Mathematical and Computational Applications

MDPI

Article

ROM-Based Inexact Subdivision Methods for PDE-Constrained Multiobjective Optimization

Stefan Banholzer [1,†], Bennet Gebken [2,†], Lena Reichle [1,†] and Stefan Volkwein [1,*,†]

[1] Department of Mathematics and Statistics, University of Konstanz, 78457 Konstanz, Germany; Stefan.Banholzer@uni-konstanz.de (S.B.); Lena.Reichle@uni-konstanz.de (L.R.)
[2] Faculty for Computer Science, Electrical Engineering and Mathematics, Paderborn University, 33098 Paderborn, Germany; bgebken@math.upb.de
* Correspondence: Stefan.Volkwein@uni-konstanz.de
† These authors contributed equally to this work.

Abstract: The goal in multiobjective optimization is to determine the so-called Pareto set. Our optimization problem is governed by a parameter-dependent semi-linear elliptic partial differential equation (PDE). To solve it, we use a gradient-based set-oriented numerical method. The numerical solution of the PDE by standard discretization methods usually leads to high computational effort. To overcome this difficulty, reduced-order modeling (ROM) is developed utilizing the reduced basis method. These model simplifications cause inexactness in the gradients. For that reason, an additional descent condition is proposed. Applying a modified subdivision algorithm, numerical experiments illustrate the efficiency of our solution approach.

Keywords: multiobjective optimization; PDE-constrained optimization; reduced-order modeling; set-oriented methods; inexact optimization

Citation: Banholzer, S.; Gebken, B.; Reichle, L.; Volkwein, S. ROM-Based Inexact Subdivision Methods for PDE-Constrained Multiobjective Optimization. *Math. Comput. Appl.* **2021**, *26*, 32. https://doi.org/10.3390/mca26020032

Academic Editors: Oliver Junge, Kathrin Padberg-Gehle, Sebastian Peitz and Oliver Schütze

Received: 25 February 2021
Accepted: 13 April 2021
Published: 15 April 2021

1. Introduction

Multiobjective optimization plays an important role in many applications, e.g., in industry, medicine, or engineering. One of the mentioned examples is the minimization of costs with simultaneous quality optimization in production or the minimization of CO_2 emission in energy generation and simultaneous cost minimization. These problems lead to multiobjective optimization problems (MOPs), where we want to achieve an optimal compromise with respect to all given objectives at the same time. Normally, the different objectives are contradictory such that there exists an infinite number of optimal compromises. The set of these compromises is called the *Pareto set*. The goal is to approximate the Pareto set in an efficient way, which turns out to be more expensive than solving a single objective optimization problem.

As multiobjective optimization problems are of great importance, there exist several algorithms to solve them. Among the most popular methods are scalarization methods, which transform MOPs into scalar problems. For example, in the weighted sum method [1–4], convex combinations of the original objectives are optimized. Another popular approach is to use non-deterministic methods like evolutionary algorithms, cf., e.g., [5]. Furthermore, as multiobjective problems are generalizations of scalar problems, some solution methods can be generalized from the scalar to the multiobjective case [6–8].

In addition to the classical methods above, there are set-based strategies for the solution of MOPs. Continuation methods [9–11] use the fact that the Pareto set is typically (the projection of) a smooth manifold. Subdivision methods [12–15] use tools from the area of dynamical systems to generate a covering of the Pareto set via hypercubes. However, especially when the objective functions and their gradients are expensive to evaluate, e.g., as an underlying PDE has to be solved for every evaluation, the computational time of these methods can quickly become very large. In the presence of PDE constraints, surrogate

models offer a promising tool to reduce the computational effort significantly [16]. Examples are dimensional reduction techniques such as the Reduced Basis (RB) Method [17,18]. In an offline phase, a low-dimensional surrogate model of the PDE is constructed by using, e.g., the greedy algorithm, cf. [17]. In the online phase, only the RB model is used to solve the PDE, which saves a lot of computing time.

In this article, we combine an extension of the set-oriented method presented in [12] based on inexact gradient evaluations of the objective functions with an RB approach and a discrete empirical interpolation method (DEIM) [19,20] for semi-linear elliptic PDEs. In order to deal with the inexactness introduced by the surrogate model, we combine the first-order optimality conditions for multiobjective optimization problems with error estimates for the RB-DEIM method and derive an additional condition for the descent direction [9] to get a tight superset of the Pareto set. This approach allows us to better control the quality of the result by controlling the errors for the objective functions independently. In order to obtain an even tighter superset of the Pareto set, we update these error estimates in our subdivision algorithm after each iteration step.

The article is organized as follows. In Section 2, we recall the basic concepts of multiobjective optimization problems and review results on descent directions with exact and inexact gradients. Furthermore, we develop a set-oriented method to solve these problems, where only inexact gradient information is utilized. In Section 3, the PDE-constrained multiobjective optimization problem and the underlying semi-linear PDE are introduced. Subsequently, we show how reduced-order modeling can be applied efficiently. In Section 4, numerical results concerning both the subdivision and the modified algorithm are presented. Finally, we give a conclusion and discuss possible future work in Section 5.

2. A Set-Oriented Method for Multiobjective Optimization with Inexact Objective Gradients

In this section, we briefly recall the basic concepts of multiobjective optimization. Furthermore, we develop a set-oriented method to solve these problems, where only inexact gradient information is utilized.

2.1. Multiobjective Optimization

Let $\mu_{\mathsf{a}}, \mu_{\mathsf{b}} \in \mathbb{R}^m$ with $\mu_{\mathsf{a}} \leq \mu_{\mathsf{b}}$ be arbitrary. We define the convex and compact parameter set $\mathcal{P}_{\mathsf{ad}} = [\mu_{\mathsf{a}}, \mu_{\mathsf{b}}] \subset \mathbb{R}^m$. Now, the goal is to solve the constrained multiobjective optimization problem

$$\min \hat{f}(\mu) \quad \text{subject to (s.t.)} \quad \mu \in \mathcal{P}_{\mathsf{ad}} \tag{1}$$

with a given objective $\hat{f} = (\hat{f}_1, \ldots, \hat{f}_k) : \mathcal{P}_{\mathsf{ad}} \to \mathbb{R}^k$ and $k > 1$.

Compared to scalar optimization, we do not have a natural total order of $\mathbb{R}^k$ for $k > 1$. Therefore, we cannot expect that there is a single point in $\mathcal{P}_{\mathsf{ad}}$ that minimizes all objectives $\hat{f}_i$ simultaneously. For this reason, we make use of the following definition.

Definition 1. (a) *A point $\bar{\mu} \in \mathcal{P}_{\mathsf{ad}}$ is called* (globally) Pareto optimal, *if there is no $\mu \in \mathcal{P}_{\mathsf{ad}}$ satisfying $\hat{f}(\mu) \lneqq \hat{f}(\bar{\mu})$. In that case we call $\bar{\mu}$ a* Pareto point.

(b) *The set of all Pareto points in $\mathcal{P}_{\mathsf{ad}}$ is called* Pareto set *and is denoted by*

$$\mathbb{P} = \left\{ \mu \in \mathcal{P}_{\mathsf{ad}} \,\middle|\, \mu \text{ is Pareto optimal} \right\}.$$

(c) *The image $\hat{f}(\mathbb{P}) \subset \mathbb{R}^k$ of the Pareto set under $\hat{f}$ is called the* Pareto front.

If $\hat{f}$ is continuously differentiable on an open set containing $\mathcal{P}_{\mathsf{ad}}$, then there exists a first-order necessary condition for Pareto optimality. To formulate this condition, we define the convex, closed, and bounded set

$$\Delta_k = \left\{ \alpha \in \mathbb{R}^k \,\middle|\, \alpha_i \geq 0 \text{ and } \sum_{i=1}^{k} \alpha_i = 1 \right\}.$$

Further, the row vector

$$\nabla \hat{f}_i(\mu) = \left(\frac{\partial \hat{f}_i}{\partial \mu_j}(\mu) \right)_{1 \le j \le m} \in \mathbb{R}^{1 \times m}$$

stands for the *gradient* of the i-th objective $\hat{f}_i$ with $i \in \{1, \ldots, k\}$.

Definition 2. *If for a given $\bar{\mu} \in \mathcal{P}_{\mathsf{ad}}$ there exists an $\bar{\alpha} = (\bar{\alpha}_i)_{1 \le i \le k} \in \Delta_k$ with*

$$\sum_{i=1}^{k} \bar{\alpha}_i \nabla \hat{f}_i(\bar{\mu})(\mu - \bar{\mu}) = (\mu - \bar{\mu})^\top D\hat{f}(\bar{\mu})^\top \bar{\alpha} \ge 0 \quad \text{for all } \mu \in \mathcal{P}_{\mathsf{ad}} \tag{2}$$

then we call $\bar{\mu}$ Pareto critical, *where*

$$D\hat{f}(\mu) = \begin{pmatrix} \nabla \hat{f}_1(\bar{\mu}) \\ \vdots \\ \nabla \hat{f}_k(\mu) \end{pmatrix} \in \mathbb{R}^{k \times m}$$

denotes the Jacobi matrix of $\hat{f}$ at μ. The set of all Pareto critical points is called the Pareto critical set, *denoted by $\mathbb{P}_c$.*

Now, we recall the first-order necessary optimality conditions for (1).

Theorem 1. *Let $\hat{f}$ be continuously differentiable and $\bar{\mu} \in \mathcal{P}_{\mathsf{ad}}$ Pareto-optimal. Then, $\bar{\mu}$ is Pareto-critical, i.e., it holds $\mathbb{P} \subset \mathbb{P}_c$. Condition (2) is called the* Karush–Kuhn–Tucker (KKT) condition *for multiobjective optimization problems.*

Proof. The claim follows from ([1], Theorem 3.25) and the specific choice of $\mathcal{P}_{\mathsf{ad}}$. □

Remark 1. (a) *Let $\bar{\mu}$ belong to the interior of $\mathcal{P}_{\mathsf{ad}}$, i.e., $\bar{\mu} \in int\,(\mathcal{P}_{\mathsf{ad}}) = (\mu_{\mathsf{a}}, \mu_{\mathsf{b}})$. Then (2) is equivalent to*

$$\sum_{i=1}^{k} \bar{\alpha}_i \nabla \hat{f}_i(\bar{\mu}) = \alpha^\top D\hat{f}(\mu) = 0 \quad \text{in } \mathbb{R}^{1 \times m} \tag{3}$$

or

$$D\hat{f}(\mu)^\top \alpha = 0 \quad \text{in } \mathbb{R}^m = \mathbb{R}^{m \times 1};$$

see also in [21].

(b) *Throughout the paper, we only calculate the Pareto critical points in the interior of $\mathcal{P}_{\mathsf{ad}}$ and make use of (3). The idea is to choose $\mathcal{P}_{\mathsf{ad}}$ sufficiently large so that we get $\mathbb{P}_c \subset int\,(\mathcal{P}_{\mathsf{ad}})$.*

(c) *Due to Theorem 1, we have*

$$\mathbb{P} \subset \mathbb{P}_c = \left\{ \mu \in \mathcal{P}_{\mathsf{ad}} \,\middle|\, \exists \alpha = \alpha(\mu) \in \Delta_k : \; D\hat{f}(\bar{\mu})^\top \alpha = 0 \text{ in } \mathbb{R}^m \right\} \subset int\,(\mathcal{P}_{\mathsf{ad}}).$$

provided $\mathbb{P}_c \subset int\,(\mathcal{P}_{\mathsf{ad}})$ holds true. ◊

2.2. Descent Direction with Exact Gradients

Next, we introduce the notion of a descent direction for the vector valued objective function $\hat{f}$ at a non-Pareto critical point $\mu \notin \mathbb{P}_c$. From now on, we assume that $\hat{f} : \mathcal{P}_{\mathsf{ad}} \to \mathbb{R}^k$ is continuously differentiable (on an open set containing $\mathcal{P}_{\mathsf{ad}}$).

Definition 3. *The vector $v \in \mathbb{R}^m$ is a* descent direction *for $\hat{f}$ in $\mu \in \mathcal{P}_{\mathsf{ad}}$, if we have*

$$\nabla \hat{f}_i(\mu) v \le 0 \quad \text{for all } i \in \{1, ..., k\}$$

and if there is at least one $i \in \{1, ..., k\}$ with $\nabla \hat{f}_i(\mu) v < 0$.

One way to compute a descent direction is to solve a constrained quadratic optimization problem in $\mathbb{R}^k$ as shown in the following theorem. For a proof we refer to the work in [8]. A similar result was shown in [22].

Theorem 2. *For given $\mu \in \operatorname{int}(\mathcal{P}_{\mathsf{ad}})$ let $\bar{\alpha} = \bar{\alpha}(\mu) \in \Delta_k$ be a (global) solution of the convex constrained quadratic minimization problem*

$$\min \frac{1}{2} \|D\hat{f}(\mu)^\top \alpha\|_2^2 \quad s.t. \quad \alpha \in \Delta_k. \tag{4}$$

Then, we have either $D\hat{f}(\mu)^\top \bar{\alpha} = 0$ or $v = -D\hat{f}(\mu)^\top \bar{\alpha} \in \mathbb{R}^m$ is a descent direction for $\hat{f}$ in μ.

Combined with a backtracking Armijo line search, the descent direction from Theorem 2 can be used to construct the steepest descent method in Algorithm 1.

