



HAL
open science

Elasticity-based mesh morphing technique with application to reduced-order modeling

Abbas Kabalan, Fabien Casenave, Felipe Bordeu, Virginie Ehrlacher,
Alexandre Ern

► **To cite this version:**

Abbas Kabalan, Fabien Casenave, Felipe Bordeu, Virginie Ehrlacher, Alexandre Ern. Elasticity-based mesh morphing technique with application to reduced-order modeling. 2024. hal-04718345

HAL Id: hal-04718345

<https://hal.science/hal-04718345v1>

Preprint submitted on 2 Oct 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Elasticity-based mesh morphing technique with application to reduced-order modeling

A. Kabalan^{1,2}, F. Casenave², F. Bordeu², V. Ehrlacher^{1,3}, A. Ern^{1,3}

¹ Cermics, Ecole nationale des ponts et chaussées, 6-8 Av. Blaise Pascal, Champs-sur-Marne, 77455 Marne-la-Vallée cedex 2, FRANCE,

² Safran Tech, Digital Sciences & Technologies, Magny-Les-Hameaux, 78114, FRANCE,

³ Inria Paris, 48 rue Barrault, CS 61534, 75647 Paris cedex, FRANCE.

Abstract The aim of this article is to introduce a new methodology for constructing morphings between shapes that have identical topology. The morphings are obtained by deforming a reference shape, through the resolution of a sequence of linear elasticity equations, onto every target shape. In particular, our approach does not assume any knowledge of a boundary parametrization. Furthermore, we demonstrate how constraints can be imposed on specific points, lines and surfaces in the reference domain to ensure alignment with their counterparts in the target domain after morphing. Additionally, we show how the proposed methodology can be integrated in an offline and online paradigm, which is useful in reduced-order modeling involving variable shapes. This framework facilitates the efficient computation of the morphings in various geometric configurations, thus improving the versatility and applicability of the approach. The methodology is illustrated on the regression problem of the drag and lift coefficients of airfoils of non-parameterized variable shapes.

1 Introduction

1.1 Background

Solving parametric partial differential equations (PDEs) for various values of parameters in a given set is a common task in industrial contexts. Examples of sets of parameters include initial and boundary values, coefficients in the PDE of interest or geometrical parameters of the domain where the PDE is posed. When the evaluation of the PDE solution is computationally expensive, model-order reduction techniques offer an efficient tool to speed up computations while maintaining accuracy.

A common situation encountered in reduced-order modeling is the following: Given a set of parameter values $\mathcal{P} \subset \mathbb{R}^p$ for some $p \in \mathbb{N}^*$, and a physical domain $\Omega_0 \subset \mathbb{R}^d$ for some $d = 2, 3$, one is interested in quickly computing an approximation of the solution $u_\mu : \Omega_0 \rightarrow \mathbb{R}$ for all $\mu \in \mathcal{P}$ of a given parametric PDE of the form $\mathcal{A}_\mu(u_\mu) = 0$ on Ω_0 with \mathcal{A}_μ some parameter-dependent differential operator, together with appropriate initial/boundary conditions. Then, the reduced-basis method [1, 2] involves constructing a low-dimensional approximation space $\mathcal{Z}_r = \text{span}\{\xi_1, \dots, \xi_r\}$ of the solution set $\mathcal{U} = \{u_\mu : \mu \in \mathcal{P}\}$, and then compute an approximation of u_μ belonging to \mathcal{Z}_r , for instance as a Galerkin approximation of the parametric PDE, enabling faster solution computations. In practice, efficient reduced-order modeling techniques employ a two-phase procedure. First one performs the offline phase, where the PDE $\mathcal{A}_\mu(u_\mu) = 0$ is solved for u_μ for some values of the parameter $\mu \in \mathcal{M}$ using the computationally expensive high-fidelity model (HFM); here, \mathcal{M} is a selected training set. Subsequently, the reduced space \mathcal{Z}_r can be constructed through approximation algorithms such as the Proper Orthogonal Decomposition (POD) [3, 4] or greedy approaches [1]. The online phase, also known as the exploitation phase, consists in computing approximations of the solution of the PDE belonging to \mathcal{Z}_r for new parameter values. This phase leverages the precomputed reduced-order basis to efficiently compute these approximations. Depending on the complexity of the PDE and its parameter dependence, more advanced strategies such as hyper-reduction [5] may be required.

The present work deals with the case where the physical domain also depends on the value of the parameter $\mu \in \mathcal{P}$. More precisely, for all $\mu \in \mathcal{P}$, we now consider $\Omega_\mu \subset \mathbb{R}^d$ to be some domain which may depend on μ , and assume that the solution of the parametric PDE is now a function $u_\mu : \Omega_\mu \rightarrow \mathbb{R}$. In such a situation, standard algorithms such as POD are not directly applicable, since the solutions u_μ are defined on different domains. The most common solution in the literature on reduced-order modeling with geometric variabilities is to find an appropriate morphing ϕ_μ from (or to) a reference geometry Ω_0 to (or from) each parametric domain Ω_μ . In this scenario, the problem can be reformulated on the reference domain Ω_0 and reduced-order modeling techniques are applied to the transformed solution set $\{u_\mu \circ \phi_\mu : \mu \in \mathcal{P}\}$. Such a task is often called a registration problem. Registration problems are also of interest when the domain does not depend on the parameter to achieve efficiency of the reduced-order modeling technique. We refer the reader to [6, 7] for some seminal works in this setting.

1.2 Related works on morphing techniques

The difficulty now lies on the efficient construction of a morphing $\phi : \Omega_0 \rightarrow \Omega$ from a reference domain $\Omega_0 \subset \mathbb{R}^d$ to a target domain $\Omega \subset \mathbb{R}^d$ that captures the target geometry accurately. Early works on model-order reduction with geometrical variability adopted the use of affine mappings [8]. However, this approach cannot be applied to general domains with curved boundaries and edges. Other commonly used techniques in computational physics to deform a geometry (or a mesh) onto another are free-form deformation (FFD) [9], radial basis function (RBF) interpolation [10], linear elasticity/harmonic mesh morphings [11], nonlinear elasticity [12, 13], only to cite a few. Numerous contributions adopted those strategies in reduced-order modeling contexts [14, 15, 16, 17]. However, all these strategies share the assumption that the geometries are parameterized and that the deformation of the nodes on the boundary is known. This way, the displacement of the nodes on $\partial\Omega_0$ can be imposed to map onto $\partial\Omega$, and the extension of the deformation to the whole domain can be determined by means of the chosen method.

In many scenarios, however, an explicit parametrization of the geometry is not available, especially in the online phase. In this situation, constructing a suitable morphing $\phi_\mu : \Omega_0 \rightarrow \Omega_\mu$ becomes more challenging. One possibility often advocated in the literature is to find the deformation of the boundary $\phi_\mu(\partial\Omega_0)$, and then leverage the knowledge of $\phi_\mu(\partial\Omega_0)$ to compute $\phi_\mu(\Omega_0)$, by using either RBF interpolation [18, 19], mesh parametrization [18], geometry registration [20, 21, 22], optimal transport [23, 24] or some other technique. These approaches require the computation of the boundary morphing before calculating the volume morphing. Another class of methods, such as the LDDMM (Large Deformation Diffeomorphic Metric Mapping) [25], proposes to find the morphing from Ω_0 to Ω_μ as a flow of diffeomorphisms solving optimal control problems. These methods are usually expensive to compute, and are often used to calculate only $\phi_\mu(\partial\Omega_0)$ [26].

