Reduced Basis Method on Parameterized Domains

Bachelor Thesis

written by
Leonardo Bohnhoff

supervised by
Prof. Dr. Ralf Hiptmair
Seminar for Applied Mathematics
ETH Zurich

Abstract

This thesis discusses the reduced basis method and implements it for the Poisson equation with homogeneous Dirichlet boundary conditions on a parametrized domain based on the C++ library LehrFEM++.

Departement of Mathematics
ETH Zurich
Switzerland
April 25, 2020

Acknowledgement

I thank Prof. Ralf Hiptmair for the support of this thesis.

Contents

1 Introduction

In this thesis the reduced basis method for parametrized elliptic boundary value problems is discussed and applied to the Poisson equation with homogeneous Dirichlet boundary conditions on a parametrized domain. The method is implemented based on the C++ finite element framework LehrFEM++ developed at ETH Zurich [1].

In section 2 the variational formulation of parametrized boundary value problems is introduced and the example problem is described.

Section 3 focuses on the reduced basis method which allows to efficiently solve a parametrized boundary value problem for a large number of different parameters. The central concept of dividing the computational procedure into offline and online stages in order to ensure efficiency and the compact representation of the solution manifold by the reduced basis are discussed.

The efficiency of the reduced basis method relies on the possibility to approximate the elements of the variational problem by affine forms, which is provided by the *Empirical Interpolation Method*. This method is outlined in section 4.

Another component of the reduced basis method is the efficient a posteriori estimation of the reduction error, that is, the difference between the Galerkin solution and its approximation provided by the reduced basis method. The methods used to estimate the reduction error are described in section 5.

Section 6 deals with the implementation of the methods outlined in the previous sections.

The last section 7 presents the results of the numerical experiments about the reduced basis method as well as the *Empiricial Interpolation Method* and the *Successive Constraint Method*, the latter being applied to estimate the reduction error.

2 Parametrized variational problems

Section 2.1 introduces the abstract formulation of parametrized stationary variational problems as in [2, Section 2.1]. In section 2.2 the example problem considered throughout this thesis is described.

2.1 Abstract formulation

We consider fields $v : \Omega \to \mathbb{R}^{d_v}$ defined over the domain $\Omega \subset \mathbb{R}^d$ with $d \in \{1, 2, 3\}$ where $d_v = 1$ for scalar fields and $d_v = d$ for vector fields. The field belongs to the space

$$\mathbb{V} = \prod_{i=1}^{d_v} \mathbb{V}_i \ , \quad \mathbb{V}_i = \{v \in H^1(\Omega) : v|_{\Gamma_i} = 0\},$$

where the Γ_i are segments of $\partial\Omega$. $\mathbb{V}$ is equipped with a scalar product $(u, v)_{\mathbb{V}}$ such that the induced norm $\|v\|_{\mathbb{V}} = \sqrt{(v, v)_{\mathbb{V}}}$ is equivalent to the $(H^1(\Omega))^{d_v}$-norm. Denoting the closed parameter space by $\mathbb{P} \subset \mathbb{R}^P$, $P \in \mathbb{N}$, we can define the parametrized field variable as $u : \mathbb{P} \to \mathbb{V}$.

We also consider a parametrized bilinear form $a(\mu, \cdot, \cdot) : \mathbb{V} \times \mathbb{V} \to \mathbb{R}$ and parametrized linear forms $l(\mu, \cdot) : \mathbb{V} \to \mathbb{R}$ and $f(\mu, \cdot) : \mathbb{V} \to \mathbb{R}$ for $\mu \in \mathbb{P}$. The abstract variational formulation can then be stated as:
Given $\mu \in \mathbb{P}$, find $u(\mu) \in \mathbb{V}$ such that

$$a(\mu; u(\mu), v) = l(\mu; v) \ , \quad \forall v \in \mathbb{V}, \tag{1}$$

and evaluate the output functional

$$s(\mu) = f(\mu; u(\mu)). \tag{2}$$

The above problem is assumed to be compliant, that is, $\forall \mu \in \mathbb{P}$, the bilinear form $a(\mu; \cdot, \cdot)$ is symmetric and $l(\mu; \cdot) = f(\mu; \cdot)$.

If the following additional conditions are fulfilled, problem (1) is well-posed:

(i) $a(\mu; \cdot, \cdot)$ is coercive and continuous with respect to $\|\cdot\|_{\mathbb{V}}$ for all $\mu \in \mathbb{P}$. In particular, by the definition of coercivity, there exists $\alpha(\mu) \geq \alpha > 0$ such that

$$a(\mu; v, v) \geq \alpha(\mu) \|v\|_{\mathbb{V}}^2 \ , \quad \forall v \in \mathbb{V},$$

where the coercivity constant is defined as

$$\alpha(\mu) = \inf_{v \in \mathbb{V}} \frac{a(\mu; v, v)}{\|v\|_{\mathbb{V}}^2} \ . \tag{3}$$

(ii) $f(\mu;\cdot)$ is continuous with respect to $\|\cdot\|_{\mathbb{V}}$ for all $\mu \in \mathbb{P}$.

The induced energy norm is denoted by

$$\|v\|_a = \sqrt{a(\mu; v, v)}\,, \quad v \in \mathbb{V},$$

which is equivalent to the $\|\cdot\|_{\mathbb{V}}$-norm.

2.2 Example problem

We consider the boundary value problem

$$\begin{aligned} -\Delta u(\mu) &= 1 \quad \text{in } \Omega(\mu) \\ u(\mu) &= 0 \quad \text{on } \partial\Omega(\mu) \end{aligned} \tag{4}$$

which depends on a parameter $\mu \in \mathbb{P} = [-1,1]^P \subset \mathbb{R}^P$ for some $P \in \mathbb{N}$. The domain is defined as

$$\Omega(\mu) = \left\{x = (r\cos(\varphi), r\sin(\varphi))^T \in \mathbb{R}^2 : 0 \le r < r_B(\mu,\varphi),\ 0 \le \varphi < 2\pi\right\}$$

where

$$r_B(\mu,\varphi) = 1 + \frac{3}{\pi^2}\sum_{l=1}^{P} l^{-2}\mu_l \sin(l\varphi). \tag{5}$$

The variational formulation of (4) reads:
Given $\mu \in \mathbb{P}$, find $u(\mu) \in H_0^1(\Omega(\mu))$ such that

$$\int_{\Omega(\mu)} \nabla u(\mu)\cdot\nabla v \mathrm{dx} = \int_{\Omega(\mu)} v \mathrm{dx}\,, \quad \forall v \in H_0^1(\Omega(\mu)). \tag{6}$$

The above variational problem is transformed to a fixed reference domain which is independent of the parameters. The reference domain is chosen to be the unit disk

$$\hat{\Omega} = \left\{x \in \mathbb{R}^2 : \|x\| < 1\right\}.$$

The transformation is given by a mapping

$$\begin{aligned} \phi(\mu): \quad \hat{\Omega} &\to \Omega(\mu) \\ \hat{x} &\mapsto x \end{aligned} \tag{7}$$

such that (6) can be expressed on the reference domain:
Given $\mu \in \mathbb{P}$, find $\hat{u}(\mu) \in H_0^1(\hat{\Omega})$ such that

$$\int_{\hat{\Omega}} \hat{\boldsymbol{\alpha}}(\mu,\hat{x})\hat{\nabla}\hat{u}(\mu)\cdot\hat{\nabla}\hat{v}\mathrm{d\hat{x}} = \int_{\hat{\Omega}} \hat{\beta}(\mu,\hat{x})\hat{v}\mathrm{d\hat{x}}\,, \quad \forall \hat{v} \in H_0^1(\hat{\Omega}) \tag{8}$$

where $\hat{u}(\mu) = u(\mu) \circ \phi(\mu)$ and the coefficient functions $\hat{\boldsymbol{\alpha}} : \mathbb{P} \times \mathbb{R}^2 \to \mathbb{R}^{2,2}$ and $\beta : \mathbb{P} \times \mathbb{R}^2 \to \mathbb{R}$ are given by [3, Equation 4.3]

$$\hat{\boldsymbol{\alpha}}(\mu, \hat{x}) = D\phi(\mu)^{-1}(\hat{x}) D\phi(\mu)^{-T}(\hat{x}) |\det(D\phi(\mu))|(\hat{x}), \tag{9}$$

$$\hat{\beta}(\mu, \hat{x}) = |\det(D\phi(\mu))|(\hat{x}), \tag{10}$$

denoting the jacobian of ϕ by $D\phi$. The mapping (7) is chosen as

$$\phi(\mu, \hat{x}) = \hat{x} + \chi(\hat{x})\big(r_B(\mu, \varphi_{\hat{x}}) - 1\big)\frac{\hat{x}}{\|\hat{x}\|}$$

with r_B defined in (5) and $\varphi_{\hat{x}} = \arg(\hat{x})$ [3, Equation 3.2]. The mollifier $\chi : \hat{\Omega} \to \mathbb{R}_{\geq 0}$ is set to [3, Equation 3.3]

$$\chi(\hat{x}) = \begin{cases} 0, & \|\hat{x}\| \leq \frac{1}{4} \\ \frac{\|\hat{x}\| - \frac{1}{4}}{1 - \frac{1}{4}}, & \frac{1}{4} < \|\hat{x}\| \leq 1. \end{cases}$$

This choice of $\phi(\mu)$ guarantees the well-posedness of (8) [3, Theorem 4.3]. Further, (8) fits the setting of section (2.1) with

$$a(\mu; \hat{u}(\mu), \hat{v}) = \int_{\hat{\Omega}} \hat{\boldsymbol{\alpha}}(\mu, \hat{x}) \hat{\nabla} \hat{u}(\mu) \cdot \hat{\nabla} \hat{v} \mathrm{d}\hat{\mathrm{x}} \,, \tag{11}$$

$$l(\mu; \hat{v}) = s(\mu) = \int_{\hat{\Omega}} \hat{\beta}(\mu, \hat{x}) \hat{v} \mathrm{d}\hat{\mathrm{x}} \,, \tag{12}$$

$\mathbb{V} = H_0^1(\hat{\Omega})$ and $\Omega = \hat{\Omega}$.

