# The role of PDE-based parameterization techniques in gradient-based IGA shape optimization applications

Hinz, Jochen; Jaeschke, Andrzej; Möller, Matthias; Vuik, Cornelis

**DOI**
10.1016/j.cma.2021.113685

**Publication date**
2021

**Document Version**
Final published version

**Published in**
Computer Methods in Applied Mechanics and Engineering

Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 378 (2021) 113685
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

# The role of PDE-based parameterization techniques in gradient-based IGA shape optimization applications

Jochen Hinz[a,*], Andrzej Jaeschke[b], Matthias Möller[c], Cornelis Vuik[c]

[a] *Institute of Mathematics, Ecole Polytechnique Fédérale de Lausanne, EPFL SB MATH, Station 8, CH-1015, Lausanne, Switzerland*
[b] *Institute of Turbomachinery, Lodz University of Technology, Wolczanska 219/223, 90-924 Lodz, Poland*
[c] *Delft Institute of Applied Mathematics, Delft University of Technology, 2628 XE Delft, The Netherlands*

Received 6 March 2020; received in revised form 13 January 2021; accepted 15 January 2021
Available online xxxx

**Abstract**

This paper proposes a shape optimization algorithm based on the principles of *Isogeometric Analysis* (IGA) in which the parameterization of the geometry enters the problem formulation as an additional PDE-constraint. Inspired by the *isoparametric principle* of IGA, the parameterization and the governing state equation are treated using the same numerical technique. This leads to a scheme that is comparatively easy to differentiate, allowing for a fully symbolic derivation of the gradient and subsequent gradient-based optimization. To improve the efficiency and robustness of the scheme, the basis is re-selected during each optimization iteration and adjusted to the current needs. The scheme is validated in two test cases.


*Keywords:* Isogeometric Analysis; Shape optimization; Elliptic Grid Generation; Parameterization techniques; Adjoint-based optimization

## 1. Introduction

Isogeometric analysis (IGA) was introduced by Hughes et al. in [1] as a numerical technique that bridges the gap between Computer Aided Design (CAD) and the numerical analysis of Partial Differential Equations (PDEs). This is accomplished by using the same function space to represent the geometry $\Omega$ and to discretize the PDE problem posed over $\Omega$. Since its birth in 2005, IGA has been successfully applied to wide variety of problems including: thermal analysis [2], linear elasticity problems [1], structural vibrations [3], incompressible flows [4] and inviscid compressible flows [5].

As a mature numerical method, it is ready to be used in more complex industrial processes. Consequently, several publications that consider the application of IGA to shape optimization problems in structural and fluid dynamics [6–11] as well as heat conduction [12] have appeared in the literature.

Most of the available CAD software generates no more than a spline-based description of the boundary contours $\partial\Omega$ of $\Omega$. Clearly, in the absence of numerical and modeling errors, the value of the shape optimization cost function is solely determined by the shape of the domain, i.e., $\partial\Omega$. When only the value of the state equation solution on

* Corresponding author.
*E-mail addresses:* jochen.hinz@epfl.ch (J. Hinz), andrzej.jaeschke@p.lodz.pl (A. Jaeschke), m.moller@tudelft.nl (M. Möller), C.Vuik@tudelft.nl (C. Vuik).

https://doi.org/10.1016/j.cma.2021.113685
0045-7825/

the boundary is relevant for the optimization, an approach based on the boundary element method (BEM) [13] constitutes a viable choice. Not only does BEM avoid computing the state equation solution in regions where it is not relevant for the optimization, it also avoids the potentially expensive and computationally delicate surface to volume problem $\partial\Omega \rightarrow \Omega$. As an example, design optimization using a T-spline BEM-isogeometric solver was proposed by Kostas et al. in [14]. As a downside of BEM, the Green's function of the underlying PDE-problem needs to be known, which then leads to an associated boundary integral equation.

Hence, in order to perform shape optimization, many practical applications still require the parameterization of interior $\Omega$ during each shape optimization iteration as well. Combining IGA and shape optimization is very appealing as the CAD definition of $\partial\Omega$ can be used directly to compute a mapping for $\Omega$, completely bypassing the need to first convert $\partial\Omega$ into a piecewise-linear curve that acts as an input for classical mesh generators. On the other hand, suitable parameterization algorithms are indispensable for generating bijective (folding-free), analysis-suitable geometry parameterizations from the boundary CAD data. Furthermore, the parametric quality of the mapping has a profound impact on the accuracy of the isogeometric analysis [15]. Therefore, besides bijectivity, practical algorithms aim at generating parameterizations of high numerical quality.

A variety of parameterization techniques have been proposed in the literature such as Coon's Patch [16], Linear Spring [17] and approaches based on (constrained and unconstrained) quality cost function optimization [17–19]. While mappings based on Coon's Patch and Linear Spring follow from a closed-form expression and are hence cheap to compute and straightforwardly differentiable, they often lead to folded (non-bijective) mappings. The same is true for unconstrained optimization. Constrained optimization approaches on the other hand typically have a higher success rate. However, this comes at the expense of a large number of (constrained) iterations (typically about $\sim$30) and the notorieties associated with nonconvex optimization, such as the danger of getting stuck in local minima. A third class of approaches attempts to generate a mapping whose inverse is composed of harmonic functions in $\Omega$. This approach is based on the observation that harmonic functions exhibit a large degree of smoothness, which benefits the numerical quality of the resulting mapping. Furthermore, it can be shown that if the parametric domain is convex, inversely harmonic mappings (IHMs) are bijective, thanks to the maximum principle [20,21]. Many approaches for approximating IHMs have been proposed in the literature [19,22], notably the PDE-based approach called Elliptic Grid Generation (EGG) [23,24]. EGG is of particular interest in shape optimization problems thanks to the parametric smoothness and bijectivity of IHMs, which is beneficial for the robustness of the parameterization pipeline.

In general, there are two main groups of shape optimization algorithms: gradient-free (like for example genetic algorithms [25]) and gradient-based methods (for example interior point methods [26,27]). The latter group generally requires fewer underlying PDE evaluations at the expense of having to compute the gradient of the objective function during each iteration. Therefore, IGA parameterization algorithms that are differentiable with respect to the design variables are desirable. Hence, the implicit differentiability, made possible by the PDE-based problem formulation, is yet another appealing feature of employing an EGG-based parameterization pipeline. An additional feature of differentiability is efficiency: as the inner control points are a smooth function of the boundary control points, there is no need for full remeshing after each iteration since cheaper mesh update strategies can be employed. This is also true for settings in which the boundary contours change as a smooth function of time.

Traditionally, IGA parameterizations are taken from tensor-product spline spaces. Unfortunately, structured spline technologies do not allow for local refinement, which may result in infeasibly-large function spaces. Therefore, unstructured spline technologies such as THB-splines [28] are gaining an increased amount of interest in the IGA community, thanks to local refinement. An EGG-based planar parameterization framework that supports THB-splines has been proposed in [29].

In order to combine the appealing features of EGG and THB-enabled local refinement, this paper adopts the parameterization framework proposed in [29] and presents an IGA-based shape optimization algorithm in which the parameterization is added to the optimization problem formulation in the form of an additional PDE-constraint. In line with the *isoparametric principle* of IGA, we numerically treat this additional constraint in the same way as the governing quantity (temperature, pressure, etc.) of the underlying optimization problem. Including the mapping explicitly as a PDE-constraint facilitates differentiation, allowing for gradient-based optimization, while also guaranteeing analysis-suitability, thanks to the bijectivity of IHMs. To improve the efficiency, the proposed algorithm employs THB-enabled adaptive local refinement strategies during every optimization iteration, resulting in a variable discretization basis. We validate the proposed methodology by presenting two test cases.

## 2. Notation

In this work, we denote vectors in boldface while matrices receive a capital letter and may furthermore be enclosed in square brackets for better readability. The $i$th entry of vector $\mathbf{x}$ is denoted by $\mathbf{x}_i$ or simply $x_i$ and similarly for the $ij$th entry of matrices. We make extensive use of vector derivatives. Here, we interchangeably use the denotation

$$[\partial_{\mathbf{t}}\mathbf{x}] \equiv \left[\frac{\partial \mathbf{x}}{\partial \mathbf{t}}\right], \quad \text{with} \quad \left[\frac{\partial \mathbf{x}}{\partial \mathbf{t}}\right]_{ij} = \frac{\partial x_i}{\partial t_j} \tag{1}$$

for the partial derivative and similarly for the total derivative. In the case of taking the derivative of a scalar, brackets are avoided. However, the argument is treated as a $1 \times 1$ matrix and hence the derivative has dimension $(1, m)$, where $m$ is the dimension of $\mathbf{t}$. When integrating locally defined quantities $u(\boldsymbol{\xi}) : \hat{\Omega} \to \mathbb{R}^n$ over the physical domain $\Omega$, we avoid mentioning the push-forward with the mapping $\mathbf{x} : \hat{\Omega} \to \Omega$ for convenience, i.e.,

$$\int_{\Omega} u \circ \mathbf{x}^{-1} \mathrm{d}S = \int_{\hat{\Omega}} u(\boldsymbol{\xi}) \det\left[\partial_{\boldsymbol{\xi}}\mathbf{x}\right] \mathrm{d}\boldsymbol{\xi} \longrightarrow \int_{\Omega} u(\mathbf{x}) \mathrm{d}S, \tag{2}$$

and assume that the reader is aware of the mathematical subtleties involved.

## 3. Problem formulation

We are considering the shape optimization problem of a planar domain $\Omega(\boldsymbol{\alpha})$ whose contours $\partial\Omega(\boldsymbol{\alpha})$ are parameterized by the $n$-tuple of design variables $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$. If the design variables are taken from the design space $\boldsymbol{\lambda}$, the optimization problem reads:

$$\begin{aligned} & J\left(u^{\boldsymbol{\alpha}}, \Omega^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}\right) \to \min_{\boldsymbol{\alpha}} \\ \text{s.t.} \quad & g_i(u^{\boldsymbol{\alpha}}, \Omega^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) \geq 0, \quad \forall i \in \{1, \ldots, N_{\neq}\} \\ & h_j(u^{\boldsymbol{\alpha}}, \Omega^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) = 0, \quad \forall j \in \{1, \ldots, N_{=}\} \\ & \boldsymbol{\alpha} \in \boldsymbol{\lambda}, \end{aligned} \tag{3}$$

where the $g_i$ and $h_j$ are problem-specific constraints. Here, $J(\cdot, \cdot, \cdot)$ denotes the objective function and $u^{\boldsymbol{\alpha}} : \Omega \to \mathbb{R}$ some state variable whose physical meaning depends on the application (temperature, pressure, etc.). We regard $u^{\boldsymbol{\alpha}}$ as a scalar quantity for convenience. However, generalizations to vectorial quantities are straightforward. Note that the dependencies of the variables contained in $J(\cdot, \cdot, \cdot)$ are concatenated in descending order, i.e., in general $u^{\boldsymbol{\alpha}} = u^{\boldsymbol{\alpha}}(\Omega^{\boldsymbol{\alpha}}(\boldsymbol{\alpha}), \boldsymbol{\alpha})$ and $\Omega^{\boldsymbol{\alpha}} = \Omega^{\boldsymbol{\alpha}}(\boldsymbol{\alpha})$. The state variable $u^{\boldsymbol{\alpha}}$ follows from a PDE-problem posed over $\Omega^{\boldsymbol{\alpha}}$ and may contain additional dependencies on $\boldsymbol{\alpha}$ (such as source terms), hence the dependency on the tuple $(\Omega^{\boldsymbol{\alpha}}, \boldsymbol{\alpha})$. Tackling (3) computationally requires introducing a bijective geometry parameterization $\mathbf{x}^{\boldsymbol{\alpha}} : \hat{\Omega} \to \Omega^{\boldsymbol{\alpha}}$, where $\hat{\Omega}$ denotes the parametric domain which is assumed to be static. Here, we restrict ourselves to geometries that are topologically equivalent to $\hat{\Omega} = (0, 1)^2$ for convenience. However, the generalization to multipatch settings is straightforward. Let

$$\mathcal{U}^f = \{v \in \mathcal{U} \mid v = f \text{ on } \partial\Omega_D^{\boldsymbol{\alpha}}\} \tag{4}$$

for some suitably-chosen vector space $\mathcal{U}$ and some $\partial\Omega_D^{\boldsymbol{\alpha}} \subseteq \partial\Omega^{\boldsymbol{\alpha}}$ on which Dirichlet data is prescribed. Deriving the weak form of the PDE-problem governing $u^{\boldsymbol{\alpha}}$ leads to

$$\text{find } u^{\boldsymbol{\alpha}} \in \mathcal{U}^{u_D^{\boldsymbol{\alpha}}} \quad \text{s.t.} \quad B\left(u^{\boldsymbol{\alpha}}, \mathbf{x}^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}, \phi\right) = 0, \quad \forall \phi \in \mathcal{U}^0, \tag{5}$$

for some differential form $B(\cdot, \cdot, \cdot, \cdot)$. Here, $u_D^{\boldsymbol{\alpha}}$ denotes the Dirichlet data as a function of the design variables. By introducing the mapping $\mathbf{x}^{\boldsymbol{\alpha}}$, the objective function takes the form

$$J(u^{\boldsymbol{\alpha}}, \Omega^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) \to J\left(u^{\boldsymbol{\alpha}}, \mathbf{x}^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}\right) \equiv J^{\boldsymbol{\alpha}}, \tag{6}$$

where $u^{\boldsymbol{\alpha}}$ satisfies (5). With the dependencies of $u^{\boldsymbol{\alpha}}$ and $\mathbf{x}^{\boldsymbol{\alpha}}$ in mind, the gradient of (6) reads:

$$\frac{\mathrm{d}J^{\boldsymbol{\alpha}}}{\mathrm{d}\boldsymbol{\alpha}} = \frac{\partial J^{\boldsymbol{\alpha}}}{\partial u^{\boldsymbol{\alpha}}}\left(\frac{\partial u^{\boldsymbol{\alpha}}}{\partial \mathbf{x}^{\boldsymbol{\alpha}}}\frac{\mathrm{d}\mathbf{x}^{\boldsymbol{\alpha}}}{\mathrm{d}\boldsymbol{\alpha}} + \frac{\partial u^{\boldsymbol{\alpha}}}{\partial \boldsymbol{\alpha}}\right) + \frac{\partial J^{\boldsymbol{\alpha}}}{\partial \mathbf{x}^{\boldsymbol{\alpha}}}\frac{\mathrm{d}\mathbf{x}^{\boldsymbol{\alpha}}}{\mathrm{d}\boldsymbol{\alpha}} + \frac{\partial J^{\boldsymbol{\alpha}}}{\partial \boldsymbol{\alpha}}. \tag{7}$$

We see that (7) requires taking the derivative of $\mathbf{x}^{\boldsymbol{\alpha}}$ with respect to $\boldsymbol{\alpha}$, while the state variable $u^{\boldsymbol{\alpha}}$ needs to be differentiable with respect to $\mathbf{x}^{\boldsymbol{\alpha}}$. These two derivatives often constitute the most challenging step in computing

the gradient because differentiating $\mathbf{x}^{\boldsymbol{\alpha}}$ or with respect to $\mathbf{x}^{\boldsymbol{\alpha}}$ can be nontrivial, depending on the parameterization technique used. On the other hand, differentiation with respect to $u^{\boldsymbol{\alpha}}$ is relatively straightforward because the implicit function theorem can be used on (5). Hence, if we take $\mathbf{x}^{\boldsymbol{\alpha}}$ as the solution of a PDE problem, differentiation is simplified, allowing for a symbolic derivation of all terms involved in (7). To this end, we adopt the principles of *Elliptic Grid Generation*, which will be the topic of the next section.