1.3 Contribution

The contributions of the paper are the following. First, we propose a method for finding a morphing from a reference domain Ω_0 to a target domain Ω_μ without a priori knowledge on the boundary parametrization. Starting from Ω_0 , the algorithm produces a sequence of morphisms $(\phi^{(m)})_{m \geq 0}$ defined on Ω_0 such that $\phi^{(0)} = \mathbf{Id}|_{\Omega_0}$, where \mathbf{Id} denotes the identity mapping from \mathbb{R}^d onto \mathbb{R}^d . For all $m \in \mathbb{N}$, denoting by $\Omega^{(m)} = \phi^{(m)}(\Omega_0)$, the morphing is updated at iteration $m + 1$ as

$$\phi^{(m+1)} = \left(\mathbf{Id}|_{\Omega_0} + \gamma^{(m)} \mathbf{u}^{(m)} \right) \circ \phi^{(m)}, \quad (1)$$

where $\mathbf{u}^{(m)} : \Omega^{(m)} \rightarrow \mathbb{R}^d$ is the solution of a linear elasticity problem posed on $\Omega^{(m)}$ and $\gamma^{(m)} > 0$ is a user-dependent parameter expected to be small. Notice that (1) may be seen as a time-discretization scheme associated with the evolution equation

$$\partial_t \phi(t) = \mathbf{u}(t) \circ \phi(t),$$

where $\mathbf{u}(t) : \mathbb{R}^d \rightarrow \mathbb{R}^d$ is a time-dependent velocity field. In the linear elasticity problem at iteration m , external forces are applied on the boundary of the current domain $\Omega^{(m)}$ to ensure that the new domain $\Omega^{(m+1)}$ is closer in a certain sense to the target domain Ω . This approach shares similarities with [27] but differs in the type of linear elasticity problems that are solved. One advantage of the present method is that one can impose certain geometrical features, such as points, lines or surfaces in Ω_0 , to be mapped onto some a priori chosen counterparts in Ω .

The second main contribution is to embed the above morphing technique in a reduced-order modeling context. Given a collection of domains $\{\Omega_i\}_{1 \leq i \leq n}$ for some $n \in \mathbb{N}^*$ which forms the training set, we compute morphisms $\phi_i : \Omega_0 \rightarrow \Omega_i$ in an offline phase using the algorithm proposed above. Then, we propose an efficient online reduced-order model to quickly compute a morphing from the reference domain Ω_0 onto a new target domain outside the training set. The efficiency of the approach is strongly linked to the use of an appropriate initial guess used as a starting point in the iterative online procedure. Finally, we provide numerical evidence that the method produces accurate results when employed in regression-based model-order reduction techniques.

1.4 Motivating example

We present in this section an example which motivates the interest of the present methodology in the context of reduced-order modeling.

Let $d = 2, 3$ and let $\{\Omega_i\}_{1 \leq i \leq n} \subset \mathbb{R}^d$ be a collection of domains in \mathbb{R}^d , where a domain in \mathbb{R}^d is understood as an open bounded connected subset of \mathbb{R}^d with piecewise smooth boundary. Assume that all the domains share the same topology. Let $\Omega_0 \subset \mathbb{R}^d$ be a fixed reference domain that shares the same topology as well. The collection $\{\Omega_i\}_{1 \leq i \leq n}$ is referred to as the training set of target domains.

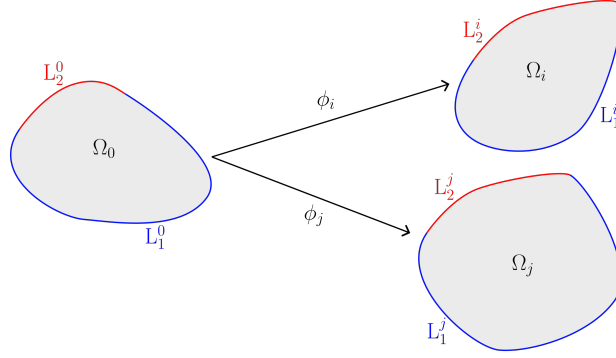


Figure 1: Reference domain Ω_0 with two samples Ω_i and Ω_j from the target dataset.

Assume now that one is actually interested in solving, for all $i \in \{1, \dots, n\}$, the following parametric (elliptic) PDE with mixed boundary conditions:

$$\begin{cases} \mathcal{L}_{\mu_i}(u_{\mu_i}) = 0 & \text{in } \Omega_i, \\ u_{\mu_i} = a & \text{on } L_1^i, \\ \nabla u_{\mu_i} \cdot \mathbf{n}_i = b & \text{on } L_2^i, \end{cases}$$

where \mathbf{n}_i is the unit normal outward vector to Ω_i , $\mu_i \in \mathcal{M}$ belongs to the set of parameter values, $u_{\mu_i} : \Omega_i \rightarrow \mathbb{R}$ is the solution to the PDE problem of interest, $a, b \in \mathbb{R}$, and L_1^i and L_2^i are open subsets of $\partial\Omega_i$ which form a partition of $\partial\Omega_i$. In other words, Dirichlet boundary conditions are enforced on L_1^i , whereas Neumann boundary conditions are enforced on L_2^i . Moreover, one wishes to construct a reduced-order modeling technique to quickly obtain numerical approximations of the problems above.

Since, for all $1 \leq i \leq n$, each solution u_{μ_i} is defined on a different domain, traditional dimensionality reduction methods such as POD are not directly applicable. One possibility is to rely on so-called registration methods to find a morphing $\phi_i : \Omega_0 \rightarrow \Omega_i$, and then apply POD on the family of functions $\{u_{\mu_i} \circ \phi_i\}_{1 \leq i \leq n}$. Moreover, we want to ensure that $\phi_i(L_1^0) = L_1^i$ and $\phi_i(L_2^0) = L_2^i$, with $\partial\Omega_0 = L_1^0 \cup L_2^0$. In this case, the boundary conditions $u_{\mu_i} \circ \phi_i|_{L_1^0} = a$ and $(\nabla u_{\mu_i} \cdot \mathbf{n}_i) \circ \phi_i|_{L_2^0} = b$ are satisfied, and the dimensionality reduction problem is expected to be simpler.

1.5 Outline of the paper

In Section 2, we present the (offline) methodology to construct a morphing from a reference domain Ω_0 onto a target domain Ω while respecting certain conditions on the morphing of the boundary. In Section 3, we show how, given a training dataset of geometries $\{\Omega_i\}_{1 \leq i \leq n}$, we can reduce the complexity of the problem of finding a morphing for a given domain outside the training dataset, so that the method can be efficient in the online phase. In both sections, we provide numerical examples to illustrate the behavior of the proposed methods. In Section 4, we present an application of the proposed morphing strategy to learn scalar outputs from simulations realized on different (non-parameterized) geometries. As an example, we predict the drag coefficient of airfoils of non-parameterized variable shapes, while underlining the advantages of the present method with respect to the MMGP method presented in [18]. In Section 5, we provide a brief summary and some concluding remarks.

2 High-fidelity morphing construction

In this section, we present the new high-fidelity methodology to construct a morphism $\phi : \Omega_0 \rightarrow \Omega$ between a reference domain $\Omega_0 \subset \mathbb{R}^d$ and a target domain $\Omega \subset \mathbb{R}^d$. In the context of model-order reduction with geometric variability, this approach is applied in the offline phase (see section 3). In what follows, we denote by $\|\cdot\|$ the Euclidean norm of \mathbb{R}^d . Moreover, we use boldface notation for vector in \mathbb{R}^d , fields taking values in \mathbb{R}^d , and sets and linear spaces composed of such fields.

2.1 Notation and preliminaries

Let Ω_0 and Ω be domains of \mathbb{R}^d . For the sake of simplicity, we mainly present the methodology in the case $d = 2$. We refer the reader to Section 2.6 for remarks about the extension to the case $d = 3$.