3 Reduced Basis Method

The procedure of the reduced basis method (RBM) consists of two stages. During the offline stage the reduced basis is determined which should accurately represent the space of solutions to the parametrized problem. In the offline stage the problem is also approximated by an affine structure to allow the rapid evaluation of the reduced basis approximation for any admissible parameter during the online stage. In section 3.1 the reduced basis approximation is introduced. In section 3.2 a way to determine the reduced basis is presented and section 3.3 discusses the online stage. The information presented in this section can be found in [2, Section 2.2 & Chapter 3].

3.1 Reduced Basis Approximation

The finite element Galerkin discretization of the abstract variational formulation (1) results in a discrete variational problem:
Given $\mu \in \mathbb{P}$, find $u_h(\mu) \in \mathbb{V}_h \subset \mathbb{V}$ such that

$$a(\mu; u_h(\mu), v_h) = l(\mu; v_h) \ , \quad \forall v_h \in \mathbb{V}_h. \tag{13}$$

The output functional is evaluated as

$$s_h(\mu) = l(\mu; u_h(\mu)).$$

Let N_h denote the dimension of $\mathbb{V}_h$ which is spanned by some basis $\{b_h^{(l)}\}_{l=1}^{N_h}$. Problem (13) is called the truth problem as it is assumed that the error $\|u(\mu) - u_h(\mu)\|_{\mathbb{V}}$ can be made as small as desired for any $\mu \in \mathbb{P}$. The discrete solution manifold is defined as the set of solutions to (13)

$$\mathcal{M}_h = \{u_h(\mu) : \mu \in \mathbb{P}\}.$$

The goal of the reduced basis method is to represent $\mathcal{M}_h$ by a low number of basis functions so that any element of $\mathcal{M}_h$ can be approximated with a small error. The set of such basis functions is called the reduced basis. Given $N_r \ll N_h$ reduced basis functions we denote them by $\{b_r^{(n)}\}_{n=1}^{N_r}$ and the subspace they span by $\mathbb{V}_r \subset \mathbb{V}_h$. The reduced basis approximation $u_r(\mu)$ is the solution to the problem:
Given $\mu \in \mathbb{P}$, find $u_r(\mu) \in \mathbb{V}_r$ such that

$$a(\mu; u_r(\mu), v_r) = l(\mu; v_r) \ , \quad \forall v_r \in \mathbb{V}_r, \tag{14}$$

which is called the reduced problem. The reduced output functional is evaluated as

$$s_r(\mu) = l(\mu; u_r(\mu)).$$

In terms of linear algebra, the truth problem (13) can be expressed as:
Given $\mu \in \mathbb{P}$, find $\mathbf{u}_h^\mu \in \mathbb{R}^{N_h}$ such that

$$\mathbf{A}_h^\mu \mathbf{u}_h^\mu = \mathbf{l}_h^\mu \,, \tag{15}$$

where

$$(\mathbf{A}_h^\mu)_{i,j} = a(\mu; b_h^{(j)}, b_h^{(i)}) \,, \quad (\mathbf{l}_h^\mu)_i = l(\mu; b_h^{(i)}) \,, \quad i,j \in \{1,\dots,N_h\}.$$

The output functional is computed by

$$s_h^\mu = \left(\mathbf{u}_h^\mu\right)^T \mathbf{l}_h^\mu$$

and the Galerkin solution is

$$u_h(\mu) = \sum_{l=1}^{N_h} \left(\mathbf{u}_h^\mu\right)_l b_h^{(l)}.$$

Analogously, the reduced problem (14) can be written as:
Given $\mu \in \mathbb{P}$, find $\mathbf{u}_r^\mu \in \mathbb{R}^{N_r}$ such that

$$\mathbf{A}_r^\mu \mathbf{u}_r^\mu = \mathbf{l}_r^\mu \,, \tag{16}$$

where

$$(\mathbf{A}_r^\mu)_{m,n} = a(\mu; b_r^{(n)}, b_r^{(m)}) \,, \quad (\mathbf{l}_r^\mu)_m = l(\mu; b_r^{(m)}) \,, \quad m,n \in \{1,\dots,N_r\}.$$

The reduced output functional is computed by

$$s_r^\mu = \left(\mathbf{u}_r^\mu\right)^T \mathbf{l}_r^\mu, \tag{17}$$

and the reduced basis approximation is

$$u_r(\mu) = \sum_{n=1}^{N_r} \left(\mathbf{u}_r^\mu\right)_n b_r^{(n)}. \tag{18}$$

Since $\mathbb{V}_r \subset \mathbb{V}_h$ the matrix $\mathbf{B} \in \mathbb{R}^{N_h, N_r}$ can be defined by the equations

$$b_r^{(n)} = \sum_{l=1}^{N_h} (\mathbf{B})_{l,n} \, b_h^{(l)} \,, \quad n \in \{1,\dots,N_r\}. \tag{19}$$

Thus, the following relations hold

$$\begin{aligned} \mathbf{A}_r^\mu &= \mathbf{B}^T \mathbf{A}_h^\mu \mathbf{B}, \\ \mathbf{l}_r^\mu &= \mathbf{B}^T \mathbf{l}_h^\mu. \end{aligned} \tag{20}$$

3.2 Reduced Basis Space Generation

The reduced basis can be generated by a greedy algorithm. The greedy algorithm is an iterative method. At each iteration the reduced basis space is enlarged by adding a new basis function to the reduced basis. The new basis function is selected in way such that the basis becomes optimal in terms of the maximum norm over $\mathbb{P}$. Concretely, given some $\mathbb{V}_r$ of dimension n, the next basis function $b_{n+1}^{(r)}$ is determined by

$$\mu_{n+1} = \arg\max_{\mu \in \mathbb{P}} \eta(\mu)$$

as $u_h(\mu_{n+1})$ where the error estimator $\eta(\mu)$ provides an upper bound on the reduction error in the energy norm, that is,

$$\|u_h(\mu) - u_r(\mu)\|_a \leq \eta(\mu) \tag{21}$$

for all $\mu \in \mathbb{P}$. Thus, one truth problem has to be solved per iteration. The computation of the error estimator is discussed in section 5. Given that it can be computed efficiently, the parameter space can be represented by a dense discrete set $\mathbb{P}_h \subset \mathbb{P}$.

Consequently, the greedy algorithm reads as follows

Input: The targeted error tolerance tol $\in \mathbb{R}$ and some $\mu_1 \in \mathbb{P}$.
Output: A reduced basis space $\mathbb{V}_r$.

Set $n = 1$.

1. Compute the solution $u_h(\mu_n)$ to (13) and set $\mathbb{V}_r \coloneqq \operatorname{span}\{u_h(\mu_1), \ldots, u_h(\mu_n)\}$.
2. For every $\mu \in \mathbb{P}_h$ compute $\eta(\mu)$ and set $\mu_{n+1} \coloneqq \arg\max_{\mu \in \mathbb{P}_h} \eta(\mu)$.
3. If $\eta(\mu_{n+1}) <$ tol terminate, otherwise set $n \coloneqq n + 1$ and go to 1.

3.3 Online stage

To solve the reduced problem (16) for some $\mu \in \mathbb{P}$ during the online stage, it is necessary to evaluate the relations (20). As these relations are operations depending on a possibly large N_h, they do not allow the fast computation of the reduced basis approximation. This difficulty can be resolved assuming

that the following affine decompositions are available

$$a(\mu; w, v) = \sum_{q=1}^{Q_a} \theta_a^{(q)}(\mu) a_q(w, v) \tag{22}$$

$$l(\mu; v) = \sum_{q=1}^{Q_l} \theta_l^{(q)}(\mu) l_q(v) \tag{23}$$

with forms

$$a_q : \mathbb{V} \times \mathbb{V} \to \mathbb{R}\,, \quad l_q : \mathbb{V} \to \mathbb{R}$$

being independent of the parameters and

$$\theta_a^{(q)} : \mathbb{P} \to \mathbb{R}\,, \quad \theta_l^{(q)} : \mathbb{P} \to \mathbb{R}.$$

If such decompositions are not given, they can be approximately established by the *Empirical Interpolation Method* which is outlined in section 4.