Algorithm 1: Steepest descent method.

Require: $\kappa \in (0,1)$, $\mu^0 \in \operatorname{int}(\mathcal{P}_{\mathsf{ad}})$ and $l = 0$;

Calculate α^0 as a solution of (4) for $\mu = \mu^0$; set $v^0 = -D\hat{f}(\mu^0)^\top \alpha^0$;

while $v^l \neq 0$ **do**

  Choose the stepsize $t_l > 0$ as maximum of the set

$$\mathcal{T}_l = \left\{ t = 2^{-j} \,\middle|\, j \in \mathbb{N},\ \mu^l + tv^l \in \operatorname{int}(\mathcal{P}_{\mathsf{ad}}) \right.$$
$$\left. \text{and } \hat{f}(\mu^l + tv^l) \leq \hat{f}(\mu^l) + \kappa t D\hat{f}(\mu^l)v^l \right\};$$

  Set $\mu^{l+1} = a(\mu^l)$ with $a(\mu^l) = \mu^l + t_l v^l$ and update $l = l + 1$;

  Calculate α^l as a solution of (4) for $\mu = \mu^l$;

  Set $v^l = -D\hat{f}(\mu^l)^\top \alpha^l$;

end

Remark 2. (a) *If Algorithm 1 terminates after a finite number l of iterations, then μ^l is a Pareto critical point.*

(b) *Assume that Algorithm 1 does not stop after a finite number of iterations. Then, every accumulation point $\bar{\mu}$ of the sequence $\{\mu^l\}_{l\in\mathbb{N}}$ generated by Algorithm 1 is a Pareto critical point. A proof based on* ([7], Theorem 1) *can be found in* ([23] Theorem 5.2.5). ◊

2.3. Descent Direction with Inexact Gradients

Suppose that we have continuously differentiable approximations $\hat{f}^\ell = (\hat{f}_i^\ell)_{1\leq i\leq k}$ of the objective function $\hat{f}$ satisfying

$$\max_{\mu\in\mathcal{P}_{\mathsf{ad}}} \|\nabla\hat{f}_i(\mu) - \nabla\hat{f}_i^\ell(\mu)\|_2 \leq \varepsilon_i \quad \text{for } i \in \{1,\ldots,k\}, \tag{5}$$

for given tolerances $\varepsilon_i \geq 0$. By $\mathbb{P} \subset \mathcal{P}_{\mathsf{ad}}$ we denote the Pareto set for $\hat{f}$ and by $\mathbb{P}^\ell \subset \mathcal{P}_{\mathsf{ad}}$ the Pareto set for $\hat{f}^\ell$. If we write $\mathbb{P}_c$, we mean the Pareto-critical set for $\hat{f}$ and $\mathbb{P}_c^\ell$ is the Pareto-critical set for $\hat{f}^\ell$. Note that, in general, we have neither $\mathbb{P}^\ell \subset \mathbb{P}$ nor $\mathbb{P} \subset \mathbb{P}^\ell$.

In this section, our goal is to compute an approximation of $\mathbb{P}$ based on the approximation $\hat{f}^\ell$ of the objective function and the error bounds ε_i. We begin by investigating the relationship between the KKT conditions of the original objective function and its approximation.

Lemma 1. *Let (5) be satisfied and $\bar{\mu} \in \operatorname{int}(\mathcal{P}_{\mathsf{ad}})$ be Pareto-critical for $\hat{J}$ with the KKT-condition vector $\bar{\alpha} \in \Delta_k$. Then, it holds that*

$$\|D\hat{J}^{\ell}(\bar{\mu})^{\top}\bar{\alpha}\|_2 \leq \sum_{i=1}^{k} \bar{\alpha}_i \varepsilon_i = \langle \bar{\alpha}, \varepsilon \rangle_2 \leq \|\varepsilon\|_{\infty} \tag{6}$$

with $\varepsilon = (\varepsilon_i)_{1 \leq i \leq k}$, where we set $\langle \bar{\alpha}, \varepsilon \rangle_2 = \bar{\alpha}^{\top}\varepsilon$.

Proof. From $\bar{\mu} \in \operatorname{int}(\mathcal{P}_{\mathsf{ad}})$ and Remark 1-a) we infer that $D\hat{J}(\bar{\mu})^{\top}\bar{\alpha} = 0$ holds. Therefore,

$$\begin{aligned}\|D\hat{J}^{\ell}(\bar{\mu})^{\top}\bar{\alpha}\|_2 = \|D\hat{J}^{\ell}(\bar{\mu})^{\top}\bar{\alpha} - D\hat{J}(\bar{\mu})^{\top}\bar{\alpha}\|_2 &= \Big\| \sum_{j=1}^{k} (\nabla \hat{J}_j^{\ell}(\bar{\mu}) - \nabla \hat{J}_j(\bar{\mu}))\bar{\alpha}_j \Big\|_2 \\ &\leq \sum_{j=1}^{k} \big\|\nabla \hat{J}_j^{\ell}(\bar{\mu}) - \nabla \hat{J}_j(\bar{\mu})\big\|_2 \bar{\alpha}_j \leq \sum_{j=1}^{k} \varepsilon_j \bar{\alpha}_j = \langle \bar{\alpha}, \varepsilon \rangle_2 \leq \|\varepsilon\|_{\infty}\end{aligned}$$

which gives the desired results. □

Based on estimate (6), we define two approximation sets for the Pareto-critical set $\mathbb{P}_c$ of $\hat{J}$.

Definition 4. *Let us introduce the two sets*

$$\mathbb{P}_1^{\ell} = \Big\{ \mu \in \operatorname{int}(\mathcal{P}_{\mathsf{ad}}) \;\Big|\; \min_{\alpha \in \Delta_k} \|D\hat{J}^{\ell}(\mu)^{\top}\alpha\|_2^2 \leq \|\varepsilon\|_{\infty}^2 \Big\} \subset \mathcal{P}_{\mathsf{ad}}$$

and

$$\mathbb{P}_2^{\ell} = \Big\{ \mu \in \operatorname{int}(\mathcal{P}_{\mathsf{ad}}) \;\Big|\; \min_{\alpha \in \Delta_k} \Big(\|D\hat{J}^{\ell}(\mu)^{\top}\alpha\|_2^2 - \langle \alpha, \varepsilon \rangle_2^2 \Big) \leq 0 \Big\} \subset \mathcal{P}_{\mathsf{ad}}.$$

Lemma 2. *It holds that*

$$\mathbb{P}_c \subset \mathbb{P}_2^{\ell}, \quad \mathbb{P}_c^{\ell} \subset \mathbb{P}_2^{\ell} \quad \textit{and} \quad \mathbb{P}_2^{\ell} \subset \mathbb{P}_1^{\ell}.$$

Proof. Let $\bar{\mu} \in \mathbb{P}_c$ be a Pareto-critical point of $\hat{J}$, then there exists $\bar{\alpha} \in \Delta_k$ with $D\hat{J}(\bar{\mu})^{\top}\bar{\alpha} = 0$. From Lemma 1, it follows that $\mathbb{P}_c \subset \mathbb{P}_2^{\ell}$.
Next, we assume that $\bar{\mu} \in \mathbb{P}_c^{\ell}$ is a Pareto-critical point of $\hat{J}^{\ell}$, then there exists $\bar{\alpha} \in \Delta_k$ with $D\hat{J}^{\ell}(\bar{\mu})^{\top}\bar{\alpha} = 0$. This implies

$$\min_{\alpha \in \Delta_k} \Big(\|D\hat{J}^{\ell}(\bar{\mu})^{\top}\alpha\|_2^2 - \langle \alpha, \varepsilon \rangle^2 \Big) \leq \|D\hat{J}^{\ell}(\bar{\mu})^{\top}\bar{\alpha}\|_2^2 - \langle \bar{\alpha}, \varepsilon \rangle_2^2 = -\langle \bar{\alpha}, \varepsilon \rangle^2 \leq 0.$$

Therefore, we have $\bar{\mu} \in \mathbb{P}_2^{\ell}$.
Let $\bar{\mu} \in \mathbb{P}_2^{\ell}$. Then, there exists $\bar{\alpha} \in \Delta_k$ with

$$\|D\hat{J}^{\ell}(\bar{\mu})^{\top}\bar{\alpha}\|_2^2 - \langle \bar{\alpha}, \varepsilon \rangle_2^2 \leq 0.$$

Thus, we get $\|D\hat{J}^{\ell}(\bar{\mu})^{\top}\bar{\alpha}\|_2^2 \leq \langle \bar{\alpha}, \varepsilon \rangle_2^2 \leq \|\varepsilon\|_{\infty}^2$ which implies $\mathbb{P}_2^{\ell} \subset \mathbb{P}_1^{\ell}$. □

Our goal is to compute the set $\mathbb{P}_2^{l}$ via a descent method like Algorithm 1. To this end, the following theorem presents a modified version of the descent direction (4), which additionally takes the error bounds ε_i into account.

Theorem 3. *Let $\varepsilon = (\varepsilon_j)_{1 \leq j \leq k}$ with $\varepsilon \geq 0$ and $\mu \in \operatorname{int}(\mathcal{P}_{\mathsf{ad}})$ be given. Assume that α_{ε} is a minimizer of the quadratic problem*

$$\min_{\alpha \in \Delta_k} \Big(\|D\hat{J}^{\ell}(\mu)^{\top}\alpha\|_2^2 - \langle \alpha, \varepsilon \rangle_2^2 \Big). \tag{7}$$

Then, we have that $\mu \in \mathbb{P}_2^\ell$ or $v_\varepsilon = -D\hat{J}^\ell(\mu)^\top \alpha_\varepsilon \in \mathbb{R}^m$ is a descent direction for $\hat{J}^\ell$ in μ.

Proof. The Lagrange functions for (7) is given as

$$L : \mathbb{R}^k \times \mathbb{R} \times \mathbb{R}^k \to \mathbb{R}, \quad (\alpha, \lambda, \varrho) \mapsto \|D\hat{J}^\ell(\mu)^\top \alpha\|_2^2 - \langle \alpha, \epsilon \rangle_2^2 + \lambda\Big(1 - \sum_{j=1}^k \alpha_j\Big) - \sum_{j=1}^k \varrho_j \alpha_j.$$

As α_ε is a minimizer of (7), we get Lagrangian multipliers $\lambda \in \mathbb{R}$ and $\varrho \in \mathbb{R}^k_{\geq 0}$ with

$$\begin{aligned} 2D\hat{J}^\ell(\mu)D\hat{J}^\ell(\mu)^\top \alpha_\varepsilon - 2\varepsilon \langle \varepsilon, \alpha_\varepsilon \rangle_2 + \lambda(-1, ..., -1)^\top - \varrho = 0, \\ \varrho_i \geq 0 \quad \text{and} \quad (\alpha_\varepsilon)_i \varrho_i = 0. \end{aligned} \tag{8}$$

If we multiply (8) with $\alpha_\varepsilon^\top$ from the left, we get

$$2\alpha_\varepsilon^\top D\hat{J}^\ell(\mu)D\hat{J}^\ell(\mu)^\top \alpha_\varepsilon - 2\langle \varepsilon, \alpha_\varepsilon \rangle_2^2 - \lambda \sum_{i=1}^m \left((\alpha_\varepsilon)_i - (\alpha_\varepsilon)_i \varrho_i\right) = 0$$

which implies

$$\lambda = 2\left(\|D\hat{J}^\ell(\mu)^\top \alpha_\varepsilon\|_2^2 - \langle \alpha_\varepsilon, \varepsilon \rangle_2^2\right).$$

First case: $\lambda \leq 0$, then $\mu \in \mathbb{P}_2^\ell$ holds and we are done.
Second case: $\lambda > 0$, then $\mu \notin \mathbb{P}_2^\ell$ holds. In this case, we show that $v_\varepsilon = -D\hat{J}^\ell(\mu)^\top \alpha_\varepsilon$ is a descent direction in μ for every objective function $\hat{J}_j^\ell$ with $j = 1, ..., k$:
Define

$$K(\mu) = \left\{ D\hat{J}^\ell(\mu)^\top \alpha \,\middle|\, \alpha \in \mathbb{R}^k,\ \alpha_i \geq 0 \text{ and } \sum_{i=1}^k \alpha_i = 1 \right\}.$$

If we can show that $w^\top v_\varepsilon < 0$ holds for every $w \in K$, we know that v_ε is a descent direction for every objective function J_i^ℓ in μ.
Choose $w \in K(\mu)$. Then, there exists an $\alpha_w \in \Delta_k$ with $w = D\hat{J}^\ell(\mu)^\top \alpha_w$, and using (8) we obtain

$$\begin{aligned} w^\top v_\varepsilon &= \left(D\hat{J}^\ell(\mu)^\top \alpha_w\right)^\top v_\varepsilon = -\alpha_w^\top D\hat{J}^\ell(\mu)D\hat{J}^\ell(\mu)^\top \alpha_\varepsilon \\ &= -\alpha_w^\top \left(\varepsilon \langle \varepsilon, \alpha_\varepsilon \rangle_2 + \frac{1}{2}\lambda(1, ..., 1) + \frac{1}{2}\varrho\right) = -\left(\langle \alpha_w, \varepsilon \rangle_2 \langle \alpha_\varepsilon, \varepsilon \rangle_2 + \frac{1}{2}\lambda + \frac{1}{2}\langle \alpha_w, \varrho \rangle_2\right) \\ &\leq -\frac{1}{2}\left(\lambda + \langle \alpha_w, \varrho \rangle_2\right) \leq -\frac{\lambda}{2} < 0. \end{aligned}$$

Therefore, v_ε is a descent direction in μ for every objective function $\hat{J}_j^\ell$, $j = 1, ..., k$. □

The descent direction v_ε from the previous theorem will be referred to as the modified descent direction. Based on this direction, we can now construct a descent method for the computation of $\mathbb{P}_2^l$, which is shown in Algorithm 2.