Let $N_p, N_l \in \mathbb{N}^*$. Let $\{\mathbf{P}_1, \dots, \mathbf{P}_{N_p}\} \subset \partial\Omega$ be a collection of N_p distinct points of $\partial\Omega$, and $\{L_1, \dots, L_{N_l}\} \subset \partial\Omega$ be a collection of disjoint, open, connected subdomains of $\partial\Omega$ with positive 1-dimensional Hausdorff measure such that $\bigcup_{k_l=1}^{N_l} \overline{L_{k_l}} = \partial\Omega$, where $\overline{L_{k_l}}$ denotes the closure of L_{k_l} . Similarly, we consider a collection $\{\mathbf{P}_1^0, \dots, \mathbf{P}_{N_p}^0\} \subset \partial\Omega_0$ of N_p distinct points of $\partial\Omega_0$ and a collection $\{L_1^0, \dots, L_{N_l}^0\} \subset \partial\Omega_0$ of disjoint, open, connected subdomains of $\partial\Omega_0$ with

positive 1-dimensional Hausdorff measure such that $\bigcup_{k_l=1}^{N_l} \overline{L_{k_l}^0} = \partial\Omega_0$. Our goal is to build a morphing such that each line L_k^0 (resp., point \mathbf{P}_k^0) in $\partial\Omega_0$ is mapped to the corresponding line L_k (resp., point \mathbf{P}_k) in $\partial\Omega$.

Let us introduce the set $\mathcal{T}_{\Omega_0} := \{\phi \in \mathbf{W}^{1,\infty}(\Omega_0), \phi^{-1} \in \mathbf{W}^{1,\infty}(\phi(\Omega_0)) : \phi \text{ is injective}\}$. We wish to find a morphing $\phi \in \mathcal{T}_{\Omega_0}$ such that

$$\phi(\Omega_0) = \Omega, \quad (2a)$$

$$\phi(\mathbf{P}_{k_p}^0) = \mathbf{P}_{k_p}, \quad \forall 1 \leq k_p \leq N_p, \quad (2b)$$

$$\phi(L_{k_l}^0) = L_{k_l}, \quad \forall 1 \leq k_l \leq N_l. \quad (2c)$$

Our aim here is to propose a new iterative method to construct a morphing $\phi \in \mathcal{T}_{\Omega_0}$ such that conditions (2a)-(2b)-(2c) are satisfied at convergence. The rest of the section is organized as follows. First, in Section 2.2, we first propose a new approach, inspired from [27], to construct a morphing satisfying only the requirement (2a). In Section 2.3, we then show how to modify the approach to take into consideration (2b)-(2c) as well.

2.2 Shape matching without constraints

The aim of this section is to propose a new approach to compute a morphism $\phi \in \mathcal{T}_{\Omega_0}$ satisfying (2a). Our starting point is the approach introduced in [27] and we comment the differences between this approach and the present one. As in [27], our approach consists in reformulating the problem as an optimization problem and formulating the algorithm as a gradient descent. Let $g \in \mathbf{H}_{\text{loc}}^1(\mathbb{R}^d)$ to be a level set function for Ω , i.e., a function such that, for all $\mathbf{x} \in \mathbb{R}^d$,

$$\begin{cases} g(\mathbf{x}) < 0 & \text{if } \mathbf{x} \in \Omega, \\ g(\mathbf{x}) > 0 & \text{if } \mathbf{x} \in \overline{\Omega}^c, \\ 0 & \text{if } \mathbf{x} \in \partial\Omega. \end{cases} \quad (3)$$

Define the functional

$$J_g : \mathcal{T}_{\Omega_0} \ni \phi \longmapsto J_g(\phi) := \int_{\phi(\Omega_0)} g(\mathbf{x}) d\mathbf{x} \in \mathbb{R}. \quad (4)$$

Since g is a level set function, the set $\mathcal{T}^* := \{\phi \in \mathcal{T}_{\Omega_0} \mid \phi(\Omega_0) = \Omega\}$ coincides with the set of global minimizers of J_g over \mathcal{T}_{Ω_0} . Thus, in order to find a morphing from Ω_0 to Ω , we can consider the following optimization problem:

$$\text{Find } \phi^* \in \arg \min_{\phi \in \mathcal{T}_{\Omega_0}} J_g(\phi). \quad (5)$$

One example of level set function g , which is commonly used in practice, is the *signed distance function* defined as

$$d_{\Omega}(\mathbf{x}) = \begin{cases} -d(\mathbf{x}, \partial\Omega) & \text{if } \mathbf{x} \in \Omega, \\ d(\mathbf{x}, \partial\Omega) & \text{if } \mathbf{x} \in \overline{\Omega}^c, \\ 0 & \text{if } \mathbf{x} \in \partial\Omega, \end{cases} \quad (6)$$

where $d(\mathbf{x}, \partial\Omega)$ is the Euclidean distance of \mathbf{x} to $\partial\Omega$.

Let us collect some auxiliary mathematical results, most of which are classical, to justify the relevance of the proposed approach. For the sake of completeness, we recall some proofs. We start with a classical lemma (see [28, Lemma 6.13]).

Lemma 1. *Let $\phi \in \mathcal{T}_{\Omega_0}$. Define the set $\mathcal{T}'_{\phi,1} := \{\mathbf{v} \circ \phi \mid \mathbf{v} \in \mathbf{W}^{1,\infty}(\phi(\Omega_0)) \text{ and } \|\mathbf{v}\|_{\mathbf{W}^{1,\infty}(\phi(\Omega_0))} < 1\} \subset \mathbf{W}^{1,\infty}(\Omega_0)$. Then, for all $\xi \in \mathcal{T}'_{\phi,1}$, we have $\phi + \xi \in \mathcal{T}_{\Omega_0}$.*

Proof. Let $\phi \in \mathcal{T}_{\Omega_0}$, $\xi \in \mathcal{T}'_{\phi,1}$ such that $\xi = \mathbf{v} \circ \phi$ for some $\mathbf{v} \in \mathbf{W}^{1,\infty}(\phi(\Omega_0))$ with $\|\mathbf{v}\|_{\mathbf{W}^{1,\infty}(\mathbb{R}^d)} < 1$. Then, $\phi + \xi = \phi + \mathbf{v} \circ \phi = (\mathbf{Id} + \mathbf{v}) \circ \phi$. Since $\|\mathbf{v}\|_{\mathbf{W}^{1,\infty}(\mathbb{R}^d)} < 1$, $(\mathbf{Id} + \mathbf{v})$ is bijective from \mathbb{R}^d to \mathbb{R}^d . In particular, it is injective on $\phi(\Omega_0)$, so that $\phi + \xi = (\mathbf{Id} + \mathbf{v}) \circ \phi$ is injective, as the composition of two injective maps. Hence, $\phi + \xi \in \mathcal{T}_{\Omega_0}$. \square

This lemma proves in particular that the set $\mathcal{T}'_{\phi} := \{\mathbf{v} \circ \phi \mid \mathbf{v} \in \mathbf{W}^{1,\infty}(\mathbb{R}^d)\}$ is included in the tangential space of \mathcal{T}_{Ω_0} at point ϕ .

The following proposition is classical in topology optimization, see, e.g., [28, 29] for a proof.

Proposition 1. *Let $g \in \mathbf{H}_{\text{loc}}^1(\mathbb{R}^d)$, let $\phi \in \mathcal{T}_0$, $\psi \in \mathcal{T}'_{\phi}$, and let \mathbf{n}_{ϕ} be the outward unit normal vector to $\phi(\partial\Omega_0)$. Then the differential of J_g at ϕ evaluated in the direction ψ reads as*

$$DJ_g(\phi)(\psi) = \int_{\phi(\partial\Omega_0)} g(\mathbf{x}) \psi \circ \phi^{-1}(\mathbf{x}) \cdot \mathbf{n}_{\phi} ds. \quad (7)$$

We define, for all $\mathbf{v} \in \mathbf{W}^{1,\infty}(\phi(\Omega_0))$,

$$\widetilde{DJ}_g(\phi)(\mathbf{v}) = \int_{\phi(\partial\Omega_0)} g(\mathbf{x})\mathbf{v}(\mathbf{x}) \cdot \mathbf{n}_\phi ds,$$

so that

$$DJ_g(\phi)(\mathbf{v} \circ \phi) = \widetilde{DJ}_g(\phi)(\mathbf{v}).$$

The proof of the following lemma is immediate using the trace theorem and the fact that $g \in \mathbf{H}_{\text{loc}}^1(\mathbb{R}^d)$.