## 4. Elliptic grid generation

Elliptic grid generation (EGG) is a PDE-based technique aimed at generating analysis-suitable geometry parameterizations $\mathbf{x}^{\boldsymbol{\alpha}} : \hat{\Omega} \rightarrow \Omega^{\boldsymbol{\alpha}}$ given only a parametric description of the boundary contours $\partial\Omega^{\boldsymbol{\alpha}}$ as a function of the state vector $\boldsymbol{\alpha}$. Let the free topological variables in $\hat{\Omega}$ be given by the tuple $\boldsymbol{\xi} = (\xi_1, \xi_2)^T = (\xi, \eta)^T$. Then, the equations of EGG read [29]:

$$A(\mathbf{x}^{\boldsymbol{\alpha}}) \colon H(\mathbf{x}_i^{\boldsymbol{\alpha}}) = 0 \quad \text{in } \hat{\Omega}, \quad \text{for } i \in \{1, 2\} \quad \text{s.t.} \quad \mathbf{x}^{\boldsymbol{\alpha}}|_{\partial\hat{\Omega}} = \partial\Omega^{\boldsymbol{\alpha}}, \tag{8}$$

where

$$H(u)_{ij} \equiv \frac{\partial^2 u}{\partial\xi_i \partial\xi_j} \quad \text{and} \quad A(\mathbf{x}^{\boldsymbol{\alpha}}) = \frac{1}{g_{11} + g_{22} + \epsilon} \begin{pmatrix} g_{22} & -g_{12} \\ -g_{12} & g_{11} \end{pmatrix}, \tag{9}$$

with $g_{ij} = \mathbf{x}^{\boldsymbol{\alpha}}_{\xi_i} \cdot \mathbf{x}^{\boldsymbol{\alpha}}_{\xi_j}$ the entries of the metric tensor and $\epsilon$ a small positive constant (typically, we take $\epsilon = 10^{-4}$). Here, $A \colon B$ denotes the Frobenius inner product between matrices $A$ and $B$. The solution of (8) is a mapping $\mathbf{x}^{\boldsymbol{\alpha}}$ whose inverse $(\mathbf{x}^{\boldsymbol{\alpha}})^{-1}$ constitutes a pair of harmonic functions on $\Omega^{\boldsymbol{\alpha}}$. As $(\mathbf{x}^{\boldsymbol{\alpha}})^{-1}$ maps into a convex parametric domain $\hat{\Omega}$, it follows from the maximum principle that $\mathbf{x}^{\boldsymbol{\alpha}}$ is a bijection between $\hat{\Omega}$ and $\Omega^{\boldsymbol{\alpha}}$, where $\mathbf{x}^{\boldsymbol{\alpha}}|_{\partial\hat{\Omega}}$ parameterizes $\partial\Omega^{\boldsymbol{\alpha}}$ [20,21]. This property justifies limiting the choice of $\mathbf{x}^{\boldsymbol{\alpha}}$ from the set of bijective parameterizations to the subset of mappings that satisfy (8).

Various alternative approaches for computing an inversely harmonic map have appeared in the literature, such as [22]. In particular, here we mention the approach based on an isogeometric boundary element method proposed in [19], which is further developed in [30] to support geometries of genus $> 0$ via a generalization of the Radó-Kneser–Choquet theorem [31,32], which forms the theoretical foundation of harmonic maps, to multiply-connected domains. While approaches based on BEM constitute a viable alternative to a PDE-based approach, in particular thanks to the support for multiply-connected domains, we base a computational approach on (8) since it allows for implicit differentiation of $\mathbf{x}^{\boldsymbol{\alpha}}$ with respect to the design variables (see (7)).

For a viable computational approach, we derive the weak counterpart of (8). Here, we adopt the approach from [24]. Given a differential function $\mathbf{x}_D^{\boldsymbol{\alpha}} : \partial\hat{\Omega} \rightarrow \mathbb{R}^2$ that parameterizes $\partial\Omega^{\boldsymbol{\alpha}}$, the mapping $\mathbf{x}^{\boldsymbol{\alpha}}$ is the solution of:

$$\text{find } \mathbf{x}^{\boldsymbol{\alpha}} \in \boldsymbol{\mathcal{V}}^{\mathbf{x}_D^{\boldsymbol{\alpha}}} \quad \text{s.t.} \quad F(\mathbf{x}^{\boldsymbol{\alpha}}, \boldsymbol{\sigma}) = 0, \quad \forall \boldsymbol{\sigma} \in \boldsymbol{\mathcal{V}}^{\mathbf{0}}, \tag{10}$$

with

$$F(\mathbf{x}^{\boldsymbol{\alpha}}, \boldsymbol{\sigma}) = \sum_{i=1}^{2} \int_{\hat{\Omega}} \boldsymbol{\sigma}_i A(\mathbf{x}^{\boldsymbol{\alpha}}) \colon H(\mathbf{x}_i^{\boldsymbol{\alpha}}) \mathrm{d}\boldsymbol{\xi}. \tag{11}$$

In (10) we used

$$\boldsymbol{\mathcal{V}}^{\mathbf{f}} \equiv \{\mathbf{v} \in V \times V \mid \mathbf{v} = \mathbf{f} \text{ on } \partial\hat{\Omega}\}, \tag{12}$$

for some suitably-chosen vector space $V$.

The discretization of (10) follows straightforwardly from replacing $V$ by the finite-dimensional $V_h^{\boldsymbol{\alpha}} \subset V$ in (12) (and similarly for $\mathcal{U}$). We denote the resulting set by $\boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha},\mathbf{f}}$. As $\mathbf{f}$ needs to be compatible with the finite-dimensional space $V_h^{\boldsymbol{\alpha}}$, the discretization may additionally require replacing the Dirichlet data $\mathbf{x}_D^{\boldsymbol{\alpha}}$ by a proper collocation $\mathbf{x}_{D,h}^{\boldsymbol{\alpha}} \in \boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha}} \setminus \boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha},\mathbf{0}}$. As such, the discretized mapping operator $\mathbf{x}_h^{\boldsymbol{\alpha}} \in \boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha},\mathbf{x}_{D,h}^{\boldsymbol{\alpha}}}$ satisfies

$$\forall \boldsymbol{\sigma}_h \in \boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha},\mathbf{0}} : \quad F(\mathbf{x}_h^{\boldsymbol{\alpha}}, \boldsymbol{\sigma}_h) = 0. \tag{13}$$

**Remark.** Due to the appearance of second order derivatives in (13), we have to assume that $\mathbf{x}_h^{\boldsymbol{\alpha}}$ is taken from a spline space with global $C^1(\hat{\Omega})$-continuity. For an approach that allows for lower regularity (and is hence compatible with simply-connected, convex multipatch domains), we refer to [33].

Since (13) results in a nonlinear root-finding problem, we tackle it with a Newton-based iterative approach. Unlike $\mathbf{x}^{\boldsymbol{\alpha}}$, its discretized counterpart $\mathbf{x}_h^{\boldsymbol{\alpha}}$ may fold due to the truncation error introduced by the numerical scheme. Grid folding can be repaired by refining $V_h^{\boldsymbol{\alpha}}$ in the affected regions and recomputing $\mathbf{x}_h^{\boldsymbol{\alpha}}$ from the enriched space. This makes using an unstructured spline technology like THB-splines particularly appealing, thanks to local refinement. For more details on the choice of $V_h^{\boldsymbol{\alpha}}$ and the algorithm that tackles the nonlinear root-finding problem (13), we refer to [29].

Optionally, we may introduce $\mathbf{U}^{\boldsymbol{\alpha}}$ as the concatenation of the state variable and the corresponding mapping operator as a function of $\boldsymbol{\alpha}$. Treating both quantities using the same numerical technique then allows for a compact reformulation of the shape optimization problem (3) that reads as follows:

$$\begin{aligned} & J\left(\mathbf{U}^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}\right) \rightarrow \min_{\boldsymbol{\alpha}} \\ \text{s.t.} \quad & g_i(\mathbf{U}^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) \geq 0, \quad \forall i \in \{1, \ldots, N_{\neq}\} \\ & h_j(\mathbf{U}^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) = 0, \quad \forall j \in \{1, \ldots, N_{=}\} \\ & \boldsymbol{\alpha} \in \boldsymbol{\lambda}, \end{aligned} \tag{14}$$

where $\mathbf{U}^{\boldsymbol{\alpha}} = (u^{\boldsymbol{\alpha}}, \mathbf{x}^{\boldsymbol{\alpha}}) \in \mathcal{U}^{u_D^{\boldsymbol{\alpha}}} \times \boldsymbol{\mathcal{V}}^{\mathbf{x}_D^{\boldsymbol{\alpha}}}$ satisfies

$$\forall \boldsymbol{\Phi} \in \mathcal{U}^0 \times \boldsymbol{\mathcal{V}}^{\mathbf{0}} : \quad G(\mathbf{U}^{\boldsymbol{\alpha}}, \boldsymbol{\Phi}, \boldsymbol{\alpha}) = 0, \tag{15}$$

with $\boldsymbol{\Phi} = (\phi, \boldsymbol{\sigma})$ and

$$G(\mathbf{U}^{\boldsymbol{\alpha}}, \boldsymbol{\Phi}, \boldsymbol{\alpha}) = B(u^{\boldsymbol{\alpha}}, \mathbf{x}^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}, \phi) + F(\mathbf{x}^{\boldsymbol{\alpha}}, \boldsymbol{\sigma}). \tag{16}$$

Here, $B(\cdot, \cdot, \cdot, \cdot)$ and $F(\cdot, \cdot)$ are taken as defined in (5) and (11), respectively. As before, a discretization of (14)–(16) follows by replacing $\boldsymbol{\mathcal{V}}$ and $\mathcal{U}$ by their suitably-chosen finite-dimensional counterparts.

## 5. Computational approach

In this section we propose a computational approach for numerically treating the optimization problem (3).

### 5.1. Discretization

We discretize the optimization problem (3) by approximating

$$J(u^{\boldsymbol{\alpha}}, \mathbf{x}^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) \simeq J(u_h^{\boldsymbol{\alpha}}, \mathbf{x}_h^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) \equiv J_h^{\boldsymbol{\alpha}}, \tag{17}$$

where $u_h^{\boldsymbol{\alpha}} \in \mathcal{U}_h^{\boldsymbol{\alpha}}$ is the solution of the discretized weak state equation (5) while $\mathbf{x}_h^{\boldsymbol{\alpha}} \in \boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha}}$ is the solution of (13) for given $\boldsymbol{\alpha}$. Here, $\Omega_h^{\boldsymbol{\alpha}}$ is parameterized by $\mathbf{x}_h^{\boldsymbol{\alpha}}$ and approximates the domain $\Omega^{\boldsymbol{\alpha}}$ whose contours are parameterized by the $\boldsymbol{\alpha}$-differentiable $\mathbf{x}_D^{\boldsymbol{\alpha}} : \partial\hat{\Omega} \rightarrow \mathbb{R}^2$ which we consider a given function. The distance

$$D(\partial\Omega_h^{\boldsymbol{\alpha}}, \partial\Omega^{\boldsymbol{\alpha}}) \equiv \left\| \mathbf{x}_{D,h}^{\boldsymbol{\alpha}} - \mathbf{x}_D^{\boldsymbol{\alpha}} \right\|_{L_2(\partial\hat{\Omega})} \tag{18}$$

serves as a measure of the approximation quality.

Likewise, we approximate the gradient by replacing $(u^{\boldsymbol{\alpha}}, \mathbf{x}^{\boldsymbol{\alpha}}) \rightarrow (u_h^{\boldsymbol{\alpha}}, \mathbf{x}_h^{\boldsymbol{\alpha}})$ in (7), i.e.,

$$\frac{\mathrm{d}J}{\mathrm{d}\boldsymbol{\alpha}} \simeq \frac{\partial J_h^{\boldsymbol{\alpha}}}{\partial u_h^{\boldsymbol{\alpha}}} \left( \frac{\partial u_h^{\boldsymbol{\alpha}}}{\partial \mathbf{x}_h^{\boldsymbol{\alpha}}} \frac{\mathrm{d}\mathbf{x}_h^{\boldsymbol{\alpha}}}{\mathrm{d}\boldsymbol{\alpha}} + \frac{\partial u_h^{\boldsymbol{\alpha}}}{\partial \boldsymbol{\alpha}} \right) + \frac{\partial J_h^{\boldsymbol{\alpha}}}{\partial \mathbf{x}_h^{\boldsymbol{\alpha}}} \frac{\mathrm{d}\mathbf{x}_h^{\boldsymbol{\alpha}}}{\mathrm{d}\boldsymbol{\alpha}} + \frac{\partial J_h^{\boldsymbol{\alpha}}}{\partial \boldsymbol{\alpha}}. \tag{19}$$

At this point, it should be noted that for given $\boldsymbol{\alpha}$, the exact evaluations of $J(\cdot, \cdot, \cdot)$ and the components of its gradient are independent of the particular choice of the coordinate system $\mathbf{x}^{\boldsymbol{\alpha}}$. As such, the quality of the approximations introduced in (17) and (19) depend solely on the numerical accuracy of $u_h^{\boldsymbol{\alpha}}$, which in turn is affected by the parametric quality of $\mathbf{x}_h^{\boldsymbol{\alpha}}$ and the distance of $\partial\Omega_h^{\boldsymbol{\alpha}}$ to the exact $\partial\Omega^{\boldsymbol{\alpha}}$.