Remark 3. (a) *If Algorithm 2 terminates after l iteration steps, then μ^l is contained in $\mathbb{P}_2^\ell$.*
(b) *Assume that Algorithm 2 does not terminate after a finite number of iteration steps. Then, every accumulation point $\bar{\mu}$ of the sequence $\{\mu^l\}_{l \in \mathbb{N}}$ generated by Algorithm 2 is in the set $\mathbb{P}_2^\ell$. A proof based on* ([7] Theorem 1) *can be found in* ([23] Theorem 5.3.5).
(c) *Note that the tolerance ε is constant for all l throughout Algorithm 2. In Section 2.5, we will adapt ε in each iteration.* ◊

Algorithm 2: Descent method with inexact gradients.

Require: $\varepsilon = (\varepsilon_j)_{1\leq j\leq k}$, $\kappa \in (0,1)$, $\mu^0 \in \operatorname{int}(\mathcal{P}_{\mathsf{ad}})$ and $l = 0$;

while $\mu_l \notin \mathbb{P}_2^\ell$ **do**

  Calculate α_ε^l as solution of (7) for $\mu = \mu^l$;

  Set $v_\varepsilon^l = -D\hat{J}^\ell(\mu^l)^\top \alpha_\varepsilon^l$;

  Choose the stepsize $t_l > 0$ as maximum of the set

$$\mathcal{T}_l = \Big\{ t = 2^{-j} \,\Big|\, j \in \mathbb{N},\ \mu^l + tv_\varepsilon^l \in \operatorname{int}(\mathcal{P}_{\mathsf{ad}}) \text{ and } \hat{J}^\ell(\mu^l + tv_\varepsilon^l) \leq \hat{J}^\ell(\mu^l) + \kappa t D\hat{J}^\ell(\mu^l)v_\varepsilon^l \Big\};$$

  Set $\mu^{l+1} = a_\varepsilon(\mu^l)$ with $a_\varepsilon(\mu^l) = \mu^l + t_l v_\varepsilon^l$ and update $l = l + 1$;

end

2.4. Subdivision Algorithm

As mentioned in the introduction, there exist set-based solution methods for MOPs which globally approximate the Pareto set via sets (instead of a finite number of points). Here, we will consider the *subdivision algorithm* [12,13,15], which computes an approximation of the Pareto set as a covering of hypercubes (or *boxes*). The idea is to start with a large box containing the Pareto set which is then iteratively subdivided into smaller boxes, while eliminating boxes that do not contain part of the Pareto set.

There are essentially two versions of the subdivision scheme: one is gradient free and, thus, is particularly useful in the case when the evaluation of gradients is computationally expensive. We refer to the work in [12], where this variant is utilized to numerically realize a reduced-order approach for a PDE-constrained multiobjective optimization problem. The other one is directly based on a dynamical systems approach and utilizes gradient information in a similar way to memetic algorithms, see in [8]. Here, we will generalize the latter to the case of inexact gradients.

For a stepsize $t_l > 0$, let us formulate a descent step of the optimization procedure by

$$\mu^{l+1} = a(\mu^l) = \mu^l + t_l v^l \quad \text{or} \quad \mu^{l+1} = a_\varepsilon(\mu^l) = \mu^l + t_l v_\varepsilon^l,$$

where v^l and v_ε^l are the descent directions given by Theorems 2 and 3, respectively, with the choice $\mu = \mu^l$. Depending on the descent step that we use, we either want to compute the Pareto-critical set $\mathbb{P}_c$ or the superset $\mathbb{P}_2^\ell$ or $\mathbb{P}_1^\ell$ of $\mathbb{P}_c$. As these sets are the sets of fixed points for their respective descent step, we want to find the subset $A_\mathbb{P} \subset \operatorname{int}(\mathcal{P}_{\mathsf{ad}})$ satisfying $a(A_\mathbb{P}) = A_\mathbb{P}$ or $a_\varepsilon(A_\mathbb{P}) = A_\mathbb{P}$.

To generate the set $A_\mathbb{P}$, we will use a subdivision method. This method produces an outer approximation of the set $A_\mathbb{P}$ in the form of a nested sequence of sets $\mathcal{B}_0, \mathcal{B}_1, \ldots \subset \mathfrak{P}(\mathcal{P}_{\mathsf{ad}})$, where $\mathfrak{P}(\mathcal{P}_{\mathsf{ad}})$ denotes the power set of $\mathcal{P}_{\mathsf{ad}}$ and each $\mathcal{B}_l$ is a subset of $\mathcal{B}_{l-1}$ in the sense that

$$\bigcup_{B\in\mathcal{B}_l} B \subset \bigcup_{B\in\mathcal{B}_{l-1}} B$$

holds and $\mathcal{B}_l$ consists of finitely many subsets B covering $A_\mathbb{P}$ for all $l \in \mathbb{N}$. For each set $\mathcal{B}_l$, we define a diameter through $\operatorname{diam}(\mathcal{B}_l) = \max_{B\in\mathcal{B}_l}(\operatorname{diam}(B))$. Algorithm 3 shows the classical subdivision method (based on Theorem 2) and our modified descent direction (based on Theorem 3).

Algorithm 3: Subdivision algorithm.

Require: $\mathcal{B}_0 \subset \mathfrak{P}(\mathcal{P}_{\mathsf{ad}})$ finite collection of subsets of $\mathcal{P}_{\mathsf{ad}}$ with $\bigcup_{B\in\mathcal{B}_0} B = \mathcal{P}_{\mathsf{ad}}$, $\theta \in (0,1)$, $l = 0$; in the setting of inexact gradients $\varepsilon_1, ..., \varepsilon_k$;

while $a_{(\varepsilon)}\big(\bigcup_{B\in\mathcal{B}_l} B\big) \neq \bigcup_{B\in\mathcal{B}_l} B$ **do**

  Subdivision:

    Construct from $\mathcal{B}_l$ a set $\hat{\mathcal{B}}_{l+1} \subset \mathfrak{P}(\mathcal{P}_{\mathsf{ad}})$ with

$$\bigcup_{B\in\hat{\mathcal{B}}_{l+1}} B = \bigcup_{B\in\mathcal{B}_l} B \text{ and } \operatorname{diam}(\hat{\mathcal{B}}_{l+1}) = \theta \operatorname{diam}(\mathcal{B}_l);$$

  Selection:

    Define the new set $\mathcal{B}_{l+1}$ by

$$\mathcal{B}_{l+1} = \big\{B \in \hat{\mathcal{B}}_{l+1} \,\big|\, \exists\, \hat{B} \in \hat{\mathcal{B}}_{l+1} \text{ with } a_{(\varepsilon)}^{-1}(B) \cap \hat{B} \neq \emptyset\big\};$$

  Set $l = l + 1$;

end

Remark 4. *In order to realize the subdivision algorithm numerically we choose a similar way as described in* ([13] Remark 2.4)*. Instead of working explicitly with the centers and radii of the boxes, these are stored within a binary tree in the subdivision step, whereby the memory requirement is noticeably reduced. The selection step is implemented using a certain number of sample points in each box. These sample points are chosen either on an a priori defined grid or randomly within the boxes. Afterwards, $a_{(\varepsilon)}$ is evaluated in these points. For more details, we refer the reader to* ([24] Section 5). ◊

2.5. Modified Subdivision Algorithm for Inexact Gradients

In Algorithm 2, we utilize the same error bounds $\varepsilon = (\varepsilon_1, ..., \varepsilon_k)$ with $\varepsilon_i \geq 0$, $i = 1, ..., k$, in each iteration step l. Note that the larger the ε, the greater the difference between $\mathbb{P}_c$ and $\mathbb{P}_2^\ell$ or $\mathbb{P}_1^\ell$. In the algorithm, we produce an outer approximation of the set $A_{\mathbb{P}}$ with a nested sequence of sets $\{\mathcal{B}_l\}_{l\in\mathbb{N}}$ by

$$\tilde{\mathcal{B}}_l = \bigcup_{B\in\mathcal{B}_l} B \subseteq \mathcal{P}_{\mathsf{ad}}.$$

As it holds that $\tilde{\mathcal{B}}_l \subset \mathcal{P}_{\mathsf{ad}}$, we have

$$\max\big\{\|\nabla \hat{f}_i(\mu) - \nabla \hat{f}_i^\ell(\mu)\|_2 \,\big|\, \mu \in \tilde{\mathcal{B}}_l\big\} \leq \max\big\{\|\nabla \hat{f}_i(\mu) - \nabla \hat{f}_i^\ell(\mu)\|_2 \,\big|\, \mu \in \mathcal{P}_{\mathsf{ad}}\big\} \quad \text{for } 1 \leq i \leq k.$$

Now we modify Algorithm 3 by utilizing the descent directions introduced in Theorem 3 and update ε after every iteration step l to generate a better approximation of the set $A_{\mathbb{P}}$. For updating ε, we use the formula

$$\varepsilon_i^l = \sup\big\{\|\nabla \hat{f}_i(\mu) - \nabla \hat{f}_i^\ell(\mu)\|_2 \,\big|\, \mu \in \tilde{\mathcal{B}}_l\big\} \quad \text{for } 1 \leq i \leq k \tag{9}$$

and set $\varepsilon^l = (\varepsilon_i^l)_{1\leq i\leq k}$. Due to the nested choice of the box coverings, we have $\varepsilon_i^{l+1} \leq \varepsilon_i^l$ for $i = 1, ..., k$ and $l \in \mathbb{N}$. In iteration step l, we generate the descent direction by computing

$$\alpha_\varepsilon^l \in \arg\min\Big\{\|D\hat{f}^\ell(\mu)^\top \alpha\|_2^2 - \langle \alpha, \varepsilon^l\rangle_2^2 \,\Big|\, \alpha \in \Delta_k\Big\}. \tag{10}$$

Then, we set

$$v_\varepsilon^l = -D\hat{f}^\ell(\mu)^\top \alpha_\varepsilon^l \quad \text{and} \quad \mu^{l+1} = a_\varepsilon^l(\mu^l) = \mu^l + t_l v_\varepsilon^l.$$

Typically, the set $\mathbb{P}_2^\ell$ is far smaller then the admissible set $\mathcal{P}_{\mathsf{ad}}$. Therefore, we expect that the error bounds in (9) become significantly smaller than the ones from (5). Therefore, we expect that we get better results with the modified function a_ε^l instead of a_ε.

3. Multiobjective Optimization of a Semi-Linear Elliptic PDE

In this section, we introduce a multiobjective parameter optimization problem governed by a semi-linear elliptic PDE. Further, we show how reduced-order modeling can be applied efficiently.

3.1. Problem Formulation

Let $\Omega \subset \mathbb{R}^{\mathfrak{d}}$, $\mathfrak{d} \in \{1,2,3\}$, be a bounded domain with Lipschitz-continuous boundary $\Gamma = \partial\Omega$. Then, we consider the problem

$$\min_{(y,\mu)} J(y,\mu) = \begin{pmatrix} J_1(y,\mu) \\ \vdots \\ J_{k-1}(y,\mu) \\ J_k(y,\mu) \end{pmatrix} = \frac{1}{2} \begin{pmatrix} \int_\Omega |y - y_1^{\mathsf{d}}|^2 \,\mathrm{d}\boldsymbol{x} \\ \vdots \\ \int_\Omega |y - y_{k-1}^{\mathsf{d}}|^2 \,\mathrm{d}\boldsymbol{x} \\ \sum_{j=1}^{m} |\mu_j - \mu_j^{\mathsf{d}}|^2 \end{pmatrix} \tag{11a}$$

subject to the elliptic boundary value problem

$$-c\Delta y + by + dy^3 = f + \sum_{i=1}^{m} \mu_i \xi_i \text{ in } \Omega, \quad c\,\frac{\partial y}{\partial \boldsymbol{n}} + y = g \text{ on } \Gamma, \tag{11b}$$

where $y \in V = H^1(\Omega)$ is the state variable and $\mu \in \mathcal{P}_{\mathsf{ad}} = [\mu_{\mathsf{a}}, \mu_{\mathsf{b}}]$ the parameter. We suppose that $g \in L^r(\Gamma)$ with $r > \mathfrak{d} - 1$, $\xi_1, ..., \xi_m, f \in H = L^2(\Omega)$, $\mu^{\mathsf{d}} = (\mu_j^{\mathsf{d}}) \in \mathbb{R}^m$, and $y_1^{\mathsf{d}}, ..., y_{k-1}^{\mathsf{d}} \in H$. Moreover, b, c, and d are non-negative constants with $c > 0$.

As Ω is a bounded connected open set with smooth boundary, it is known that V is a Hilbert space endowed with the inner product

$$\langle \varphi, \psi \rangle_V = \int_\Omega \nabla\varphi \cdot \nabla\psi \,\mathrm{d}\boldsymbol{x} + \int_\Gamma \varphi\psi \,\mathrm{d}\boldsymbol{s} \quad \text{for } \varphi, \psi \in V$$

and the induced norm $\|\varphi\|_V = \langle \varphi, \varphi \rangle_V^{1/2}$ for $\varphi \in V$, see ([25], p. 133) for instance.

