S. Kaulmann[a] · B. Haasdonk[a]

Online Greedy Reduced Basis Construction Using Dictionaries

Stuttgart, March 2013

[a]Institute of Applied Analysis and Numerical Simulation,
University of Stuttgart, Pfaffenwaldring 57, 70569 Stuttgart, Germany
{sven.kaulmann, haasdonk}@mathematik.uni-stuttgart.de.

Abstract The Reduced Basis method is a means for model order reduction for parametrized partial differential equations. In the last decades it has found broad application for problems with multi-query or real-time character. While the method has shown to be performing well for numerous different fields of applications, problems with high parameter dimension or high sensitivity with respect to the parameter still pose major challenges. In our contribution, we present a new basis generation algorithm that is particularly fit to these kinds of problems: Instead of building the reduced basis during the *offline* phase we build a large *dictionary* of basis vector candidates and compute a small parameter-adapted basis from that *dictionary* with a Greedy procedure during the *online* phase.

Keywords Greedy algorithms · reduced basis methods · model order reduction · dictionary

VI International Conference on Adaptive Modeling and Simulation
ADMOS 2013
J. P. Moitinho de Almeida, P. Díez, C. Tiago and N. Parés (Eds)

ONLINE GREEDY REDUCED BASIS CONSTRUCTION USING DICTIONARIES

Sven Kaulmann[*] **and Bernard Haasdonk**[*]

[*]Institute for Applied Analysis and Numerical Simulation,
University of Stuttgart,
Pfaffenwaldring 57, 70569 Stuttgart, Germany
{`sven.kaulmann|haasdonk`}`@mathematik.uni-stuttgart.de`.

Key words: Greedy algorithms, reduced basis methods, model order reduction, dictionary

Abstract. The Reduced Basis method is a means for model order reduction for parametrized partial differential equations. In the last decades it has found broad application for problems with multi-query or real-time character. While the method has shown to be performing well for numerous different fields of applications, problems with high parameter dimension or high sensitivity with respect to the parameter still pose major challenges. In our contribution, we present a new basis generation algorithm that is particularly fit to these kinds of problems: Instead of building the reduced basis during the *offline* phase we build a large *dictionary* of basis vector candidates and compute a small parameter-adapted basis from that *dictionary* with a Greedy procedure during the *online* phase.

1 Introduction and Motivation

As numerical simulations find more and more use in real-world scenarios and industrial applications, demands concerning efficiency and reliability increase as well. Especially scenarios that call for real-time simulations or multi-query evaluations of partial differential equations (PDEs) often require means of model order reduction. Examples for such scenarios are optimal control and optimization settings.

The Reduced Basis (RB) method [6] provides model order reduction for a special class of PDEs, so-called parametrized partial differential equations, (in the weak form) given as

$$B_h(u_h(\boldsymbol{\mu}), v_h; \boldsymbol{\mu}) = L_h(v_h; \boldsymbol{\mu}) \quad \forall v_h \in X_h, \tag{1}$$

for $u_h(\boldsymbol{\mu}) \in X_h$, $\boldsymbol{\mu} \in \mathcal{P} \subset \mathbb{R}^p$ and a suitable given discrete function space X_h. Here, $B_h : X_h \times X_h \times \mathcal{P} \to \mathbb{R}$ denotes a given parametrized bilinear form and $L_h : X_h \times \mathcal{P} \to \mathbb{R}$

denotes a given parametrized linear form. We will assume B_h to be coercive and symmetric in the sequel.

Such equations arise, for example, in the context of heat diffusion. In this case, the parameter $\boldsymbol{\mu}$ could model the diffusion coefficient. The RB method now reduces the complexity of Equation 1 from $\mathcal{N} = \dim(X_h)$ to $N \in \mathbb{N}$, $N \ll \mathcal{N}$, by introducing a low-dimensional surrogate $X_N, \dim(X_N) = N$ of the high-dimensional discretization space X_h. This space X_N is the span of solutions of (1) for a given set of parameter values:

$$X_N = \langle \{u_h(\boldsymbol{\mu}_1), \ldots, u_h(\boldsymbol{\mu}_N)\} \rangle, \qquad \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_N \in \mathcal{P}.$$

