis. As a matter of

Step	N	N_{SD}	$\epsilon_{\max}$	$\epsilon_{\min}$	$\bar{\epsilon}$	σ	μ_1	μ_2
0	8	2,2,2,2	1.38	0.33	0.72	0.26	0	0
1	12	3,3,3,3	0.43	0.12	0.25	$8.50e{-2}$	0	0.33
2	16	4,4,4,4	0.41	0.12	0.23	$6.97e{-2}$	0.33	0
3	20	5,5,5,5	0.33	0.12	0.21	$6.16e{-2}$	0.33	0.42
4	24	6,6,6,6	0.3	0.1	0.18	$4.87e{-2}$	0.42	0
5	24	6,6,6,6	0.26	0.1	0.18	$4.32e{-2}$	0.42	0
6	24	6,6,6,6	0.26	0.1	0.18	$4.32e{-2}$	0.42	0
7	24	6,6,6,6	0.26	0.1	0.18	$4.32e{-2}$	0.42	0
8	24	6,6,6,6	0.26	0.1	0.18	$4.32e{-2}$	0.42	0
9	24	6,6,6,6	0.26	0.1	0.18	$4.32e{-2}$	0.42	0

Table 6.6: POD-Greedy algorithm: Extension step, total basis size N, basis size per subdomain, maximum error, minimum error, mean error for the training parameter set $\mathcal{M}_{\text{train}}$, standard deviation of error, components one and two of $\boldsymbol{\mu}$ used for extension.

Step	N	N_{SD}	$\epsilon_{\max}$	$\epsilon_{\min}$	$\bar{\epsilon}$	σ	μ_1	μ_2
0	8	2,2,2,2	1.38	0.3	0.71	0.27	0	0
1	12	3,3,3,3	0.45	$6.15e{-2}$	0.25	$9.70e{-2}$	0	0.33
2	16	4,4,4,4	0.52	$6.15e{-2}$	0.25	0.1	0.33	0
3	20	5,5,5,5	0.42	$6.15e{-2}$	0.23	$9.23e{-2}$	0.33	0.42
4	24	6,6,6,6	0.33	$6.15e{-2}$	0.18	$6.86e{-2}$	0.42	0
5	28	7,7,7,7	0.3	$6.15e{-2}$	0.16	$5.81e{-2}$	$8.33e{-2}$	0.42
6	28	7,7,7,7	0.24	$6.16e{-2}$	0.15	$4.46e{-2}$	$8.33e{-2}$	0.42
7	28	7,7,7,7	0.24	$6.16e{-2}$	0.15	$4.46e{-2}$	$8.33e{-2}$	0.42
8	28	7,7,7,7	0.24	$6.16e{-2}$	0.15	$4.46e{-2}$	$8.33e{-2}$	0.42
9	28	7,7,7,7	0.24	$6.16e{-2}$	0.15	$4.46e{-2}$	$8.33e{-2}$	0.42

Table 6.7: POD-Greedy algorithm: Extension step, total basis size N, basis size per subdomain, maximum error, minimum error, mean error for the training parameter set $\mathcal{M}_{\text{train}}$, standard deviation of error, components one and two of $\boldsymbol{\mu}$ used for extension.

fact the error between the high dimensional and the (reconstructed) reduced solution is pretty small. However, our error estimator computes the error between the reduced and weak solution and overestimates – as demonstrated above – this error largely. Therefore, in this case starting with extension step six, beginning with a certain richness of the reduced basis and thus small projection error to the reduced space, the error for the base functions themselves starts to shadow the projection error. Henceforward the basis is not enlarged anymore. To enrich the basis further we would need to refine the grid until the estimated error for the high dimensional solutions to all parameters drops below our