3.2. The Parameter Dependent Semi-Linear Elliptic PDE

In this subsection, we study the state equation (11b). First, we define the nonlinear operator $\mathcal{A} : V \to V'$ by

$$\langle \mathcal{A}(y), \varphi \rangle_{V',V} = \int_\Omega c\nabla y \cdot \nabla\varphi + \left(by + dy^3\right)\varphi \,\mathrm{d}\boldsymbol{x} + \int_\Gamma y\varphi \,\mathrm{d}\boldsymbol{s} \quad \text{for } y, \varphi \in V.$$

Recall that $\mathfrak{d} \in \{1,2,3\}$ implies $V \hookrightarrow L^6(\Omega)$, cf. ([26] Section 7). Therefore, the operator $\mathcal{A}$ is well defined. Moreover, for $\mu \in \mathcal{P}_{\mathsf{ad}}$ the functional $b_\mu \in V'$ is given by

$$\langle b_\mu, \varphi \rangle_{V',V} = \int_\Omega \Big(f + \sum_{i=1}^{m} \mu_i \xi_i\Big)\varphi \,\mathrm{d}\boldsymbol{x} + \int_\Gamma g\varphi \,\mathrm{d}\boldsymbol{s} \quad \text{for } \varphi \in V.$$

Now, we define a weak solution to the state equation (11b).

Definition 5. *A* weak solution *of* (11b) *is a function* $y \in V$ *satisfying*

$$\langle \mathcal{A}(y), \varphi \rangle_{V',V} = \langle b_\mu, \varphi \rangle_{V',V} \quad \textit{for all } \varphi \in V. \tag{12}$$

The following result is proved, e.g., in ([26] Section 4.2.3).

Proposition 1. *For a fixed parameter $\mu \in \mathbb{R}^m$, there exists a unique solution $y \in V$ to (11b). This solution is even continuous on $\overline{\Omega}$, and for a constant c_∞ it holds that*

$$\|y\|_V + \|y\|_{C(\overline{\Omega})} \leq c_\infty \Big(\|g\|_{L^r(\Gamma)} + \Big\| f + \sum_{i=1}^m \mu_i \xi_i \Big\|_H \Big).$$

Remark 5. *We define the state space $\mathcal{Y} = V \cap C(\overline{\Omega})$, which is a Banach space endowed with the natural norm*

$$\|\varphi\|_{\mathcal{Y}} = \|\varphi\|_V + \|\varphi\|_{C(\overline{\Omega})} \quad \text{for } \varphi \in \mathcal{Y}.$$

Motivated by Proposition 1, we define the parameter-to-state mapping $\mathcal{S} : \mathcal{P}_{\mathsf{ad}} \to \mathcal{Y}$ as follows: For a given parameter $\mu \in \mathcal{P}_{\mathsf{ad}}$, the function $y = \mathcal{S}(\mu) \in \mathcal{Y}$ is the solution to (11b). It follows by standard arguments that $\mathcal{S}$ is continuously Fréchet-differentiable, see ([23] Sections 2 and 4). ♢

3.3. Reduced Formulation and Adjoint Approach

Utilizing the parameter-to-state mapping $\mathcal{S}$, we define the *reduced cost functional*

$$\hat{J}(\mu) = J(\mathcal{S}(\mu), \mu) = \begin{pmatrix} \hat{J}_1(\mu) \\ \vdots \\ \hat{J}_k(\mu) \end{pmatrix} \quad \text{for } \mu \in \mathcal{P}_{\mathsf{ad}}$$

with

$$\hat{J}_i(\mu) = \frac{1}{2} \int_\Omega \left| (\mathcal{S}(\mu))(\boldsymbol{x}) - y_i^{\mathsf{d}}(\boldsymbol{x}) \right|^2 \mathrm{d}\boldsymbol{x} \quad \text{for } 1 \leq i \leq k-1$$

and $\hat{J}_k(\mu) = \sum_{j=1}^m |\mu_j - \mu_j^{\mathsf{d}}|^2/2$. Now, the *reduced problem* is given as

$$\min \hat{J}(\mu) \quad \text{s.t.} \quad \mu \in \mathcal{P}_{\mathsf{ad}}. \tag{13}$$

If $\bar{\mu} \in \mathcal{P}_{\mathsf{ad}}$ is a locally optimal solution to (13), then the pair $(\mathcal{S}(\bar{\mu}), \bar{\mu})$ is a locally optimal solution to (11). Conversely, if $(\mathcal{S}(\bar{\mu}), \bar{\mu}) \in \mathcal{Y} \times \mathcal{P}_{\mathsf{ad}}$ solves (11) locally, the parameter $\bar{\mu}$ is a locally optimal solution to (13).

To apply the subdivision algorithm, the reduced objective function $\hat{J}$ has to be Fréchet-differentiable, following immediately from the fact that the parameter-to-state operator $\mathcal{S}$ is Fréchet-differentiable, cf. Remark 5. The gradient of $\hat{J}$ can be expressed by introducing adjoint variables. For that purpose, we define the operators $\mathcal{B}_i : V \times V \to V'$, $1 \leq i \leq k-1$, as

$$\langle \mathcal{B}_i(p,y), \varphi \rangle_{V',V} = \int_\Omega c \nabla p \cdot \nabla \varphi + \left(b + 3dy^2\right) p\varphi \,\mathrm{d}\boldsymbol{x} + \int_\Gamma p\varphi \,\mathrm{d}\boldsymbol{x} - \int_G (y - y_i^{\mathsf{d}})\varphi \,\mathrm{d}\boldsymbol{x}$$

for $p, y, \varphi \in V$. Next, we define adjoint variables.

Definition 6. *Let a parameter $\mu \in \mathcal{P}_{\mathsf{ad}}$ be given and $y(\mu) = \mathcal{S}(\mu) \in \mathcal{Y}$. For every $i \in \{1, \ldots, k-1\}$, we call the solution $p_i(\mu) \in V$ to*

$$\langle \mathcal{B}_i(p_i(\mu), y(\mu)), \varphi \rangle_{V',V} = 0 \quad \textit{for all } \varphi \in V \tag{14}$$

the adjoint variable associated with the objective $\hat{J}_i$.

Remark 6. (a) *Notice that (14) is the weak formulation of the elliptic problem*

$$\begin{aligned} -c\Delta p_i(\mu) + \left(b + 3dy(\mu)^2\right) p_i(\mu) &= y(\mu) - y_i^{\mathsf{d}} && \textit{in } \Omega, \\ c\frac{\partial p_i}{\partial \boldsymbol{n}}(\mu) + p_i(\mu) &= 0 && \textit{on } \Gamma \end{aligned} \tag{15}$$

for $i = 1, \dots, k-1$.

(b) *Applying the Lax–Milgram lemma (see, e.g.,* ([26] Section 2)*) one can show that* (14) *has a unique solution* $p_i(\mu) \in V$ *for all* $\mu \in \mathcal{P}_{\mathsf{ad}}$ *and* $i = 1, \dots, k-1$. ◊

Now, we can express the gradient of the reduced cost functional as follows:

$$\nabla \hat{J}_i(\mu) = \begin{pmatrix} \int_\Omega \xi_1 p_i(\mu)\,\mathrm{d}\boldsymbol{x} \\ \vdots \\ \int_\Omega \xi_m p_i(\mu)\,\mathrm{d}\boldsymbol{x} \end{pmatrix} \text{ for } 1 \le i \le k-1, \quad \nabla \hat{J}_k(\mu) = \mu - \mu^{\mathsf{d}},$$

where $y(\mu) \in \mathcal{Y}$ and $p_i(\mu) \in V$, $i \in \{1, \dots, k-1\}$, solve (12) and (14), respectively.

3.4. Finite Element (FE) Galerkin Discretization

Let us briefly introduce a standard finite element (FE) method based on a triangulation of the spatial domain Ω. Here, we utilize piecewise linear FE ansatz functions $\varphi_1, \dots, \varphi_N \in V$, which are linearly independent. We define the finite-dimensional subspace $V^N = \operatorname{span}\{\varphi_1, ..., \varphi_N\} \subset V$ supplied with the same topology as in V.

Next, we replace (12) by a FE Galerkin scheme: For each $\mu \in \mathcal{P}_{\mathsf{ad}}$, the FE solution $y^N(\mu) \in V^N$ solves

$$\langle \mathcal{A}(y^N(\mu)), \varphi_i \rangle_{V',V} = \langle b_\mu, \varphi_i \rangle_{V',V} \quad \text{for } i = 1, \dots, N. \tag{16}$$

It follows by the same arguments as for (12) that the FE problem (16) has a unique solution $y^N = y^N(\mu)$ for every $\mu \in \mathcal{P}_{\mathsf{ad}}$. Therefore, the parameter-to-state FE mapping $\mathcal{S}^N : \mathcal{P}_{\mathsf{ad}} \to V^N$, $\mu \mapsto \mathcal{S}^N(\mu) = y^N(\mu)$ is well defined. For $y^N \in V^N$ there exist coefficients y_i^N, $i = 1, \dots, N$, satisfying

$$y^N = \sum_{i=1}^{N} \mathrm{y}_i^N \varphi_i \quad \text{for } \boldsymbol{x} \in \Omega. \tag{17}$$

Inserting (17) into (16), we can express (16) as a nonlinear algebraic system. For that purpose, we introduce the $N \times N$-matrices

$$\mathrm{K} = \left(\left(\int_\Omega \nabla\varphi_j \cdot \nabla\varphi_i \,\mathrm{d}\boldsymbol{x}\right)\right), \quad \mathrm{M} = \left(\left(\int_\Omega \varphi_j\varphi_i \,\mathrm{d}\boldsymbol{x}\right)\right), \quad \mathrm{Q} = \left(\left(\int_\Gamma \varphi_j\varphi_i \,\mathrm{d}\boldsymbol{s}\right)\right),$$

the N-vectors

$$\mathrm{y}^N = (\mathrm{y}_i^N), \quad \mathrm{F}_\mu = \left(\int_\Omega \left(f + \sum_{j=1}^{m} \mu_j \xi_j\right)\varphi_i \,\mathrm{d}\boldsymbol{x}\right), \quad \mathrm{G} = \left(\int_\Gamma g\varphi_i \,\mathrm{d}\boldsymbol{s}\right),$$

and the nonlinearity function $\mathrm{H} : \mathbb{R}^N \to \mathbb{R}^N$ given as

$$\mathrm{H}(\mathrm{v}) = \left(\int_\Omega \left(\textstyle\sum_{j=1}^N \mathrm{v}_j\varphi_j\right)^3 \varphi_i \,\mathrm{d}\boldsymbol{x}\right) \quad \text{for } \mathrm{v} = (\mathrm{v}_1, \dots, \mathrm{v}_N) \in \mathbb{R}^N.$$

Inserting (17) into (16), we end up with the following nonlinear system:

$$(c\mathrm{K} + b\mathrm{M} + \mathrm{Q})\mathrm{y}^N + d\mathrm{H}(\mathrm{y}^N) = \mathrm{F}_\mu + \mathrm{G} \in \mathbb{R}^N. \tag{18}$$

Remark 7. *The difficulty of* (18) *lies in the fact that we cannot assemble* $\mathrm{H}(\mathrm{y}^N)$ *efficiently in terms of a matrix-vector multiplication. Therefore, we use* mass lumping *to compute* $\mathrm{H}(\mathrm{y}^N)$ *approximately. For an introduction into mass lumping see, e.g.,* ([27] Section 15) or ([28] Section 17.2). *With that we can write* (18) *as a root finding problem of*

$$(c\mathrm{K} + b\mathrm{M} + \mathrm{Q})\mathrm{y}^N + d\tilde{M}(\mathrm{y}^N)^3 = \mathrm{F}_\mu + \mathrm{G} \in \mathbb{R}^N \tag{19}$$

with $(\mathrm{y}^N)^3 = ((\mathrm{y}_i^N)^3)_{1\le i\le N}$ *and the lumped mass matrix* $\tilde{M}$ *defined by*

$$\tilde{M}_{kj} := \delta_{kj} \sum_{l=1}^{N} M_{kl}.$$

Further details can be found in [23,29]. *Let us refer to in* [30], *where mass lumping is utilized in optimal control.* ◊

Now, the FE objectives are given as

$$\hat{J}_i^N(\mu) = \frac{1}{2}\int_\Omega \left|\mathcal{S}^N(\mu) - y_i^{\mathsf{d}}\right|^2 \mathrm{d}\boldsymbol{x} \text{ for } i = 1, ..., k-1, \quad \hat{J}_k^N(\mu) = \sum_{j=1}^{m} \left|\mu_j - \mu_j^{\mathsf{d}}\right|^2.$$

3.5. Reduced-Order Modelling (ROM)

To generate the Pareto-critical set of the MOP (11), we need to evaluate the reduced objectives $\hat{J}_i$, $i = 1, \ldots, k$, and their gradients many times. Therefore, the state and adjoint equations have to be solved numerically very often. Therefore, the use of ROM is a suitable option. In this paper, we will use the Reduced Basis (RB) method. The main idea is to construct a low-dimensional (i.e., $\ell \ll N$) subspace V^ℓ of the FE space V^N spanned by FE solutions of the state and adjoint equation for appropriately chosen parameters $\mu \in \mathcal{P}_{\mathsf{ad}}$. Here, this strategy is realized by greedy algorithms. We refer to the works in [17,18,31] for a general explanation and to ([23] Section 3.2) for our specific problem (11). This is an iterative procedure where in each iteration FE solutions of the state and adjoint equation at a specific parameter value are added to the basis. An essential ingredient of greedy algorithms is the choice of an error indicator $\eta_\ell(\mu)$. Here, we use the maximal true error between the FE and the ROM gradients, i.e., we set

$$\eta_\ell(\mu) := \max_{1\le i\le k} \|\nabla \hat{J}_i^N(\mu) - \nabla \hat{J}_i^\ell(\mu)\|_2. \tag{20}$$

Our subdivision scheme is based on gradient information. To be able to generate $\mathbb{P}_1^\ell$ or $\mathbb{P}_2^\ell$ for the approximation of $\mathbb{P}$, we have to ensure that the approximated objective function $\hat{J}^\ell$ based on the ROM model satisfies the inequality in (5). The idea is to generate a new basis element in every step until the maximum error $\eta_\ell(\mu)$ on a discrete training set S_{train}, which approximates $\mathcal{P}_{\mathsf{ad}}$ good enough, is smaller than a tolerance ε_{tol}. For more details, see Algorithm 4.