Galerkin-projection of Equation 1 then leads to a reduced dimensional equation system

$$\boldsymbol{A}_N(\boldsymbol{\mu}) \cdot \boldsymbol{u}_N(\boldsymbol{\mu}) = \boldsymbol{b}_N(\boldsymbol{\mu}), \tag{2}$$

where $\boldsymbol{u}_N(\boldsymbol{\mu}) \in \mathbb{R}^N$, $\boldsymbol{A}_N : \mathcal{P} \to \mathbb{R}^{N \times N}$, $(\boldsymbol{A}_N(\boldsymbol{\mu}))_{i,j} = B_h(\varphi_i, \varphi_j; \boldsymbol{\mu})$, $\boldsymbol{b}_N : \mathcal{P} \to \mathbb{R}^N$, $(\boldsymbol{b}_N(\boldsymbol{\mu}))_i = L_h(\varphi_i; \boldsymbol{\mu})$. Here, $\Phi = \{\varphi_1, \ldots \varphi_N\}$ denotes an orthonormal basis of the space X_N.

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

Offline phase During this phase, all $\mathcal{N}$-dependent computations are performed. This phase may be very expensive as a certain amount of solutions $u_h(\boldsymbol{\mu}_i)$ needs to be computed.

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

While the computation of X_N can clearly be done during the offline phase, the assembly of the equation system (2) needs to be done for every new given parameter $\boldsymbol{\mu}$ during the online phase. As this requires evaluations of B_h at the solutions $u_h(\boldsymbol{\mu}_i)$, the online phase would hence depend on $\mathcal{N}$. We thus make the assumption of parameter separability:

Assumption 1. *Assume B_h, L_h to be parameter separable, that is*

$$B_h(u, v; \boldsymbol{\mu}) = \sum_{q=1}^{Q_B} \Theta_B^q(\boldsymbol{\mu}) B_h^q(u, v), \qquad L_h(u; \boldsymbol{\mu}) = \sum_{q=1}^{Q_L} \Theta_L^q(\boldsymbol{\mu}) L_h^q(u), \quad \forall u, v \in X_h, \tag{3}$$

for given numbers $Q_B, Q_L \in \mathbb{N}$, parameter-dependent functions $\Theta_B^q, \Theta_L^q : \mathcal{P} \to \mathbb{R}$ and parameter-independent bilinear and, respectively, linear forms $B_h^q : X_h \times X_h \to \mathbb{R}$, $L_h^q : X_h \to \mathbb{R}$.

Using Assumption 1, the assembly of the system (2) can be done in two steps: During the offline phase, after computing the reduced basis space X_N, project the parameter-independent components of B_h and L_h to X_N:

$$\begin{aligned}(\boldsymbol{A}_N^q)_{i,j} &= B_h^q(\varphi_i, \varphi_j), \quad 1 \le q \le Q_B, 1 \le i, j \le N,\\ (\boldsymbol{b}_N^q)_i &= L_h^q(\varphi_i), \quad 1 \le q \le Q_L, 1 \le i \le N.\end{aligned}$$

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

$$\boldsymbol{A}_N(\boldsymbol{\mu}) = \sum_{q=1}^{Q_B} \Theta_B^q(\boldsymbol{\mu}) \boldsymbol{A}_N^q, \qquad\qquad \boldsymbol{b}_N(\boldsymbol{\mu}) = \sum_{q=1}^{Q_L} \Theta_L^q(\boldsymbol{\mu}) \boldsymbol{b}_N^q.$$

1.1 Error Estimation

One crucial ingredient of the reduced basis method is *a posteriori* error estimation. Error estimation is used for basis construction during the offline phase and for certification by approximation quality control during the online phase. In our work we use a residual-based estimator shortly outlined in this paragraph. For more details we refer to [1, 6].

Definition 1.1. For B_h given from (1) we denote by $\|\cdot\|_{\boldsymbol{\mu}} : X_h \to [0, \infty)$ the parameter dependent *energy* norm

$$\|u\|_{\boldsymbol{\mu}} = \sqrt{B_h(u, u; \boldsymbol{\mu})}. \tag{4}$$

We introduce the residual and its Riesz-representative.

Definition 1.2. For a given function $u \in X_h$ let the *residual* $r_h[u] : X_h \times \mathcal{P} \to \mathbb{R}$ be given by

$$r_h[u](v; \boldsymbol{\mu}) := L_h(v; \boldsymbol{\mu}) - B_h(u, v; \boldsymbol{\mu}) \quad \forall v \in X_h.$$

Its *Riesz-representative* $r_u(\boldsymbol{\mu}) \in X_h$, given a parameter $\overline{\boldsymbol{\mu}} \in \mathcal{P}$, is defined as the solution to

$$B_h(r_u(\boldsymbol{\mu}), v; \overline{\boldsymbol{\mu}}) = r_h[u](v; \boldsymbol{\mu}) \quad \forall v \in X_h.$$

Using the Riesz-representative we can now state our error estimator.