Lemma 2. *Let $\phi \in \mathcal{T}_{\Omega_0}$. The linear functional $\widetilde{DJ}_g(\phi) : \mathbf{W}^{1,\infty}(\phi(\Omega_0)) \rightarrow \mathbb{R}$ can be uniquely extended to a continuous linear form defined on the space $\mathbf{H}^1(\phi(\Omega_0))$.*

The following proposition is at the heart of the new method we propose in the present work.

Proposition 2. *Assume that $d = 2$. Let $\alpha > 0$ and let $\phi \in \mathcal{T}_{\Omega_0}$ be such that*

- $\phi(\Omega_0)$ is a domain of \mathbb{R}^2 ;
- the 1-dimensional Hausdorff measure of $\partial\phi(\Omega_0)$ is positive;
- for all $\mathbf{p} \in \mathbb{R}^2$ and all $r > 0$, $\phi(\Omega_0) \neq B(\mathbf{p}, r)$, where $B(\mathbf{p}, r)$ denotes the open ball of \mathbb{R}^2 with center \mathbf{p} and radius r .

For all $\mathbf{u} \in \mathbf{H}^1(\phi(\Omega_0))$, let $\varepsilon(\mathbf{u})$ and $\sigma(\mathbf{u})$ to be the linearized strain and stress tensors defined by

$$\begin{aligned} \varepsilon(\mathbf{u}) &:= \frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^T), \\ \sigma(\mathbf{u}) &:= \frac{E}{(1+\nu)}\varepsilon(\mathbf{u}) + \frac{E\nu}{(1+\nu)(1-\nu)}\text{Tr}(\varepsilon(\mathbf{u}))I, \end{aligned}$$

with $E > 0$ and $-1 < \nu < \frac{1}{2}$ are, respectively, called the Young modulus and the Poisson ratio. Then the bilinear form

$$a_\phi : \mathbf{H}^1(\phi(\Omega_0)) \times \mathbf{H}^1(\phi(\Omega_0)) \ni (\mathbf{u}, \mathbf{v}) \longmapsto a_\phi(\mathbf{u}, \mathbf{v}) := \int_{\phi(\Omega_0)} \sigma(\mathbf{u}) : \varepsilon(\mathbf{v}) d\mathbf{x} + \alpha \int_{\phi(\partial\Omega_0)} (\mathbf{u} \cdot \mathbf{n}_\phi)(\mathbf{v} \cdot \mathbf{n}_\phi) ds \quad (8)$$

defines an inner product on $\mathbf{H}^1(\phi(\Omega_0))$.

Proof. We only need to show the positive definiteness of a_ϕ . Let $\mathbf{u} \in \mathbf{H}^1(\phi(\Omega_0))$ be such that $a_\phi(\mathbf{u}, \mathbf{u}) = 0$. Let us prove that $\mathbf{u} = \mathbf{0}$. From the definition (8) of a_ϕ , we infer that $\varepsilon(\mathbf{u}) = 0$ in $\phi(\Omega_0)$ and $\mathbf{u} \cdot \mathbf{n}_\phi = 0$ on $\phi(\partial\Omega_0)$. Thus, since $\phi(\Omega_0)$ is connected, there exists $M \in \mathbb{R}^{2 \times 2}$ with $M^T = -M$ and $\mathbf{b} \in \mathbb{R}^2$ such that $\mathbf{u}(\mathbf{x}) = M\mathbf{x} + \mathbf{b}$, for all $\mathbf{x} \in \phi(\Omega_0)$.

Let us now prove that necessarily $M = 0$ and $\mathbf{b} = \mathbf{0}$. Reasoning by contradiction, let us first assume that $M \neq 0$. Then, there exists $m \in \mathbb{R} \setminus \{0\}$ and $\mathbf{y} = (y_1, y_2) \in \mathbb{R}^2$ such that for all $\mathbf{x} = (x_1, x_2) \in \phi(\Omega_0)$,

$$\mathbf{u}(\mathbf{x}) = m \begin{pmatrix} x_2 - y_2 \\ -(x_1 - y_1) \end{pmatrix}.$$

Since $\mathbf{u} \cdot \mathbf{n}_\phi = 0$ at $\phi(\partial\Omega_0)$, we inform that $\mathbf{n}_\phi(\mathbf{x}) = \pm \frac{(\mathbf{x}-\mathbf{y})}{\|\mathbf{x}-\mathbf{y}\|}$ for all $\mathbf{x} \in \phi(\partial\Omega_0)$ such that $\mathbf{x} \neq \mathbf{y}$. Since the boundary of $\phi(\Omega_0)$ is piecewise \mathcal{C}^1 , the following holds:

- Either $\mathbf{n}_\phi(\mathbf{x}) = \frac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|}$ for all $\mathbf{x} \in \phi(\partial\Omega_0)$ and hence $\phi(\Omega_0)$ has to be equal to $B(\mathbf{y}, r)$ for some $r > 0$, which is not possible by assumption;
- Or $\mathbf{n}_\phi(\mathbf{x}) = \frac{-(\mathbf{x}-\mathbf{y})}{\|\mathbf{x}-\mathbf{y}\|}$ for all $\mathbf{x} \in \phi(\partial\Omega_0)$ has to be equal to $\overline{B(\mathbf{y}, r)}^c$ for some $r > 0$, which cannot be since $\phi(\Omega_0)$ is a bounded set.

Hence, $M = 0$ and $\mathbf{u}(\mathbf{x}) = \mathbf{b}$ for all $\mathbf{x} \in \phi(\Omega_0)$. Thus, we obtain $\mathbf{b} \cdot \mathbf{n}_\phi = 0$ on $\phi(\partial\Omega_0)$ which is not possible if $\mathbf{b} \neq \mathbf{0}$ since $\phi(\partial\Omega_0)$ would then be a hyperplane orthogonal to \mathbf{b} . Hence, $\mathbf{b} = \mathbf{0}$, and this completes the proof. \square

The high-fidelity algorithm we propose to compute a morphism $\phi \in \mathcal{T}_{\Omega_0}$ satisfying (2a) is a particular gradient descent algorithm to minimize the functional J_g for some level set function $g \in \mathbf{H}_{\text{loc}}^1(\mathbb{R}^d)$ over \mathcal{T}_{Ω_0} , all the iterations being guaranteed to be well-defined when $d = 2$. More precisely, the algorithm is an iterative algorithm which computes a sequence of morphisms $(\phi^{(m)})_{m \geq 0}$ as follows. The starting point is $\phi^{(0)} = \text{Id}$. At iteration $m \in \mathbb{N}$, knowing $\phi^{(m)}$, the next iterate $\phi^{(m+1)}$ is computed as

$$\phi^{(m+1)} = (\text{Id} + \gamma^{(m)}\mathbf{u}^{(m)}) \circ \phi^{(m)}, \quad (9)$$

where $\gamma^{(m)}$ is some positive (small) constant and $\mathbf{u}^{(m)}$ is computed as follows: Given some finite-dimensional subspace $\mathbf{V}^{(m)} \subset \mathbf{W}^{1,\infty}(\Omega_0^{(m)})$ where $\Omega^{(m)} := \phi^{(m)}(\Omega_0)$, $\mathbf{u}^{(m)} \in \mathbf{V}^{(m)}$ is defined as the unique solution to

$$\forall \mathbf{v}^{(m)} \in \mathbf{V}^{(m)}, \quad a_{\phi^{(m)}}(\mathbf{u}^{(m)}, \mathbf{v}^{(m)}) = -\widetilde{D}J_g(\phi^{(m)})(\mathbf{v}^{(m)}). \quad (10)$$