As V^ℓ is a subset of V, we endow V^ℓ with the V-topology. Due to $\psi_j \in V^N$ $(1 \le j \le \ell)$, there exists a coefficient matrix $\Psi \in \mathbb{R}^{N\times\ell}$ such that

$$\psi_j(\boldsymbol{x}) = \sum_{i=1}^{N} \Psi_{ij}\varphi_i(\boldsymbol{x}) \quad \text{for } \boldsymbol{x} \in \Omega. \tag{21}$$

Now, we replace (16) by an RB Galerkin scheme: For each $\mu \in \mathcal{P}_{\mathsf{ad}}$, the RB solution $y^\ell(\mu) \in V^\ell$ solves

$$\langle \mathcal{A}(y^\ell(\mu)), \psi_i\rangle_{V',V} = \langle b_\mu, \psi_i\rangle_{V',V} \quad \text{for } i = 1, \ldots, \ell. \tag{22}$$

We suppose that (22) has a unique solution $y^\ell = y^\ell(\mu)$ for every $\mu \in \mathcal{P}_{\mathsf{ad}}$. Therefore, the parameter-to-state RB mapping $\mathcal{S}^\ell : \mathcal{P}_{\mathsf{ad}} \to V^\ell$, $\mu \mapsto \mathcal{S}^\ell(\mu) = y^\ell(\mu)$ is well defined.

Inserting (21) and $y^\ell(\mu) = \sum_{i=1}^N \mathrm{y}_i^\ell(\mu)\psi_i$ into (22), we derive the nonlinear algebraic system (cf. (18))

$$\left(c\mathrm{K}^\ell + b\mathrm{M}^\ell + \mathrm{Q}^\ell\right) + d\Psi^\top \mathrm{H}\left(\Psi \mathrm{y}^\ell\right) = \mathrm{F}_\mu^\ell + \mathrm{G}^\ell \in \mathbb{R}^l \quad (\ell \ll N) \tag{23}$$

with the $\ell \times \ell$ matrices $\mathrm{K}^\ell = \Psi^\top \mathrm{K}\Psi$, $\mathrm{M}^\ell = \Psi^\top \mathrm{M}\Psi$, and $\mathrm{Q}^\ell = \Psi^\top \mathrm{Q}\Psi$ and the ℓ-vectors $\mathrm{F}_\mu^\ell = \Psi^\top \mathrm{F}_\mu$ and $\mathrm{G}^\ell = \Psi^\top \mathrm{G}$.

Algorithm 4: Greedy algorithm.

Require: Tolerance $\varepsilon_{tol} > 0$, error indicator $\eta_\ell(\mu)$, discrete training set S_{train};
Return : RB space V^ℓ and basis Ψ_ℓ;
Set $V_0^\ell := \{0\}$, $\Psi_0 := \emptyset$, $\ell := 0$, $M := 0$;
while $\max_{\mu \in S_{train}} \eta_\ell(\mu) > \varepsilon_{tol}$ *and* $M \leq |S_{train}|$ **do**
  Set $\mu_{M+1} \in \operatorname{argmax} \{\eta_\ell(\mu) : \mu \in S_{train}\}$;
  $\tilde{\psi}^p_{M+1} := p_N(\mu_{M+1})$ and $\tilde{\psi}^y_{M+1} := y_N(\mu_{M+1})$;
  Perform Gram–Schmidt orthonormalization (GS) of $\tilde{\psi}^p_{M+1}$ against Ψ_M and get ψ^p_{M+1};
  Set $\Psi_{M+1} := \{\Psi_M, \psi^p_{M+1}\}$, $V^\ell_{M+1} := {V^N}^\ell_M \oplus \{\psi^p_{M+1}\}$;
  if $\tilde{\psi}^y_{M+1}$ *is linearly independent of* V^ℓ_{M+1} **then**
    Perform GS orthonormalization of $\tilde{\psi}^y_{M+1}$ against Ψ_{M+1} and get ψ^y_{M+1};
    Set $\Psi_{M+2} := \{\Psi_{M+1}, \psi^y_{M+1}\}$, $V^\ell_{M+2} := {V^N}^\ell_{M+1} \oplus \{\psi^y_{M+1}\}$ and $M := M + 2$;
  else
    Set $M := M + 1$;
  end
end

Remark 8. *As mentioned in Remark 7, we apply mass lumping to evaluate the nonlinear function* H *more efficiently. With that we can write* (23) *as a root finding problem of*

$$\left(c\mathrm{K}^\ell + b\mathrm{M}^\ell + \mathrm{Q}^\ell\right)\mathrm{y}^\ell + d\tilde{\mathrm{M}}^\ell\left(\Psi \mathrm{y}^\ell\right)^3 = \mathrm{F}^\ell_\mu + \mathrm{G}^\ell \in \mathbb{R}^l \quad (\ell \ll N) \tag{24}$$

with $(\Psi y^\ell)^3 = (((\Psi y^\ell)^3_i))_{1 \leq i \leq N}$ *and the* $\ell \times N$ *matrix* $\tilde{\mathrm{M}}^\ell = \Psi^\top \tilde{\mathrm{M}}$. *However, in the RB Galerkin scheme the evaluation of the nonlinearity is still as costly as in the FE case. Here, discrete empirical interpolation (DEIM) is applied, cf.* [19,20]. *We skip the detailed description here and refer the reader to* ([23] Section 3.2). ◊

3.6. Convergence Analysis

We prove the convergence of the RB solution with mass lumping to the weak solution of the state and adjoint equation.

Remark 9. *As the FE space is a finite dimensional space, the error for the state and adjoint equation converges for increasing dimension of the RB space to zero. Therefore, the RB solution converges for increasing accuracy in the Greedy algorithm to the FE solution. We skip a detailed description of the proof here and refer the reader to* [23, Section 3.2]. ◊

Theorem 4. *Let* $(S_j)_j \subset \mathcal{P}_{\mathsf{ad}}$ *with* $S_j \subseteq S_{j+1}$ *be a growing sequence of training sets and choose a monotone sequence* $(\varepsilon_j)_j \subset \mathbb{R}_{>0}$ *satisfying*

$$\varepsilon_j \to 0 \text{ for } j \to \infty \quad \text{and} \quad \varepsilon_j \geq \varepsilon_{j+1}.$$

Then, we get

$$\limsup_{j\to\infty} \left\{ \|y(\mu) - y^\ell(\mu)\|_V \,\middle|\, \mu \in \mathcal{P}_{\mathsf{ad}} \right\} = 0, \quad \limsup_{j\to\infty} \left\{ \|p(\mu) - p^\ell(\mu)\|_V \,\middle|\, \mu \in \mathcal{P}_{\mathsf{ad}} \right\} = 0.$$

Proof. Let $\mu \in \mathcal{P}_{\mathsf{ad}}$ be an arbitrary parameter. It holds that

$$\|y(\mu) - y^\ell(\mu)\|_V \leq \|y(\mu) - y^N(\mu)\|_V + \|y^N(\mu) - y^\ell(\mu)\|_V$$

with $N > \ell$. From Theorem A1 (see Appendix A), we infer that the first summand

converges to zero for increasing N. For the second summand, we refer the reader to ([23] Section 3.2.5); in this section we proved the convergence of the RB solution to the FE solution. For the adjoint equation, we can do the same and the claim follows. □

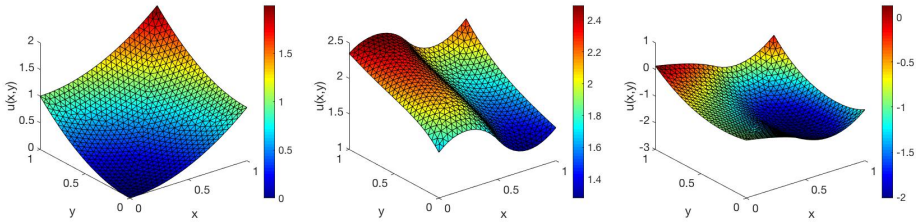
4. Numerical Experiments

In this section, we use our algorithm to solve multiobjective optimization problems with PDE constraints and interpret the numerical results. All computations were executed on a computer with a 2.9 GHz Intel Core i7 CPU, 8 GB of RAM, and an Intel HD Graphic 4000 1536 MB GPU. The algorithms were implemented in Matlab R2017b. For the subdivision method, we used the implementation from https://math.uni-paderborn.de/en/ag/chair-of-applied-mathematics/research/software.

In this example, we will numerically investigate the application of the modified subdivision algorithm presented in Section 2.5 to the PDE-constrained multiobjective optimization problem using the RB-DEIM solver from Section 3.5. For the underlying PDE, we set $\mathfrak{d} = 2$, $\Omega = (0,1)^2$ with elements $\boldsymbol{x} = (x_1, x_2)$, $\mathcal{P}_{\mathsf{ad}} = [-2,2]^2$, and $b = c = d = 1$; the right-hand side $f(\boldsymbol{x}) = x_1^2 + x_2^2 - 4 + (x_1^2 + x_2^2)^3$, $m = 2$, $\xi_1(\boldsymbol{x}) = -25 \cdot \mathbb{1}_{x_1 > 0.5}(\boldsymbol{x})$, and $\xi_2(\boldsymbol{x}) = 25 \cdot \mathbb{1}_{x_1 \leq 0.5}(\boldsymbol{x})$; and the boundary condition $g(\boldsymbol{x}) = 2 \cdot \mathbb{1}_{x_1=1}(\boldsymbol{x}) + 2 \cdot \mathbb{1}_{x_2=1}(\boldsymbol{x}) + x_1^2 + x_2^2$. This leads to the following PDE:

$$-\Delta y + y + y^3 = f + \mu_1 \xi_1 + \mu_2 \xi_2 \text{ in } \Omega, \quad \frac{\partial y}{\partial \boldsymbol{n}} + y = g \text{ on } \partial\Omega. \tag{25}$$

In Figure 1, the corresponding solutions of the state equation are shown for three values of μ.



(**a**) (**b**) (**c**)

Figure 1. FE solutions of (25) for parameters (**a**) $\mu = (0,0)$, (**b**) $\mu = (0,1)$ and (**c**) $\mu = (1,0)$.