Theorem 1.1 (Residual based a posteriori error estimate)**.** *Given parameters $\boldsymbol{\mu}, \overline{\boldsymbol{\mu}} \in \mathcal{P}$, the energy norm of the Riesz-representative to a given reduced approximation $u_N(\boldsymbol{\mu})$ is an efficient a posteriori error estimate in the sense that*

$$\frac{1}{\gamma_{\overline{\boldsymbol{\mu}}}(\boldsymbol{\mu})} \left\|r_{u_N(\boldsymbol{\mu})}(\boldsymbol{\mu})\right\|_{\overline{\boldsymbol{\mu}}} \quad \le \quad \|u_h(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})\|_{\overline{\boldsymbol{\mu}}} \quad \le \quad \frac{1}{\alpha_{\overline{\boldsymbol{\mu}}}(\boldsymbol{\mu})} \left\|r_{u_N(\boldsymbol{\mu})}(\boldsymbol{\mu})\right\|_{\overline{\boldsymbol{\mu}}}.$$

Here we used the constants $\alpha_{\overline{\boldsymbol{\mu}}}, \gamma_{\overline{\boldsymbol{\mu}}} \in \mathbb{R}$,

$$\alpha_{\overline{\boldsymbol{\mu}}} = \inf_{u \in X_h} \frac{B_h(u,u;\overline{\boldsymbol{\mu}})}{\|u\|_{\overline{\boldsymbol{\mu}}}^2}, \quad \gamma_{\overline{\boldsymbol{\mu}}} = \sup_{u,v \in X_h} \frac{B_h(u,v;\overline{\boldsymbol{\mu}})}{\|u\|_{\overline{\boldsymbol{\mu}}} \|v\|_{\overline{\boldsymbol{\mu}}}}$$

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

Remark 1. *The constants $\alpha_{\overline{\boldsymbol{\mu}}}$,$\gamma_{\overline{\boldsymbol{\mu}}}$ can be bound by easily computable constants using the Min-Theta approach [6].*

Remark 2. *In the following we will always assume the parameter $\overline{\boldsymbol{\mu}} \in \mathcal{P}$ to be given and will use it without further notice.*

1.2 Efficient Evaluation of the Error Estimator

As a preparation of evaluations of the error estimator in a reduced space X_N we compute the components $r_L^q \in X_h, q \in \{1,\dots,Q_L\}$

$$B_h(r_L^q, v; \overline{\boldsymbol{\mu}}) = L_h^q(v) \quad \forall v \in X_h, \tag{5}$$

and the components $r_B^q \in X_h, q \in \{1,\dots,Q_B \cdot N\}$ with

$$B_h(r_B^{(j-1)\cdot N+i}, v; \overline{\boldsymbol{\mu}}) = B_h^j(\varphi_i, v) \quad \forall v \in X_h, \tag{6}$$

where $\Phi = \{\varphi_i | 1 \le i \le N\}$ is a basis of X_N. For the sake of simplicity of the following presentation, we collect the energy products of all Riesz-representatives in one matrix $\mathbf{G} \in \mathbb{R}^{Q_r \times Q_r}, Q_r = Q_L + Q_B N$:

$$\mathbf{G} = \begin{pmatrix} G_1 & G_2 \\ G_3 & G_4 \end{pmatrix}, \tag{7}$$

where

$$\begin{aligned} (G_1)_{i,j} &= B_h(r_L^i, r_L^j, \overline{\boldsymbol{\mu}}), \quad (G_2)_{i,j} = B_h(r_L^i, r_B^j, \overline{\boldsymbol{\mu}}), \\ (G_3)_{i,j} &= B_h(r_B^i, r_L^j, \overline{\boldsymbol{\mu}}), \quad (G_4)_{i,j} = B_h(r_B^i, r_B^j, \overline{\boldsymbol{\mu}}). \end{aligned}$$

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

$$(\boldsymbol{\Theta}_r(\boldsymbol{\mu}, u_N))_k = \begin{cases} \Theta_L^k(\boldsymbol{\mu}), & k \le Q_L \\ -(u_N)_i \Theta_B^j(\boldsymbol{\mu}), & \text{else} \end{cases} \tag{8}$$

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

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

1.3 Summary

Using the offline-online splitting, RB methods gain impressive complexity reductions for a wide range of applications such as elliptic stationary problems [6], parabolic instationary problems [3] and hyperbolic problems [4]. The Greedy-type algorithm used for basis construction during the offline phase [6] usually yields small bases that at the same time guarantee a small error $\Delta_{X_N}(\boldsymbol{\mu})$.