Given a reduced basis space, the affine decompositions are used to precompute the following matrices and vectors during the offline stage

$$\begin{aligned} \mathbf{A}_r^{(q)} &= \mathbf{B}^T \mathbf{A}_h^{(q)} \mathbf{B}\,, && q \in \{1, \ldots, Q_a\}, \\ \mathbf{l}_r^{(q)} &= \mathbf{B}^T \mathbf{l}_h^{(q)}\,, && q \in \{1, \ldots, Q_l\}, \end{aligned}$$

where

$$(\mathbf{A}_h^{(q)})_{ij} = a_q(\mu; b_h^{(j)}, b_h^{(i)})\,, \quad (\mathbf{l}_h^{(q)})_i = l_q(\mu; b_h^{(i)})\,, \quad i, j \in \{1, \ldots, N_h\}. \tag{24}$$

Then, during the online stage, due to the relations (20), the linear systems of equations (16) is built as

$$\mathbf{A}_r^{\mu} = \sum_{q=1}^{Q_a} \theta_a^{(q)}(\mu) \mathbf{A}_r^{(q)} \tag{25}$$

$$\mathbf{l}_r^{\mu} = \sum_{q=1}^{Q_l} \theta_l^{(q)}(\mu) \mathbf{l}_r^{(q)} \tag{26}$$

for any $\mu \in \mathbb{P}$.

4 Empirical interpolation method

In general, the *Empirical Interpolation Method* (EIM) provides the approximation of some parametrized function by an affine decomposition. In section 4.1 the EIM for the case of a scalar function is presented while section 4.2 treats the case a matrix function which differs slightly from the scalar case. Section 4.3 discusses the application of the EIM to the RBM 3. The methods discussed in this section are found in [2, Chapter 5].

4.1 Scalar EIM

We consider a parametrized function

$$f : \mathbb{P} \times \Omega \to \mathbb{R}$$

such that $f_\mu \coloneqq f(\mu, \cdot) \in C^0(\overline{\Omega})$ for all $\mu \in \mathbb{P}$. The aim is to approximate this function by a sum of separable terms

$$f_\mu(x) \approx \mathrm{I}_Q[f_\mu](x) \coloneqq \sum_{q=1}^{Q} c_q(\mu) h_q(x)\,, \quad \mu \in \mathbb{P},\, x \in \Omega,$$

with basis functions $h_q : \Omega \to \mathbb{R}$ and coefficients $c_q : \mathbb{P} \to \mathbb{R}$. The interpolant $\mathrm{I}_Q[f_\mu]$ of f_μ is obtained by interpolating f_μ at interpolation points $\{x_i\}_{i=1}^{Q} \subset \Omega$ where the basis functions and the interpolations points are to be determined. Assuming they are given, the coefficients can be calculated from the interpolation conditions

$$\mathrm{I}_Q[f_\mu](x_i) = f_\mu(x_i)\,, \quad i = 1, \ldots, Q\,. \tag{27}$$

Conditions (27) are equivalent to the linear systems of equations

$$\mathbf{T}\mathbf{c}^\mu = \mathbf{f}^\mu \tag{28}$$

with

$$(\mathbf{T})_{i,q} = h_q(x_i)\,, \quad (\mathbf{c}^\mu)_q = c_q(\mu)\,, \quad (\mathbf{f}^\mu)_i = f_\mu(x_i)\,, \quad i, q \in \{1, \ldots, Q\}.$$

The basis functions and interpolation points are selected empirically by a greedy algorithm where in each iteration a new basis function and a new interpolation point are determined. Denoting the discrete set of samples by $\mathbb{P}_{\mathrm{EIM}} \subset \mathbb{P}$, the algorithm is given by

Input: A function family $f_\mu : \Omega \to \mathbb{R}$, the targeted error tolerance $\mathrm{tol}_{\mathrm{EIM}}$ and $1 \leq p \leq \infty$ specifying the norm $\|\cdot\|_{L^p(\Omega)}$ to be applied.

Output: The basis functions $\{h_i\}_{i=1}^Q$ and the interplation points $\{x_i\}_{i=1}^Q$.

Set $i = 1$. By definition $\mathrm{I}_0[f_\mu] = 0$.

1. Find

$$\mu_i = \arg\sup_{\mu \in \mathbb{P}_{\mathrm{EIM}}} \|f_\mu - \mathrm{I}_{i-1}[f_\mu]\|_{L^p(\Omega)}$$

   and use μ_i to find

$$x_i = \arg\sup_{x \in \Omega} |f_{\mu_i}(x) - \mathrm{I}_{i-1}[f_{\mu_i}](x)|.$$

2. Use μ_i and x_i to build the new basis function

$$h_i = \frac{f_{\mu_i} - \mathrm{I}_{i-1}[f_{\mu_i}]}{f_{\mu_i}(x_i) - \mathrm{I}_{i-1}[f_{\mu_i}](x_i)}. \tag{29}$$

3. Compute the error

$$e_i = \left\| \|f_\mu - \mathrm{I}_{i-1}[f_\mu]\|_{L^p(\Omega)} \right\|_{L^\infty(\mathbb{P}_{\mathrm{EIM}})} \tag{30}$$

4. If $e_i < \mathrm{tol}_{\mathrm{EIM}}$ terminate, otherwise set $i \coloneqq i + 1$ and go to 1.

By the choice of (29) it holds

$$(\mathbf{T})_{i,q} = \begin{cases} 1\,, & i = q \\ 0\,, & 1 \leq i < q \leq Q\,, \end{cases}$$

such that $\mathbf{T}$ is invertible and the linear system of equations (28) is uniquely solvable.

4.2 Matrix EIM

Similarly as in section 4.1, we consider a function

$$\boldsymbol{F} : \mathbb{P} \times \Omega \to \mathbb{R}^{K,L}$$

for which we seek an approximation of the form

$$\boldsymbol{F}_\mu(x) \coloneqq \boldsymbol{F}(\mu, x) \approx \mathrm{I}_Q[\boldsymbol{F}_\mu] \coloneqq \sum_{q=1}^{Q} c_q(\mu)\boldsymbol{H}_q(x)\,, \quad \mu \in \mathbb{P},\, x \in \Omega,$$

with $\boldsymbol{H}_q : \Omega \to \mathbb{R}^{K,L}$ and $c_q : \mathbb{P} \to \mathbb{R}$.

In contrast to the scalar case, given the set of basis functions, the interpolation conditions are not merely specified by the interpolation points $\{x_i\}_{i=1}^Q$, but additionally by index pairs $\{(k_i, l_i)\}_{i=1}^Q$ where $1 \le k_i \le K$ and $1 \le l_i \le L$ for the iteration numbers $i \in \{1, \dots, Q\}$. The tripels $\{(x_i, k_i, l_i)\}_{i=1}^Q$ can be encoded in interpolation functionals $\{\sigma_i\}_{i=1}^Q$ which are defined by

$$\begin{aligned} \sigma_i : \quad & \{\Omega \to \mathbb{R}^{K,L}\} \to \mathbb{R} \\ & \{x \mapsto \boldsymbol{M}(x)\} \mapsto (\boldsymbol{M})_{k_i,l_i}(x_i). \end{aligned} \tag{31}$$

Thus, the interpolation conditions read

$$\sigma_i\big(\mathrm{I}_Q[\boldsymbol{F}_\mu]\big) = \sigma_i(\boldsymbol{F}_\mu)\,, \quad i = 1, \dots, Q\,,$$

which again determine the interpolant $\mathrm{I}_Q[\boldsymbol{F}_\mu]$ of $\boldsymbol{F}_\mu$.

Consequently, the adapted greedy algorithm reads

Input: A function family $\boldsymbol{F}_\mu : \Omega \to \mathbb{R}^{K,L}$, the targeted error tolerance $\mathrm{tol}_{\mathrm{EIM}}$ and $1 \leq p \leq \infty$ specifying the norm $\|\cdot\|_{L^p(\Omega)}$ to be applied.

Output: The basis functions $\{\boldsymbol{H}_i\}_{i=1}^{Q}$ and the interplation functionals $\{\sigma_i\}_{i=1}^{Q}$.

Set $i = 1$. By definition $\mathrm{I}_0[\boldsymbol{F}_\mu] = 0$.