In other words, $\mathbf{u}^{(m)} \in \mathbf{V}^{(m)}$ is the unique solution to the following linear elasticity problem:

$$\forall \mathbf{v}^{(m)} \in \mathbf{V}^{(m)}, \quad \int_{\Omega^{(m)}} \sigma(\mathbf{u}^{(m)}) : \varepsilon(\mathbf{v}^{(m)}) d\mathbf{x} + \alpha \int_{\Omega^{(m)}} (\mathbf{u}^{(m)} \cdot \mathbf{n}^{(m)})(\mathbf{v}^{(m)} \cdot \mathbf{n}^{(m)}) ds = - \int_{\Omega^{(m)}} g(\mathbf{x}) \mathbf{v}^{(m)}(\mathbf{x}) \cdot \mathbf{n}^{(m)} ds, \quad (11)$$

where $\mathbf{n}^{(m)}$ denotes the outward unit normal vector to $\Omega^{(m)}$. In particular, when the level set function g is chosen to be the distance function d_Ω , problem (11) reads as

$$\forall \mathbf{v}^{(m)} \in \mathbf{V}^{(m)}, \quad \int_{\Omega^{(m)}} \sigma(\mathbf{u}^{(m)}) : \varepsilon(\mathbf{v}^{(m)}) d\mathbf{x} + \alpha \int_{\Omega^{(m)}} (\mathbf{u}^{(m)} \cdot \mathbf{n}^{(m)})(\mathbf{v}^{(m)} \cdot \mathbf{n}^{(m)}) ds = - \int_{\Omega^{(m)}} d_\Omega(\mathbf{x}) \mathbf{v}^{(m)}(\mathbf{x}) \cdot \mathbf{n}^{(m)} ds. \quad (12)$$

In what follows, we refer to this procedure as the **signed distance algorithm**. An illustration is shown in Figure 2.

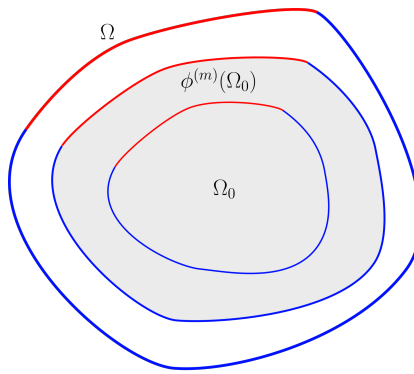


Figure 2: Example of reference domain Ω_0 , target domain Ω , and intermediate domain $\phi^{(m)}(\Omega_0)$.

Remark 1 (Comparison with [27]). *Let us comment on the differences between the approach we propose and the one in [27]. In [27], the authors consider a similar iterative algorithm with the difference that the gradient direction $\mathbf{u}^{(m)}$ is obtained as the solution of a problem of the form*

$$\forall \mathbf{v}^{(m)} \in \widetilde{\mathbf{V}}^{(m)}, \quad \int_{\Omega^{(m)}} \sigma(\mathbf{u}^{(m)}) : \varepsilon(\mathbf{v}^{(m)}) d\mathbf{x} = - \int_{\Omega^{(m)}} d_\Omega(\mathbf{x}) \mathbf{v}^{(m)}(\mathbf{x}) \cdot \mathbf{n}^{(m)} ds, \quad (13)$$

where $\widetilde{\mathbf{V}}^{(m)}$ is a finite-dimensional subspace of $\mathbf{H}_{0,\omega^{(m)}}^1(\Omega^{(m)}) := \{\mathbf{u} \in \mathbf{H}^1(\Omega^{(m)}) : \mathbf{u} = \mathbf{0} \text{ on } \omega^{(m)}\}$ and $\omega^{(m)} \subset \Omega^{(m)}$ is an open subdomain of $\Omega^{(m)}$ with positive measure. Our motivation for considering problems of the form (12) instead of problems of the form (13) is twofold:

- On the one hand, in the present approach, there is no need for the choice of a subdomain $\omega^{(m)}$ of $\Omega^{(m)}$, which in particular avoids to have null displacements into some arbitrarily chosen region of the domain $\Omega^{(m)}$;
- on the other hand, the second term on the left-hand side of (12) may be seen as a Tikhonov regularization term to select a solution $\mathbf{u}^{(m)}$ such that its normal component is as small as possible on the boundary of $\Omega^{(m)}$. This allows, in particular close to convergence, to allow tangential displacements along the boundary of the domain. This feature turns out to be particularly useful in our numerical tests.

Remark 2 (Parameter $\gamma^{(m)}$). *For simplicity, in our numerical tests, the sequence of parameters $(\gamma^{(m)})_{m \geq 0}$ is chosen to be equal to some constant value γ . Let us highlight, however, that, if for all $m \in \mathbb{N}$, we choose $\gamma^{(m)} \leq \min\left(\gamma_0, \frac{1}{2\|\mathbf{u}^{(m)}\|_{\mathbf{W}^{1,\infty}(\mathbb{R}^d)}}\right)$ for some $\gamma_0 > 0$, then it is guaranteed by Lemma 1 that $\phi^{(m)} \in \mathcal{T}_{\Omega_0}$ for all $m \in \mathbb{N}$.*

Remark 3 (Gradient descent). *The present procedure is a gradient descent algorithm for the resolution of the optimization problem (5). Indeed, $\boldsymbol{\psi}^{(m)} := \mathbf{u}^{(m)} \circ \phi^{(m)}$ is the unique solution of*

$$\forall \boldsymbol{\xi}^{(m)} \in \mathbf{V}_0^{(m)}, \quad c_{\phi^{(m)}}(\boldsymbol{\psi}^{(m)}, \boldsymbol{\xi}^{(m)}) = DJ_g(\phi^{(m)})(\boldsymbol{\xi}^{(m)}),$$

where for all $\phi \in \mathcal{T}_{\Omega_0}$,

$$c_\phi : \mathbf{H}^1(\Omega_0) \times \mathbf{H}^1(\Omega_0) \ni (\psi, \xi) \mapsto c_\phi(\psi, \xi) := a_\phi(\psi \circ \phi^{-1}, \xi \circ \phi^{-1})$$

and

$$\mathbf{V}_0^{(m)} := \left\{ \mathbf{v} \circ \phi^{(m)} : \mathbf{v} \in \mathbf{V}^{(m)} \right\} \subset \mathbf{H}^1(\Omega_0).$$

The function $\psi^{(m)} \in \mathcal{T}'_{\phi^{(m)}}$ is a gradient descent direction, computed with respect to the inner product $c_{\phi^{(m)}}$ on $\mathbf{V}_0^{(m)}$, and (9) amounts to update $\phi^{(m+1)}$ as $\phi^{(m+1)} = \phi^{(m)} + \gamma^{(m)}\psi^{(m)}$.

2.3 Shape matching with constraints

The goal of this section is to propose a variant of the iterative procedure presented in the previous section so as to enforce matching conditions concerning points and lines as in (2b)-(2c). The variant proposed here consists in computing at each iteration $m \in \mathbb{N}$ a displacement field $\mathbf{u}^{(m)} \in \mathbf{V}^{(m)}$ solution to

$$\forall \mathbf{v}^{(m)} \in \mathbf{V}^{(m)}, \quad a_{\phi^{(m)}}(\mathbf{u}^{(m)}, \mathbf{v}^{(m)}) = b_{\phi^{(m)}}(\mathbf{v}^{(m)}), \quad (14)$$

for some continuous linear functional $b_{\phi^{(m)}} : \mathbf{H}^1(\phi^{(m)}(\Omega_0)) \rightarrow \mathbb{R}$ encoding the constraints (2b)-(2c). The updated morphing $\phi^{(m+1)}$ is again defined by (9).