In [23], we have already observed that the error between the FE- and RB-DEIM-solution of the state and adjoint equation decreases if the FE grids get finer. We skip the detailed description here and refer the reader to ([23] Section 5).
Notice that $y(\boldsymbol{x}) = x_1^2 + x_2^2$ solves (25) for $\mu = (0,0)$. For the FE-solver, we used linear finite elements with $\Delta H_{\max} = 0.04$ and the finite elements have 762 degrees of freedom. We choose the following two objective functions:

$$\hat{J}_1(\mu) = \frac{1}{2} \int_\Omega |y(\mu) - y^{\mathsf{d}}|^2 \, \mathrm{d}\boldsymbol{x}, \quad \hat{J}_2(\mu) = \frac{1}{2} \sum_{j=1}^{2} |\mu_j - \mu_j^{\mathsf{d}}|^2$$

with $\mu^{\mathsf{d}} = (1,1)$. For the desired state y^{d}, we take the FE solution for $\mu = 0$, i.e., $y^{\mathsf{d}} = y^N(0)$. Thus, $y^{\mathsf{d}} = \sum_{i=1}^{N} \mathrm{y}_i^{\mathsf{d}} \varphi_i$ is a piecewise linear approximation of $x_1^2 + x_2^2$. The associated FE objectives are now given as

$$\hat{J}_1^N(\mu) = \frac{1}{2} (\mathrm{y}^N(\mu) - \mathrm{y}^{\mathsf{d}})^\top \mathrm{M} (y^N(\mu) - \mathrm{y}^{\mathsf{d}}), \quad \hat{J}_2^N(\mu) = \frac{1}{2} \sum_{j=1}^{2} |\mu_j - \mu_j^{\mathsf{d}}|^2$$

with $\mathrm{y}^{\mathsf{d}} = (\mathrm{y}_i^{\mathsf{d}})_{1\le i\le N}$. The gradients are

$$\nabla \hat{J}_1^N(\mu) = \begin{pmatrix} \mathrm{p}^N(\mu)^\top \bar{F}_1 \\ \mathrm{p}^N(\mu)^\top \bar{F}_2 \end{pmatrix}, \quad \nabla \hat{J}_2^N(\mu) = \mu - \mu^{\mathsf{d}}$$

where $p^N(\mu) = \sum_{j=1}^N \mathrm{p}_j^N(\mu)\varphi_j$ is the FE solution to (15), $\mathrm{p}^N(\mu) = (\mathrm{p}_j^N(\mu))_{1\le j\le N}$ and $\bar{F}_i = (\int_\Omega \varphi_j \xi_i \,\mathrm{d}\boldsymbol{x})_{1\le j\le N}$. The associated RB objective functions have the form

$$\hat{J}_1^\ell(\mu) = \frac{1}{2}\big(\mathrm{y}^\ell(\mu) - \mathrm{y}^{\mathsf{d},\ell}\big)^\top \mathrm{M}^\ell \big(\mathrm{y}^\ell(\mu) - \mathrm{y}^{\mathsf{d},\ell}\big), \quad \hat{J}_2^\ell(\mu) = \frac{1}{2}\sum_{j=1}^m |\mu_j - \mu_j^{\mathsf{d}}|^2$$

with $y^{\mathsf{d},\ell} = \Psi^\top \mathrm{y}^{\mathsf{d}}$. In [23], we have already observed good agreement between the approximated Pareto critical sets with the FE- and RB-DEIM-solver, if the error between the gradients is sufficiently small. However, we cannot always guarantee this agreement. Thus, we will instead use the supersets $\mathbb{P}_1^\ell$ and $\mathbb{P}_2^\ell$ from Section 2.3 for the approximation of $\mathbb{P}$ (and $\mathbb{P}_c$), which is only based on the reduced objective function $\hat{J}^\ell$ and the error bounds ε. To generate $\mathbb{P}_1^\ell$, we compute the steepest descent direction for all components of $\hat{J}^\ell$. Similar to Algorithm 1, we first calculate α_k as solution of (4) for μ_k. Then, the descent direction is $v_k := -D\hat{J}^\ell(\mu_k)^\top \alpha_k$. As a stopping condition, we choose $\sigma = \|\varepsilon\|_\infty$ and set $a(\mu_k) = \mu_k$ if it holds that

$$\|D\hat{J}^\ell(\mu_k)^\top \alpha_k\|_2 = \|v_k\|_2 < \sigma = \|\varepsilon\|_\infty.$$

To generate $\mathbb{P}_2^\ell$, we use Algorithm 2.
To save computational time during the modified subdivision algorithm, we calculate

$$\max_{1\le i\le k} \|\nabla \hat{J}_i^N(\mu) - \nabla \hat{J}_i^\ell(\mu)\|_2$$

before the algorithm starts (i.e., in an offline phase) on a training set $\mathcal{P}_{\mathsf{train}}$, which approximates $\mathcal{P}_{\mathsf{ad}}$ and store these errors. During the subdivision algorithm, we use them to generate the new ε_{i+1}^l faster and without calculating the FE solution again.
To see a significant difference between the modified subdivision algorithm and the subdivision algorithm and $\mathbb{P}_2^\ell$ and $\mathbb{P}_1^\ell$, we need to have a big error between the gradients. To achieve this, we choose a rough training set

$$S_{train} = \big\{(-2 + 0.51k, -2 + 0.427j)^\top \,\big|\, (k,j) \in \{0,\dots,7\} \times \{0,\dots,9\}\big\}$$

with $|S_{train}| = 80$. For the Greedy algorithm, we choose the tolerance $\varepsilon_{tol} = 0.06$ and the true error $\eta_\ell(\mu) := \max_{1\le i\le k} \|\nabla \hat{J}_i^N(\mu) - \nabla \hat{J}_i^\ell(\mu)\|_2$ as error indicator, for the DEIM algorithm we choose the tolerance $\bar{\varepsilon} = 0.5$.
With these settings, we generate an RB-basis with six elements and a DEIM-basis with 18 elements. This leads to the following estimations for the gradients of $\hat{J}^N$ and $\hat{J}^\ell$:

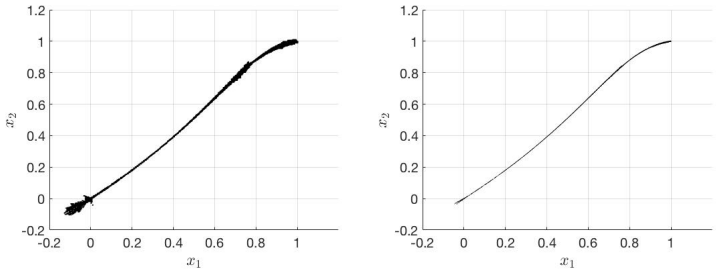
$$\begin{aligned} \max_{\mu\in\mathcal{P}_{\mathsf{ad}}} \|\nabla \hat{J}_1^N(\mu) - \nabla \hat{J}_1^\ell(\mu)\|_2 &\approx \max_{\mu\in\mathcal{P}_{\mathsf{train}}} \|\nabla \hat{J}_1^N(\mu) - \nabla \hat{J}_1^\ell(\mu)\|_2 \approx 0.0491 = \varepsilon_1, \\ \max_{\mu\in\mathcal{P}_{\mathsf{ad}}} \|\nabla \hat{J}_2^N(\mu) - \nabla \hat{J}_2^\ell(\mu)\|_2 &= \max_{\mu\in\mathcal{P}_{\mathsf{ad}}} \|\nabla \hat{J}_2(\mu) - \nabla \hat{J}_2^\ell(\mu)\|_2 = 0 = \varepsilon_2. \end{aligned}$$

To generate these estimations, we chose a training set $\mathcal{P}_{\mathsf{train}} \subset \mathcal{P}_{\mathsf{ad}}$ with 3105 equally distributed test points and generate the error for these points. Thus, we have

$$\varepsilon^0 = \begin{pmatrix} 0.0491 \\ 0 \end{pmatrix}.$$

Now, we test the subdivision and the modified subdivision algorithm with the following conditions. To compute the descent direction, we use the Matlab function *fmincon* and

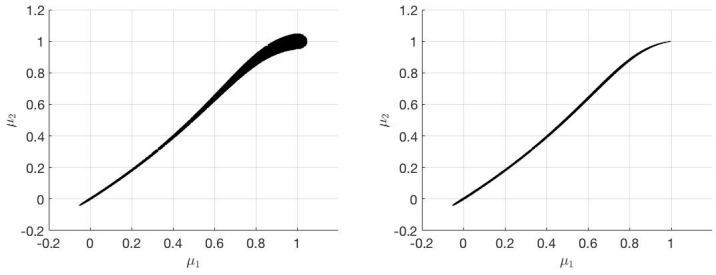
solve (4) or (7). The algorithm stops when the box size is small enough, which is after 25 iteration steps in our case. In every step, we halve the boxes. We choose five sample points in each box during the first five iteration steps, four sample points in the next five iteration steps, three sample points for the iteration steps ten to 14, two sample points for the next five steps, and one sample points for the last five iteration steps, see Remark 4. Figure 2 shows the generated approximated Pareto sets for the FE-solver after 20 and 25 iteration steps.



(**a**) (**b**)

Figure 2. Pareto-critical set $\mathbb{P}_c$ in iteration step (**a**) 20 and (**b**) 25 for the FE-solver.

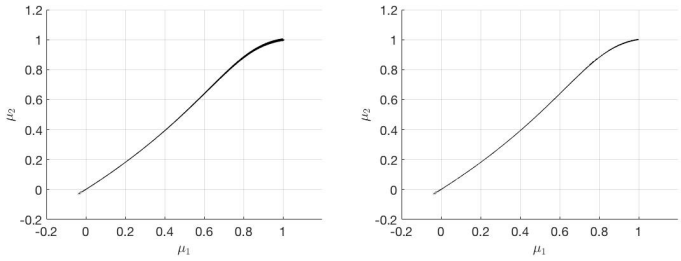
The results with the subdivision algorithm and RB-DEIM-solver are shown in Figure 3.



(**a**) (**b**)

Figure 3. (**a**) $\mathbb{P}_1^\ell$ and (**b**) $\mathbb{P}_2^\ell$ generated with the subdivision algorithm.

The modified subdivision algorithm and RB-DEIM-solver lead to the results plotted in Figure 4.



(**a**) (**b**)

Figure 4. (**a**) $\mathbb{P}_1^\ell$ and (**b**) $\mathbb{P}_2^\ell$ generated with the modified subdivision algorithm.

The runtime, number of boxes and number of function and gradient evaluations needed in iteration step 10, 15, 20, and 25 are shown in Tables 1–5. The total runtime and number of function and gradient evaluations needed for the different methods and the speed-up are shown in Table 6.

Table 1. The performance of the subdivision algorithm with the FE-solver and the steepest descent method.

FE				
It. Step	**Time [s]**	**# Grad. Solves**	**# Del. Boxes**	**# Boxes**
10	370.7	6872	30	64
15	1077.1	16786	166	322
20	1239.1	20092	938	1388
25	3364.2	64721	2567	4749

Table 2. Subdivision algorithm with the steepest descent method.

RB-DEIM					
It. Step	**Time [s]**	**# Grad. Solves**	**# Del. Boxes**	**# Boxes**	ε
10	18.8	6809	32	64	$(0.0491, 0)$
15	42.4	14279	168	326	$(0.0491, 0)$
20	44.3	13781	622	3412	$(0.0491, 0)$
25	848.6	196405	149	97551	$(0.0491, 0)$

Table 3. Subdivision algorithm with the modified descent direction.

RB-DEIM					
It. Step	**Time [s]**	**# Grad. Solves**	**# Del. Boxes**	**# Boxes**	ε
10	21.0	6792	31	63	$(0.0491, 0)$
15	43.1	15546	167	315	$(0.0491, 0)$
20	38.0	12655	878	1438	$(0.0491, 0)$
25	289.5	64606	233	30607	$(0.0491, 0)$

Table 4. Modified subdivision algorithm with the steepest descent method.

RB-DEIM					
It. Step	**Time [s]**	**# Grad. Solves**	**# Del. Boxes**	**# Boxes**	ε
10	18.5	6829	32	64	$(0.041, 0)$
15	44.1	16307	170	318	$(0.0084, 0)$
20	45.6	16677	919	1357	$(0.0072, 0)$
25	121.2	29811	560	12230	$(0.0072, 0)$

Table 5. Modified subdivision algorithm with the modified descent direction.

RB-DEIM					
It. Step	**Time [s]**	**# Grad. Solves**	**# Del. Boxes**	**# Boxes**	ε
10	18.7	6805	31	263	$(0.041, 0)$
15	44.1	16459	168	318	$(0.0084, 0)$
20	49.7	18266	937	1329	$(0.0072, 0)$
25	109.7	37170	1615	4129	$(0.0043, 0)$

Table 6. Total runtime, number of function, and gradient evaluations for the different methods and the speed-up.

Algorithm	Solver	Method	Time [min]	Speed-Up	# Grad. Solves
subdivision	FE	steep. desc.	495.83	-	610728
subdivision	RB-DEIM	steep. desc.	41.77	11.9	622034
subdivision	RB-DEIM	mod. desc.	24.77	20.0	390362
mod. subdivision	RB-DEIM	steep. desc.	21.00	23.6	372551
mod. subdivision	RB-DEIM	mod. desc.	21.91	22.6	412667

The Greedy algorithm and DEIM together take 14.06 s in the offline phase. To ensure the error ε (cf. (5)) for the gradients on the training set $\mathcal{P}_{\text{train}}$, it takes 151.13 s. It follows from Figures 3 and 4 that $\mathbb{P}_2^\ell$ is significantly smaller than $\mathbb{P}_1^\ell$ for the subdivision algorithm as well as for the modified subdivision algorithm. Therefore, we have a much better approximation for $\mathbb{P}_c$ if we choose $\mathbb{P}_2^\ell$ instead of $\mathbb{P}_1^\ell$. If we compare the two different subdivision algorithms, we notice that with the modified subdivision algorithm we have a better approximation of $\mathbb{P}_c$ than with the subdivision algorithm.

The reason for this is that in the modified subdivision algorithm we have a monotonically decreasing sequence ε^l. In Figure 5, the error between $\nabla \hat{f}_1^N$ and $\nabla \hat{f}_1^\ell$ is plotted on the parameter set $\mathcal{P}_{\text{ad}}$. The black markers are some points on the Pareto critical set $\mathbb{P}_c$. It turns out that the difference between the two gradients is significantly smaller near $\mathbb{P}_c$ than in other regions of $\mathcal{P}_{\text{ad}}$. Due to this, in the modified subdivision algorithm, the sequence ε^l decreases noticeably so that it is useful to update ε after each iteration step. As we have already mentioned, $\mathbb{P}_2^\ell$ is a better approximation of $\mathbb{P}_c$. If $\mathbb{P}_2^\ell$ is generated by the modified subdivision algorithm, the result is even better. Comparing Figures 2b and 4b, there is no significant difference between the two sets. Therefore, the modified subdivision algorithm yields a good approximation $\mathbb{P}_2^\ell$ for $\mathbb{P}_c$, although the error ε_1 is not small.