1. Find

$$\mu_i = \underset{\mu \in \mathbb{P}_{\mathrm{EIM}}}{\arg\sup} \sup_{\substack{1 \leq k \leq K \\ 1 \leq l \leq L}} \|(\boldsymbol{F}_\mu)_{k,l} - (\mathrm{I}_{i-1}[\boldsymbol{F}_\mu])_{k,l}\|_{L^p(\Omega)}$$

   and use μ_i to find

$$\sigma_i = \underset{\sigma \in \Lambda}{\arg\sup} \, |\sigma\big(\boldsymbol{F}_{\mu_i} - \mathrm{I}_{i-1}[\boldsymbol{F}_{\mu_i}]\big)|, \tag{32}$$

   where Λ denotes the set of all possible interpolation functionals. An equivalent formulation of (32) reads

$$(x_i, k_i, l_i) = \underset{\substack{(x,k,l) \in \\ \Omega \times \{1,\dots,K\} \times \{1,\dots,L\}}}{\arg\sup} |(\boldsymbol{F}_{\mu_i})_{k,l}(x) - (\mathrm{I}_{i-1}[\boldsymbol{F}_{\mu_i}])_{k,l}(x)|.$$

2. Use μ_i and σ_i to build the new basis function

$$\boldsymbol{H}_i = \frac{\boldsymbol{F}_{\mu_i} - \mathrm{I}_{i-1}[\boldsymbol{F}_{\mu_i}]}{\sigma_i\big(\boldsymbol{F}_{\mu_i} - \mathrm{I}_{i-1}[\boldsymbol{F}_{\mu_i}]\big)}. \tag{33}$$

3. Compute the error

$$e_i = \left\| \sup_{\substack{1 \leq k \leq K \\ 1 \leq l \leq L}} \|(\boldsymbol{F}_\mu)_{k,l} - (\mathrm{I}_{i-1}[\boldsymbol{F}_\mu])_{k,l}\|_{L^p(\Omega)} \right\|_{L^\infty(\mathbb{P}_{\mathrm{EIM}})} \tag{34}$$

4. If $e_i < \mathrm{tol}_{\mathrm{EIM}}$ terminate, otherwise set $i \coloneqq i + 1$ and go to 1.

Using the definitions

$$(\mathbf{T})_{i,q} = \sigma_i(\boldsymbol{H}_q)\,, \quad (\mathbf{c}^\mu)_q = c_q(\mu)\,, \quad (\mathbf{F}^\mu)_i = \sigma_i(\boldsymbol{F}_\mu)\,, \quad i,q \in \{1,\dots,Q\}.$$

and the linearity of the interpolation functionals, the coefficients for some parameter $\mu \in \mathbb{P}$ can be obtained by solving the linear systems of equations

$$\mathbf{T}\mathbf{c}^\mu = \mathbf{F}^\mu\,. \tag{35}$$

Again, due to the construction (33), it holds

$$(\mathbf{T})_{i,q} = \begin{cases} 1\,, & i = q \\ 0\,, & 1 \le i < q \le Q\,, \end{cases}$$

ensuring the invertability of $\mathbf{T}$.

4.3 EIM and RBM

To derive the affine decompositions (22) and (23) for our example problem, the EIM is applied to the coefficient functions of the forms (11) and (12)

$$\hat{\beta}(\mu,\hat{x}) \approx \sum_{q=1}^{Q_l} \theta_l^{(q)}(\mu) h_q(\hat{x})$$
$$\hat{\boldsymbol{\alpha}}(\mu,\hat{x}) \approx \sum_{q=1}^{Q_a} \theta_a^{(q)}(\mu) \boldsymbol{H}_q(\hat{x})$$

where $\mu \in \mathbb{P}$ and $\hat{x} \in \hat{\Omega}$. We then obtain the parameter-independent forms

$$a_q(\hat{w},\hat{v}) = \int_{\hat{\Omega}} \boldsymbol{H}_q(\hat{x}) \hat{\nabla}\hat{w} \cdot \hat{\nabla}\hat{v} \mathrm{d}\hat{\mathrm{x}}\,, \qquad q \in \{1,\dots,Q_a\}$$
$$l_q(\hat{v}) = \int_{\hat{\Omega}} h_q(\hat{x}) \hat{v} \mathrm{d}\hat{\mathrm{x}}\,, \qquad q \in \{1,\dots,Q_l\}$$

with $\hat{w},\hat{v} \in H_0^1(\hat{\Omega})$.

5 Error estimation

As mentioned in section 3.2, the generation of the reduced basis space in the offline stage of the RBM relies on the availability of an estimator bounding the reduction error from above. The results presented in this section can be found in [2, Chapter 4].

5.1 Error bounds

We recall from (36) that we seek an error estimator $\eta(\mu)$ which guarantees that

$$\|u_h(\mu) - u_r(\mu)\|_a \leq \eta(\mu) \tag{36}$$

for all $\mu \in \mathbb{P}$.

The error estimation makes use of the discrete coercivity constant defined by

$$\alpha_h(\mu) = \inf_{v_h \in \mathbb{V}_h} \frac{a(\mu; v_h, v_h)}{\|v_h\|_{\mathbb{V}}^2}. \tag{37}$$

As $\mathbb{V}_h \subset \mathbb{V}$, it holds that $\alpha(\mu) \leq \alpha_h(\mu)$ with the continuous coercivity constant $\alpha(\mu)$ given in (3). We also introduce the residual as

$$r(\mu; v_h) = f(\mu; v_h) - a(\mu; u_r(\mu), v_h)\,, \quad \forall v_h \in \mathbb{V}_h.$$

By the Riesz representation theorem, the unique Riesz representation $\hat{r}_h(\mu) \in \mathbb{V}_h$ of $r(\mu; \cdot)$ fulfills

$$(\hat{r}_h(\mu), v_h)_{\mathbb{V}} = r(\mu; v_h)\,, \quad \forall v_h \in \mathbb{V}_h,$$

as well as

$$\|\hat{r}_h(\mu)\|_{\mathbb{V}} = \|r(\mu; \cdot)\|_{\mathbb{V}_h'} = \sup_{v_h \in \mathbb{V}_h} \frac{r(\mu; v_h)}{\|v_h\|_{\mathbb{V}}}.$$

Further, assuming the existence of a lower bound on the discrete coercivity constant $\alpha_{LB}(\mu) \leq \alpha_h(\mu)$, we can state the following theorem.

Theorem 1. *For a compliant problem it holds*

$$\|u_h(\mu) - u_r(\mu)\|_a \leq \frac{\|\hat{r}_h(\mu)\|_{\mathbb{V}}}{\alpha_{LB}^{1/2}(\mu)}\,, \quad \forall \mu \in \mathbb{P}.$$

Thus, the required error estimator is obtained as

$$\eta(\mu) = \frac{\|\hat{r}_h(\mu)\|_{\mathbb{V}}}{\alpha_{LB}^{1/2}(\mu)}. \tag{38}$$

It remains to find efficient ways to compute the quantities constituting the error estimator, that is, with computational effort independent of the supposedly large truth dimension N_h. This is an essential goal since it allows a large training set P_h for the generation of the reduced basis. Sections 5.1.1 and 5.1.2 describe the computations of the residual in the dual norm $\|\hat{r}_h(\mu)\|_{\mathbb{V}}$ and the lower bound $\alpha_{LB}(\mu)$, respectively.

5.1.1 Residual computation

We start by plugging the affine decompositions (22) and (23) as well as the reduced basis approximation (18) into the residual

$$r(\mu; v_h) = \sum_{q=1}^{Q_l} \theta_l^{(q)}(\mu) l_q(v_h) - \sum_{q=1}^{Q_a} \sum_{n=1}^{N_r} \theta_a^{(q)}(\mu) \left(\mathbf{u}_r^{\mu}\right)_n a_q(b_r^{(n)}, v_h).$$

Let $Q_r = Q_l + Q_a N_r$. Defining

$$\mathbf{r}^{\mu} = \left(\theta_l^{(1)}(\mu), \ldots, \theta_l^{(Q_l)}(\mu), -((\mathbf{u}_r^{\mu})^T \theta_a^{(1)}(\mu), \ldots, -((\mathbf{u}_r^{\mu})^T \theta_a^{(Q_a)}(\mu)\right)^T \in \mathbb{R}^{Q_r} \tag{39}$$

and

$$R = \left(l_1, \ldots, l_{Q_l}, A_1, \ldots, A_{Q_a}\right)^T \in (\mathbb{V}_h')^{Q_r}$$

where

$$A_q = \left(a_q(b_1^{(r)}, \cdot), \ldots, a_q(b_{N_r}^{(r)}, \cdot)\right) \in (\mathbb{V}_h')^{N_r}$$

the residual can be written as

$$r(\mu; v_h) = \sum_{q=1}^{Q_r} (\mathbf{r}^{\mu})_q \, R_q(v_h).$$

Using the Riesz representation $\hat{r}_h^{(q)}$ of R_q we conclude that

$$\hat{r}_h(\mu) = \sum_{q=1}^{Q_r} (\mathbf{r}^{\mu})_q \hat{r}_h^{(q)}$$

and thus

$$\|\hat{r}_h(\mu)\|_{\mathbb{V}}^2 = \sum_{q=1}^{Q_r} \sum_{p=1}^{Q_r} (\mathbf{r}^{\mu})_q (\mathbf{r}^{\mu})_p (\hat{r}_h^{(q)}, \hat{r}_h^{(p)})_{\mathbb{V}}.$$