We numerically treat (3) based on a *variable basis approach* (VBA) rather than a *static basis approach* (SBA). In SBA, $u_h^{\boldsymbol{\alpha}}$ and $\mathbf{x}_h^{\boldsymbol{\alpha}}$ are constructed from the static tuple $(\mathcal{U}_h, \boldsymbol{\mathcal{V}}_h)$, while in VBA the tuple $(\mathcal{U}_h^{\boldsymbol{\alpha}}, \boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha}})$ may be chosen differently during each iteration and is tuned to the current needs. We make a choice based on the following principles:

A. $\left\| \mathbf{x}_{D,h}^{\boldsymbol{\alpha}} - \mathbf{x}_D^{\boldsymbol{\alpha}} \right\|_{L_2(\partial\hat{\Omega})}$, with $\mathbf{x}_{D,h}^{\boldsymbol{\alpha}} \in \boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha}} \setminus \boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha},\mathbf{0}}$ is sufficiently small;

B. $\mathbf{x}_h^{\boldsymbol{\alpha}} \in \boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha}}$, resulting from $\mathbf{x}_{D,h}^{\boldsymbol{\alpha}}$ in combination with (13), is a bijection and preferably of high numerical quality;
C. $u_h^{\boldsymbol{\alpha}} \in \mathcal{U}_h^{\boldsymbol{\alpha}}$ approximates $u^{\boldsymbol{\alpha}}$ well.

As such, for given $\boldsymbol{\alpha}$, we select the tuple $(\mathcal{U}_h^{\boldsymbol{\alpha}}, \boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha}})$ such that points A to C are satisfied with a minimal number of degrees of freedom (DOFs).

In SBA, a necessary condition for local optimality follows straightforwardly from the discretized counterpart of (3) over the static tuple $(\mathcal{U}_h, \boldsymbol{\mathcal{V}}_h)$. In contrast, VBA necessitates basing such a condition on (3) *before* discretization. Hence, numerical assessment of local optimality in (3) is obligatory, due to the approximate nature of $J_h^{\boldsymbol{\alpha}}$ and its gradient. This may be regarded as a drawback since it can generate false positives/negatives caused by the truncation error at the current iterate. On the other hand, VBA allows for $\boldsymbol{\alpha}$-specific feature-based basis selection for approximating both $\mathbf{x}^{\boldsymbol{\alpha}}$ and $u^{\boldsymbol{\alpha}}$, leading to a highly flexible scheme. When performing shape optimization in combination with EGG, a static $\boldsymbol{\mathcal{V}}_h$ may be inappropriate for particular choices of $\boldsymbol{\alpha}$ which results in grid-folding (impeding the evaluation of $J_h^{\boldsymbol{\alpha}}$), hence justifying VBA-enabled feature-based basis selection in applications which are geometrically complex.

**Remark.** If we regard the truncation error $\tau(\boldsymbol{\alpha})$ in $u^{\boldsymbol{\alpha}} = u_h^{\boldsymbol{\alpha}} + \tau(\boldsymbol{\alpha})$ as a random variable drawn from some probability distribution, above methodology possesses many properties reminiscent of stochastic gradient descent [34]. As such, the convergence tolerance should be designed with the expected magnitude of $\tau(\boldsymbol{\alpha})$ (and its contribution to the gradient) in mind and hence taken generously. Here, we regard this as a minor shortcoming since we consider complex and highly nonconvex, nonlinear optimization problems in which the model error as well as the notorieties associated with nonconvex optimization (such as the danger of getting stuck in local minima) pose a greater threat to solution quality than the truncation error in practice. Furthermore, in most practical applications, a particular state vector need not be optimal in order to be considered adequate.

### 5.2. Gradient-based optimization using an adjoint formulation

In the following, we present a scheme that is suitable for gradient-based optimization, where all terms involved are assembled from expressions that have been derived fully symbolically. For given $\boldsymbol{\alpha}$, we assume that a suitable tuple $(\mathcal{U}_h^{\boldsymbol{\alpha}}, \boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha}})$ has been chosen based on principles A to C (see Section 5.1). Particular methodologies for satisfying these principles depend on the application and are discussed in Section 6. In the following, the operator $[\cdot]$ returns the canonical basis of a vector space, which we assume to be clear from context. Reminiscent of (12), we introduce

$$\boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha},\mathbf{f}} \equiv \{\mathbf{v} \in \boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha}} \mid \mathbf{v} = \mathbf{f} \text{ on } \partial\hat{\Omega}\} \quad \text{and} \quad \mathcal{U}_h^{\boldsymbol{\alpha},f} \equiv \{v \in \mathcal{U}_h^{\boldsymbol{\alpha}} \mid v = f \text{ on } \partial\hat{\Omega}_D\}, \tag{20}$$

where, as before, $\mathbf{f} \in \boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha}}$ and $f \in \mathcal{U}_h^{\boldsymbol{\alpha}}$ by assumption. In (20), $\partial\hat{\Omega}_D \subseteq \partial\hat{\Omega}$ refers to the preimage of $\partial\Omega_{D,h}^{\boldsymbol{\alpha}} \subseteq \partial\Omega_h^{\boldsymbol{\alpha}}$ under the mapping $\mathbf{x}_h^{\boldsymbol{\alpha}}$. As opposed to (4), here $\mathcal{U}_h^{\boldsymbol{\alpha}}$ is defined in the static parametric domain. In the following, we assume that $\partial\Omega_D^{\boldsymbol{\alpha}} = \emptyset$ for convenience, i.e., $u_h^{\boldsymbol{\alpha}}$ is free of Dirichlet data or the data is enforced with Nitsche's method [35]. This significantly simplifies the expression for the gradient. Eq. (20) allows for the decomposition into *boundary* ($\mathcal{B}$) and *inner* ($\mathcal{I}$) bases:

$$\left[\boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha}}\right] = \left[\boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha},\mathcal{B}}\right] \cup \left[\boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha},\mathcal{I}}\right], \quad \text{with} \quad \boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha},\mathcal{I}} = \boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha},\mathbf{0}} \quad \text{and} \quad \boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha},\mathcal{B}} = \boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha}} \setminus \boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha},\mathcal{I}}, \tag{21}$$

Given $\mathbf{x}_{D,h}^{\boldsymbol{\alpha}} \in \boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha},\mathcal{B}}$ (see Section 5.1), we introduce the mapping

$$\mathbf{x}_h^{\boldsymbol{\alpha}} = \mathbf{x}_0^{\boldsymbol{\alpha}} + \mathbf{x}_{D,h}^{\boldsymbol{\alpha}}, \quad \text{with} \quad \mathbf{x}_0^{\boldsymbol{\alpha}} = \sum_{\boldsymbol{\sigma}_i \in \left[\boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha},\mathcal{I}}\right]} c_i^{\mathcal{I}} \boldsymbol{\sigma}_i \quad \text{and} \quad \mathbf{x}_{D,h}^{\boldsymbol{\alpha}} = \sum_{\boldsymbol{\sigma}_j \in \left[\boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha},\mathcal{B}}\right]} c_j^{\mathcal{B}} \boldsymbol{\sigma}_j, \tag{22}$$

where the $c_j^{\mathcal{B}}$ are known. We introduce the vector of weights $\mathbf{c}_{\mathcal{A}}$ (where the subscript $\mathcal{A}$ stands for *all*), which is the concatenation of the vectors $\mathbf{c}_{\mathcal{I}}$ and $\mathbf{c}_{\mathcal{B}}(\boldsymbol{\alpha})$, containing the $c_i^{\mathcal{I}}$ and $c_j^{\mathcal{B}}$, respectively. Similarly, we introduce

$$u_h^{\boldsymbol{\alpha}} = \sum_{\phi_i \in [\mathcal{U}_h^{\boldsymbol{\alpha}}]} d_i \phi_i \quad \text{with the corresponding vector of weights} \quad (\mathbf{d}_{\mathcal{A}})_i = d_i. \tag{23}$$

With the introduction of the tuple $(\mathbf{c}_{\mathcal{A}}, \mathbf{d}_{\mathcal{A}})$, the discrete objective function is rewritten in the form

$$J_h^{\boldsymbol{\alpha}}(\mathbf{x}_h^{\boldsymbol{\alpha}}, u_h^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) \longrightarrow J_h^{\boldsymbol{\alpha}}(\mathbf{c}_{\mathcal{A}}, \mathbf{d}_{\mathcal{A}}, \boldsymbol{\alpha}), \tag{24}$$

with

$$\mathbf{c}_{\mathcal{A}} = \mathbf{c}_{\mathcal{A}}(\mathbf{c}_{\mathcal{I}}, \mathbf{c}_{\mathcal{B}}), \quad \mathbf{c}_{\mathcal{I}} = \mathbf{c}_{\mathcal{I}}(\mathbf{c}_{\mathcal{B}}), \quad \mathbf{c}_{\mathcal{B}} = \mathbf{c}_{\mathcal{B}}(\boldsymbol{\alpha}) \quad \text{and} \quad \mathbf{d}_{\mathcal{A}} = \mathbf{d}_{\mathcal{A}}(\mathbf{c}_{\mathcal{A}}, \boldsymbol{\alpha}). \tag{25}$$

With above concatenated dependencies in mind, the transposed gradient approximation reads:

$$\frac{\mathrm{d}J_h^{\boldsymbol{\alpha}\,T}}{\mathrm{d}\boldsymbol{\alpha}} = \left[\frac{\mathrm{d}\mathbf{c}_{\mathcal{A}}}{\mathrm{d}\boldsymbol{\alpha}}\right]^T \left( \left[\frac{\partial \mathbf{d}_{\mathcal{A}}}{\partial \mathbf{c}_{\mathcal{A}}}\right]^T \frac{\partial J_h^{\boldsymbol{\alpha}\,T}}{\partial \mathbf{d}_{\mathcal{A}}} + \frac{\partial J_h^{\boldsymbol{\alpha}\,T}}{\partial \mathbf{c}_{\mathcal{A}}} \right) + \left[\frac{\mathrm{d}\mathbf{d}_{\mathcal{A}}}{\mathrm{d}\boldsymbol{\alpha}}\right]^T \frac{\partial J_h^{\boldsymbol{\alpha}\,T}}{\partial \mathbf{d}_{\mathcal{A}}} + \frac{\partial J_h^{\boldsymbol{\alpha}\,T}}{\partial \boldsymbol{\alpha}}, \tag{26}$$

where we denoted matrix quantities in square brackets.

Introducing the discrete EGG residual vector $\mathbf{F}_h^{\boldsymbol{\alpha}}$ with

$$\left(\mathbf{F}_h^{\boldsymbol{\alpha}}\right)_i = F(\mathbf{x}_h^{\boldsymbol{\alpha}}, \boldsymbol{\sigma}_i), \quad \text{for} \quad \boldsymbol{\sigma}_i \in \left[\mathcal{V}_h^{\boldsymbol{\alpha},\mathcal{I}}\right] \quad \text{and} \quad F(\cdot,\cdot) \text{ as defined in (11)}, \tag{27}$$

we use the implicit function theorem [36] to derive an expression for the gradient of $\mathbf{c}_{\mathcal{A}}$. We have:

$$\left[\frac{\mathrm{d}\mathbf{c}_{\mathcal{A}}}{\mathrm{d}\boldsymbol{\alpha}}\right]^T = \left[\left[\frac{\mathrm{d}\mathbf{c}_{\mathcal{I}}}{\mathrm{d}\boldsymbol{\alpha}}\right]^T, \left[\frac{\partial \mathbf{c}_{\mathcal{B}}}{\partial \boldsymbol{\alpha}}\right]^T\right], \quad \text{with} \quad \left[\frac{\mathrm{d}\mathbf{c}_{\mathcal{I}}}{\mathrm{d}\boldsymbol{\alpha}}\right] = -\left[\frac{\partial \mathbf{F}_h^{\boldsymbol{\alpha}}}{\partial \mathbf{c}_{\mathcal{I}}}\right]^{-1} \left[\frac{\partial \mathbf{F}_h^{\boldsymbol{\alpha}}}{\partial \mathbf{c}_{\mathcal{B}}}\right] \left[\frac{\partial \mathbf{c}_{\mathcal{B}}}{\partial \boldsymbol{\alpha}}\right]. \tag{28}$$

Similarly, we define the residual vector $\mathbf{B}_h^{\boldsymbol{\alpha}}$ of the discretized weak state equation (see (5)), with entries

$$\left(\mathbf{B}_h^{\boldsymbol{\alpha}}\right)_i = B\left(u_h^{\boldsymbol{\alpha}}, \mathbf{x}_h^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}, \phi_i\right), \quad \text{for} \quad \phi_i \in \left[\mathcal{U}_h^{\boldsymbol{\alpha}}\right]. \tag{29}$$

Implicit differentiation yields

$$\left[\frac{\partial \mathbf{d}_{\mathcal{A}}}{\partial \mathbf{c}_{\mathcal{A}}}\right] = -\left[\frac{\partial \mathbf{B}_h^{\boldsymbol{\alpha}}}{\partial \mathbf{d}_{\mathcal{A}}}\right]^{-1} \left[\frac{\partial \mathbf{B}_h^{\boldsymbol{\alpha}}}{\partial \mathbf{c}_{\mathcal{A}}}\right] \quad \text{and} \quad \left[\frac{\mathrm{d}\mathbf{d}_{\mathcal{A}}}{\mathrm{d}\boldsymbol{\alpha}}\right] = -\left[\frac{\partial \mathbf{B}_h^{\boldsymbol{\alpha}}}{\partial \mathbf{d}_{\mathcal{A}}}\right]^{-1} \left[\frac{\partial \mathbf{B}_h^{\boldsymbol{\alpha}}}{\partial \boldsymbol{\alpha}}\right]. \tag{30}$$

Substituting in (26) leads to

$$\frac{\mathrm{d}J_h^{\boldsymbol{\alpha}\,T}}{\mathrm{d}\boldsymbol{\alpha}} = \left[\frac{\mathrm{d}\mathbf{c}_{\mathcal{A}}}{\mathrm{d}\boldsymbol{\alpha}}\right]^T \mathbf{b} - \left[\frac{\partial \mathbf{B}_h^{\boldsymbol{\alpha}}}{\partial \boldsymbol{\alpha}}\right]^T \mathbf{a} + \frac{\partial J_h^{\boldsymbol{\alpha}\,T}}{\partial \boldsymbol{\alpha}}, \tag{31}$$

with

$$\mathbf{a} = \left[\frac{\partial \mathbf{B}_h^{\boldsymbol{\alpha}}}{\partial \mathbf{d}_{\mathcal{A}}}\right]^{-T} \frac{\partial J_h^{\boldsymbol{\alpha}\,T}}{\partial \mathbf{d}_{\mathcal{A}}} \quad \text{and} \quad \mathbf{b} = -\left[\frac{\partial \mathbf{B}_h^{\boldsymbol{\alpha}}}{\partial \mathbf{c}_{\mathcal{A}}}\right]^T \mathbf{a} + \frac{\partial J_h^{\boldsymbol{\alpha}\,T}}{\partial \mathbf{c}_{\mathcal{A}}}. \tag{32}$$