In this contribution, we investigate problems with very high sensitivity with respect to the parameter $\boldsymbol{\mu}$ that yield unfeasibly large reduced bases. For these kinds of problems, we introduce a new method for model order reduction that uses a large *dictionary* of basis vector candidates to build a small, parameter-adapted basis during the online phase. Our method holds some similarity with the *locally adaptive Greedy method* introduced in [5]. As a main difference, our method does not use proximity in parameter space as an indicator for well-suited basis candidates in the basis construction but directly measures function similarity via error estimation. This will always yield ideal basis sizes.

Further ideas about Greedy methods and dictionary approaches can be found in [7].

In Section 2 we present our dictionary construction algorithm. Section 3 is dedicated to the online basis construction procedure. Finally we present some preliminary numerical results in Section 4.

2 Dictionary Construction and Offline Data Computation

During the offline phase of our new method, we construct a "dictionary" $\mathcal{D}$ with size D of basis vector candidates $\varphi_i \in X_h$:

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

For the experiments presented in Section 4 we use a pretty straightforward algorithm to construct the dictionary: We choose a finite subset $\mathcal{S} \subset \mathcal{P}$, referred to as the *training set* in the sequel, and compute

$$\mathcal{D} = \{u_h(\boldsymbol{\mu}) \in X_h \,|\boldsymbol{\mu} \in \mathcal{S}\}. \tag{9}$$

This idea restricts the size of the training set to a certain extent as its size is directly linked to the size of the dictionary. We will comment on more elaborate methods in Section 5.

Together with the dictionary we compute the matrices $\left\{\mathbf{A}_D^q \in \mathbb{R}^{D\times D}|1 \leq q \leq Q_B\right\}$ and the vectors $\left\{\mathbf{b}_D^q \in \mathbb{R}^D|1 \leq q \leq Q_L\right\}$,

$$\begin{aligned}(\mathbf{A}_D^q)_{i.j} &= B_h^q(\varphi_i, \varphi_j), \quad \varphi_i, \varphi_j \in \mathcal{D},\\ (\mathbf{b}_D^q)_i &= L_h^q(\varphi_i), \quad \varphi_i \in \mathcal{D}\end{aligned} \tag{10}$$

that will be needed for reduced simulation during the online phase. Furthermore, we compute the matrix $\mathbf{G} \in \mathbb{R}$ from Section 1.2 for $\Phi = \mathcal{D}$.

3 Online Basis Construction

In this section we describe the Greedy algorithm that is used to construct a space $X_N(\boldsymbol{\mu})$ from the dictionary for a given parameter $\boldsymbol{\mu} \in \mathcal{P}$. As a means to this end we define a so-called *indicator function* $\eta_\Delta : X_h \times \mathcal{P} \to [0, \infty)$ that indicates the reduction of the error from Theorem 1.1 for a given Parameter $\boldsymbol{\mu}^* \in \mathcal{P}$ in a given space X_N if X_N is enlarged with $\varphi \in X_h$:

$$\eta_\Delta(\varphi, \boldsymbol{\mu}^*; X_N) = \Delta_{X_N}(\boldsymbol{\mu}^*) - \Delta_{X_N \oplus \langle\{\varphi\}\rangle}(\boldsymbol{\mu}^*). \tag{11}$$

Using this indicator for selection of basis extension candidates from the dictionary in an iterative basis construction algorithm will yield ideal basis sizes.