On the level of linear algebra, the above steps can be summarized and grouped into offline and online stages as follows. First, let a reduced basis

space $\mathbb{V}_r$ represented by the matrix B defined in (19) be given. During the offline stage assemble the matrices

$$\mathbf{R} = \left(\mathbf{l}_h^{(1)}, \ldots, \mathbf{l}_h^{(Q_f)}, \mathbf{A}_h^{(1)}\mathbf{B}, \ldots, \mathbf{A}_h^{(Q_a)}\mathbf{B}\right)^T$$

and

$$\mathbf{G} = \mathbf{R}^T \mathbf{M}_h^{-1} \mathbf{R} \tag{40}$$

where $(\mathbf{M}_h)_{i,j} = (b_h^{(j)}, b_h^{(i)})_{\mathbb{V}}$ for $i, j \in \{1, \ldots, N_h\}$. Then, during the online stage, for any $\mu \in \mathbb{P}$, compute

$$\|\hat{r}_h(\mu)\|_{\mathbb{V}} = \sqrt{(\mathbf{r}^\mu)^T \mathbf{G} \mathbf{r}^\mu},$$

where $r(\mu)$ is given in (39). The cost of the last operation does not depend on N_h.

5.1.2 Successive constraint method

The discrete coercivity constant $\alpha_h(\mu)$ defined in (37) can be computed as the smallest eigenvalue of the generalized eigenvalue problem

$$\mathbf{A}_h^\mu \mathbf{v}_h = \lambda \mathbf{M}_h \mathbf{v}_h,$$

where

$$(\mathbf{A}_h^\mu)_{i,j} = a(\mu; b_h^{(j)}, b_h^{(i)}), \quad (\mathbf{M}_h)_{i,j} = (b_h^{(j)}, b_h^{(i)})_{\mathbb{V}}, \quad i, j \in \{1, \ldots, N_h\}.$$

It can be used in place of the lower bound $\alpha_{LB}(\mu)$ in the error estimator (38), but solving generalized eigenvalue problems for the entire training set P_h is associated with a too high computational cost. The *Successive Constraint Method* (SCM) solves a few generalized eigenvalue problems during its offline stage and allows to compute $\alpha_{LB}(\mu)$ for any $\mu \in \mathbb{P}$ with a cost independent of N_h during its online stage.

By the affine decomposition of the bilinear form (22) the discrete coercivity constant is expanded as

$$\alpha_h(\mu) = \inf_{v_h \in \mathbb{V}_h} \sum_{q=1}^{Q_a} \theta_q^{(q)} \frac{a_q(v_h, v_h)}{\|v_h\|_{\mathbb{V}}^2}.$$

The SCM rewrites the above minimization problem as

$$\alpha_h(\mu) = \min_{\mathbf{y} \in \mathcal{Y}} S(\mu, \mathbf{y}),$$

where

$$S: \quad \mathbb{P} \times \mathbb{R}^{Q_a} \to \mathbb{R}$$
$$(\mu, \mathbf{y}) \mapsto S(\mu, \mathbf{y}) = \sum_{q=1}^{Q_a} \theta_q^{(a)}(\mathbf{y})_q$$

and

$$\mathcal{Y} = \Big\{ \mathbf{y} \in \mathbb{R}^{Q_a} \,\Big|\, \exists v_h \in \mathbb{V}_h : (\mathbf{y})_q = \frac{a_q(v_h, v_h)}{\|v_h\|_{\mathbb{V}}^2}, \quad q \in \{1, \ldots, Q_a\} \Big\}.$$

Lower and upper bounds $\alpha_{LB}(\mu) \leq \alpha_h(\mu) \leq \alpha_{UB}(\mu)$ can be established by respectively enlarging and restricting $\mathcal{Y}$ using the sets $\mathcal{Y}_{UB} \subset \mathcal{Y} \subset \mathcal{Y}_{LB}(\mu)$, such that

$$\alpha_{LB}(\mu) = \min_{\mathbf{y} \in \mathcal{Y}_{LB}} S(\mu, \mathbf{y}) \quad \text{and} \quad \alpha_{UB}(\mu) = \min_{\mathbf{y} \in \mathcal{Y}_{UB}} S(\mu, \mathbf{y}).$$

The SCM applies a greedy algorithm to construct at each iteration with index n the sets $\mathcal{Y}_{LB}^{(n)}$ and $\mathcal{Y}_{UB}^{(n)}$ such that

$$\mathcal{Y}_{UB}^{(1)} \subset \cdots \subset \mathcal{Y}_{UB}^{(n)} \subset \mathcal{Y} \subset \mathcal{Y}_{LB}^{(n)}(\mu) \subset \cdots \subset \mathcal{Y}_{LB}^{(1)}(\mu).$$

Thus, the indication for the sharpness of the bounds

$$1 - \frac{\alpha_{LB}(\mu)}{\alpha_{UB}(\mu)}$$

decreases with every iteration.

The set $\mathcal{Y}_{UB}^{(n)}$ is constructed given that at iteration n a set of discrete coercivity constants $\{\alpha_h(\mu_i)\}_{i=1}^n$ related to another set $\mathbb{C}_n = \{\mu_i\}_{i=1}^n \subset \mathbb{P}$ is known. It is then chosen as

$$\mathcal{Y}_{UB}^{(n)} = \Big\{ \mathbf{y}^{(i)} \in \mathbb{R}^{Q_a} : 1 \leq i \leq n \Big\}$$

where

$$(\mathbf{y}^{(i)})_q = \frac{a_q(v_h^{(i)}, v_h^{(i)})}{\left\| v_h^{(i)} \right\|_{\mathbb{V}}^2}$$

and $v_h^{(i)}$ denotes the eigenfunction associated with $\alpha_h(\mu_i)$ of the generalized eigenvalue problem

$$a(\mu_i; v_h^{(i)}, v_h) = \alpha_h(\mu_i)(v_h^{(i)}, v_h)_{\mathbb{V}}, \quad \forall v_h \in \mathbb{V}_h. \tag{41}$$

The construction of the set $\mathcal{Y}_{LB}^{(n)}(\mu)$ requires $\{\alpha_h(\mu_i)\}_{i=1}^n$ as well as lower bounds $\{\alpha_{LB}^{(n-1)}(\mu)\}_{\mu\in\mathbb{P}_{\text{SCM}}}$ from the previous iteration where $\mathbb{P}_{\text{SCM}} \subset \mathbb{P}$ denotes the discrete set of parameter samples used by the SCM. Further, it requires the mapping

$$\mathbb{P}_M(\mu; A) = \begin{cases} M \text{ closest points in A to } \mu \in A\,, & |A| > M \\ A\,, & |A| \leq M \end{cases}$$

with the discrete set $A \subset \mathbb{R}^P$. The enlarged space is then defined for some $M_e, M_p \in \mathbb{N}$ by

$$\begin{aligned} \mathcal{Y}_{LB}^{(n)}(\mu) = \Big\{ \mathbf{y} \in \mathcal{B} \Big| \quad & S(\tilde{\mu}, \mathbf{y}) \geq \alpha_h(\tilde{\mu})\,, \quad \forall \tilde{\mu} \in \mathbb{P}_{M_e}(\mu; \mathbb{C}_n), \\ & S(\tilde{\mu}, \mathbf{y}) \geq \alpha_{LB}^{(n-1)}(\tilde{\mu})\,, \quad \forall \tilde{\mu} \in \mathbb{P}_{M_p}(\mu; \mathbb{P}_{\text{SCM}} \backslash \mathbb{C}_n) \Big\} \end{aligned}$$

where the set $\mathcal{B}$ can be precomputed once as

$$\mathcal{B} = \prod_{q=1}^{Q_a} \left[\inf_{v_h \in \mathbb{V}_h} \frac{a_q(v_h, v_h)}{\|v_h\|_{\mathbb{V}}^2}, \sup_{v_h \in \mathbb{V}_h} \frac{a_q(v_h, v_h)}{\|v_h\|_{\mathbb{V}}^2} \right]$$

by solving generalized eigenvalue problems involving the bilinear forms $a_q(\cdot, \cdot)$.

We can now describe offline stage of the SCM as follows

Input: The targeted error tolerance tol_{SCM} and some initial set $\mathbb{C}_1 = \{\mu_1\}$ containing one point $\mu_1 \in \mathbb{P}_{\text{SCM}}$.

Output: The points $\mathbb{C}_n = \{\mu_i\}_{i=1}^n$ with corresponding constants $\{\alpha_h(\mu_i)\}_{i=1}^n$, the vectors $\{\mathbf{y}^{(i)}\}_{i=1}^n$ and the set of bounds $\{\alpha_{LB}^{(n)}(\mu)\}_{\mu\in\mathbb{P}_{\text{SCM}}}$.