Vector $\mathbf{a}$ is computed by solving the following linear system:

$$\left[\frac{\partial \mathbf{B}_h^{\boldsymbol{\alpha}}}{\partial \mathbf{d}_{\mathcal{A}}}\right]^T \mathbf{a} = \frac{\partial J_h^{\boldsymbol{\alpha}\,T}}{\partial \mathbf{d}_{\mathcal{A}}}. \tag{33}$$

Vector $\mathbf{b}$ then follows from substituting $\mathbf{a}$ in (32). Furthermore, we have

$$\left[\frac{\mathrm{d}\mathbf{c}_{\mathcal{A}}}{\mathrm{d}\boldsymbol{\alpha}}\right]^T \mathbf{b} = \left[\left[\frac{\mathrm{d}\mathbf{c}_{\mathcal{I}}}{\mathrm{d}\boldsymbol{\alpha}}\right]^T, \left[\frac{\partial \mathbf{c}_{\mathcal{B}}}{\partial \boldsymbol{\alpha}}\right]^T\right] \begin{bmatrix}\mathbf{b}_{\mathcal{I}} \\ \mathbf{b}_{\mathcal{B}}\end{bmatrix} = \left[\frac{\partial \mathbf{c}_{\mathcal{B}}}{\partial \boldsymbol{\alpha}}\right]^T \mathbf{q}, \tag{34}$$

where

$$\mathbf{q} = -\left[\frac{\partial \mathbf{F}_h^{\boldsymbol{\alpha}}}{\partial \mathbf{c}_{\mathcal{B}}}\right]^T \mathbf{e} + \mathbf{b}_{\mathcal{B}} \quad \text{with} \quad \mathbf{e} = \left[\frac{\partial \mathbf{F}_h^{\boldsymbol{\alpha}}}{\partial \mathbf{c}_{\mathcal{I}}}\right]^{-T} \mathbf{b}_{\mathcal{I}}. \tag{35}$$

We compute the matrix–vector product

$$\mathbf{e} = \left[\frac{\partial \mathbf{F}_h^{\boldsymbol{\alpha}}}{\partial \mathbf{c}_{\mathcal{I}}}\right]^{-T} \mathbf{b}_{\mathcal{I}} \quad \text{from the solution of} \quad \left[\frac{\partial \mathbf{F}_h^{\boldsymbol{\alpha}}}{\partial \mathbf{c}_{\mathcal{I}}}\right]^T \mathbf{e} = \mathbf{b}_{\mathcal{I}}. \tag{36}$$

Finally, it should be noted that the matrix $[\partial_{\boldsymbol{\alpha}} \mathbf{c}_{\mathcal{B}}]$ in (34) depends on the operator

$$\pi^{\boldsymbol{\alpha}} : C^0(\mathbb{R}^2, \partial\hat{\Omega}) \to \mathcal{V}_h^{\boldsymbol{\alpha},\mathcal{B}}, \quad \text{with} \quad \pi^{\boldsymbol{\alpha}}(\mathbf{x}_D^{\boldsymbol{\alpha}}) = \mathbf{x}_{D,h}^{\boldsymbol{\alpha}} \tag{37}$$

and typically involves a sparse matrix–matrix inverse product. Upon transposing, the order of multiplication is reversed and the transposed inverse moves to the front. Therefore, we can treat matrix–vector products of the form $[\partial_{\boldsymbol{\alpha}} \mathbf{c}_{\mathcal{B}}]^T \mathbf{k}$ by inverting a sparse linear system and subsequent multiplication by a sparse matrix. We will present concrete examples of this step in Section 6. In the following, we recapitulate all the necessary steps for computing the tuple $\left(J_h^{\boldsymbol{\alpha}}, \mathrm{d}_{\boldsymbol{\alpha}} J_h^{\boldsymbol{\alpha}}\right)$ for given $\boldsymbol{\alpha}$.

**S.1**: Choose an appropriate spline space tuple $(\mathcal{U}_h^{\boldsymbol{\alpha}}, \mathcal{V}_h^{\boldsymbol{\alpha}})$.
**S.2**: Compute $\mathbf{x}_{D,h}^{\boldsymbol{\alpha}}$ from $\mathbf{x}_D^{\boldsymbol{\alpha}}$ using $\pi^{\boldsymbol{\alpha}}$.
**S.3**: Solve the nonlinear root-finding problem $\mathbf{F}_h^{\boldsymbol{\alpha}}(\mathbf{c}_{\mathcal{I}}) = \mathbf{0}$, yielding the analysis-suitable mapping $\mathbf{x}_h^{\boldsymbol{\alpha}}$.
**S.4**: Solve the root-finding problem $\mathbf{B}_h^{\boldsymbol{\alpha}} = \mathbf{0}$ using a suitable numerical algorithm. This yields the state variable $u_h^{\boldsymbol{\alpha}}$.
**S.5**: Substitute $(\mathbf{x}_h^{\boldsymbol{\alpha}}, u_h^{\boldsymbol{\alpha}}, \boldsymbol{\alpha})$ in $J(\cdot,\cdot,\cdot)$ to compute $J_h^{\boldsymbol{\alpha}}$.
**S.6**: Compute $\mathbf{a} \to \mathbf{b} \to \mathbf{e} \to \mathbf{q}$ and finally $\mathrm{d}_{\boldsymbol{\alpha}} J_h^{\boldsymbol{\alpha}}$ using (31) to (36).

Due to the approximate nature of the tuple $(u_h^{\boldsymbol{\alpha}}, \mathbf{x}_h^{\boldsymbol{\alpha}})$, we allow for a small amount of slack in the assessment of numerical feasibility, i.e., we replace

$$\begin{aligned} g_i(u^{\boldsymbol{\alpha}}, \mathbf{x}^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) \geq 0 \quad &\longrightarrow \quad g_i(u_h^{\boldsymbol{\alpha}}, \mathbf{x}_h^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) \geq \mu \quad \forall i \in \{1, \ldots, N_{\neq}\} \\ h_j(u^{\boldsymbol{\alpha}}, \mathbf{x}^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) = 0 \quad &\longrightarrow \quad -\mu \leq h_j(u_h^{\boldsymbol{\alpha}}, \mathbf{x}_h^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) \leq \mu \quad \forall j \in \{1, \ldots, N_{=}\}, \end{aligned} \tag{38}$$

with $\mu > 0$ in (3). The procedure that carries out **S.1** to **S.6**, along with the relaxed constraints, is passed to a gradient-based optimization routine (such as IPOPT).

### 5.3. Gradient assembly costs

In the following, we analyze the computational costs of assembling the gradient. The majority of the costs result from assembling sparse matrices, as well as solving sparse linear systems, such as in (33). In order to compute $\mathbf{x}_h^{\boldsymbol{\alpha}}$, we simultaneously assemble the quantities

$$\mathbf{F}_h^{\boldsymbol{\alpha}}(\mathbf{c}_{\mathcal{I}}^i) \quad \text{and} \quad \left[\frac{\partial \mathbf{F}_h^{\boldsymbol{\alpha}}}{\partial \mathbf{c}_{\mathcal{I}}^i}\right] \tag{39}$$

during a joint element loop at the beginning of the $i$th Newton iteration (see Section 4). As such, this routine automatically yields the matrix $\left[\partial_{\mathbf{c}_{\mathcal{I}}} \mathbf{F}_h^{\boldsymbol{\alpha}}\right]$ at the last step. If $\mathbf{B}_h^{\boldsymbol{\alpha}}$ is linear, assembling $\left[\partial_{\mathbf{d}_{\mathcal{A}}} \mathbf{B}_h^{\boldsymbol{\alpha}}\right]$ is a precursor to computing $u_h^{\boldsymbol{\alpha}}$ and hence available. If $\mathbf{B}_h^{\boldsymbol{\alpha}}$ is nonlinear, we recommend basing an iterative algorithm on Newton's method and computing the residual and its derivative in tandem, as in (39). As such, additional cost factors are assembling $\left[\partial_{\mathbf{c}_{\mathcal{A}}} \mathbf{B}_h^{\boldsymbol{\alpha}}\right]$ and solving a number of sparse linear equations. Due to the nonlinear nature of $\mathbf{F}_h^{\boldsymbol{\alpha}}$, the cost of computing $\mathrm{d}_{\boldsymbol{\alpha}} J_h^{\boldsymbol{\alpha}}$ is of the same order as a discrete evaluation of $J(\cdot,\cdot,\cdot)$ (regardless of the length of $\boldsymbol{\alpha}$).

Finally, we note that the discrete constraint gradients associated with the $g_i$ and $h_j$ are efficiently computed by replacing $J_h^{\boldsymbol{\alpha}}$ by the corresponding term in (26) and repeating steps (31) to (36). Hereby, the required matrices can be reused from the assembly of the gradient.

### 5.4. Memory-saving strategies in large-scale applications

In light of enabling large-scale optimization as well as the prospect of extending the presented methodology to volumetric applications, in the following, we discuss ways to avoid the memory-consuming assembly of the matrices involved in computing $u_h^{\boldsymbol{\alpha}}$, $\mathbf{x}_h^{\boldsymbol{\alpha}}$ and $J_h^{\boldsymbol{\alpha}}$ and their derivatives.

Memory-saving strategies are based on the observation that matrices only appear in the form of matrix–vector products during the assembly of $\mathrm{d}_{\boldsymbol{\alpha}} J_h^{\boldsymbol{\alpha}}$. Let $\mathbf{B} = \mathbf{B}(\ldots, \mathbf{q}, \ldots)$. Then, we have

$$\left[\frac{\partial \mathbf{B}(\ldots, \mathbf{q}, \ldots)}{\partial \mathbf{q}}\right] \mathbf{a} \simeq \frac{B(\ldots, \mathbf{q} + \epsilon \mathbf{a}, \ldots) - B(\ldots, \mathbf{q}, \ldots)}{\epsilon}, \tag{40}$$

for $\epsilon > 0$ small. As such, in steps (31) to (36), matrix–vector products can be approximated using (40). Since Krylov-subspace (KS) methods such as GMRES [37] only require matrix–vector products, we combine a KS-method with (40) for solving linear systems as they appear in, e.g., Eq. (33). Reminiscent of *Newton–Krylov* [38], this principle

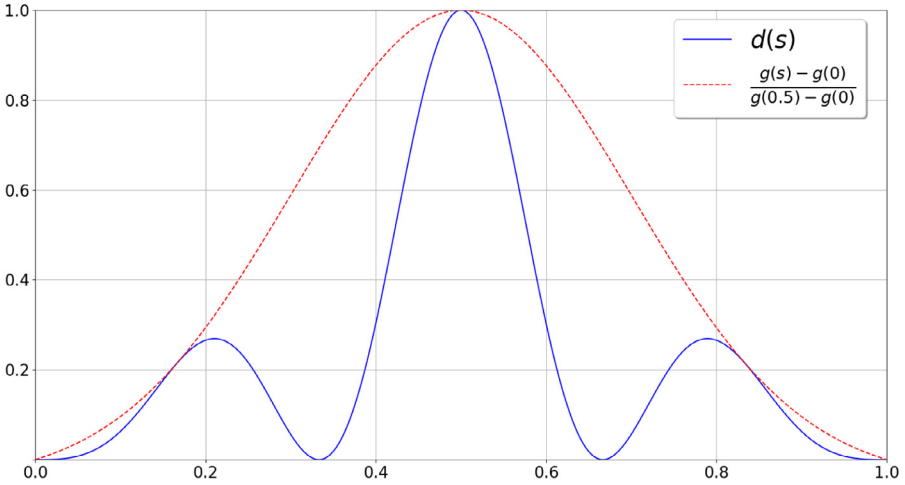


**Fig. 1.** Plot of $d(s)$ (see (41)) and the corresponding envelope function over the interval $s \in [0, 1]$.

may be extended to the computation of $u_h^{\boldsymbol{\alpha}}$ and $\mathbf{x}_h^{\boldsymbol{\alpha}}$, hence completely bypassing matrix assembly in steps **S.1** to **S.6**. Hereby, we regard the cumulative error contribution to $\mathrm{d}_{\boldsymbol{\alpha}} J_h^{\boldsymbol{\alpha}} \simeq \mathrm{d}_{\boldsymbol{\alpha}} J^{\boldsymbol{\alpha}}$ as negligible compared to other sources (such as the model error and the truncation resulting from the numerical scheme). The optimal choice of $\epsilon$ is discussed in [38].

## 6. Examples

In this section we apply the methodology from Section 5.1 to selected test cases. We consider the first example a validation test case, in which the exact minimizer can be computed exactly (up to machine precision). Hereby, we compare the results of VBA (see Section 5) to the exact minimum. Furthermore, we compare the VBA results to those resulting from taking the tuple $(\mathcal{U}_h^{\boldsymbol{\alpha}}, \mathcal{V}_h^{\boldsymbol{\alpha}})$ static (SBA). In the second case, we consider the design of a cooling element, whereby the plausibility of the outcome can only be assessed using physical reasoning.

Both examples have been carefully selected in order to be geometrically challenging. We implemented the scheme from Section 5 in the open-source Python library *Nutils* [39].