Algorithm 1. *Given a parameter $\boldsymbol{\mu}^* \in \mathcal{P}$, an error tolerance $\varepsilon > 0$, a desired basis size $N \in \mathbb{N}$, $n = 0$ and $X_0 = \{0\}$ we now repeat the following steps to construct a parameter-fit reduced basis space $X_N(\boldsymbol{\mu}^*)$ from a precomputed dictionary $\mathcal{D}$:*

1. *Evaluate the error estimator $\Delta_{X_n}(\boldsymbol{\mu}^*)$. If $\Delta_{X_n}(\boldsymbol{\mu}^*) < \varepsilon$ or $n \geq N$ set $X_N(\boldsymbol{\mu}^*) = X_n(\boldsymbol{\mu}^*)$ and finish, else go on with Step 2.*

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

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

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

4. *Set $n = n + 1$ and enrich the reduced space: $X_n = X_{n-1} \oplus \langle\{\varphi_{\max}\}\rangle$.*

Clearly, in a naive implementation, Step 2, which includes reduced simulation in the space $X_n \oplus \langle\{\varphi\}\rangle$ and evaluation of the error estimator for **all** dictionary elements $\varphi \in \mathcal{D}$, will be too costly to be applicable, especially for large dictionaries $\mathcal{D}$ ($\mathcal{O}(DN^4)$). We will therefore now point out how Algorithm 1 can be performed with a complexity of $\mathcal{O}(|\mathcal{D}| \cdot N^3)$ which will be favorable over the standard Greedy RB approach where the complexity is also cubic in the basis size but bases are usually a lot larger than with our method.

3.1 Simultaneous Reduced Simulation and Indicator Evaluation

As a first step for evaluation of the indicator η_Δ (11) we need to compute all reduced solutions $u_{n,\varphi} \in X_n \oplus \langle\{\varphi\}\rangle$ for all dictionary elements $\varphi \in \mathcal{D}$ and a given space X_n.

Proposition 3.1. *The solution $\mathbf{u}_{n,\varphi}(\boldsymbol{\mu})$ of Equation (2) in the space $X_n \oplus \langle\{\varphi\}\rangle$ for a given parameter $\boldsymbol{\mu} \in \mathcal{P}$ is given by*

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

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

Proof. We define the matrix $\mathbf{A}_n \in \mathbb{R}^{n\times n}$ and the vectors $\mathbf{u}_n, \mathbf{b}_n \in \mathbb{R}^n$ for the space X_n as in (2). Let

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

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

$$\mathbf{A}_{n,\varphi}\cdot\mathbf{u}_{n,\varphi} = \mathbf{b}_{n,\varphi}, \tag{12}$$

where

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

Multiplication of Equation (12) with the invertible block diagonal matrix $\operatorname{diag}(\mathbf{A}_n^{-1}, 1)$ then yields:

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

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

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

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

Which can be rewritten in the form

$$\mathbf{u}_{n,\varphi} = \begin{pmatrix}\mathbf{u}_n\\ 0\end{pmatrix} + \frac{\sigma(\varphi)-\boldsymbol{\beta}(\varphi)\mathbf{u}_n}{\gamma(\varphi)-\boldsymbol{\beta}(\varphi)\mathbf{A}_n^{-1}\boldsymbol{\alpha}(\varphi)}\cdot\begin{pmatrix}-\mathbf{A}_n^{-1}\boldsymbol{\alpha}(\varphi)\\ 1\end{pmatrix}. \tag{13}$$

$\square$

Using Proposition 3.1, only one matrix-vector multiplication and two vector-vector multiplications are needed for the computation of one reduced solution in Step (2) in Algorithm 1. Only once per loop iteration in Algorithm 1, the matrix $\mathbf{A}_n$ needs to be inverted. The quantities $\boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma, \sigma$ and $\mathbf{A}_n$ can be extracted from $\mathbf{A}_D(\boldsymbol{\mu})$,$\mathbf{b}_D(\boldsymbol{\mu})$ where

$$\mathbf{A}_D(\boldsymbol{\mu}) = \sum_{q=1}^{Q_B} \Theta_B^q(\boldsymbol{\mu})\mathbf{A}_D^q \in \mathbb{R}^{D\times D}, \quad \mathbf{b}_D^q(\boldsymbol{\mu}) = \sum_{q=1}^{Q_L} \Theta_L^q(\boldsymbol{\mu})\mathbf{b}_D^q \in \mathbb{R}^D. \tag{14}$$

Here $\{\mathbf{A}_D^q | 1 \le q \le Q_B\}, \{\mathbf{b}_D^q | 1 \le q \le Q_L\}$ are the precomputed quantities from Section 2.