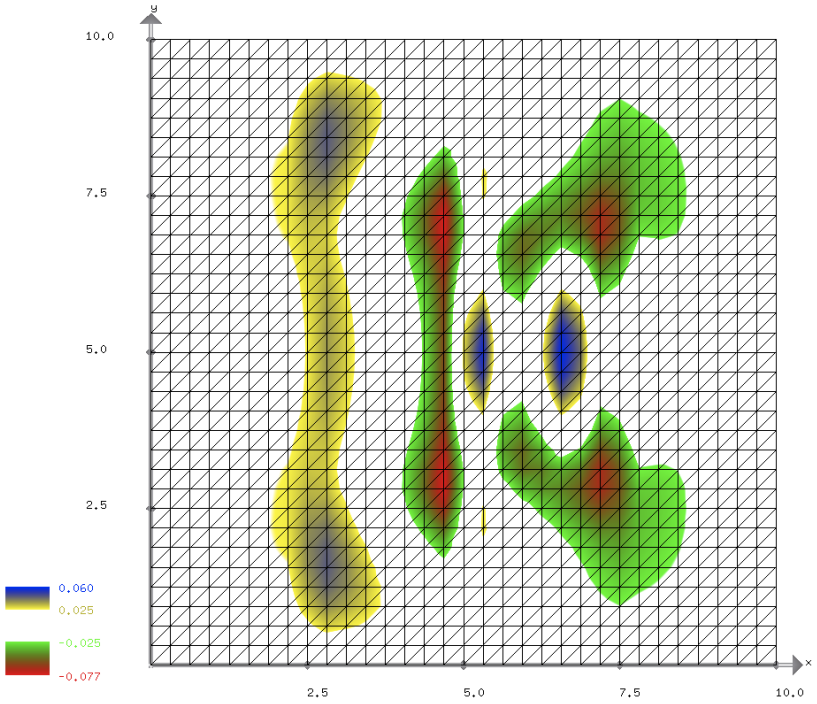

Figure 6.3: Difference between the high dimensional and reconstructed reduced solution for $\mu = (0.43, 0.43, 0.43)$ after extension step four in Table 6.8. The coarse cell intersections are located at $x = 2.5$, $x = 5.0$ and $x = 7.5$. In all white regions the absolute value of the difference is smaller than 0.025.

desired accuracy Δ which is to be reached by the basis generation algorithm (see 3.3).

The same behaviour can be observed for the POD-Greedy algorithm on the finer grid in Table 6.7.

While this first problem can be avoided easily by refining the grid until the estimated error for the basis functions themselves drops below a certain border, the next weakness is more severe and of methodic nature.

In Tables 6.8, 6.9 we see results for a basis generation loop using the full POD-Greedy algorithm for POD percentages 99.99% and 100%, respectively. For these computations we used $N_S = 3$, $\boldsymbol{\mu} \in [0.1, 0.5]^3$ and a triangulation with width $h = 0.44$.

In Table 6.8 we notice that from time to time the estimated error, especially the maximum error, increases severely from one extension to the next. Occasionally, the largest error in the current basis is even reached for parameters where the respective solutions belong to the basis already. Our experiments showed that in these cases the error almost only stems from the measurement of the jumps, that is from the terms $\eta_2^e(p_N)$ in Equation (4.9). This is not a bug or imprecision in the implementation of the error estimator. We validated this behavior by explicitly measuring the jumps in the respective reduced solutions. In Figure 6.3 we visualize the difference between the

Step	N	N_{SD}	$\epsilon_{\max}$	$\epsilon_{\min}$	$\bar{\epsilon}$	σ	μ_1	μ_2	μ_3
0	8	2/2/2/2	0.96	0.31	0.48	0.11	0.1	0.1	0.1
1	12	3/3/3/3	0.89	$9.84e{-2}$	0.41	0.18	0.1	0.43	0.43
2	16	4/4/4/4	1.17	$9.84e{-2}$	0.43	0.23	0.43	0.43	0.1
3	16	4/4/4/4	0.69	$9.49e{-2}$	0.3	0.12	0.43	0.1	0.43
4	16	4/4/4/4	1.8	0.38	0.63	0.23	0.43	0.43	0.43
5	17	4/4/5/4	3.53	0.82	1.29	0.43	0.43	0.43	0.43
6	17	4/4/5/4	0.58	$9.29e{-2}$	0.27	$9.43e{-2}$	0.43	0.1	0.43
7	17	4/4/5/4	0.59	0.15	0.3	$9.01e{-2}$	0.43	0.43	0.43
8	17	4/4/5/4	0.57	0.12	0.28	$8.76e{-2}$	0.43	0.1	0.43
9	17	4/4/5/4	0.7	0.18	0.32	$9.36e{-2}$	0.43	0.43	0.43
10	17	4/4/5/4	0.59	0.12	0.29	$9.25e{-2}$	0.43	0.1	0.43
11	17	4/4/5/4	1.3	0.33	0.53	0.15	0.43	0.43	0.43