### 6.1. A validation example with known exact solution

We are considering the example of a domain fenced-off by four parametric curves that are given by an envelope function multiplied by a cosine, whereby the amplitude of the cosine is a degree of freedom in $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)$. We define the function

$$d(s) = \underbrace{\left(\frac{g(s) - g(0)}{g(0.5) - g(0)}\right)}_{\text{envelope function}} \underbrace{\left(\frac{1 - \cos(\omega \pi s)}{2}\right)}_{\text{trigonometric component}}, \quad \text{where} \quad g(s) = \exp\left(-\frac{\left(s - \frac{1}{2}\right)^2}{2\sigma^2}\right), \tag{41}$$

with $(\omega, \sigma) = (6, 0.2)$. Note that $d(0) = d(1) = 0$ and $d(0.5) = 1$, $d(s)$ and the envelope function are depicted in Fig. 1. Let $\partial\hat{\Omega} = \bar{\gamma}_S \cup \bar{\gamma}_E \cup \bar{\gamma}_N \cup \bar{\gamma}_W$, where the $\gamma_\beta$, $\beta \in \{S, E, N, W\}$ denote the southern, eastern, northern, and western boundary of $\partial\hat{\Omega}$, respectively. The contour parameterization $\mathbf{x}_D^{\boldsymbol{\alpha}} : \partial\hat{\Omega} \to \partial\Omega^{\boldsymbol{\alpha}}$ reads:

$$\mathbf{x}_D^{\boldsymbol{\alpha}}(\boldsymbol{\xi}) = \begin{cases} (\xi_1, \alpha_1 d(\xi_1))^T & \boldsymbol{\xi} \in \bar{\gamma}_S \\ (1 - \alpha_2 d(\xi_2), \xi_2)^T & \boldsymbol{\xi} \in \bar{\gamma}_E \\ (\xi_1, 1 - \alpha_3 d(\xi_1))^T & \boldsymbol{\xi} \in \bar{\gamma}_N \\ (\alpha_4 d(\xi_2), \xi_2)^T & \boldsymbol{\xi} \in \bar{\gamma}_W \end{cases}, \tag{42}$$

while

$$\boldsymbol{\lambda} = \{\boldsymbol{\alpha} \in \mathbb{R}^4 \mid \mathbf{0} \leq \boldsymbol{\alpha} \leq \tfrac{2}{5}\mathbb{1}\}, \tag{43}$$

where $\mathbb{1}$ is a vector of ones.

Here, we base $\pi^{\boldsymbol{\alpha}} : C^0(\mathbb{R}^2, \partial\hat{\Omega}) \to \boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha},\mathcal{B}}$ (see Section 5) on an $L_2(\partial\hat{\Omega})$ projection. For given $\boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha},\mathcal{B}}$, we hence have

$$\left[\frac{\partial \mathbf{c}_{\mathcal{B}}}{\partial \boldsymbol{\alpha}}\right]^T = -\left[\frac{\partial \mathbf{D}^{\boldsymbol{\alpha}}}{\partial \boldsymbol{\alpha}}\right]^T \left[\frac{\partial \mathbf{D}^{\boldsymbol{\alpha}}}{\partial \mathbf{c}_{\mathcal{B}}}\right]^{-T}, \quad \text{where} \quad \mathbf{D}_i^{\boldsymbol{\alpha}} = \int_{\partial\hat{\Omega}} \boldsymbol{\sigma}_i^{\mathcal{B}} \cdot \left(\mathbf{x}_D^{\boldsymbol{\alpha}} - \mathbf{x}_{D,h}^{\boldsymbol{\alpha}}(\mathbf{c}_{\mathcal{B}})\right) \mathrm{d}\gamma \tag{44}$$

and $\boldsymbol{\sigma}_i^{\mathcal{B}} \in \left[\boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha},\mathcal{B}}\right]$. In (44), we take matrix–vector products in the same way as in Section 5. We base our state variable residual on the following PDE problem:

$$-\Delta_{\mathbf{x}^{\boldsymbol{\alpha}}} u^{\boldsymbol{\alpha}} = -\Delta_{\mathbf{x}^{\boldsymbol{\alpha}}} f^{\boldsymbol{\alpha}} \quad \text{in } \hat{\Omega}, \quad \text{s.t.} \quad u^{\boldsymbol{\alpha}}\big|_{\partial\hat{\Omega}} = f^{\boldsymbol{\alpha}}, \quad \text{where} \quad f^{\boldsymbol{\alpha}} = \det\left[\frac{\partial \mathbf{x}^{\boldsymbol{\alpha}}}{\partial \boldsymbol{\xi}}\right] \tag{45}$$

and $\Delta_{\mathbf{x}^{\boldsymbol{\alpha}}}$ denotes the Laplace–Beltrami operator corresponding to $\mathbf{x}^{\boldsymbol{\alpha}}$. Clearly, the exact solution of (45) satisfies $u^{\boldsymbol{\alpha}} = f^{\boldsymbol{\alpha}}$. We derive the weak form of (45) and implement the boundary conditions using Nitsche's method. This leads to

$$\left(\mathbf{B}_h^{\boldsymbol{\alpha}}\right)_i = \left(\nabla(u_h^{\boldsymbol{\alpha}} - f^{\boldsymbol{\alpha}}), \nabla\phi_i\right)_{\Omega_h^{\boldsymbol{\alpha}}} - \int_{\partial\Omega_h^{\boldsymbol{\alpha}}} \phi_i \frac{\partial u_h^{\boldsymbol{\alpha}}}{\partial \mathbf{n}} \mathrm{d}\gamma - \int_{\partial\Omega_h^{\boldsymbol{\alpha}}} (u_h^{\boldsymbol{\alpha}} - f^{\boldsymbol{\alpha}}) \frac{\partial \phi_i}{\partial \mathbf{n}} \mathrm{d}\gamma + \eta_i \int_{\partial\Omega_h^{\boldsymbol{\alpha}}} (u_h^{\boldsymbol{\alpha}} - f^{\boldsymbol{\alpha}})\phi_i \mathrm{d}\gamma, \tag{46}$$

with $\phi_i \in \left[\mathcal{U}_h^{\boldsymbol{\alpha}}\right]$. Here, $\partial/\partial\mathbf{n}$ denotes the outward normal derivative with respect to $\Omega_h^{\boldsymbol{\alpha}}$ and $\eta_i \gg 1$ is a penalty parameter. We use

$$\eta_i = \begin{cases} cI_i^{-1} & I_i > 0 \\ 0 & \text{else} \end{cases}, \quad \text{where} \quad I_i = \int_{\partial\Omega_h^{\boldsymbol{\alpha}}} \phi_i \mathrm{d}\gamma \quad \text{and} \quad c = 10^3. \tag{47}$$

The objective function reads:

$$J^{\boldsymbol{\alpha}} = \int_{\hat{\Omega}} u^{\boldsymbol{\alpha}} \mathrm{d}S + \frac{1}{2}\|\boldsymbol{\alpha}\|^2 \implies J_h^{\boldsymbol{\alpha}} = \int_{\hat{\Omega}} u_h^{\boldsymbol{\alpha}} \mathrm{d}S + \frac{1}{2}\|\boldsymbol{\alpha}\|^2 \tag{48}$$

and there are no further constraints. Since $u^{\boldsymbol{\alpha}} = \partial_{\boldsymbol{\xi}} \mathbf{x}^{\boldsymbol{\alpha}}$, we have

$$\int_{\hat{\Omega}} u^{\boldsymbol{\alpha}} \mathrm{d}S = \mathrm{Area}(\Omega^{\boldsymbol{\alpha}}) = 1 - \sum_i \alpha_i A, \quad \text{where} \quad A = \int_{[0,1]} d(s) \mathrm{d}s. \tag{49}$$

We compute the exact value of $A$ up to machine precision, which yields $A \simeq 0.2374$. The exact minimum over $\boldsymbol{\alpha} \in \boldsymbol{\lambda}$ is assumed at $\boldsymbol{\alpha}* = A\mathbb{1}$ and yields $J^{\alpha*} \simeq 0.8873$. The contours of the resulting domain $\Omega^{\alpha*}$ are depicted in Fig. 2.

For increasingly fine $(\mathcal{U}_h^{\boldsymbol{\alpha}}, \boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha}})$, the minimum of the discretized optimization problem should converge to the exact minimum, allowing us to test the consistency of the scheme.

In the following, we discuss how to choose the tuple $(\mathcal{U}_h^{\boldsymbol{\alpha}}, \boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha}})$ during each iteration. We start by dividing $\hat{\Omega}$ into a structured set of elements, resulting from the bivariate knot vector $\boldsymbol{\Xi}^{p_1,p_2} = \Xi^{p_1} \times \mathcal{H}^{p_2}$, where the $p_i$ denote the polynomial degree. Here, we restrict ourselves to bicubic bases, i.e., $p_1 = p_2 = 3$, with maximum regularity. With points A to C (see Section 5) in mind, we repeatedly refine the $\phi_i \in \left[V_h^{\boldsymbol{\alpha},\mathcal{B}}\right]$, where, as before

$$V_h^{\boldsymbol{\alpha}} \times V_h^{\boldsymbol{\alpha}} = \boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha}} \quad \text{and} \quad V_h^{\boldsymbol{\alpha},\mathcal{B}} = \{\phi \in V_h^{\boldsymbol{\alpha}} \mid \phi \neq 0 \text{ on } \partial\hat{\Omega}\}$$

until the Dirichlet data is approximated sufficiently well. Refinement of some $\phi_i \in \left[V_h^{\boldsymbol{\alpha},\mathcal{B}}\right]$ entails replacing its supporting elements by their finer counterparts from the next level in the element hierarchy, leading to a finer element segmentation of $\hat{\Omega}$ and an associated canonical THB-spline basis with a locally increased number of DOFs. For more information on THB-refinement we refer to [28,40].

Here, $V_h^{\boldsymbol{\alpha}}$ is initialized to the coarse-grid basis resulting from $\boldsymbol{\Xi}^{p_1,p_2}$. Let the $i$th contribution to the projection residual be denoted by $r_i(\mathbf{x}_{D,h}^{\boldsymbol{\alpha}})$, where

$$R(\mathbf{x}_{D,h}^{\boldsymbol{\alpha}})^2 = \frac{1}{2}\int_{\partial\hat{\Omega}} \left\|\mathbf{x}_D^{\boldsymbol{\alpha}} - \mathbf{x}_{D,h}^{\boldsymbol{\alpha}}\right\|^2 \mathrm{d}\gamma = \frac{1}{2}\sum_i \int_{\partial\hat{\Omega}} \phi_i \left\|\mathbf{x}_D^{\boldsymbol{\alpha}} - \mathbf{x}_{D,h}^{\boldsymbol{\alpha}}\right\|^2 \mathrm{d}\gamma \equiv \frac{1}{2}\sum_i r_i^2(\mathbf{x}_{D,h}^{\boldsymbol{\alpha}}). \tag{50}$$

Here, we made use of the fact that the $\phi_i$ form a partition of unity on $\partial\hat{\Omega}$.

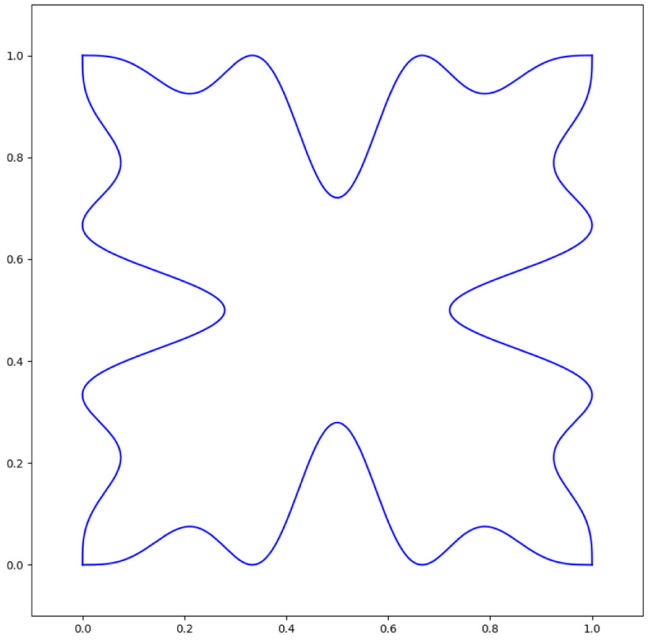


**Fig. 2.** The contours of the domain $\Omega^{\boldsymbol{\alpha}*}$ that correspond to the exact minimizer $\boldsymbol{\alpha}*$.

We refine $\phi_i \in \left[V_h^{\boldsymbol{\alpha},\mathcal{B}}\right]$ whenever $r_i(\mathbf{x}_{D,h}^{\boldsymbol{\alpha}})$ exceeds a threshold $\mu_i$. The threshold is of the form

$$\mu_i = \frac{\mu}{\sqrt{\|\phi_i\|_{L_2(\partial\hat{\Omega})}}}, \tag{51}$$

where $\mu$ is a small positive constant that tunes the accuracy of the approximation $\mathbf{x}_{D,h}^{\boldsymbol{\alpha}} \simeq \mathbf{x}_D^{\boldsymbol{\alpha}}$ on $\partial\hat{\Omega}$.

As a next step, we compute $\mathbf{x}_h^{\boldsymbol{\alpha}}$ using the methodology from Section 4. As the choice of $\boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha}} = V_h^{\boldsymbol{\alpha}} \times V_h^{\boldsymbol{\alpha}}$, at this point, is solely based on accurately resolving the boundary contours, it may be too optimistic (in terms of the number of inner DOFs) for computing a folding-free mapping. In the case of folding, we apply a posteriori refinement to *defective* elements, i.e., elements $\mathcal{E} \subset \hat{\Omega}$ on which

$$\det\left[\frac{\partial \mathbf{x}_h^{\boldsymbol{\alpha}}}{\partial \boldsymbol{\xi}}\right](\mathbf{p}) < 0$$

for some $\mathbf{p} \in \mathcal{E}$, by refining all $\phi_i \in \left[V_h^{\boldsymbol{\alpha}}\right]$ that are non-vanishing on $\mathcal{E}$. The defective mapping is prolonged to the refined space and serves as an initial guess for recomputing it from the enriched space. This step may be repeated until $\boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha}}$ is such that $\mathbf{x}_h^{\boldsymbol{\alpha}} \in \boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha}}$ is folding-free. Note that, although the proposed methodology is robust in practice, it may lead to over-refinement. The methodology may be combined with the refinement strategies proposed in [29], which avoid over-refinement.

**Remark.** Here, we base the selection of $\boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha}}$ on a posteriori strategies, which necessitates recomputing $\mathbf{x}_h^{\boldsymbol{\alpha}}$ after each refinement. Choosing the coarse-grid basis properly (i.e., not too coarse), we typically did not encounter more than $1-2$ a posteriori refinements in the cases considered in this work. Fortunately, the defective mappings can be used as an initial guess for the recomputed one, significantly reducing computational costs.

Reliable a priori refinement strategies are however desirable and constitute a topic for future research.

After achieving bijectivity, additional refinement can be performed in order to further improve the quality of the mapping. A posteriori strategies that rely on the *Winslow* functional [41] are discussed in [29].