In the sequel, we will outline how to efficiently evaluate the indicator function η_Δ for all possible extensions in Step (2) of Algorithm 1. When evaluating the indicator $\eta_\Delta(\varphi, \mu^*; X_n)$ for all dictionary elements φ we need to evaluate $\Delta_{X_n \oplus \langle\{\varphi\}\rangle}(\boldsymbol{\mu}^*)$ for all $\varphi \in \mathcal{D}$. The next proposition proofs that these values can be computed simultaneously for the whole dictionary.

Proposition 3.2. *For suitable choice of matrices* $\mathbf{g}_1 \in \mathbb{R}^{(n+1)\times D}$, $\mathbf{g}_2 \in \mathbb{R}^{1\times D}$ *the vector* $\Delta \in \mathbb{R}^D$ *with*

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

contains the squared error estimators for all possible basis extensions:

$$\Delta = \left(\Delta_{X_n \oplus \langle\{\varphi_1\}\rangle}(\boldsymbol{\mu}^*), \quad \cdots, \quad \Delta_{X_n \oplus \langle\{\varphi_D\}\rangle}(\boldsymbol{\mu}^*) \right). \tag{15}$$

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

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

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

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

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

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

$$\overline{\mathbf{G}} = \overline{\mathbf{G}}(\boldsymbol{\mu}^*) = \mathbf{S}(\boldsymbol{\mu}^*)^\intercal \cdot \mathbf{G} \cdot \mathbf{S}(\boldsymbol{\mu}^*) \in \mathbb{R}^{(D+1)\times(D+1)}, \tag{16}$$

with $\mathbf{G}$ as computed in Section 2. For the exposition of the rest of the simultaneous indicator evaluation we need some additional notation:

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

Assume a basis $\Phi \subset \mathcal{D}$ of the space X_n to be given. Let $I_\Phi \subset \mathbb{N}$ be an index set for Φ and $I_\mathcal{D} \subset \mathbb{N}$ be an index set for $\mathcal{D}$. Additionally we use the vectors $\mathbf{u}_{n,\varphi} = \mathbf{u}_{n,\varphi}(\boldsymbol{\mu}^*)$ from Section 3.1.

Using the above notation we can define

$$\begin{aligned}
\mathbf{g}_1 = {} & \overline{\mathbf{G}}_{[1, I_\Phi \mathbin{+\!\!+} 1],[1, I_\Phi \mathbin{+\!\!+} 1]} \cdot \begin{pmatrix} 1 & \cdots & 1 \\ -(\mathbf{u}_{n,\varphi_1})_1 & \cdots & -(\mathbf{u}_{n,\varphi_D})_1 \\ \vdots & \ddots & \vdots \\ -(\mathbf{u}_{n,\varphi_1})_n & \cdots & -(\mathbf{u}_{n,\varphi_D})_n \end{pmatrix} \\
& + \overline{\mathbf{G}}_{[1, I_\Phi \mathbin{+\!\!+} 1], I_\mathcal{D} \mathbin{+\!\!+} 1} \cdot \begin{pmatrix} -(\mathbf{u}_{n,\varphi_1})_{n+1} & & 0 \\ & \ddots & \\ 0 & & -(\mathbf{u}_{n,\varphi_1})_{n+1} \end{pmatrix}, \\
\mathbf{g}_2 = {} & \begin{pmatrix} 1 & \cdots & 1 \end{pmatrix} \cdot \left(\overline{\mathbf{G}}^\intercal_{I_\mathcal{D} \mathbin{+\!\!+} 1, [1, I_\Phi \mathbin{+\!\!+} 1]} \circ \begin{pmatrix} 1 & \cdots & 1 \\ -(\mathbf{u}_{n,\varphi_1})_1 & \cdots & -(\mathbf{u}_{n,\varphi_D})_1 \\ \vdots & \ddots & \vdots \\ -(\mathbf{u}_{n,\varphi_1})_n & \cdots & -(\mathbf{u}_{n,\varphi_D})_n \end{pmatrix} \right) \\
& + \begin{pmatrix} \overline{\mathbf{G}}_{2,2}, & \cdots, & \overline{\mathbf{G}}_{D+1,D+1} \end{pmatrix} \circ \begin{pmatrix} -(\mathbf{u}_{n,\varphi_1})_{n+1}, & \cdots, & -(\mathbf{u}_{n,\varphi_D})_{n+1} \end{pmatrix}.
\end{aligned}$$