Set $i = 1$ and $\alpha_{LB}^{(0)}(\mu) = 0$ for all $\mu \in \mathbb{P}_{\text{SCM}}$. Compute $\alpha_h(\mu_1)$ and $\mathbf{y}^{(1)}$.

1. For all $\mu \in \mathbb{P}_{\text{SCM}}$
   (a) Find $\alpha_{UB}^{(i)}(\mu) = \min_{\mathbf{y}\in\mathcal{Y}_{UB}^{(i)}} S(\mu, \mathbf{y})$.
   (b) Find $\alpha_{LB}^{(i)}(\mu) = \min_{\mathbf{y}\in\mathcal{Y}_{LB}^{(i)}(\mu)} S(\mu, \mathbf{y})$.
   (c) Compute $e^{(i)}(\mu) = 1 - \frac{\alpha_{LB}^{(i)}(\mu)}{\alpha_{UB}^{(i)}(\mu)}$
2. Choose $\mu_{i+1} = \arg\max_{\mu\in\mathbb{P}_{\text{SCM}}} e^{(i)}(\mu)$.
3. If $e^{(i)}(\mu_{i+1}) \leq \text{tol}_{\text{SCM}}$, terminate, otherwise go to 4.
4. Compute $\alpha_h(\mu_{i+1})$ out of (41) where the left hand side matrix $\mathbf{A}_h^{\mu_{i+1}}$ is assembled as

$$\mathbf{A}_h^{\mu_{i+1}} = \sum_{q=1}^{Q_a} \theta_a^{(q)}(\mu_{i+1}) \mathbf{A}_h^{(q)}.$$

   Calculate $\mathbf{y}^{(i+1)}$. Then set $i \coloneqq i + 1$ and go to 1.

Then, during the online stage, given $\{\alpha_h(\mu_i)\}_{i=1}^n$ and $\{\alpha_{LB}^{(n)}(\mu)\}_{\mu\in\mathbb{P}_{\text{SCM}}}$ from the offline stage, the desired lower bound is computed for any $\mu \in \mathbb{P}$ as

$$\alpha_{LB}(\mu) = \min_{\mathbf{y}\in\mathcal{Y}_{LB}^{(n)}(\mu)} S(\mu, \mathbf{y})$$

with the additional constraint $S(\mu, \mathbf{y}) \geq 0$. Computing lower bounds in the offline and online stages means solving linear programs with Q_a variables and $2Q_a + M_e + M_p$ constraints.

6 Implementation

The procedure of the RBM builds upon two independent components, namely the computation of the truth solution and the transformation of the truth problem into an affine form. The implementation of these two basic components is outlined in section 6.1 and 6.2, respectively. In section 6.3 the implementation of the reduced basis method is discussed.

6.1 Truth problem

The solution to the truth problem (15) is computed by the class `TruthSolver`. It is tailored to the example problem (8) and thus needs to be provided with the necessary information about it, for example, the coefficient functions $\hat{\boldsymbol{\alpha}}$ and $\hat{\beta}$. This information is collected in a struct called `TruthData` which is passed to the constructor of `TruthSolver`. The information encapsulated in `TruthData` can also be retrieved from objects of type `TruthSolver`.

6.2 EIM

The EIM can be applied to a scalar function by the function `ApplyScalarEIM`. To decompose a matrix function $\boldsymbol{F}_\mu : \hat{\Omega} \to \mathbb{R}^{2,2}$ the function `ApplyMatrixEIM` is used. Both of these functions are overloaded as they either use the $L^2(\hat{\Omega})$-norm or the $L^\infty(\hat{\Omega})$-norm depending on whether a mesh together with a quadrature rule for the computation of the $L^2(\hat{\Omega})$-norm are provided. They return objects of type `ScalarEIMBasisProvider` and `MatrixEIMBasisProvider`, respectively. The class `ScalarEIMBasisProvider` stores as members the interpolation points and the basis functions of the interpolant which are determined by the EIM. It provides the evaluation of the basis functions and computes the coefficients of the interpolant such that the interpolant can be retrieved. The class `MatrixEIMBasisProvider` offers the same functionality but in place of the interpolation points it stores the interpolation functionals. Further, the pair `ApplyMatrixEIM` and `MatrixEIMBasisProvider` is implemented for symmetric functions as the coefficient function $\hat{\boldsymbol{\alpha}}$ defined in (9) is symmetric. A symmetric function reduces the amount of work carried out by the EIM since three instead of four components need to be determined to make up

the basis functions

$$\begin{aligned}\mathrm{I}_Q[\boldsymbol{F}_\mu](\hat{x}) &= \sum_{q=1}^{Q} c_q(\mu)\boldsymbol{H}_q(\hat{x}) \\ &= \sum_{q=1}^{Q} c_q(\mu)\Bigg((\boldsymbol{H}_q)_{1,1}(\hat{x}) \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \quad + (\boldsymbol{H}_q)_{1,2}(\hat{x}) \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \\ &\qquad\qquad + (\boldsymbol{H}_q)_{2,2}(\hat{x}) \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \Bigg).\end{aligned}$$

Both classes precompute and store as a member the matrix $\mathbf{T}$ defined in (28) for scalar functions and in (35) for matrix functions.

6.3 RBM

The class `ReducedBasisHandler` takes care of the reduced basis. Together with the function `PerformGreedyAlgorithm` it is in charge of the construction of the reduced basis space and it provides the reduced basis approximation.

We recall that the EIM as well as the offline stage of the SCM have to be performed only once during the offline stage of the RBM. Thus, upon construction, objects of type `ReducedBasisHandler` perform the EIM and the offline stage of the SCM. They have access to the coefficient functions $\hat{\boldsymbol{\alpha}}$ and $\hat{\beta}$ since they are given a reference to an object of type `TruthSolver`. The functions `ApplyScalarEIM` and `ApplyMatrixEIM` are applied and the resulting objects of type `ScalarEIMBasisProvider` and `MatrixEIMBasisProvider` are stored as members of the class. The latter objects are also used to assemble and store the matrices $\{\mathbf{A}_h^{(q)}\}_{q=1}^{Q_a}$ and vectors $\{\mathbf{l}_h^{(q)}\}_{q=1}^{Q_l}$ of the affine decomposition. Similarly, the quantities supplied by the offline stage of the SCM are stored.

Once an object of type `ReducedBasisHandler` is created, it is passed to the function `PerformGreedyAlgorithm` to assemble the reduced basis by the greedy algorithm. The class `ReducedBasisHandler` stores the set of basis functions determined during the greedy algorithm and provides a method to add a new basis function. Whenever this method is called, the new basis function is computed by using the object of type `TruthSolver` and the offline stage of the residual computation is rerun, that is, the matrix $\mathbf{G}$ given in (40) is recomputed.

Computing the error estimator for the training set $\mathbb{P}_h$ in each iteration of the greedy algorithm corresponds to the online stage of both the residual computation and the SCM. The error estimator is provided by the class

`ReducedBasisHandler` as it stores the quantities related to the offline stages of the SCM and the residual computation.

When the greedy algorithm terminates, the object of type `ReducedBasisHandler` holds the reduced basis space and provides methods to compute the reduced output functional (17) as well as the reduced basis approximation (18) which is provided by its coefficients with respect to the truth basis $\{b_h^{(l)}\}_{l=1}^{N_h}$.

To solve the linear programs occurring in the SCM, the implementation relies on the numerical C++ library PNL [4].

7 Numerical experiments

In this section the results of some numerical experiments for the example problem outlined in 2.2 are presented. In all experiments, the number of parameter samples of the different algorithms are chosen as $|\mathbb{P}_{\text{EIM}}| = P \cdot 100$ and $|\mathbb{P}_{\text{SCM}}| = |\mathbb{P}_h| = P \cdot 500$ where P stands for the number of parameters. The samples are generated uniformly at random. Further, the error tolerances are chosen as $\text{tol}_{\text{EIM}} = 10^{-4}$ and $\text{tol}_{\text{SCM}} = 0.05$. A triangular mesh $\mathcal{M}$ over the unit disk $\hat{\Omega}$ is used that counts $N_h = 13071$ degrees of freedom. We choose linear Lagrangian finite elements to compute the truth solution $u_h(\mu) \in \mathbb{V}_h = \mathcal{S}^0_{1,0}(\mathcal{M})$. The parameters of the SCM are set as $M_e = M_p = 10$.

7.1 Truth problem

It is important to ensure the correct implementation of the truth problem such that the error between the truth solution and the exact solution can be made small. Otherwise the reduced basis approximation will converge to a solution deviating from the exact solution.