Table 6.8: POD-Greedy algorithm for POD accuracy 99.99%: Extension step, total basis size N, basis size per subdomain, maximum error, minimum error, mean error for the training parameter set $\mathcal{M}_{\text{train}}$, standard deviation of error, components one, two and three of $\boldsymbol{\mu}$ used for extension.

high dimensional and the reconstructed reduced solution for $\boldsymbol{\mu} = (0.43, 0.43, 0.43)$ after extension step four in Table 6.8. We see, that the error is mainly located on the subdomain intersections which also seconds what was said before. The problem here is that the POD step in the algorithm destroys the continuity properties of the basis functions over the subdomain intersections, especially for decreasing POD percentages. In the results shown in Table 6.9 the effect of drastically increasing jump errors is much weaker due to the higher POD accuracy. All in all the error decay is much more stable in this case.

6.5 Multiscale Capabilities

For this test we used model problem 6.1.4 with $\boldsymbol{\mu} \in [0.1, 1.0]^2$, $\varepsilon = 1.0$ and the uniform basis generation algorithm (see Section 5.6.1) to build a reduced basis of size 23. The grid had 32768 entities and was divided into 4 equally large coarse cells vertically.

In Figure 6.4 we show the high dimensional solution (first column) and the reconstructed low dimensional solution (second column) for $\boldsymbol{\mu} = (0.1, 1.0)$ (first row) and $\boldsymbol{\mu} = (1.0, 0.1)$ (second row). The third row shows a closeup on the solutions from the second row.

We see that while we were able to reduced the number of degrees of freedom from 16641 for the high dimensional scheme to 23 for the low dimensional one, the reconstructed low dimensional solution captures the macroscopic behavior, as well as the dependence on the parameter nearly perfectly. Also, as we see in the closeup, the oscillations on the small scale are resolved correctly by the low dimensional scheme as expected.