Upon completion, we are in the possession of an analysis-suitable $\mathbf{x}_h^{\boldsymbol{\alpha}} : \hat{\Omega} \to \Omega_h^{\boldsymbol{\alpha}}$ from the appropriately refined $\boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha}}$. As a next step, we choose a suitable space $\mathcal{U}_h^{\boldsymbol{\alpha}}$. Heuristically, there exists a strong correlation between the

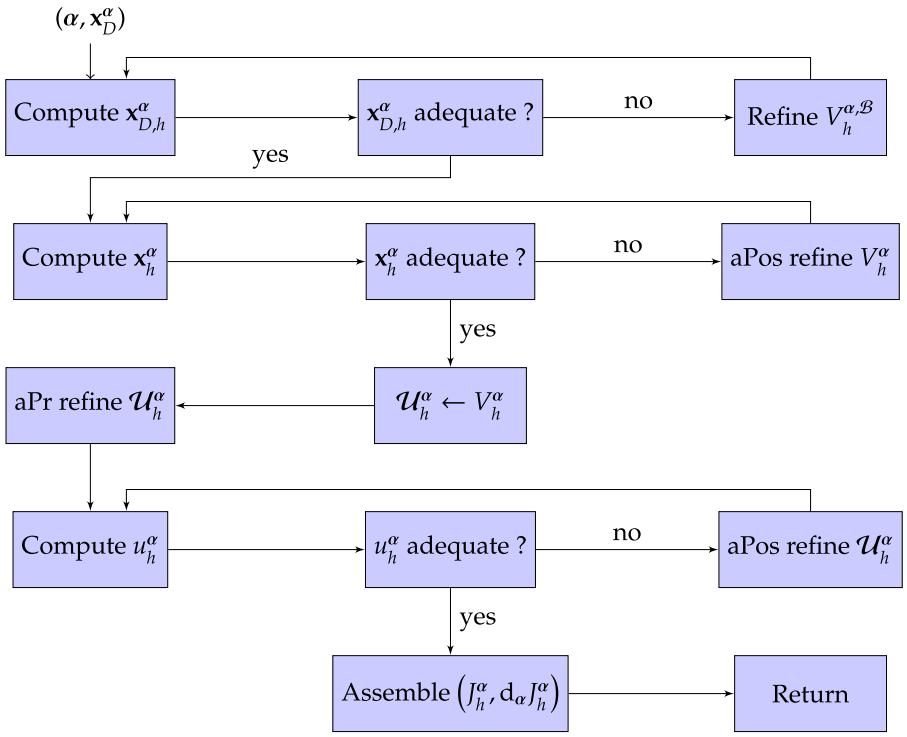


**Fig. 3.** Block diagram summarizing all the steps required for computing the tuple $\left(J_h^{\boldsymbol{\alpha}}, \mathrm{d}_{\boldsymbol{\alpha}} J_h^{\boldsymbol{\alpha}}\right)$.

regions in which $V_h^{\boldsymbol{\alpha}}$ has been refined in order to yield an analysis-suitable $\mathbf{x}_h^{\boldsymbol{\alpha}}$ and the regions that ought to be refined in order to accurately approximate $u^{\boldsymbol{\alpha}}$. As such, we initialize $\mathcal{U}_h^{\boldsymbol{\alpha}}$ to the current choice of $V_h^{\boldsymbol{\alpha}}$. In this work, we always base $\mathcal{U}_h^{\boldsymbol{\alpha}}$ on $V_h^{\boldsymbol{\alpha}}$ or a (possibly repeated) uniform h-refinement thereof. However, for more flexibility, we briefly recapitulate possible feature-based refinement strategies.

In all cases, plausible a priori (aPr) strategies refine elements that are too large (on $\Omega_h^{\boldsymbol{\alpha}}$), while a posteriori (aPos) strategies depend on the underlying PDE problem. In the case of (45), aPos refinement can be based on the *strong residual norm*

$$m_{\mathrm{SR}} = \int_{\Omega_h^{\boldsymbol{\alpha}}} \left(\Delta u_h^{\boldsymbol{\alpha}} - \Delta f^{\boldsymbol{\alpha}}(\mathbf{x}_h^{\boldsymbol{\alpha}})\right)^2 \mathrm{d}S + \mathcal{F}(u_h^{\boldsymbol{\alpha}}|_{\partial\hat{\Omega}}), \tag{52}$$

where $\mathcal{F}(\cdot) : \mathcal{U}_h^{\boldsymbol{\alpha},\mathcal{B}} \to \mathbb{R}^+$ is a suitably-chosen penalty term that gauges how well the boundary condition is resolved by Nitsche's method. Note that (52) requires that the entries of $\mathbf{x}_h^{\boldsymbol{\alpha}}$ are elements from $C^2(\hat{\Omega})$, which is satisfied if we utilize bicubics with maximum regularity. Eq. (52) may then be decomposed into the basis function wise contributions (as in (50)) or serve as a cost function for *dual weighted residual* (DWR) based aPos refinement [42].

Alternatively, the *weak residual norm* $m_{\mathrm{WR}}$, with

$$m_{\mathrm{WR}} = \sum_i c_{\mathrm{WR,i}}^2 \quad \text{and} \quad c_{\mathrm{WR,i}} = B(u_h^{\boldsymbol{\alpha}}, \mathbf{x}_h^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}, \psi_i) \tag{53}$$

may be utilized. Here, the $\psi_i$ are taken from a space $\bar{\mathcal{U}} \supset \mathcal{U}_h^{\boldsymbol{\alpha}}$ that results from uniformly refining $\mathcal{U}_h^{\boldsymbol{\alpha}}$ in $p$ or $h$. For more details, we refer to [17].

Upon completion of an adequate state variable approximation $u_h^{\boldsymbol{\alpha}}$, we are in the position to assemble the tuple $\left(J_h^{\boldsymbol{\alpha}}, \mathrm{d}_{\boldsymbol{\alpha}} J_h^{\boldsymbol{\alpha}}\right)$ utilizing the principles from Section 5. All the required steps are summarized in Fig. 3.

In the following, we present the results of a computational approach for various values of $\mu$ (see Eq. (51)) and $u_{\mathrm{ref}}$, where $u_{\mathrm{ref}}$ refers to the number of aPr $h$-refinements of $\mathcal{U}_h^{\boldsymbol{\alpha}}$ with respect to $V_h^{\boldsymbol{\alpha}}$. For this, the procedure that corresponds to Fig. 3 has been passed to a SLSQP [43] routine. In all cases, we use the initial guess $\boldsymbol{\alpha}_0 = \mathbf{0}$.

**Table 1**
Tables showing $|\min J_h^{\boldsymbol{\alpha}} - \min J^{\boldsymbol{\alpha}}|$ (a) and the required number of iterations until convergence is reached (b) for various combinations of $(\mu, u_{\text{ref}})$.

(**a**) Table showing $|\min J_h^{\boldsymbol{\alpha}} - \min J^{\boldsymbol{\alpha}}|$.

| $\mu$ | $u_{\text{ref}}$ | | |
|---|---|---|---|
| | 0 | 1 | 2 |
| $10^{-2}$ | $8.2 \times 10^{-3}$ | $4.6 \times 10^{-4}$ | $7.7 \times 10^{-6}$ |
| $10^{-4}$ | $3.7 \times 10^{-3}$ | $1.6 \times 10^{-4}$ | $5.5 \times 10^{-6}$ |
| $10^{-6}$ | $2.7 \times 10^{-4}$ | $1.3 \times 10^{-5}$ | $5.2 \times 10^{-7}$ |

(**b**) #iterations required until convergence.

| $\mu$ | $u_{\text{ref}}$ | | |
|---|---|---|---|
| | 0 | 1 | 2 |
| $10^{-2}$ | 4 | 2 | 3 |
| $10^{-4}$ | 4 | 3 | 3 |
| $10^{-6}$ | 2 | 3 | 3 |

**Table 2**
Tables showing the average of the number of DOFs involved in computing $\mathbf{x}_h^{\boldsymbol{\alpha}}$ (a) and $u_h^{\boldsymbol{\alpha}}$ (b) (over all iterations) for various combinations of $(\mu, u_{\text{ref}})$.

(**a**) Average #DOFs for $\mathbf{x}_h^{\boldsymbol{\alpha}}$.

| $\mu$ | $u_{\text{ref}}$ | | |
|---|---|---|---|
| | 0 | 1 | 2 |
| $10^{-2}$ | 483.5 | 389 | 452 |
| $10^{-4}$ | 735.5 | 676 | 676 |
| $10^{-6}$ | 1145 | 1460 | 1460 |

(**b**) Average #DOFs for $u_h^{\boldsymbol{\alpha}}$.

| $\mu$ | $u_{\text{ref}}$ | | |
|---|---|---|---|
| | 0 | 1 | 2 |
| $10^{-2}$ | 241.75 | 625 | 2641 |
| $10^{-4}$ | 367.75 | 1169 | 4337 |
| $10^{-6}$ | 572.5 | 2737 | 10 609 |

**Table 3**
Tables showing $|\min J_h^{\boldsymbol{\alpha}} - \min J^{\boldsymbol{\alpha}}|$ (a) and the required number of iterations until convergence is reached (b) in the SBA case for various combinations of $(n_e, u_{\text{ref}})$.

(**a**) Table showing $|\min J_h^{\boldsymbol{\alpha}} - \min J^{\boldsymbol{\alpha}}|$.

| $n_e$ | $u_{\text{ref}}$ | | |
|---|---|---|---|
| | 0 | 1 | 2 |
| 12 | $8.4 \times 10^{-3}$ | $5.4 \times 10^{-4}$ | $6.5 \times 10^{-6}$ |
| 16 | $7.2 \times 10^{-3}$ | $4.1 \times 10^{-4}$ | $1.1 \times 10^{-5}$ |
| 23 | $4.7 \times 10^{-3}$ | $2.4 \times 10^{-4}$ | $8.3 \times 10^{-6}$ |

(**b**) #iterations required until convergence.

| $n_e$ | $u_{\text{ref}}$ | | |
|---|---|---|---|
| | 0 | 1 | 2 |
| 12 | 4 | 3 | 2 |
| 16 | 4 | 3 | 2 |
| 23 | 4 | 3 | 2 |

Table 1 shows the discrepancy between the exact objective function minimum and its numerical approximation $|\min J_h^{\boldsymbol{\alpha}} - \min J^{\boldsymbol{\alpha}}|$ (a) and the required number of iterations until convergence is reached (b), for all possible combinations of $\mu = (10^{-2}, 10^{-4}, 10^{-6})$ and $u_{\text{ref}} = (0, 1, 2)$. In all cases, we initialized $\left[V_h^{\boldsymbol{\alpha}}\right]$ to a bicubic B-spline basis with 7 elements per coordinate direction and maximum regularity. Finally, Tables 3 and 4 show the corresponding results from a static basis approach. Hereby, $n_e$ denotes the number of elements we used per coordinate direction. Their values have been carefully selected to yield roughly the same number of DOFs associated with both $\mathbf{x}_h^{\boldsymbol{\alpha}}$ and $u_h^{\boldsymbol{\alpha}}$ as the average number of DOFs in the VBA-results (see Table 2).

Table 1(a) clearly demonstrates the consistency of the scheme, whereby discrepancies as low as $\sim 0.5 \times 10^{-6}$ are achieved. Comparing the VBA to the SBA results, Tables 1(a) and 3(a) demonstrate that VBA outperforms SBA in terms of accuracy, where an up to $\sim$ 10-fold error reduction of VBA over SBA can be observed. The total number of iterations required until convergence is achieved is comparable for VBA and SBA and never exceeds the number of four iterations.

### 6.2. Designing a cooling element

We are considering the design of a cooling element of dimension $\Delta x = 2$ and $\Delta y = 1$. In this example, there are four active coolers whose positions can slide in the direction tangential to $\partial\Omega^{\boldsymbol{\alpha}}$ and to a lesser extend in the normal direction (see Fig. 4). Further degrees of freedom are their radii $R_i$. Hence, the state vector is given by $\boldsymbol{\alpha} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, R_1, R_2, R_3, R_4)$, which is comprised of 12 DOFs. The surface cooling rate for the $i$th active

**Table 4**
Tables showing the number of DOFs involved in computing $\mathbf{x}_h^{\boldsymbol{\alpha}}$ (a) and $u_h^{\boldsymbol{\alpha}}$ (b) in the SBA case for various combinations of $(n_e, u_{\text{ref}})$.

(**a**) #DOFs for $\mathbf{x}_h^{\boldsymbol{\alpha}}$.

| $n_e$ | $u_{\text{ref}}$ | | |
|---|---|---|---|
| | 0 | 1 | 2 |
| 12 | 450 | 450 | 450 |
| 16 | 722 | 722 | 722 |
| 23 | 1352 | 1352 | 1352 |

(**b**) #DOFs for $u_h^{\boldsymbol{\alpha}}$.

| $n_e$ | $u_{\text{ref}}$ | | |
|---|---|---|---|
| | 0 | 1 | 2 |
| 12 | 225 | 729 | 2601 |
| 16 | 361 | 1225 | 4489 |
| 23 | 676 | 2401 | 9025 |

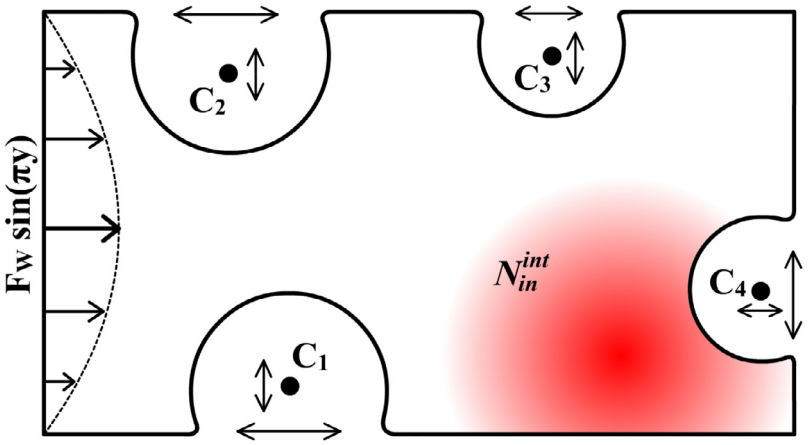


**Fig. 4.** The cooling element design template. Here, the centers of the active coolers are depicted by small black dots. Their positions constitute degrees of freedom in the design space, as well as their radii.