The method of manufactured solutions is applied for validation [5, Section 5.8]. We define the solution on $\Omega(\mu)$ as

$$u(x_1, x_2) = (x_1^2 + x_2^2)\sin(x_1)\sin(x_2)$$

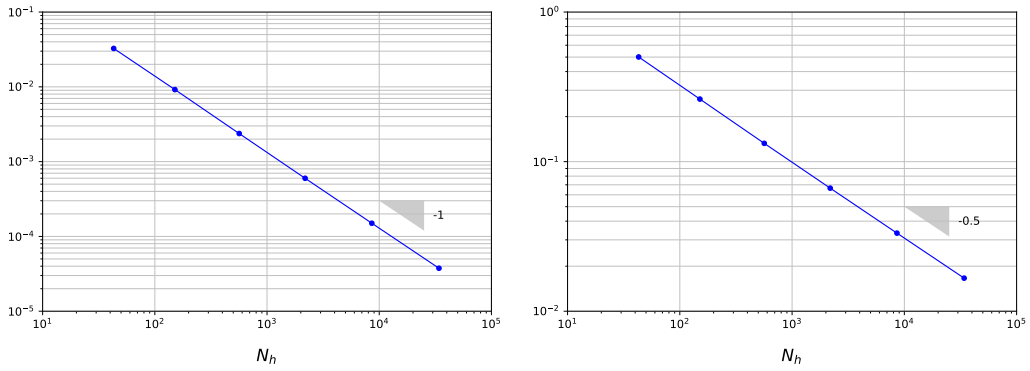
and compute the discretization errors

$$\|\hat{u} - \hat{u}_h(\mu)\|_{L^2(\hat{\Omega})} \quad \text{and} \quad |\hat{u} - \hat{u}_h(\mu)|_{H^1(\hat{\Omega})}$$

on a sequence of regularly refined meshes over the reference domain $\hat{\Omega}$ for the parameters $\mu = (-0.6, 0.1, 0.9)$. We expect algebraic convergence

$$\|\hat{u} - \hat{u}_h(\mu)\|_{L^2(\hat{\Omega})} = \mathcal{O}(N_h^{-1}) \quad \text{and} \quad |\hat{u} - \hat{u}_h(\mu)|_{H^1(\hat{\Omega})} = \mathcal{O}(N_h^{-\frac{1}{2}})$$

due to the regular mesh refinement and the 2D problem. Figure 1 reports the results. The corresponding algebraic convergence rates are estimated as 1.01 in the $L^2(\hat{\Omega})$-norm and as 0.51 in the $H^1(\hat{\Omega})$-semi-norm, matching the expected rates.



(a) (b)

Figure 1: Convergence of the discretization error in the $L^2(\hat{\Omega})$-norm (a) and in the $H^1(\hat{\Omega})$-semi-norm (b)

7.2 EIM

The convergence of the EIM is tested for $P = 3$. Figure 2 compares the convergence of the interpolation errors (30) and (34) for scalar and matrix functions as well as for the norms $\|\cdot\|_{L^2(\hat{\Omega})}$ and $\|\cdot\|_{L^\infty(\hat{\Omega})}$. It can be observed that for both the scalar and matrix case the EIM using the $L^2(\hat{\Omega})$-norm requires less basis functions to reach the prescribed accuracy. Assuming an exponential decay of the interpolation error of the form

$$e = O(e^{-\gamma Q}), \quad \text{with rate } \gamma > 0,$$

where Q denotes the number of basis functions, the rates of the four combinations can be approximated as

	$\|\cdot\|_{L^2(\hat{\Omega})}$	$\|\cdot\|_{L^\infty(\hat{\Omega})}$
scalar EIM	0.39	0.37
matrix EIM	0.24	0.22

Due to the above result, the EIM with the $L^2(\hat{\Omega})$-norm is applied in the remaining experiments.

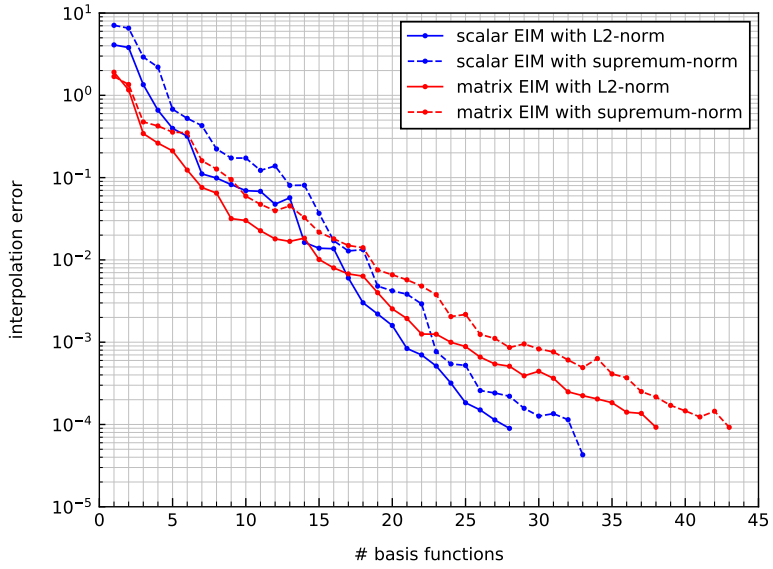


Figure 2: Convergence of the interpolation error with respect to the number of basis functions Q

7.3 SCM

The offline stage of the SCM is tested for $P = 3$. Figure 3 shows the convergence of the accuracy of the estimated lower and upper bounds on the discrete coercivity constant, that is, for each iteration i of the SCM greedy procedure it reports the error quantity

$$\max_{\mu \in \mathbb{P}_{\text{SCM}}} \left\{ 1 - \frac{\alpha_{LB}^{(i)}(\mu)}{\alpha_{UB}^{(i)}(\mu)} \right\},$$

where a smaller value indicates sharper bounds. We recall that one generalized eigenvalue problem needs to be solved per iteration. It can be seen that after 20 iterations the convergence starts to slow down significantly. The algorithm could be stopped at this point since already reasonable bounds have been obtained.

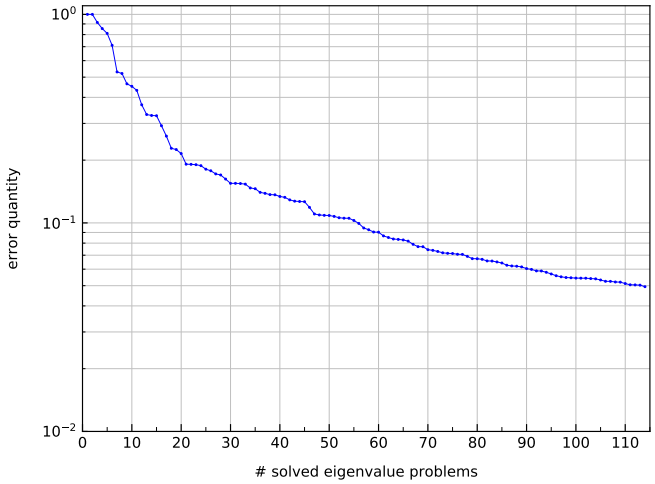


Figure 3: Convergence of the error quantity with respect to the number of solved generalized eigenvalue problems

7.4 RBM

In this section the reduction error of the RBM is examined. We are interested in the convergence of the relative reduction error in the energy norm

$$\max_{\mu \in P_h} \frac{\|u_h(\mu) - u_r(\mu)\|_a}{\|u_h(\mu)\|_a}$$

for different numbers of parameters $P = 1, 2, 3, 4$. The results of this comparison are shown in figure 4 where the targeted relative error is 10^{-4}. The number of reduced basis functions required to achieve the targeted accuracy increases significantly with an increasing number of parameters.

Additionally, figure 5 provides a visual comparison between the truth solution, the reduced basis approximation and their absolute difference $|u_h(\mu) - u_r(\mu)|$. The targeted error tolerance of the greedy algorithm is set as $\text{tol} = 10^{-4}$. The functions are plotted on the original domain $\Omega(\mu)$ depending on $P = 3$ arbitrarily chosen parameters $\mu = (-0.6, 0.1, 0.9)$. The illustrations confirm the expected accuracy of the reduced basis approximation.