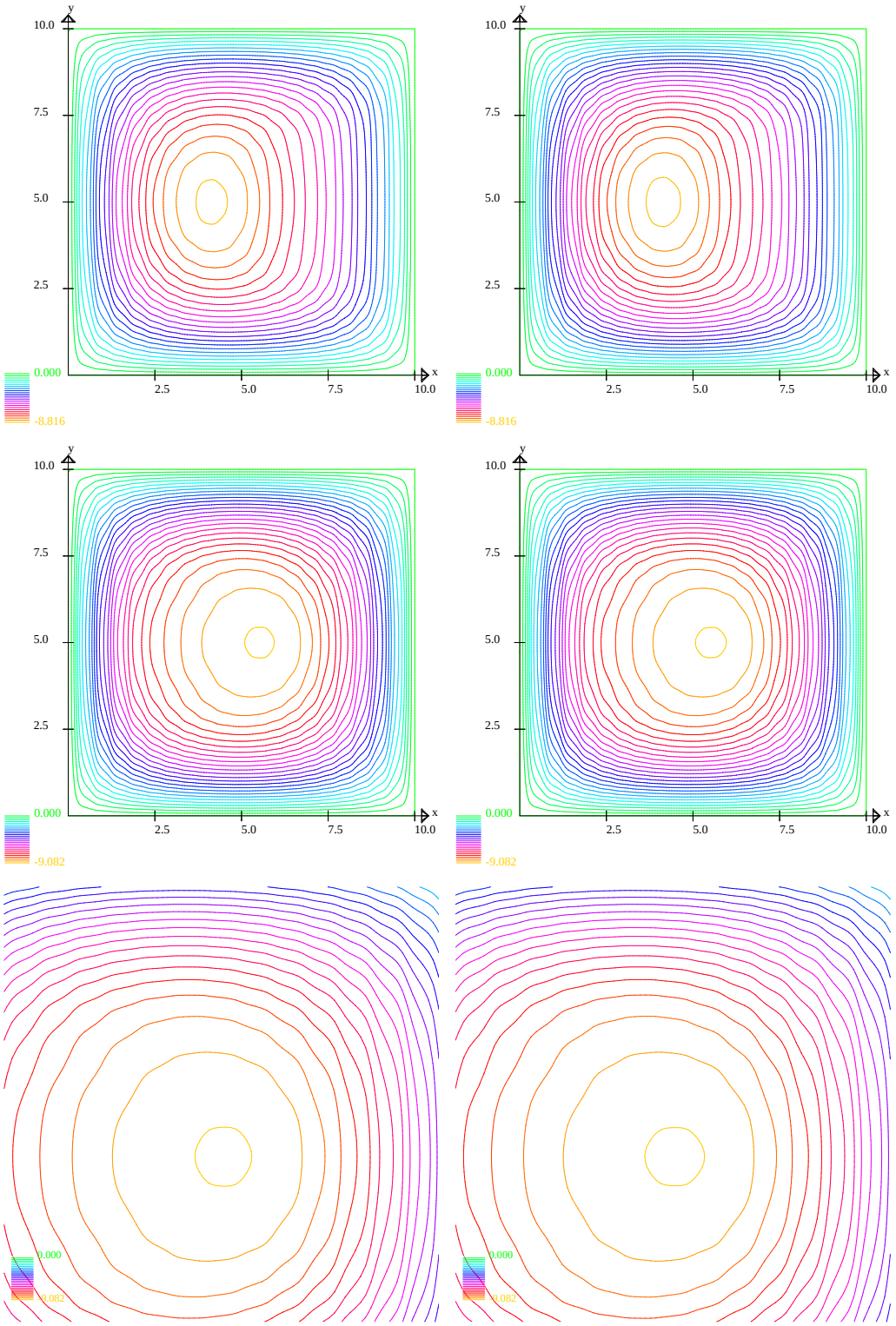

Figure 6.4: High dimensional solution (left) and reconstructed low dimensional solution (right) for $\mu = (0.1, 1.0)$, $\mu = (1.0, 0.1)$ for the test described in 6.5. Last row: closeup on row two.

Step	N	N_{SD}	$\epsilon_{\max}$	$\epsilon_{\min}$	$\bar{\epsilon}$	σ	μ_1	μ_2	μ_3
0	8	2/2/2/2	0.96	0.31	0.48	0.11	0.1	0.1	0.1
1	12	3/3/3/3	0.89	$9.84e{-2}$	0.41	0.18	0.1	0.43	0.43
2	16	4/4/4/4	1.17	$9.84e{-2}$	0.43	0.23	0.43	0.43	0.1
3	20	5/5/5/5	0.69	$9.49e{-2}$	0.3	0.12	0.43	0.1	0.43
4	24	6/6/6/6	0.34	$8.99e{-2}$	0.21	$5.17e{-2}$	0.43	0.43	0.43
5	27	7/7/7/6	0.4	$1.37e{-3}$	0.1	$8.45e{-2}$	0.43	0.1	0.43
6	27	7/7/7/6	0.37	$7.94e{-3}$	$9.09e{-2}$	$8.30e{-2}$	0.43	0.1	0.43
7	28	7/7/7/7	0.33	$1.99e{-3}$	$5.68e{-2}$	$6.78e{-2}$	0.43	0.1	0.43
8	30	8/7/7/8	0.28	$4.86e{-3}$	0.13	$5.52e{-2}$	0.43	0.1	0.43
9	33	9/8/8/8	0.25	$3.69e{-3}$	0.13	$4.17e{-2}$	0.43	0.1	0.43
10	35	9/9/8/9	0.25	$9.57e{-3}$	0.14	$4.28e{-2}$	0.43	0.1	0.43
11	36	10/9/8/9	0.24	$1.48e{-2}$	0.13	$4.62e{-2}$	0.43	0.1	0.43

Table 6.9: POD-Greedy algorithm for POD accuracy 100%: Extension step, total basis size N, basis size per subdomain, maximum error, minimum error, mean error for the training parameter set $\mathcal{M}_{\text{train}}$, standard deviation of error, components one, two and three of $\boldsymbol{\mu}$ used for extension.

Extension Step	N	Error Estimator Assemblation (s)	Training time (s)
0	8	166	3
1	12	548	3
2	16	1,163	3
3	20	2,049	4
4	24	3,217	5
5	27	4,615	7
6	27	6,034	7
7	28	6,098	7
8	30	6,252	7
9	33	6,875	8
10	35	8,814	9
11	36	9,671	9

Table 6.10: Runtimes for the POD-Greedy algorithm.