Let us focus more specifically on the case where $d = 2$ for the sake of clarity. Then, for all $\phi \in \mathcal{T}_{\Omega_0}$ and all $\mathbf{v} \in \mathbf{H}^1(\phi(\Omega_0))$, the quantity $b_\phi(\mathbf{v})$ is defined as the sum of two terms:

$$b_\phi(\mathbf{v}) = b_\phi^p(\mathbf{v}) + b_\phi^l(\mathbf{v}),$$

where b_ϕ^p (resp., b_ϕ^l) is a point-matching (resp., line-matching) linear form.

On the one-hand, the point-matching linear form is defined as follows. For all $1 \leq k_p \leq N_p$, we consider a neighborhood $N_{k_p}^0$ of $\mathbf{P}_{k_p}^0$ in $\partial\Omega_0$ (which is taken to be small), and define the following linear form on $\mathbf{H}^1(\phi(\Omega_0))$:

$$b_\phi^p(\mathbf{v}) := \beta_1 \sum_{k_p=1}^{N_p} \int_{\phi(N_{k_p}^0)} (\mathbf{P}_{k_p} - \phi(\mathbf{P}_{k_p}^0)) \cdot \mathbf{v} ds, \quad (15)$$

with the user-dependent parameter $\beta_1 > 0$. The aim of this term is to force each point $\mathbf{P}_{k_p}^0$ to match with its corresponding point \mathbf{P}_{k_p} at convergence of the scheme. Notice that $\phi(\mathbf{P}_{k_p}^0)$ is well defined since $\phi \in \mathbf{W}^{1,\infty}(\Omega_0)$.

On the other hand, the line-matching linear form is defined as follows. For any bounded closed subset $A \subset \mathbb{R}^2$, we denote by 1_A its characteristic function and $\Pi_A(\mathbf{x})$ denotes one minimizer of the following minimization problem:

$$\Pi_A(\mathbf{x}) \in \arg \min_{\mathbf{y} \in A} \|\mathbf{x} - \mathbf{y}\|. \quad (16)$$

Such an element is not uniquely defined in general, in which case one has to make a choice among all minimizers of (16). Then we define the *vector distance function* $\mathbf{D}_\phi^{\partial\Omega} : \phi(\partial\Omega_0) \rightarrow \mathbb{R}^2$ as follows:

$$\mathbf{D}_\phi^{\partial\Omega} : \phi(\partial\Omega_0) \ni \mathbf{x} \mapsto \mathbf{D}_\phi^{\partial\Omega}(\mathbf{x}) := \sum_{k_l=1}^{N_l} (\Pi_{L_{k_l}}(\mathbf{x}) - \mathbf{x}) 1_{\phi(L_{k_l}^0)}(\mathbf{x}) \in \mathbb{R}^2. \quad (17)$$

An illustration of $\mathbf{D}_\phi^{\partial\Omega}$ is shown in Figure 3. The linear form $b_\phi^l : \mathbf{H}^1(\phi(\Omega_0)) \rightarrow \mathbb{R}$ is then defined as follows:

$$\forall \mathbf{v} \in \mathbf{H}^1(\phi(\Omega_0)), \quad b_\phi^l(\mathbf{v}) := \beta_2 \int_{\phi(\partial\Omega_0)} (\mathbf{D}_\phi^{\partial\Omega} \cdot \mathbf{n}_\phi) (\mathbf{v} \cdot \mathbf{n}_\phi) ds = \beta_2 \sum_{i=1}^{N_l} \int_{\phi(L_i^0)} ((\Pi_{L_{k_l}} - \text{Id}) \cdot \mathbf{n}_\phi) (\mathbf{v} \cdot \mathbf{n}_\phi) ds, \quad (18)$$

for some user-dependent parameter $\beta_2 > 0$.

In what follows, we refer to this procedure as the **vector distance algorithm**.

Remark 4 (Alternative definition). *An alternative definition for b_ϕ^l is*

$$\forall \mathbf{v} \in \mathbf{H}^1(\phi(\Omega_0)), \quad b_\phi^l(\mathbf{v}) := \beta_2 \int_{\phi(\partial\Omega_0)} \mathbf{D}_\phi^{\partial\Omega} \cdot \mathbf{v} ds. \quad (19)$$

Numerical tests were also performed with this alternative definition. Altogether, the quality of the resulting transported mesh was observed to be better when using (18) rather than (19).

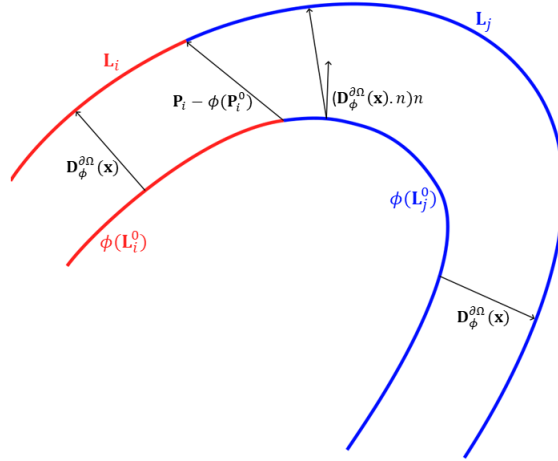


Figure 3: Visual representation of (17). $\mathbf{D}_{\phi}^{\partial\Omega}(\mathbf{x})$ is the vector that points from $\mathbf{x} \in \phi(L_i^0)$ to its projection on L_i .

2.4 Implementation details

The aim of this section is to give some details about the practical implementation of the procedures described in the two previous sections.

In the initialization step, assuming that Ω_0 is a polygonal domain, a conforming mesh \mathcal{M}_0 of the domain Ω_0 is chosen, typically employing simplicial mesh cells. At each iteration $m \in \mathbb{N}$, the mesh \mathcal{M}_0 is transformed into a mesh $\mathcal{M}^{(m)}$ of $\Omega^{(m)} = \phi^{(m)}(\Omega_0)$ via the diffeomorphism $\phi^{(m)}$. The finite-dimensional space $\mathbf{V}^{(m)}$ is then chosen at each iteration as the classical \mathbb{P}_1 finite-element space associated with the mesh $\mathcal{M}^{(m)}$. In principle, the transported mesh $\mathcal{M}^{(m)}$ could contain ill-shaped elements. In such a situation, one could potentially introduce a new mesh \mathcal{M}_m of $\Omega^{(m)}$. This was not needed in the numerical tests presented below.

Note that a full mesh of the target domain Ω is actually not required. However, we still need to use a boundary mesh of $\partial\Omega$, which is denoted by $\partial\mathcal{M}$, for the computation of the signed distance function d_{Ω} (see (6)) or the vector distance function $\mathbf{D}_{\phi^{(m)}}^{\partial\Omega}$ (see (17)).

For each vertex \mathbf{x} in the boundary mesh $\phi^{(m)}(\partial\mathcal{M}_0)$, we compute $d_{\Omega}(\mathbf{x})$ by determining the projection of \mathbf{x} onto $\partial\mathcal{M}$. This is achieved by determining the closest node on $\partial\mathcal{M}$ to \mathbf{x} (using a KD-tree structure for example), identifying boundary elements sharing this node (forming candidate elements), and projecting \mathbf{x} onto these elements. On the other hand, to compute $\mathbf{D}_{\phi^{(m)}}^{\partial\Omega}(\mathbf{x})$, we determine the index $1 \leq k_l \leq N_l$ such that $\mathbf{x} \in \phi^{(m)}(L_{k_l}^0) \subset \phi^{(m)}(\partial\mathcal{M}_0) \subset \mathbb{R}^2$, and compute the projection of \mathbf{x} onto $\overline{L_{k_l}}$. Notice that computing $\mathbf{D}_{\phi^{(m)}}^{\partial\Omega}(\mathbf{x})$ tends to be less costly from a computational point of view than computing the signed distance $d_{\Omega}(\mathbf{x})$, since we do not have to determine the position of \mathbf{x} relative to $\partial\Omega$ to determine the sign of $d_{\Omega}(\mathbf{x})$. Once the matching term is evaluated, we can compute $\mathbf{u}_{\Omega}^{(m)}$ by solving the variational problem (10) or (14).