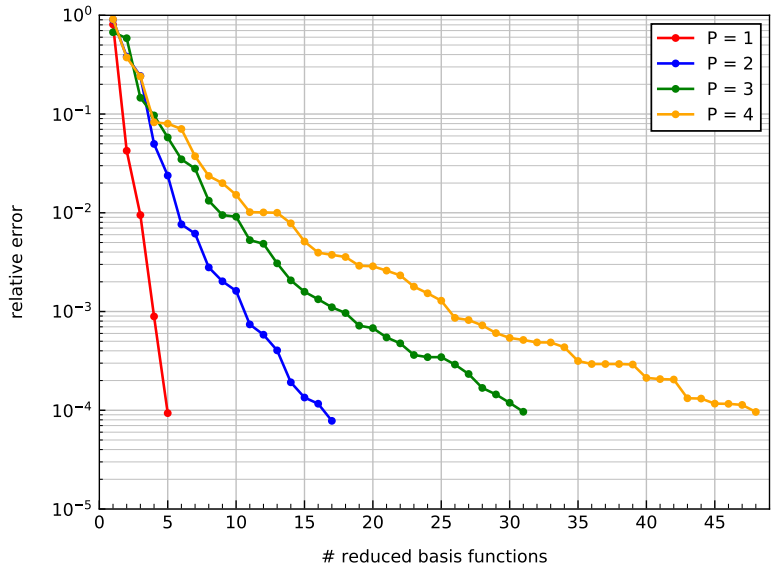
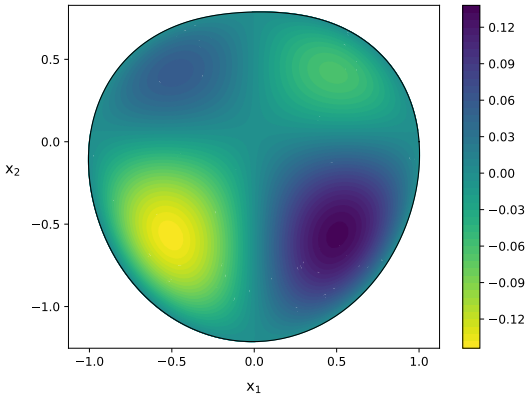
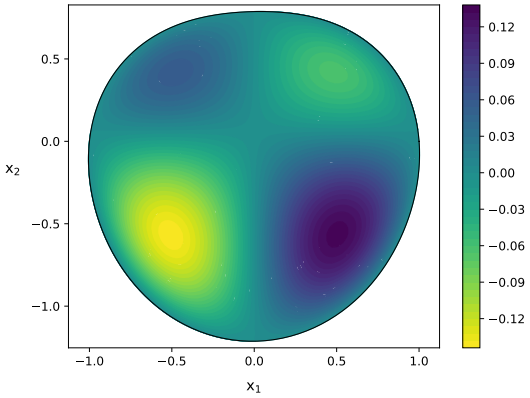


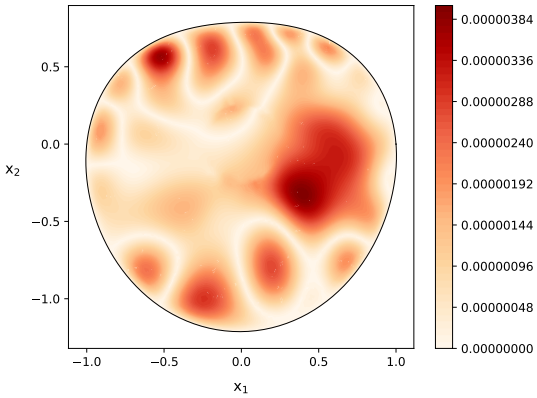
Figure 4: The maximal relative reduction error with respect to the number of reduced basis functions for different numbers of parameters



(a) Truth solution



(b) Reduced basis approximation



(c) Error

Figure 5: Visual comparison between the truth solution and its reduced basis approximation

8 Conclusion

The results of the numerical experiments suggest that the reduced basis method was implemented correctly for the considered example problem on a parametrized domain. The L^2-norm seems to be more suitable to be used by the EIM than the L^∞-norm. We observed that increasing the number of parameters of the example problem decreases the speed of convergence of the reduction error, at least with regard to the first four parameters.

References

[1] "LehrFEM++." `https://github.com/craffael/lehrfempp`, 2020.

[2] J. S. Hesthaven, G. Rozza, and B. Stamm, *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*. Springer, Cham, 2016.

[3] R. Hiptmair, L. Scarabosio, C. Schillings, and C. Schwab, "Large deformation shape uncertainty quantification in acoustic scattering," *Advances in Computational Mathematics*, vol. 44, no. 5, pp. 1475 – 1518, 2018.

[4] "PNL." `https://github.com/pnlnum/pnl`, 2020.

[5] R. Hiptmair, "Numerical methods for partial differential equations," 2018. `https://www.sam.math.ethz.ch/~grsam/NUMPDE/NUMPDE18.pdf`.

Appendix

Running the code on your PC

The code performing the experiments described in section 7 is available from the following link:
`https://gitlab.ethz.ch/bohnhofl/rbm`

In order to run the code, it is necessary to have the ARPACK library installed.

The documentation of the code is made by Doxygen and can be found in:
`rbm/thesis/documentation/html/index.html`

Having compiled the code, figure 4 from section 7.4 can be generated by:
```
cd rbm/thesis/build/experiments/reduced_basis_convergence
make thesis.experiments.reduced_basis_convergence.run
```

Figure 4 is then located at:
`rbm/thesis/build/experiments/reduced_basis_convergence/plots`

The components of figure 5 are generated by:
```
cd rbm/thesis/build/experiments/reduced_basis_visualisation
make thesis.experiments.reduced_basis_visualisation.run
```

They will be located at:
`rbm/thesis/build/experiments/reduced_basis_visualisation/plots`

Running the code on the Euler cluster

Conducting the experiments for the parameters specified in section 7 will take a considerable amount of time. Therefore, they were conducted on the Euler cluster of ETH Zurich. To this end, the code had to be adjusted to the environment of the cluster. Both experiments from section 7.2 and 7.3 lasted for around 18 hours. The experiments related to figures 4 and 5 took around 6 days (in total) and 1 day, respectively. The code provided above is not the adjusted version that can run on the cluster. The adjusted version differs from the above version in some respects. Instead of using the Arpack support module provided by Eigen, it uses the Spectra library to solve sparse generalized eigenvalue problems. It has not been corrected by clang-tidy. Further differences become apparent when going through the instructions listed below.

The adjusted version is available from the following link:
`https://gitlab.ethz.ch/bohnhofl/rbm_euler`

In the following, instructions for conducting the main experiment (i.e. generating figure 4) on the Euler cluster are given (valid for April 2020):

1. Clone the adjusted version onto your PC.
2. In the following files, replace the word "bohnhofl" by "your_username" (your ETH-username):

   ```
   rbm_euler/thesis/utility/test/main.cc
   rbm_euler/thesis/problem/test/main.cc
   rbm_euler/thesis/method/test/main.cc
   rbm_euler/thesis/experiments/empirical_interpolation_method/main.cc
   rbm_euler/thesis/experiments/successive_constraint_method/main.cc
   rbm_euler/thesis/experiments/reduced_basis_visualisation/main.cc
   rbm_euler/thesis/experiments/reduced_basis_convergence/dP1/main.cc
   rbm_euler/thesis/experiments/reduced_basis_convergence/dP2/main.cc
   rbm_euler/thesis/experiments/reduced_basis_convergence/dP3/main.cc
   rbm_euler/thesis/experiments/reduced_basis_convergence/dP4/main.cc
   rbm_euler/thesis/experiments/reduced_basis_convergence/dP5/main.cc
   rbm_euler/thesis/experiments/reduced_basis_convergence/main.cc
   rbm_euler/thesis/experiments/reduced_basis_convergence/plot.py
   ```

3. Transfer the code from your PC to your home directory of the Euler cluster using `scp`:

   ```
   scp -r rbm_euler your_username@euler.ethz.ch:
   ```

4. Switch to the new software stack and load the required modules:

   ```
   env2lmod
   module load gcc/8.2.0
   module load cmake/3.11.1
   module load python/3.6.4
   module load eigen
   module load boost
   ```

5. Create a build directory and compile the code:

```
cd rbm_euler/thesis
mkdir build
cd build
cmake ..
make
```

6. Run the code for generating the data contained in figure 4 by submitting the following jobs:

```
cd rbm_euler/thesis/experiments/reduced_basis_convergence/dP1
bsub -n 1 -W 6:00 ./thesis.experiments.reduced_basis_convergence.dP1.main

cd rbm_euler/thesis/experiments/reduced_basis_convergence/dP2
bsub -n 1 -W 24:00 ./thesis.experiments.reduced_basis_convergence.dP2.main

cd rbm_euler/thesis/experiments/reduced_basis_convergence/dP3
bsub -n 1 -W 48:00 ./thesis.experiments.reduced_basis_convergence.dP3.main

cd rbm_euler/thesis/experiments/reduced_basis_convergence/dP4
bsub -n 1 -W 120:00 -R "rusage[mem=16384]" ./thesis.experiments.reduced_basis_convergence.dP4.main
```

7. (After 4 days) regularly check the progress of the jobs by:

```
bjobs
```

8. When the jobs have finished, the convergence data for $P = 1, \ldots, 4$ are stored in the files `rbm_euler/thesis/build/experiments/reduced_basis_convergence/dPP/output_PP.dat`. Generate figure 4 by:

```
cd rbm_euler/thesis/build/experiments/reduced_basis_convergence
python /cluster/home/your_username/rbm_euler/thesis/experiments/reduced_basis_convergence/plot.py
```

9. Figure 4 is then located at:

```
/cluster/home/your_username/rbm_euler/thesis/build/experiments/reduced_basis_convergence/plots
```