6.6 Runtimes

This section deals with the time needed for the different parts of our implementation during a typical basis generation and simulation process.

In Table 6.10 we present runtimes for the POD-Greedy basis generation loop introduced

N	t_{high} (ms)	t_{low} (ms)	t_{recons} (ms)	Factor	Error
4	65.74	1.47	$7.41e{-2}$	43	$2.24e{-2}$
8	67.26	1.57	$7.41e{-2}$	41	$4.59e{-3}$
19	64.19	2.49	1.07	18	$4.86e{-4}$

Table 6.11: Runtimes for the high and low dimensional algorithms and the reconstruction on a grid with size 2048.

in Table 6.9. In the third column we see the time needed for the assemblation of the error estimator matrices (4.19)-(4.22) in the extension step indicated in column one. We see that even for the rather coarse grid that we used in this example, the time for the assemblation of the online matrices is considerable, ranging from 166 seconds on for the smallest basis to 9671 seconds or about 2.5 hours for the largest basis. Still, this does not form a restriction in the applicability of the error estimator as the whole generation process takes place offline where time is no, or at least not a big, concern.

The evaluation of the error estimator, on the contrary, is really quick. The time needed for the reduced simulation and evaluation of the error estimator for all parameters in the training set together ranges from three to nine seconds during the basis generation. In this case we have $N_S = 3$ and six training parameters in each direction giving a total of 216 possible combinations for $\mu_1, \ldots, \mu_3$ and even more combinations for the parameters in λ (see (6.4)). Thus, the mean training time (reduced simulation and evaluation of error estimator for all parameters in the training set) of six seconds is pretty impressive.

In Table 6.11 we compare runtimes for the high dimensional algorithm to those for the reduced simulation and reconstruction for different sizes of the reduced basis on a triangulation with 2048 elements ($h \approx 0.44$). The test problem in this case was the same as for the POD-Greedy test, namely test case 6.1.3 with $N_S = 3$, $\boldsymbol{\mu} \in [0.1, 0.5]^3$. For each basis we computed the runtimes for 27 different parameters and then build the mean values to get an impression of typical runtimes. In column one we give the basis size N, in column two the runtime for the high dimensional algorithm (2.13), in column three the time needed for a solution of the reduced system (3.13), in column four the runtime for the reconstruction step, that is the projection from the reduced space back to the high dimensional space and in column five the factor between the high dimensional part and the low dimensional part which comprises the reduced simulation and reconstruction. In the last column we give the largest relative error (error divided by the norm of the reconstructed solution) between the high dimensional and reconstructed reduced solution over all test parameters. As expected, the runtime for the high dimensional algorithm mainly stays the same for all computations while the low dimensional algorithm gets more expensive from smaller to larger basis sizes. Still, even for the largest basis, the speedup by a factor of 18 is quite acceptable considering the small error of about $5 \cdot 10^{-4}$ and the small grid size.

This factor of speedup, as expected, gets much better with a larger grid size as the complexity of the low dimensional algorithm does not depend on the grid size.

N	t_{high} (ms)	t_{low} (ms)	t_{recons} (ms)	Factor	Error
4	7,219.41	1.77	29.33	232	$2.39e{-2}$
8	7,292.07	1.84	51.11	138	$5.28e{-3}$
19	7,213.48	3.39	91.19	76	$5.66e{-4}$

Table 6.12: Runtimes for the high and low dimensional algorithms and the reconstruction on a grid with size 131072.

In Table 6.12 we demonstrate this with runtimes for the same computations as before on a grid with 131072 elements ($h \approx 0.055$). Here the factor ranges from 76 to 232 as the computational cost for the high dimensional algorithm is much larger in this case. Again, even the smallest speedup of 76 is pretty convincing considering the small additional error of at most $5.6 \cdot 10^{-4}$.

7 Outlook

In this work we have introduced a new multiscale technique combining standard methods from the multiscale community with techniques from the reduced basis community. Especially in the numerical experiments we have seen the advantages but also the flaws of the approach. There are different means that can and will be applied to improve the concept.