The value of the parameter $\gamma^{(m)}$ can be adjusted throughout the iterations, however it must remain sufficiently small to ensure that $\phi^{(m+1)}$ belongs to \mathcal{T}_{Ω_0} (see Lemma 1). In the numerical tests presented below, the value $\gamma^{(m)}$ is chosen to be equal to some constant value $\gamma > 0$ which is specified below.

Let us introduce here two quantities that will be used to measure the geometric error and thus to assess the quality of a given morphing $\phi \in \mathcal{T}_{\Omega_0}$, and to define the stopping criterion of the two iterative algorithms we have presented in the previous sections. The first one is based on the use of the signed distance function (and is thus suitable to define a convergence criterion for the signed distance algorithm):

$$\Delta_1(\phi, \Omega_0, \Omega) := \sup\{d(\mathbf{x}, \partial\Omega) : \mathbf{x} \in \partial\phi(\Omega_0)\} = \sup\{|d_{\Omega}(\mathbf{x})| : \mathbf{x} \in \partial\phi(\Omega_0)\} = \|d_{\Omega}\|_{L^{\infty}(\partial\phi(\Omega_0))}. \quad (20)$$

The second one relies on the vector distance function (and is thus used for the definition of a convergence criterion for the vector distance algorithm):

$$\Delta_2(\phi, \Omega_0, \Omega) := \sup\{\|\mathbf{D}_{\phi}^{\partial\Omega}(\mathbf{x})\| : \mathbf{x} \in \partial\phi(\Omega_0)\} = \|\mathbf{D}_{\phi}^{\partial\Omega}\|_{L^{\infty}(\partial\phi(\Omega_0))}. \quad (21)$$

More precisely, given a stopping threshold $\epsilon > 0$, the iterative algorithms presented above are run until the first iteration $M \geq 0$ such that $\Delta_i(\phi^{(M)}) < \epsilon$ for $i = 1, 2$.

After convergence, we perform one final correction step, by computing a finite element approximation of the unique solution $\mathbf{u}^* \in \mathbf{H}^1(\phi^{(M)}(\Omega_0))$ of

$$\begin{cases} -\operatorname{div}(\sigma(\mathbf{u}^*)) = 0, & \text{in } \phi^{(M)}(\Omega_0), \\ \mathbf{u}^* = \mathbf{D}_{\phi^{(M)}}^{\partial\Omega}, & \text{on } \partial\phi^{(M)}(\Omega_0). \end{cases} \quad (22)$$

The final morphism is set to be $\phi^* := (\mathbf{Id} + \mathbf{u}^*) \circ \phi^{(M)}$. This guarantees that $\phi^*(\partial\Omega_0)$ coincides with $\partial\Omega$. An illustration of the output of this final correction step is presented in Figure 4.

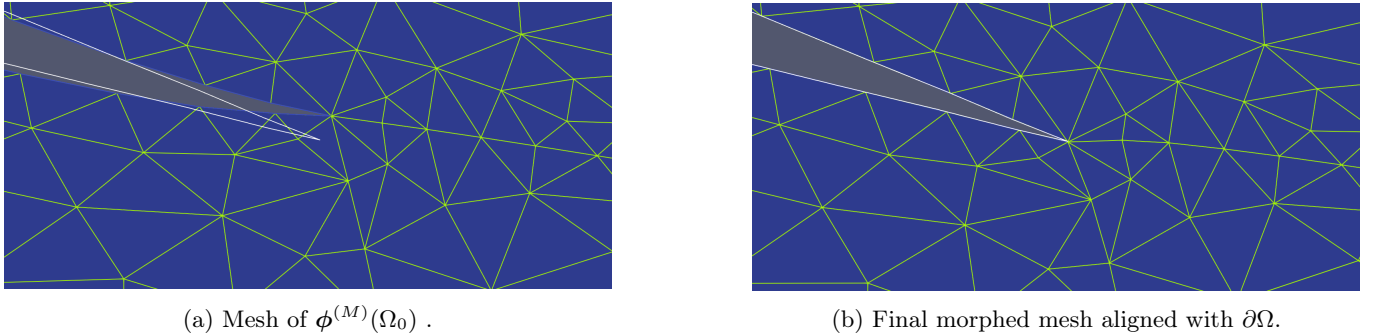


Figure 4: Illustration of the final correction step.

2.5 Numerical results

In this section, we present numerical results obtained with the procedures described in the previous sections on two two-dimensional test cases. All the results were obtained using the Muscat library [30].

2.5.1 Tensile2D dataset

The first example is taken from the dataset in [31]. For all $R > 0$, the set $B(R)$ is defined as $B(R) := \{(x, y) \in \mathbb{R}^2 / (x - 1)^2 + y^2 \leq R^2\} \cup \{(x, y) \in \mathbb{R}^2 / (x + 1)^2 + y^2 \leq R^2\}$. We consider the reference domain $\Omega_0 := [-1, 1]^2 \setminus B(0.5)$ and the target domain $\Omega := [-1, 1]^2 \setminus B(0.2)$, both shown in Figure 5a and 5b, respectively. We consider $N_p = 4$ control points in $\partial\Omega_0$ having coordinates $(-1, 0.5), (-1, -0.5), (1, 0.5), (-1, -0.5)$. We consider $N_l = 4$ control lines on $\partial\Omega_0$ which consist of the two half-circles (highlighted in red on the reference domain in Figure 5) and the top and bottom parts of the boundary (highlighted in green). In Figure 5c, we show a mesh of the reference domain Ω_0 superimposed with the boundary of the target domain Ω . The mesh has approximately 9000 elements.

The aim of the following tests is to highlight the advantages of the vector distance algorithm with respect to the distance function algorithm. The vector distance algorithm is run with the parameters $E := 1, \nu := 0.3, \alpha := 200, \gamma := 8, \beta_1 := 1$ and $\beta_2 := 0$. The convergence is obtained after 145 iterations with a tolerance $\epsilon = 10^{-3}$ and a stopping criterion based on Δ_2 . The evolution of the deformed mesh is shown in Figure 6. For comparison, we show in Figure 7 the evolution of the mesh using the signed distance algorithm, with the parameters $E := 1, \nu := 0.3, \alpha := 200, \gamma := 8$. The convergence is attained after 180 iterations with a stopping criterion based on Δ_1 and the same value of ϵ as above. When using the signed distance function, the half-circles in $\partial\Omega_0$ are not mapped onto the half-circles in $\partial\Omega$. Moreover, we observe that the mesh used for the reference domain needs to be sufficiently refined near the boundary for the algorithm to converge correctly, as already noted in [27]. In our implementation, the vector distance algorithm typically takes around 37 seconds to converge, whereas the signed distance algorithm takes around 71 seconds. As mentioned in Section 2.4, this is due to the fact that calculating the signed distance function is more expensive than calculating the vector distance function.