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

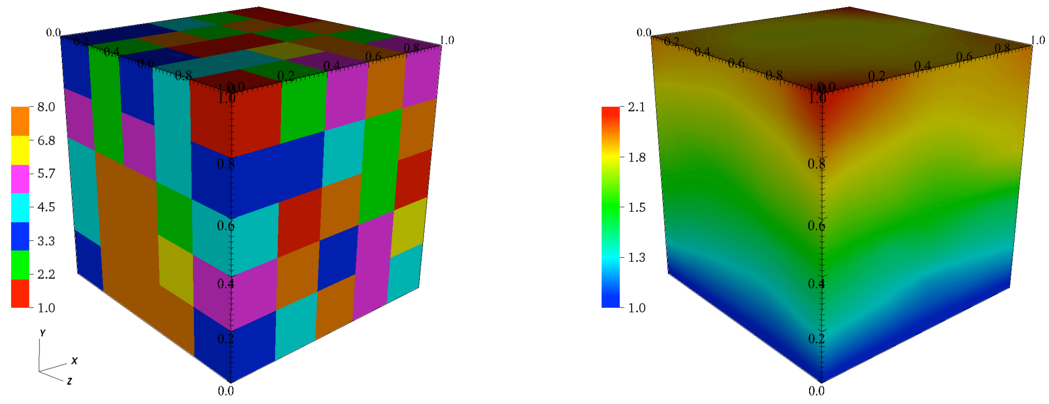

Figure 1: Heat diffusion coefficient $\lambda(\boldsymbol{\mu}) : \Omega \to \mathbb{R}$ (left), solution $u_h(\boldsymbol{\mu})$ for $\boldsymbol{\mu} = (1, 2, 3, 4, 5, 6, 7, 8)$ (right).

4 Experiments

In this section we present some preliminary numerical results for the method introduced in this paper. All tests were performed using our C++ library DUNErb, based on the *Distributed and Unified Numerics Environment* (DUNE). Both packages can be found online[1].

For our tests, we solve the heat equation on the unit cube $\Omega = [0,1]^3$. The problem statement is as follows: Find $u(\boldsymbol{\mu}) \in H_0^1(\Omega)$ such that

$$\begin{aligned} -\nabla \cdot (\lambda(\mathbf{x})\nabla u(\mathbf{x})) &= 1 \quad \text{in } \Omega, \\ u(\mathbf{x}) &= 0 \quad \text{on } \Gamma_D = [0,1] \times 0 \times [0,1], \\ \lambda(\mathbf{x})\nabla u(\mathbf{x}) \cdot \mathbf{n} &= 0 \quad \text{on } \partial\Omega \setminus \Gamma_D, \end{aligned} \tag{17}$$

where $\boldsymbol{\mu} \in \mathcal{P} = (0, 10]^8$. The heat diffusion coefficient $\lambda(\boldsymbol{\mu}) : \Omega \to \mathbb{R}$ has the form

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

where, as usual, $(\boldsymbol{\mu})_i \in \mathbb{R}$ denotes the i-th component of the vector $\boldsymbol{\mu}$ and the functions $\chi_i : \Omega \to \{0,1\}$ denote the characteristic functions for the eight subdomains of Ω sketched in the left plot in Figure 1. Furthermore, the right plot shows a typical solution for a given parameter.

4.1 Offline Phase

We discretize Ω using 1000 cubes, X_h is a linear discontinuous galerkin space with 4000 degrees of freedom. The training set $\mathcal{S} \subset \mathcal{P}$ is given by a lognormal distribution

[1]http://users.dune-project.org/

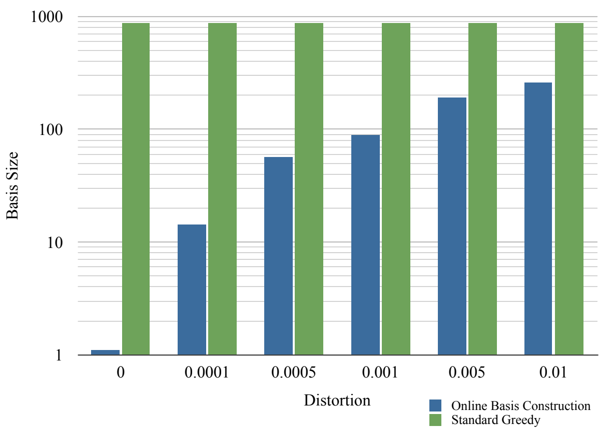

Figure 2: Mean basis size N during the online phase for the standard Greedy method and our online basis construction algorithm for different values of ρ