cooler $\mathcal{C}_i$ reads:

$$h^i_-(\mathbf{x}) = \frac{1}{20}\frac{R_i^3}{\|\mathbf{x}-\mathbf{x}_i\|^2}\left(u_h^{\boldsymbol{\alpha}}(\mathbf{x}) - T_\infty\right), \tag{54}$$

where $T_\infty = 0$ denotes the ambient temperature. A heat source delivers a constant heat influx given by

$$N_{\text{tot}} = N_{\text{in}}^W + N_{\text{in}}^{\text{int}}, \tag{55}$$

where $N_{\text{in}}^W$ denotes the influx at the western boundary, while $N_{\text{in}}^{\text{int}}$ denotes the influx delivered directly to the cooling element through an additional source term which satisfies

$$N_{\text{in}}^{\text{int}} = \int_{\Omega^{\boldsymbol{\alpha}}} A \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}_0\|^2}{2\sigma^2}\right)\mathrm{d}S, \tag{56}$$

where $A = \frac{N_{\text{tot}}}{4\pi\sigma^2}$, $\mathbf{x}_0 = (1.5, 0.25)^T$ and $\sigma = 0.1$. Note that changing the domain of integration from $\Omega^{\boldsymbol{\alpha}}$ to $\mathbb{R}^2$ in (56) yields a value of $N_{\text{in}}^{\text{int}} = N_{\text{tot}}/2$. As $N_{\text{tot}}$ is a constant quantity, we necessarily have $N_{\text{in}}^W = N_{\text{tot}} - N_{\text{in}}^{\text{int}}$. The surface heat flux density $h_W : \Gamma_W^{\boldsymbol{\alpha}} \to \mathbb{R}$ at the western boundary $\Gamma_W^{\boldsymbol{\alpha}} \subset \partial\Omega^{\boldsymbol{\alpha}}$ is of the form $h_W(y) = F_W(\Omega^{\boldsymbol{\alpha}})\sin(\pi y)$. Therefore, we have

$$N_{\text{in}}^W = N_{\text{tot}} - N_{\text{in}}^{\text{int}} = \int_{\Gamma_W^{\boldsymbol{\alpha}}} F_W \sin(\pi y)\mathrm{d}\gamma. \tag{57}$$

Hence

$$F_W(\Omega^{\boldsymbol{\alpha}}) = \frac{N_{\text{tot}}\pi}{2}\left(1 - \int_{\Omega^{\boldsymbol{\alpha}}} \frac{1}{4\pi\sigma^2}\exp\left(-\frac{\|\mathbf{x}-\mathbf{x}_0\|^2}{2\sigma^2}\right)\mathrm{d}S\right). \tag{58}$$

The relationship between $u^{\boldsymbol{\alpha}}$ and the (uniform) temperature of the heat source $T^{\boldsymbol{\alpha}}$ reads:

$$N_{\text{tot}} = A_1(\Omega^{\boldsymbol{\alpha}}) \int_{\Gamma_W^{\boldsymbol{\alpha}}} (T^{\boldsymbol{\alpha}} - u^{\boldsymbol{\alpha}}) \sin(\pi y) \mathrm{d}\gamma + A_2(\Omega^{\boldsymbol{\alpha}}) \int_{\Omega^{\boldsymbol{\alpha}}} (T^{\boldsymbol{\alpha}} - u^{\boldsymbol{\alpha}}) \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_0\|^2}{2\sigma^2}\right) \mathrm{d}S, \tag{59}$$

where

$$A_1(\Omega^{\boldsymbol{\alpha}}) = \frac{\pi}{2}\left(1 - \frac{W(\Omega^{\boldsymbol{\alpha}})}{4\pi\sigma^2}\right) \quad \text{and} \quad A_2(\Omega^{\boldsymbol{\alpha}}) = \frac{W(\Omega^{\boldsymbol{\alpha}})}{8\pi^2\sigma^4}, \tag{60}$$

with

$$W(\Omega^{\boldsymbol{\alpha}}) = \int_{\Omega^{\boldsymbol{\alpha}}} \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_0\|^2}{2\sigma^2}\right) \mathrm{d}S. \tag{61}$$

Inverting (59) gives:

$$T^{\boldsymbol{\alpha}}(u^{\boldsymbol{\alpha}}, \Omega^{\boldsymbol{\alpha}}) = \frac{N_{\text{tot}} + A_1(\Omega^{\boldsymbol{\alpha}}) \int_{\Gamma_W^{\boldsymbol{\alpha}}} u^{\boldsymbol{\alpha}} \sin(\pi y) \mathrm{d}\gamma + A_2(\Omega^{\boldsymbol{\alpha}}) \int_{\Omega^{\boldsymbol{\alpha}}} u^{\boldsymbol{\alpha}} \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}_0\|^2}{2\sigma^2}\right) \mathrm{d}S}{\frac{2}{\pi} A_1(\Omega^{\boldsymbol{\alpha}}) + W(\Omega^{\boldsymbol{\alpha}}) A_2(\Omega^{\boldsymbol{\alpha}})}. \tag{62}$$

**Remark.** The rationale behind $A_1(\Omega^{\boldsymbol{\alpha}})$ and $A_2(\Omega^{\boldsymbol{\alpha}})$ in (62) can be understood as follows: given $N_{\text{tot}} = 1$, suppose the cooling element width were to be contracted to $\Delta x \to 0$. We have $\lim_{\Delta x \to 0} W(\Omega^{\boldsymbol{\alpha}}) = 0$. In the limit, the temperature of the heat source should be fully determined by the first term on the right hand side of (59), which is the case because $\lim_{\Delta x \to 0}(A_1, A_2) = (\frac{\pi}{2}, 0)$. As such, a constant influx of $1 = N_{\text{tot}} = N_{\text{in}}^W$ means

$$T^{\boldsymbol{\alpha}} - u^{\boldsymbol{\alpha}}|_{\Gamma_W^{\boldsymbol{\alpha}}} = 1.$$

Conversely, suppose $\Omega^{\boldsymbol{\alpha}}$ were to be replaced by $\mathbb{R}^2$. Then, the dependency is divided equally among both terms since for

$$W(\Omega^{\boldsymbol{\alpha}} = \mathbb{R}^2) = 2\pi\sigma^2, \quad \text{we have} \quad (A_1, A_2) = \left(\frac{\pi}{4}, \frac{1}{4\pi\sigma^2}\right).$$

So, for $T^{\boldsymbol{\alpha}} - u^{\boldsymbol{\alpha}} = 1$, both terms contribute the same factor of $\frac{1}{2}$ to the right hand side of (59).

The weak state equation is based on the following PDE-problem:

$$\begin{aligned} -d\Delta u^{\boldsymbol{\alpha}} &= -f u^{\boldsymbol{\alpha}} + A \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_0\|^2}{2\sigma^2}\right) \quad \text{in } \Omega^{\boldsymbol{\alpha}} \\ \text{s.t.} \quad d \left.\frac{\partial u^{\boldsymbol{\alpha}}}{\partial \mathbf{n}}\right|_{\partial\Omega^{\boldsymbol{\alpha}}} &= \begin{cases} -h_{\text{cooling}} + F_W \sin(\pi y) & \mathbf{x} \in \bar{\Gamma}_W^{\boldsymbol{\alpha}} \\ -h_{\text{cooling}} & \mathbf{x} \in \partial\Omega^{\boldsymbol{\alpha}} \setminus \bar{\Gamma}_W^{\boldsymbol{\alpha}} \end{cases}, \end{aligned} \tag{63}$$

where

$$h_{\text{cooling}} = \sum_{i=1}^{4} h_-^i \quad \text{and} \quad f = 10^{-3} \quad \text{denotes the internal dissipation rate.} \tag{64}$$

The $i$th entry of the discretized weak state equation residual reads:

$$\begin{aligned} \left(\mathbf{B}_h^{\boldsymbol{\alpha}}\right)_i &= d\left(\nabla u_h^{\boldsymbol{\alpha}}, \nabla\phi_i\right)_{\Omega_h^{\boldsymbol{\alpha}}} + \int_{\Omega_h^{\boldsymbol{\alpha}}} f u^{\boldsymbol{\alpha}} \phi_i \mathrm{d}S \\ &+ \int_{\Omega_h^{\boldsymbol{\alpha}}} A \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_0\|^2}{2\sigma^2}\right) \left(\frac{\pi}{2}\bar{\phi}_i - \phi_i\right) \mathrm{d}S + \sum_{j=1}^{4} \int_{\partial\Omega_h^{\boldsymbol{\alpha}}} \phi_i h_-^j \mathrm{d}\gamma - \frac{\pi}{2} N_{\text{tot}} \bar{\phi}_i, \end{aligned} \tag{65}$$

with $A$ as in (56), $d = 0.8$,

$$\bar{\phi}_i = \int_{\Gamma_W^{\boldsymbol{\alpha}}} \phi_i \sin(\pi y) \mathrm{d}\gamma \quad \text{and} \quad \phi_i \in \left[\mathcal{U}_h^{\boldsymbol{\alpha}}\right]. \tag{66}$$

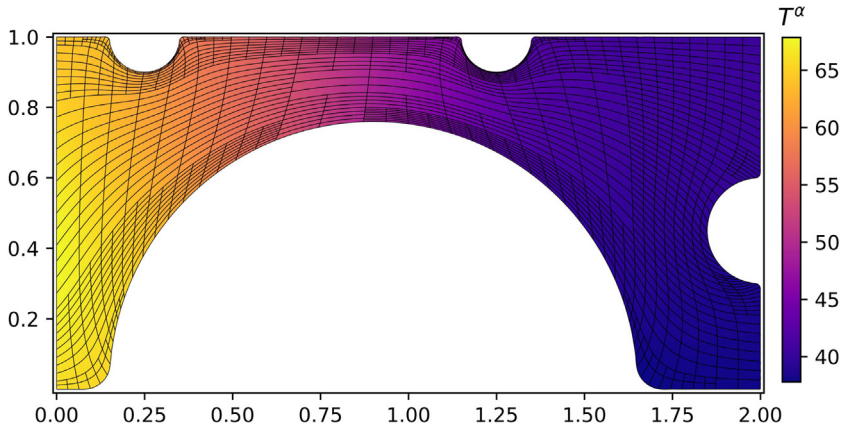


**Fig. 5.** The initial guess passed to the minimization routine.

We are minimizing the manufacturing costs of the cooling element such that the heat source temperature does not exceed the value of $T_{\max} = 80$. The problem reads:

$$\begin{aligned} & J\left(u^{\boldsymbol{\alpha}}, \Omega^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}\right) \rightarrow \min_{\boldsymbol{\alpha}} \\ \text{s.t.} \quad & T_{\max} - T^{\boldsymbol{\alpha}} \geq 0 \\ & \boldsymbol{\alpha} \in \boldsymbol{\lambda}, \end{aligned} \tag{67}$$

where

$$J\left(u^{\boldsymbol{\alpha}}, \Omega^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}\right) = \int_{\Omega^{\boldsymbol{\alpha}}} 1 \mathrm{d}S + \sum_{i=1}^{4} C_{\mathrm{CE}} R_i^2, \quad \text{with} \quad C_{\mathrm{CE}} = \frac{100}{\pi}. \tag{68}$$

Furthermore, the feasible design space $\boldsymbol{\lambda}$ is the space of all $\boldsymbol{\alpha}$ such that the active coolers do not overlap and the genus of $\Omega^{\boldsymbol{\alpha}}$ does not change (allowing for shape optimization without topology changes). This leads to a total of 30 (partly nonlinear) inequalities.

A major challenge is deciding where to place the active coolers and what radii to use. Increasing the radius means additional cooling but also additional manufacturing costs and decreased cooling element area, decreasing the heat capacity and the channel heat conductivity. Furthermore, placing a cooler close to the internal heat source (see (56)) reduces the amount of internal influx, increasing the influx amplitude $F_W$ (see (57)) at $\Gamma_W^{\boldsymbol{\alpha}}$ for compensation.

We are considering the case $N_{\mathrm{tot}} = 10$ and follow the same approach as in Section 6.1 with $\mu = 0.5 \times 10^{-3}$ and $u_{\mathrm{ref}} = 1$. Since $\mathbf{x}^{\boldsymbol{\alpha}}$ is a continuous function of the input state vector, we improve the efficiency by storing the tuples $(\boldsymbol{\alpha}^i, \mathbf{c}_{\mathcal{A}}^i, \boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha},i})$ (see Section 5) after each iteration. Whenever some $\boldsymbol{\alpha}^i$ with $\|\boldsymbol{\alpha} - \boldsymbol{\alpha}^i\| < \epsilon$ is found in the database, the corresponding mapping $\mathbf{x}_h^{\boldsymbol{\alpha},i} \in \boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha},i}$ is prolonged to the coarsest element segmentation of $\hat{\Omega}$ that is compatible with both $\boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha},i}$ and the current $\boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha}}$. Upon completion, it is restricted to $\boldsymbol{\mathcal{V}}_h^{\boldsymbol{\alpha}}$, which yields the vector $\mathbf{c}_{\mathcal{A}}^R = \left(\mathbf{c}_{\mathcal{B}}^R, \mathbf{c}_{\mathcal{I}}^R\right)^T$. The weights corresponding to the inner DOFs, $\mathbf{c}_{\mathcal{I}}^R$, are extracted and then used as an initial guess for the root-finding problem (13). We have noticed this to lead to a tremendous speedup, in particular during the last iterations, in which $\boldsymbol{\alpha}$ varies only slightly. Hereby, the required number of iterations is reduced from typically four to as few as one.

**Remark.** This principle may be extended to higher than zeroth-order database interpolation.

Here, we use $\epsilon = 0.05$. A feasible initial guess is created by picking one of the coolers and increasing its radius until $T^{\boldsymbol{\alpha}} < T_{\max}$. The initial design is depicted in Fig. 5. As in Section 6.1, the routine that computes $J_h^{\boldsymbol{\alpha}}$, $\mathrm{d}_{\boldsymbol{\alpha}} J_h^{\boldsymbol{\alpha}}$ and the constraints is passed to an SLSQP optimizer.

Figs. 6(a) to 6(d) show the cooling element after 4, 7, 10 and 13 iterations. Convergence is reached after 15 iterations and the corresponding design is depicted in Fig. 7. The final design reduces the manufacturing costs from the initial $J_h^{\boldsymbol{\alpha}} = 10.66$ to $J_h^{\boldsymbol{\alpha}} = 6.29$.