The error estimator will be revised. The localization step in the development of the estimator turned out to be disadvantageous. This could be replaced by directly measuring the (discrete) H^{-1} norm of the residual.

A revision of the basis generation algorithm will be done, too. The POD step turned out to be destroying too much of the continuity properties of the ansatz functions. Oversampling techniques and the usage of another bilinear form in the POD step seem to be promising improvements for this step of the new approach.

Bibliography

[ABCM01] D. N. Arnold, F. Brezzi, B. Cockburn, and L. D. Marini. Unified analysis of discontinuous galerkin methods for elliptic problems. *SIAM J. Numer. Anal.*, 39(5):1749–1779 (electronic), 2001.

[AEJ08] J. E. Aarnes, Y. Efendiev, and L. Jiang. Mixed multiscale finite element methods using limited global information. *Multiscale Model. Simul.*, 7(2):655–676, 2008.

[Alt06] H. W. Alt. Lineare funktionalanalysis: Eine anwendungsorientierte einführung. page 431, Jan 2006.

[Arn82] D. N. Arnold. An interior penalty finite element method with discontinuous elements. *SIAM Journal on Numerical Analysis*, Jan 1982.

[BMM+99] F. Brezzi, M. Manzini, D. Marini, P. Pietra, and A. Russo. Discontinuous finite elements for diffusion problems. *Atti Convegno in onore di F. Brioschi (Milan 1997), Istituto Lombardo, Accademia di Scienze e Lettere*, pages 197–217, 1999.

[EEL+07] W. E, B. Engquist, X. Li, W. Ren, and E. Vanden-Eijnden. Heterogeneous multiscale methods: A review, Jan 2007.

[EH07] Y. Efendiev and T. Hou. Multiscale finite element methods for porous media flows and their applications. *Appl. Numer. Math.*, 57(5-7):577–596, 2007.

[Eva10] L. C. Evans. *Partial Differential Equations*. Jan 2010.

[HO08] B. Haasdonk and M. Ohlberger. Reduced basis method for finite volume approximations of parametrized linear evolution equations. *M2AN, Math. Model. Numer. Anal.*, 42(2):277–302, 2008.

[HS96] T. Hughes and J. Stewart. A space-time formulation for multiscale phenomena. *Journal of Computational and Applied Mathematics*, 74(1-2), Nov 1996.

[Hug95] T. Hughes. Multiscale phenomena: Green's functions, the dirichlet-to-neumann formulation, subgrid scale models, bubbles, and the origins of stabilized methods. *Comput Method Appl M*, 127(1-4):387–401, Jan 1995.

[HW97] 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*, Jan 1997.

[HWC99] T. Y. Hou, X.-H. Wu, and Z. Cai. Convergence of a multiscale finite element method for elliptic problems with rapidly oscillating coefficients. *Mathematics of Computation*, Jan 1999.

[JLT03] P. Jenny, S. Lee, and H. Tchelepi. Multi-scale finite-volume method for elliptic problems in subsurface flow simulation. *Journal of computational physics*, 187(1):47–67, Jan 2003.

[Jol02] I. Jolliffe. *Principal Component Analysis*. Jan 2002.

[MR02] Y. Maday and E. M. Rønquist. A reduced-basis element method. *C. R. Math. Acad. Sci. Paris*, 335(2):195–200, 2002.

[MR04] Y. Maday and E. Ronquist. The reduced basis element method: Application to a thermal fin problem. *SIAM J. Sci. Comput.*, 26(1):240–258, Jan 2004.

[Ohl05] M. Ohlberger. A posterior error estimates for the heterogenoeous mulitscale finite element method for elliptic homogenization problems. *SIAM Multiscale Mod. Simul.*, 1(4):1–88–114, Dec 2005.

[PR07] A. Patera and G. Rozza. *Reduced Basis Approximation and a Posteriori Error Estimation for Parametrized Partial Differential Equations*. 2007.

[Riv08] B. Rivière. *Discontinuous Galerkin methods for solving elliptic and parabolic equations*. Jan 2008.

[Whe78] M. Wheeler. An elliptic collocation-finite element method with interior penalties. *SIAM Journal on Numerical Analysis*, Jan 1978.

Erklärung

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.

Münster, den 01.03.2011

Sven Kaulmann