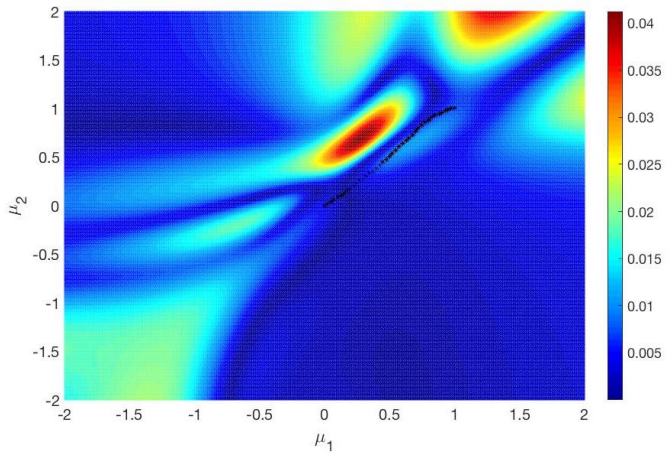


Figure 5. Difference $\|\nabla \hat{f}_1^N(\mu) - \nabla \hat{f}_1^\ell(\mu)\|_2$ for $\mu \in \mathcal{P}_{\text{ad}}$.

Regarding the computational time (cf. Tables 1–6), we notice that in the first 20 iterations the four methods take almost the same time. In these iteration steps, the FE-solver takes between 19 and 30 times more time for one iteration step than the four RB-based methods. Only in the subsequent iteration steps a significant difference in the four RB-based methods appears. For these iteration steps, the FE-solver takes between 3.9 and 30.8 times as much time as one of the four methods for one iteration step. We notice that the computation of $\mathbb{P}_1^\ell$ takes around two times as much time as the computation of $\mathbb{P}_2^\ell$ with the subdivision algorithm. The main reason for this is the much larger number of function and gradient evaluations that we have for $\mathbb{P}_1^\ell$. This, in turn, can be attributed to the significantly larger number of boxes for $\mathbb{P}_1^\ell$. If we use the modified subdivision algorithm, the computation of $\mathbb{P}_2^\ell$ takes slightly more time than the computation of $\mathbb{P}_1^\ell$. The main

reason for this is again the larger number of function and gradient evaluations. Unlike the previous case, this can not be lead back to the number of boxes, but probably to the calculation of the modified descent method, which requires more function and gradient evaluations. Nevertheless, the difference in the computational time is marginal (a factor about 1.04) for the modified subdivision algorithm. For $\mathbb{P}_1^\ell$, we notice another behavior: Here, the generation of $\mathbb{P}_1^\ell$ with the modified subdivision algorithm is 2 times faster than with the subdivision algorithm. This is mainly because of the larger number of function evalutions, which is due to the larger number of boxes. The FE-solver takes approximately 23.6 times as much time as the computation of $\mathbb{P}_1^\ell$ and approximately 22.6 times as much as the time as the computation of $\mathbb{P}_2^\ell$ with the modified subdivsion. This is another advantage of the modified subdivision algorithm. As we get a better approximation in this algorithm, we have fewer boxes in the iteration steps, and therefore we have a smaller number of function and gradient evaluations than we have for the subdivision algorithm. Finally, we see that the modified subdivision algorithm works better and faster than the subdivision algorithm. As $\mathbb{P}_2^\ell$ is a tighter approximation of $\mathbb{P}_c$, it is better to generate $\mathbb{P}_2^\ell$ rather than $\mathbb{P}_1^\ell$.

5. Conclusions

In this article, we present a way to solve multiobjective parameter optimization problems of semilinear elliptic PDEs by combining an RB approach and DEIM with the set-oriented method based on gradient evaluations. To deal with the error introduced by the surrogate model, we derived an additional condition for the descent direction, which allows us to consider the errors for the objective functions independently and derive a superset $\mathbb{P}_2^\ell$ of the Pareto-critical set $\mathbb{P}_c$. To get an even tighter superset, we update these error bounds in our subdivision algorithm after each iteration step. To summarize the numerical results, we first investigated the influence of the error bounds for the gradients of the objective functions. By individually adapting the components of the error bounds, we obtained a tighter covering of the Pareto critical set. When we additionally adjusted the error bounds in each iteration step, the result became even tighter and almost coincided with the exact solution of the MOP (solution with the FE-solver and the steepest descent method). Furthermore, we compared the computational time for each method. The FE-solver needed between 11.9 times and 23.6 times more time than the four different RB-based methods we presented in this work. For future work, it could be interesting to improve the results in ([23] Section 3.2.4) and to develop an efficient a posteriori error estimator for the error in the objectives and their gradients, cf., e.g., [32–34]. These error bounds can then be used in a weak greedy algorithm and beyond that for the error bounds which are needed in the subdivision algorithm.

Author Contributions: The authors contributed equally to this work. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded partially by Deutsche Forschungsgemeinschaft grant number Priority Programme 1962.

Acknowledgments: The authors gratefully acknowledge partial support by the German DFG-Priority Program 1962. Furthermore, S. Banholzer acknowledges his partial funding by the Landesgraduiertenförderung of Baden-Württemberg.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MOP	Multiobjective optimization problem
DEIM	Discrete Empirical Interpolation Method
FE	Finite Element
GS	Gram–Schmidt
KKT	Karush–Kuhn–Tucker
PDE	Partial-differential equation
POD	Proper Orthogonal Decomposition
RB	Reduced Basis
ROM	Reduced Order Modeling

Appendix A. Mass Lumping

We follow the work in [35]. Let $\Omega \subset \mathbb{R}^{\mathfrak{d}}$, $\mathfrak{d} \in \{1,2,3\}$ be an open, bounded Lipschitz domain, e.g., $\Omega = (0,1)^{\mathfrak{d}}$. We set $H = L^2(\Omega)$ and $V = H^1(\Omega)$, where V is endowed with the inner product

$$\langle \varphi, \varphi \rangle_V = \int_\Omega \nabla\varphi \cdot \nabla\varphi \, \mathrm{d}\boldsymbol{x} + \int_\Gamma \varphi\varphi \, \mathrm{d}\boldsymbol{s} \quad \text{for } \varphi, \varphi \in V$$

and its induced norm. Let $\mathcal{T}^N$ denote an underlying triangulation of Ω and $V(\mathcal{T}^N) = \{z^1, ..., z^N\}$ denote the set of interior vertices of $\mathcal{T}^N$, and let $V(\tau)$ be the set of vertices belonging to $\tau \in \mathcal{T}^N$ and $V^N = \text{ span}\{\varphi^1, .., \varphi^N\}$ be the FE space generated by first order Lagrange elements. It holds that $\varphi^j(z^l) = \delta_{jl}$.

Define for $z^j \in V(\mathcal{T}^N)$ a lumped mass region Ω^j by joining the centroids of the triangles, which have z^j as a common vertex, to the midpoint of the edges, which have z^j as a common extremity. We define the two sets

$$V^N = \left\{ v \in C(\overline{\Omega}) \,\middle|\, v|_{\tau_j} \in P_1 \text{ for all } \tau_i \in \mathcal{T}^N \right\} \subset V,$$
$$H^N = \left\{ v \in L^\infty(\Omega) \,\middle|\, v = \sum_{j=1}^N \mathbb{1}_{\Omega^j} v^j,\ v^j \in \mathbb{R} \right\} \subset H$$

and the operator

$$\mathcal{R}^N : C(\overline{\Omega}) \to H^N, \quad \mathcal{R}^N(\omega) = \sum_{j=1}^N \mathbb{1}_{\Omega^j} \omega(z^j).$$

We consider the following semi-linear elliptic partial differential equation:

$$-\Delta y + f(y) = h \text{ in } \Omega, \quad \frac{\partial y}{\partial \boldsymbol{n}} + y = g \text{ on } \Gamma = \partial\Omega. \tag{A1}$$

A weak solution of (A1) is a function $y \in \mathcal{Y} = V \cap L^\infty(\Omega)$ such that

$$\int_\Omega \nabla y \cdot \nabla\varphi + f(y)\varphi \, \mathrm{d}\boldsymbol{x} + \int_\Gamma y\varphi \, \mathrm{d}\boldsymbol{s} = \int_\Omega h\varphi \, \mathrm{d}\boldsymbol{x} + \int_\Gamma g\varphi \, \mathrm{d}\boldsymbol{s} \quad \text{for all } \varphi \in V. \tag{A2}$$

The function f is supposed to be sufficiently smooth, bounded, and measurable for a fixed y; monotonically increasing; and satisfies $f \geq 0$. Moreover, $h \in H$ and $g \in L^r(\Gamma)$ with $s > \mathfrak{d} - 1$. Then, there exists a unique solution $y \in \mathcal{Y}$ of (A1), c.f. ([26], Section 4.2.3), for instance. This solution is even continuous, and for a constant c_∞ it holds that

$$\|y\|_V + \|y\|_{C(\overline{\Omega})} \leq c_\infty \left(\|h\|_H + \|g\|_{L^r(\Gamma)} \right).$$

The lumped mass finite element approximation of (A1) is to find $y^N \in V^N$ such that

$$\int_\Omega \nabla y^N \cdot \nabla \varphi + f(\mathcal{R}^N(y^N))\varphi^N \,\mathrm{d}\boldsymbol{x} + \int_\Gamma y^N \varphi \,\mathrm{d}\boldsymbol{s} = \int_\Omega h\varphi \,\mathrm{d}\boldsymbol{x} + \int_\Gamma g\varphi \,\mathrm{d}\boldsymbol{s} \tag{A3}$$

holds for all $\varphi \in V^N$ and for $\varphi^N = \mathcal{R}^N(\varphi)$.

Remark A1. *For any $v \in C(\overline{\Omega})$, it holds that $\mathcal{R}^N(v) = \sum_{j=1}^N \mathbb{1}_{\Omega^j} v(z^j)$. Therefore, $R^N(v) \to v$ for $n \to \infty$, which implies*

$$\|R^N(v) - v\|_H \to 0 \quad \textit{for } n \to \infty.$$

◊

Theorem A1. *Assume that $y \in V$ and $y^N \in V^N$ are solutions of (A2) and (A3), respectively. Then, it holds that*

$$\|y - y^N\|_V \to 0 \quad \textit{for } N \to \infty.$$

Proof. Because y solves (A2), it holds that $y \in C(\overline{\Omega})$. Let $y^p \in V^N$ be the interpolation polynomial of y in V^N. Then, it follows that $\mathcal{R}^N(y^p) = \mathcal{R}^N(y)$. Utilizing (A3), we derive

$$-\int_\Omega \nabla y^N \cdot \nabla y^p - \int_\Gamma y^N y^p \,\mathrm{d}\boldsymbol{s} = \int_\Omega f(\mathcal{R}^N(y^N))\mathcal{R}^N(y) - hy^p \,\mathrm{d}\boldsymbol{x} - \int_\Gamma g y^p \,\mathrm{d}\boldsymbol{s}. \tag{A4}$$

Let $\varepsilon > 0$ be an arbitrary tolerance. For N large enough, we have $\|y - y^p\|_V < \varepsilon$. Furthermore, we have $\|R^N(y) - y\|_H < \varepsilon$ and $\|R^N(y^p - y^N) - (y^p - y^N)\|_H < \varepsilon$. From (A4), we conclude

$$\begin{aligned}
\|y^N - y\|_V^2 &= \int_\Omega \left|\nabla(y^N - y)\right|^2 \mathrm{d}\boldsymbol{x} + \int_\Gamma |y^N - y|^2 \,\mathrm{d}\boldsymbol{s} \\
&= \int_\Omega \Big(\nabla(y^N - y) \cdot \nabla(y^p - y) + \nabla(y^N - y) \cdot \nabla y^N + \nabla(y - y^N) \cdot \nabla y^p\Big) \,\mathrm{d}\boldsymbol{x} \\
&\quad + \int_\Gamma \Big((y^N - y)(y^p - y) + (y^N - y)y^N + (y - y^N)y^p\Big) \,\mathrm{d}\boldsymbol{s} \\
&= \int_\Omega \Big(\nabla(y^N - y) \cdot \nabla(y^p - y) + \nabla(y^N - y) \cdot \nabla y^N + \nabla y \cdot \nabla y^p\Big) \,\mathrm{d}\boldsymbol{x} \\
&\quad + \int_\Gamma \Big((y^N - y)(y^p - y) + (y^N - y)y^N + yy^p\Big) \,\mathrm{d}\boldsymbol{s} \\
&\quad + \int_\Omega \Big(f(\mathcal{R}^N(y^N))\mathcal{R}^N(y) - hy^p\Big) \,\mathrm{d}\boldsymbol{x} - \int_\Gamma g y^p \,\mathrm{d}\boldsymbol{s}.
\end{aligned}$$