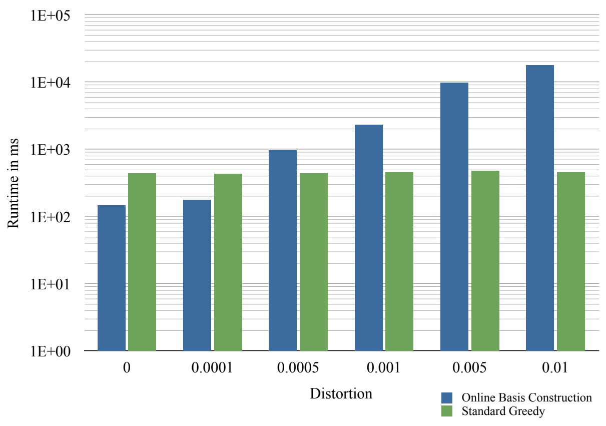

Figure 3: Mean total runtime during the online phase for the standard Greedy method and our online basis construction algorithm for different values of ρ

centered at 2 in each component. We generate a traditional [6] Greedy-basis Φ_G with a training set $\mathcal{S}_G$ with $|\mathcal{S}_G| = 1000$ and a tolerance of 10^{-5} and a dictionary $\mathcal{D}$ using the approach described in Section 2 using a training set $\mathcal{S}_\mathcal{D} \supset \mathcal{S}_G$ with $|\mathcal{S}_\mathcal{D}| = 2000$. While the generation of the traditional Greedy-basis takes more than 9 hours and produces about 600 megabytes of data, the generation of the dictionary takes only one hour but produces about 1.1 gigabytes of data.

4.2 Online Phase

Using both the basis Φ_G and the dictionary $\mathcal{D}$ we run online simulations on the test set $\mathcal{T} = \mathcal{S}_G + \rho\mathcal{R}$ where $\mathcal{R}$ contains random numbers in $(0,1]^8$ and $\rho \in \mathbb{R}$ denotes a distortion scale. For our algorithm, we use the same error tolerance as for the standard Greedy algorithm: $\varepsilon = 10^{-5}$.

Figure 2 shows the resulting basis sizes N for the standard Greedy method (which is fixed by the basis construction during the offline phase, here: $N = 871$) and the basis size N resulting from Algorithm 1 for different values of ρ. We see that, especially for small disturbances of the training parameters, our online basis generation algorithm yields substantially smaller bases.

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

5 Outlook

In our future work we will investigate two different dictionary construction algorithms: the “Offline–Greedy”-algorithm and the “Randomized Offline–Greedy”-algorithm. The “Offline–Greedy”-algorithm will use Algorithm 1 with a small maximum basis size N during the offline phase to iteratively find the parameter $\boldsymbol{\mu}$ worst approximated in the current dictionary and enrich the dictionary with $u_h(\boldsymbol{\mu})$. This algorithm will hopefully build up a dictionary that combines good approximation quality with small online basis sizes, even for large distortions ρ.

REFERENCES

[1] Felix Albrecht, Bernard Haasdonk, Sven Kaulmann, and Mario Ohlberger. The Localized Reduced Basis Multiscale Method. In Angela Handlovičová, Zuzana Minarechová, and Daniel Ševčovič, editors, *Algoritmy 2012*, pages 393–403. Slovak University of Technology in Bratislava, Publishing House of STU, April 2012.

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

[3] Bernard Haasdonk and Mario Ohlberger. Reduced basis method for finite volume approximations of parametrized linear evolution equations. *M2AN. Mathematical Modelling and Numerical Analysis*, 42(2):277–302, 2008.

[4] Bernard Haasdonk and Mario Ohlberger. Reduced basis method for explicit finite volume approximations of nonlinear conservation laws. In *Hyperbolic problems: theory, numerics and applications*, pages 605–614. Amer. Math. Soc., Providence, RI, 2009.

[5] Yvon Maday and Benjamin Stamm. Locally adaptive greedy approximations for anisotropic parameter reduced basis spaces. *arXiv.org*, math.NA, April 2012.

[6] Anthony T. Patera and Gianluigi Rozza. Reduced Basis Approximation and a Posteriori Error Estimation for Parametrized Partial Differential Equations. *Version 1.0, Copyright MIT 2006, to appear in (tentative rubric) MIT Pappalardo Graduate Monographs in Mechanical Engineering.*

[7] Vladimir Temlyakov. *Greedy approximation*, volume 20 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 2011.