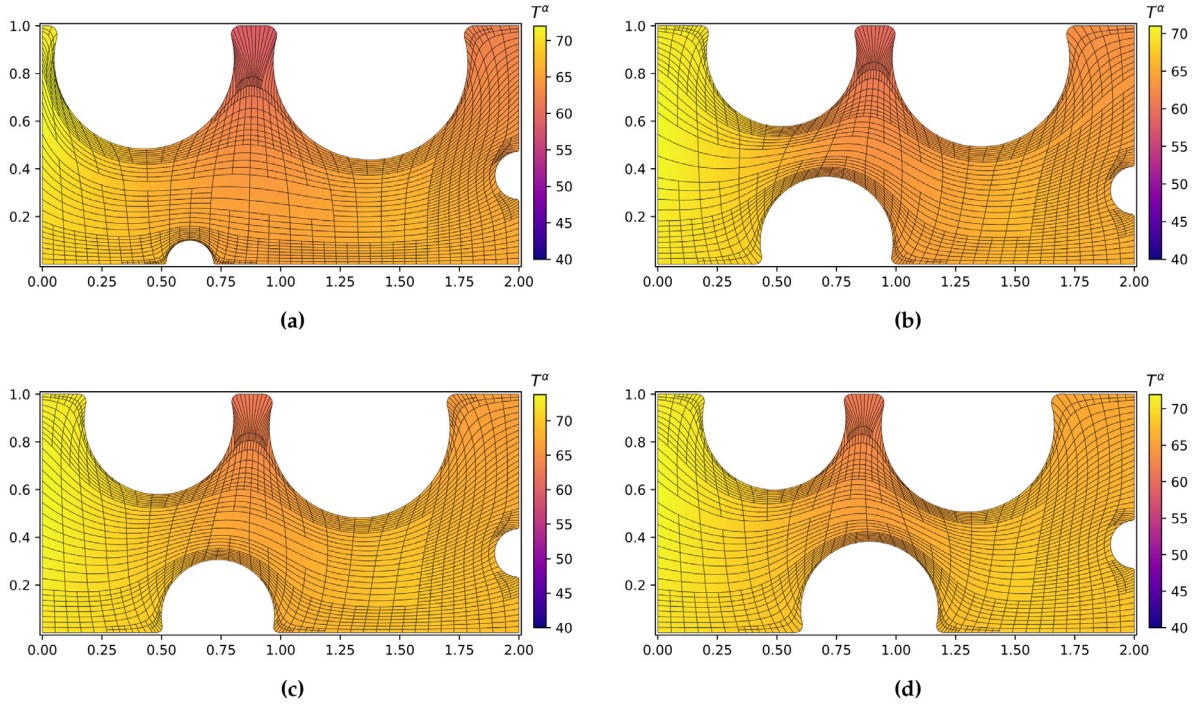


**Fig. 6.** The cooling element after 4(a), 7(b), 10(c) and 13(d) iterations.

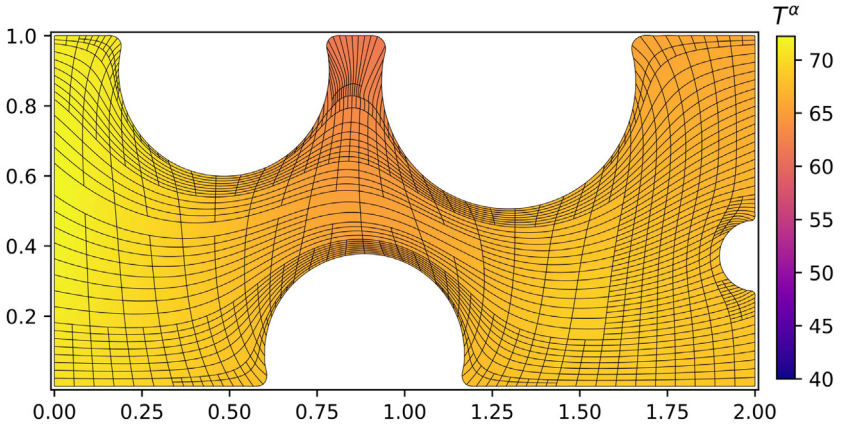


**Fig. 7.** The final cooling element design after 15 iterations.

A striking difference between the initial and all intermediate designs is the improved heat conductivity within the channel, leading to a more homogeneous temperature (and one that is higher on average). This is not surprising. As the cooling efficiency is linear in the difference between the temperature at the boundaries and the ambient temperature $T_\infty = 0$, a higher average temperature implies higher average cooling efficiency.

The final design places a modestly-sized cooler $\mathcal{C}_1$ at the center of the southern boundary and a similarly-sized cooler $\mathcal{C}_2$ at the western part of the northern boundary. To its right, a slightly larger cooler $\mathcal{C}_3$ is placed while a small cooler $\mathcal{C}_4$ is placed at eastern boundary close to $y = 0.4$.

Compared to the initial design, one big cooler has been replaced by several modestly-sized ones, improving the channel heat conductivity and by that the cooling cost efficiency. The slightly larger size of $\mathcal{C}_3$ compared to $\mathcal{C}_2$ can be explained by the internal heat source centered at $\mathbf{x}_0 = (1.5, 0.25)^T$. The small radius of $\mathcal{C}_4$ may be explained by

the fact that increasing its size reduces the amount of internal influx area, leading to a larger influx at $\Gamma_W^{\boldsymbol{\alpha}}$ instead. As such, we regard the final design as plausible, adding more credibility to the proposed numerical scheme.

## 7. Conclusions

In this manuscript, we proposed an IGA-based shape optimization algorithm in which the parameterization is added to the problem formulation in the form of an additional PDE-constraint. This has enabled us to derive a fully symbolical expression for the gradient of the objective function, allowing for gradient-based optimization. The discretization of the equations has been accomplished with the so-called *variable basis approach* (VBA) in which a new THB-spline basis is chosen during each iteration based on the current requirements, such as accurately resolving the geometry contours and particular features of the state equation solution. This leads to a highly flexible scheme in which folding due to numerical truncation is automatically repaired through THB-enabled local refinement.

We have tested the scheme by applying it to two examples. In the first example, we compared the numerical solution to the known exact solution and concluded that the scheme is consistent. Comparing the VBA-approach to an approach in which the basis is taken static (SBA) furthermore revealed that VBA-enabled feature-based refinement leads to a $\sim$10-fold error reduction over SBA at a comparable total number of DOFs. This discrepancy may be further increased by employing more proficient a priori and a posteriori refinement techniques. In the second example, we considered the design of a cooling element. Unlike in the first example, the exact minimizer was unknown, however, the optimization routine converged to a design that we consider plausible. In both cases, the scheme succeeded in fully automatically parameterizing a wide range of geometries which would be too complex for other symbolically-differentiable parameterization strategies (such as Coon's Patch) at the expense of leading to a nonlinear problem.

Finally, we briefly discussed possible memory-saving strategies for large-scale optimization and (possible) future implementations of the scheme with support for volumetric applications. Furthermore, the scheme is straightforwardly enhanced to support multipatch parameterizations by adopting the mixed FEM EGG algorithm introduced in [33].

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

The authors gratefully acknowledge the research funding which was partly provided by the MOTOR project that has received funding from the European Unions Horizon 2020 research and innovation program under grant agreement no. 678727.

The research visit of Mr. A. Jaeschke at Delft University of Technology that allowed this cooperation was funded by The Polish National Agency for Academic Exchange under grant agreement no. PPN/IWA/2018/1/00032.

## References

[1] T.J.R. Hughes, J.A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, CMAME 194 (2005) 4135–4195.

[2] Z. Kacprzyk, K. Ostapska-Łuczkowska, Isogeometric analysis as a new FEM formulation-simple problems of steady state thermal analysis, Procedia Eng. 91 (2014) 87–92.

[3] J.A. Cottrell, A. Reali, Y. Bazilevs, T.J. Hughes, Isogeometric analysis of structural vibrations, Comput. Methods Appl. Mech. Engrg. 195 (41–43) (2006) 5257–5296.

[4] Y. Bazilevs, T. Hughes, NURBS-based isogeometric analysis for the computation of flows about rotating components, Comput. Mech. 43 (1) (2008) 143–150.

[5] A. Jaeschke, Isogeomeric Analysis for Compressible Flows with Application in Turbomachinery (Master's thesis), TU Delft, 2015.

[6] W.A. Wall, M.A. Frenzel, C. Cyron, Isogeometric structural shape optimization, Comput. Methods Appl. Mech. Engrg. 197 (33–40) (2008) 2976–2988.

[7] X. Qian, Full analytical sensitivities in NURBS based isogeometric shape optimization, Comput. Methods Appl. Mech. Engrg. 199 (29–32) (2010) 2059–2071, http://dx.doi.org/10.1016/j.cma.2010.03.005.

[8] Y. Wang, Z. Wang, Z. Xia, L. Poh, Structural design optimization using isogeometric analysis: A comprehensive review, CMES - Comput. Model. Eng. Sci. 117 (3) (2018) 455–507, http://dx.doi.org/10.31614/cmes.2018.04603.

[9] N.D. Manh, A. Evgrafov, A.R. Gersborg, J. Gravesen, Isogeometric shape optimization of vibrating membranes, Comput. Methods Appl. Mech. Engrg. 200 (13–16) (2011) 1343–1353.
[10] P. Nørtoft, J. Gravesen, Isogeometric shape optimization in fluid mechanics, Struct. Multidiscip. Optim. 48 (5) (2013) 909–925.
[11] B.-U. Park, Y.-D. Seo, O. Sigmund, S.-K. Youn, Shape optimization of the stokes flow problem based on isogeometric analysis, Struct. Multidiscip. Optim. 48 (5) (2013) 965–977.
[12] M. Yoon, S.-H. Ha, S. Cho, Isogeometric shape design optimization of heat conduction problems, Int. J. Heat Mass Transfer 62 (2013) 272–285.
[13] S.A. Sauter, C. Schwab, Boundary element methods, in: Boundary Element Methods, Springer, 2010, pp. 183–287.
[14] K. Kostas, A. Ginnis, C. Politis, P. Kaklis, Ship-hull shape optimization with a t-spline based bem-isogeometric solver, Comput. Methods Appl. Mech. Engrg. 284 (2015) 611–622, http://dx.doi.org/10.1016/j.cma.2014.10.030.
[15] G. Xu, B. Mourrain, R. Duvigneau, A. Galligo, Optimal analysis-aware parameterization of computational domain in isogeometric analysis, in: International Conference on Geometric Modeling and Processing, Springer, 2010, pp. 236–254.
[16] G. Farin, D. Hansford, Discrete coons patches, Comput. Aided Geom. Design 16 (7) (1999) 691–700.
[17] J. Gravesen, A. Evgrafov, D.-M. Nguyen, P. Nørtoft, Planar parametrization in isogeometric analysis, in: International Conference on Mathematical Methods for Curves and Surfaces, Springer, 2012, pp. 189–212.
[18] G. Xu, B. Mourrain, R. Duvigneau, A. Galligo, Parameterization of computational domain in isogeometric analysis: methods and comparison, Comput. Methods Appl. Mech. Engrg. 200 (23–24) (2011) 2021–2031.
[19] A. Falini, J. Špeh, B. Jüttler, Planar domain parameterization with THB-splines, Comput. Aided Geom. Design 35 (2015) 95–108.
[20] T. Radó, Aufgabe 41, Jahresber. Dtsch. Math.-Ver. 35 (1926) 49.
[21] H. Kneser, Losung der aufgabe 41, Jahresber. Dtsch. Meth. (1926) 123–124.
[22] T. Nguyen, B. Jüttler, Parameterization of contractible domains using sequences of harmonic maps, in: International Conference on Curves and Surfaces, Springer, 2010, pp. 501–514.
[23] B.N. Azarenok, Generation of structured difference grids in two-dimensional nonconvex domains using mappings, Comput. Math. Math. Phys. 49 (5) (2009) 797–809.
[24] J. Hinz, M. Möller, C. Vuik, Elliptic grid generation techniques in the framework of isogeometric analysis applications, Comput. Aided Geom. Design (2018).
[25] J.H. Holland, Genetic algorithms and adaptation, in: Adaptive Control of Ill-Defined Systems, Springer, 1984, pp. 317–333.
[26] A. Wächter, L.T. Biegler, On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming, Math. Program. 106 (1) (2006) 25–57.
[27] L.T. Biegler, V.M. Zavala, Large-scale nonlinear programming using ipopt: An integrating framework for enterprise-wide dynamic optimization, Comput. Chem. Eng. 33 (3) (2009) 575–582.
[28] C. Giannelli, B. Jüttler, H. Speleers, THB-splines: The truncated basis for hierarchical splines, Comput. Aided Geom. Design 29 (7) (2012) 485–498.
[29] J. Hinz, M. Abdelmalik, M. Möller, Goal-oriented adaptive THB-spline schemes for PDE-based planar parameterization, 2020, arXiv:2001.08874.
[30] A. Falini, B. Jüttler, THB-splines multi-patch parameterization for multiply-connected planar domains via template segmentation, J. Comput. Appl. Math. 349 (2019) 390–402.
[31] P. Duren, W. Hengartner, Harmonic mappings of multiply connected domains, Pacific J. Math. 180 (2) (1997) 201–220.
[32] G. Choquet, Sur un type de transformation analytique généralisant la représentation conforme et définie au moyen de fonctions harmoniques, Bull. Sci. Math. 69 (2) (1945) 156–165.
[33] J. Hinz, M. Möller, C. Vuik, An IGA framework for PDE-based planar parameterization on convex multipatch domains, 2019, arXiv preprint arXiv:1904.03009.
[34] L. Bottou, Large-scale machine learning with stochastic gradient descent, in: Proceedings of COMPSTAT'2010, Springer, 2010, pp. 177–186.
[35] A. Hansbo, P. Hansbo, An unfitted finite element method, based on nitsche's method, for elliptic interface problems, Comput. Methods Appl. Mech. Engrg. 191 (47–48) (2002) 5537–5552.
[36] S.G. Krantz, H.R. Parks, The Implicit Function Theorem: History, Theory, and Applications, Springer Science & Business Media, 2012.
[37] Y. Saad, M.H. Schultz, Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (3) (1986) 856–869.
[38] D.A. Knoll, D.E. Keyes, Jacobian-free newton–krylov methods: a survey of approaches and applications, J. Comput. Phys. 193 (2) (2004) 357–397.
[39] G. van Zwieten, J. van Zwieten, C. Verhoosel, E. Fonn, T. van Opstal, W. Hoitinga, Nutils, 2019, http://dx.doi.org/10.5281/zenodo.3243447. URL https://doi.org/10.5281/zenodo.3243447.
[40] E. van Brummelen, T. Demont, G. van Zwieten, An adaptive isogeometric analysis approach to elasto-capillary fluid-solid interaction, Internat. J. Numer. Methods Engrg. (2020).
[41] A. Charakhch'yan, S. Ivanenko, A variational form of the winslow grid generator, J. Comput. Phys. 136 (2) (1997) 385–398.
[42] R. Rannacher, Adaptive finite element methods in flow computations, in: Recent Advances in Adaptive Computation, Vol. 383, Contemporary Mathematics, 2004, pp. 176–183.
[43] D. Kraft, A software package for sequential quadratic programming, in: Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt Fur Luft- und Raumfahrt, 1988.