Utilizing $\|y - y^p\|_V < \varepsilon$, $\|R^N(y) - y\|_H < \varepsilon$, and $\|R^N(y^p - y^N) - (y^p - y^N)\|_H < \varepsilon$, it follows that

$$\begin{aligned}
\int_\Omega \Big(f(\mathcal{R}^N(y^N))\mathcal{R}^N(y)\Big) \,\mathrm{d}\boldsymbol{x} &= \langle f(\mathcal{R}^N(y^N)), \mathcal{R}^N(y)\rangle_H \\
&= \langle f(\mathcal{R}^N(y^N)) - f(\mathcal{R}^N(y)), \mathcal{R}^N(y) - \mathcal{R}^N(y^N)\rangle_H \\
&\quad + \langle f(\mathcal{R}^N(y^N)), \mathcal{R}^N(y^N)\rangle_H + \langle f(\mathcal{R}^N(y)), \mathcal{R}^N(y) - \mathcal{R}^N(y^N)\rangle_H \\
&\le \langle f(\mathcal{R}^N(y^N)), \mathcal{R}^N(y^N)\rangle_H + \langle f(\mathcal{R}^N(y)), \mathcal{R}^N(y) - \mathcal{R}^N(y^N)\rangle_H \\
&= \langle f(\mathcal{R}^N(y)), \mathcal{R}^N(y - y^N) - (y - y^N)\rangle_H + \langle f(y), y - y^N\rangle_H \\
&\quad + \langle f(\mathcal{R}^N(y)) - f(y), (y - y^N)\rangle_H + \langle f(\mathcal{R}^N(y^N)), \mathcal{R}^N(y^N)\rangle_H \\
&\le c\big(\|y - y^p\|_H + \|\mathcal{R}^N(y^p - y^N) - (y^p - y^N)\|_H\big) + \langle f(y), y - y^N\rangle_H \\
&\quad + c\,\|\mathcal{R}^N(y) - y\|_H \|y - y^N\|_H + \langle f(\mathcal{R}^N(y^N)), \mathcal{R}^N(y^N)\rangle_H \\
&\le c\Big(2\varepsilon + \varepsilon\,\|y - y^N\|_H\Big) + \langle f(y), y - y^N\rangle_H + \langle f(\mathcal{R}^N(y^N)), \mathcal{R}^N(y^N)\rangle_H.
\end{aligned}$$

Inserting this estimation into the equality before thus gives

$$\begin{aligned}\|y-y^N\|_V^2 &\leq \varepsilon\|y-y^N\|_V^2+\frac{\|y^p-y\|_V^2}{4\varepsilon}+\langle\nabla y^N,\nabla(y^N-y)\rangle_{H^\mathfrak{d}}+\langle\nabla y^p,\nabla y\rangle_{H^\mathfrak{d}}\\&\quad+\langle y^N-y,y^p-y\rangle_{L^2(\Gamma)}+\langle y^N,y^N-y\rangle_{L^2(\Gamma)}+\langle y^p,y\rangle_{L^2(\Gamma)}-\langle y^p,h\rangle_H\\&\quad-\langle y^p,g\rangle_{L^2(\Gamma)}+c\varepsilon+c\varepsilon\|y-y^N\|_H+\langle g,y-y^N\rangle_{L^2(\Gamma)}-\langle y,y-y^N\rangle_{L^2(\Gamma)}\\&\quad+\langle h,y-y^N\rangle_H-\langle\nabla y,\nabla(y-y^N)\rangle_{H^\mathfrak{d}}+\langle f(\mathcal{R}^N(y^N)),\mathcal{R}^N(y^N)\rangle_H\\&\leq c\varepsilon\|y-y^N\|_V^2+\frac{\varepsilon^2}{4\varepsilon}+c\varepsilon\langle\nabla y^N,\nabla y\rangle_{H^\mathfrak{d}}+\langle\nabla y^p,\nabla y\rangle_{H^\mathfrak{d}}\\&\quad+\langle g,y-y^p\rangle_{L^2(\Gamma)}+\langle h,y-y^p\rangle_H+\langle y^N,y^p-y\rangle_{L^2(\Gamma)}-\langle\nabla y,\nabla(y-y^N)\rangle_{H^\mathfrak{d}}\\&\leq c\varepsilon+c\varepsilon\|y-y^N\|_V^2\end{aligned}$$

with $H^\mathfrak{d}=\otimes_{i=1}^{\mathfrak{d}}H$. In the last inequality, we used the fact that $y^p\to y$ for $N\to\infty$. Therefore, we get

$$\|y-y^N\|_V^2\leq c_0\varepsilon$$

and it follows $\|y-y^N\|_V\to 0$ for $N\to\infty$. □

Remark A2. *For $y^N\in V^N$ and $j=1,\ldots,N$ we find*

$$\begin{aligned}\int_\Omega f(\mathcal{R}^N(y^N))\mathcal{R}^N(\varphi^j)\,\mathrm{d}\boldsymbol{x}&=\sum_{i=1}^N\int_{\Omega^i}f\Big(\sum_{l=1}^N y_l^N\mathbb{1}_{\Omega_l}\Big)\mathbb{1}_{\Omega^j}\,\mathrm{d}\boldsymbol{x}\\&=\int_{\Omega^j}f\Big(\sum_{l=1}^N y_l^N\mathbb{1}_{\Omega_l}\Big)\,\mathrm{d}\boldsymbol{x}=f(y_j^N)\int_{\Omega^j}\mathbb{1}_{\Omega_j}\,\mathrm{d}\boldsymbol{x}\\&=f(y_j^N)\sum_{\tau\in\mathcal{T}^N:z^j\in V(\tau)}\frac{|\tau|}{n+1}=\tilde{M}_{jj}f(y_j^N).\end{aligned}$$

For the last step we use

$$\tilde{M}_{jj}=\sum_{k=1}^N M_{jk}=\sum_{\tau\in\mathcal{T}^N:z^j\in V(\tau)}\int_\tau\varphi_j\,\mathrm{d}\boldsymbol{x}=\sum_{\tau\in\mathcal{T}^N:z^j\in V(\tau)}\frac{|\tau|}{n+1}.$$

A detailed proof for this equality can be found in ([36], Appendix A). ◊

References

1. Ehrgott, M. *Multicriteria Optimization*; Springer: Berlin/Heidelberg, Germany, 2005.
2. Iapichino, L.; Ulbrich, S.; Volkwein, S. Multiobjective PDE-constrained optimization using the reduced-basis method. *Adv. Comput. Math.* **2017**, *43*, 945–972. [CrossRef]
3. Miettinen, K. *Nonlinear Multiobjective Optimization*; Kluwer Academic Publishers: Dordrecht, The Netherlands, 1999.
4. Zadeh, L. Optimality and non-scalar-valued performance criteria. *IEEE Trans. Autom. Control* **1963**, *8*, 59–60. [CrossRef]
5. Deb, K. *Multi-Objective Optimization Using Evolutionary Algorithms*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2001.
6. Fliege, J.; Graña Drummond, L.M.; Svaiter, B.F. Netwon's Method for multiobjective optimization. *SIAM J. Optim.* **2008**, *20*, 602–626. [CrossRef]
7. Fliege, J.; Svaiter, B.F. Steepest descent methods for multicriteria optimization. *Math. Methods Oper. Res.* **2000**, *51*, 479–494. [CrossRef]
8. Schäffler, S.; Schultz, R.; Weinzierl, K. Stochastic method for the solution of unconstrained vector optimization problems. *J. Optim. Theory Appl.* **2002**, *114*, 209–222. [CrossRef]
9. Banholzer, S.; Gebken, B.; Dellnitz, M.; Peitz, S.; Volkwein, S. ROM-based multiobjective optimization of elliptic PDEs via numerical continuation. *arXiv* **2019**, arXiv:1906.09075v1.
10. Hillermeier, C. *Nonlinear Multiobjective Optimization: A Generalized Homotopy Approach*; Birkhäuser: Cambridge, MA, USA, 2001.

11. Schütze, O.; Dell'Aere, A.; Dellnitz, M. On continuation methods for the numerical treatment of multi-objective optimization problems. In *Practical Approaches to Multi-Objective Optimization*; Branke, J., Deb, K., Miettinen, K., Steuer, R.E., Eds.; Dagstuhl Seminar Proceedings: Dagstuhl, Deutschland, 2005.
12. Beermann, D.; Dellnitz, M.; Peitz, S.; Volkwein, S. Set-oriented multiobjective optimal control of PDEs using proper orthogonal decomposition. In *Reduced-Order Modeling (ROM) for Simulation and Optimization: Powerful Algorithms as Key Enablers for Scientific Computing*; Springer International Publishing: Cham, Switzerland, 2018; pp. 47–72.
13. Dellnitz, M.; Schütze, O.; Hestermeyer, T. Covering Pareto sets by multilevel subdivision techniques. *J. Optim. Theory Appl.* **2005**, *124*, 113–136. [CrossRef]
14. Jahn, J. Multiobjective search algorithm with subdivision technique. *Comput. Optim. Appl.* **2006**, *35*, 161–175. [CrossRef]
15. Schütze, O.; Witting, K.; Ober-Blöbaum, S.; Dellnitz, M. Set oriented methods for the numerical treatment of multiobjective optimization problems. In *EVOLVE—A Bridge between Probability, Set Oriented Numerics and Evolutionary Computation*; Tantar, E., Tantar, A.-A., Bouvry, P., Del Moral, P., Legrand, P., Coello Coello, C.A., Schütze, O., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; Volume 447, pp. 187–219.
16. Schilders, W.H.; Van der Vorst, H.A.; Rommes, J. *Model Order Reduction*; Springer: Berlin/Heidelberg, Germany, 2008.
17. Hesthaven, J.S.; Rozza, G.; Stamm, B. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*; SpringerBriefs in Mathematics: Heidelberg, Germany, 2016.
18. Patera, A.T.; Rozza, G. *Reduced Basis Approximation and A Posteriori Error Estimation for Parametrized Partial Differential Equations*. MIT Pappalardo Graduate Monographs in Mechanical Engineering: Cambridge, MA, USA, 2007.
19. Chaturantabut, S.; Sorensen, D.C. Nonlinear model reduction via discrete empirical interpolation. *SIAM J. Sci. Comput.* **2010**, *32*, 2737–2764. [CrossRef]
20. Chaturantabut, S.; Sorensen, D.C. A state space estimate for POD-DEIM nonlinear model reduction. *SIAM J. Numer. Anal.* **2012**, *50*, 46–63. [CrossRef]
21. Gebken, B.; Peitz, S.; Dellnitz, M. A descent method for equality and inequality constrained multiobjective optimization problems. In *Numerical and Evolutionary Optimization—NEO 2017*; Trujillo, L., Schütze, O., Maldonado, Y., Valle, P., Eds.; Springer: Cham, Switzerland, 2017.
22. Graña Drummond, L.M.; Svaiter, B.F. A steepest descent method for vector optimization. *J. Comput. Appl. Math.* **2005**, *175*, 395–414. [CrossRef]
23. Reichle, L. Set-Oriented Multiobjective Optimal Control of Elliptic Non-Linear Partial Differential Equations Using POD Objectives and Gradient. Master's Thesis, University of Konstanz, Konstanz, Germany, 2020. Available online: http://nbn-resolving.de/urn:nbn:de:bsz:352-2-1h6pp1cxptbap6 (accessed on 15 April 2021).
24. Dellnitz, M.; Hohmann, A. A subdivision algorithm for the computation of unstable manifolds and global attractors. *Numer. Math.* **1997**, *75*, 293–316. [CrossRef]
25. Dautray, R.; Lions, J.-L. *Mathematical Analysis and Numerical Methods for Science and Technology. Volume 2: Functional and Variational Methods*; Springer-Verlag: Berlin/Heidelberg, Germany, 2000.
26. Tröltzsch, F. *Optimal Control of Partial Differential Equations: Theory, Methods and Applications*; American Mathematical Society: Providence, RI, USA, 2010; Volume 112.
27. Thomée, V. *Galerkin Finite Element Methods for Parabolic Problems*; Springer: Berlin/Heidelberg, Germany, 2006.
28. Zienkiewicz, O.C.; Taylo, R.L. *The Finite Element Method*, 5th ed.; Butterworth-Hienemann: Oxford, UK, 2000; Volume 3.
29. Brenner, S.; Scott, R. *The Mathematical Theory of Finite Element Methods*; Springer: Berlin/Heidelberg, Germany, 2008.
30. Rösch, A.; Wachsmuth, G. Mass lumping for the optimal control of elliptic partial differential equations. *SIAM J. Numer. Anal.* **2017**, *55*, 1412–1436. [CrossRef]
31. Quarteroni, A.; Manzoni, A.; Negri, F. *Reduced Basis Methods for Partial Differential Equations*; Springer: Berlin/Heidelberg, Germany, 2016.
32. Grepl, M.A.; Maday, Y.; Nguyen, N.C.; Patera, A.T. Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations. *ESAIM Math. Model. Numer. Anal.* **2007**, *41*, 575–605. [CrossRef]
33. Hinze, M.; Korolev, D. Reduced basis methods for quasilinear elliptic PDEs with applications to permanent magnet synchronous motors. *arXiv* **2020**, arXiv:2002.04288.
34. Rogg, S.; Trenz, S.; Volkwein, S. Trust-region POD using a-posteriori error estimation for semilinear parabolic optimal control problems. *Konstanz. Schriften Math.* **2017**, *359*. Available online: https://kops.uni-konstanz.de/handle/123456789/38240 (accessed on 15 April 2021).
35. Zeng, J.; Yu, H. Error estimates of the lumped mass finite element method for semilinear elliptic problems. *J. Comput. Appl. Math.* **2012**, *236*, 993–2004. [CrossRef]
36. Bernreuther, M. RB-Based PDE-Constrained Non-Smooth Optimization. Master's Thesis, University of Konstanz, Konstanz, Germany, 2019. Available online: http://nbn-resolving.de/urn:nbn:de:bsz:352-2-t4k1djyj77yn3 (accessed on 15 April 2021).