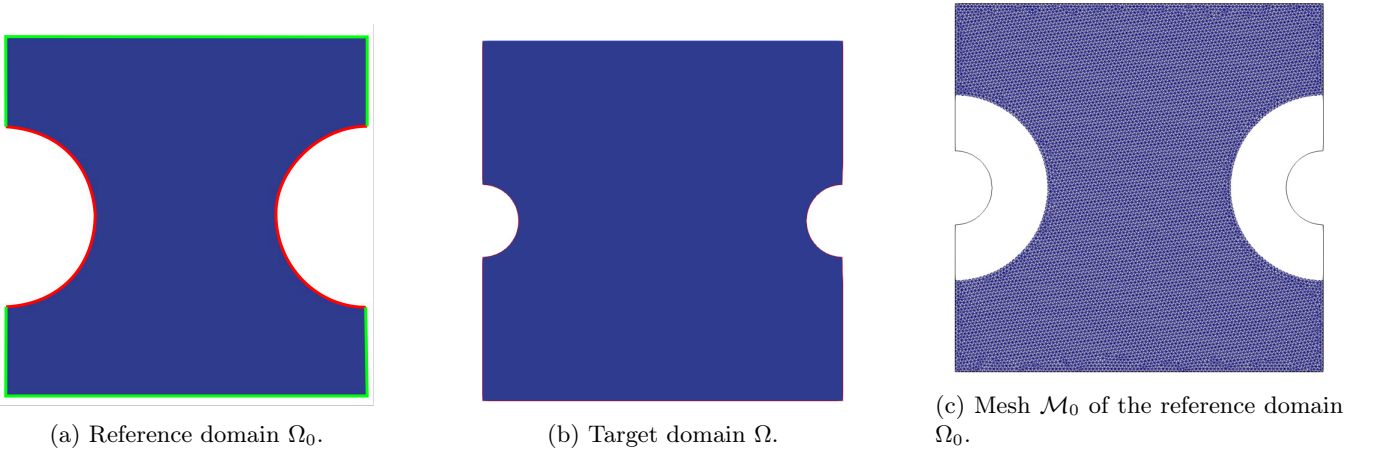


Figure 5: Reference and target domains, with the partition used on the boundary of the reference domain.

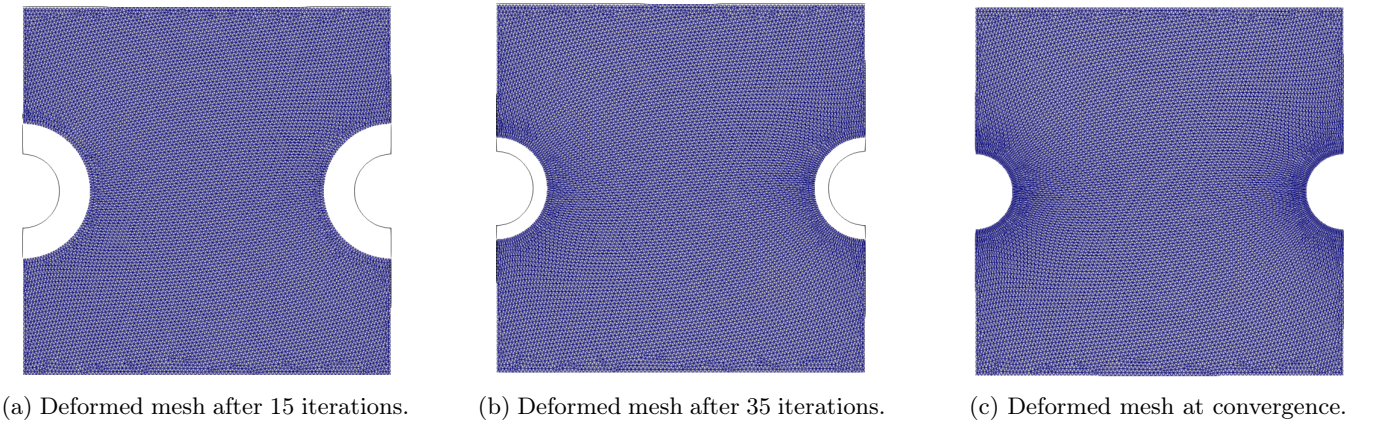


Figure 6: Evolution of the mesh using the vector distance algorithm.

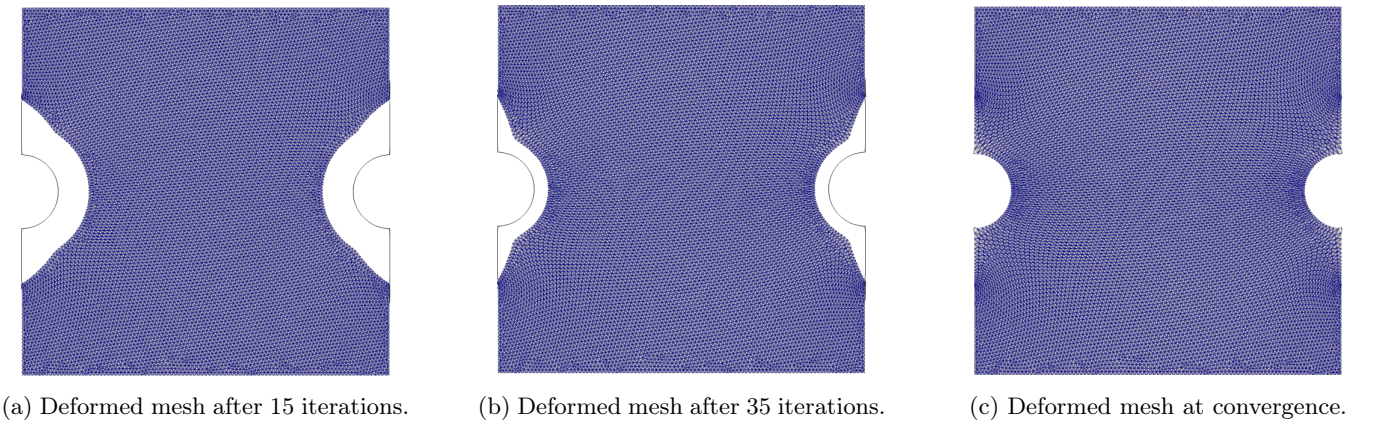
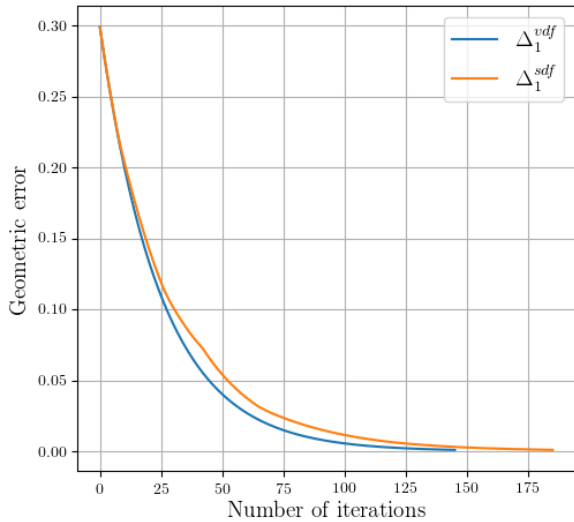
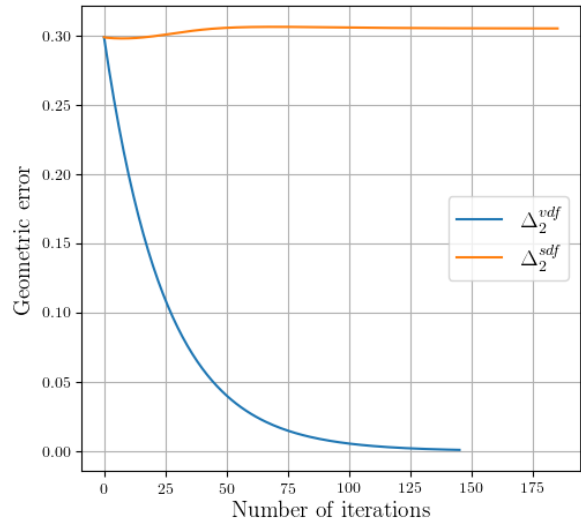


Figure 7: Evolution of the mesh using the signed distance algorithm.

Figure 8 illustrates the behaviour of the geometric error Δ_1 (a) and Δ_2 (b) as a function of the number of iterations of the chosen algorithm (signed distance function (*sdf*) or vector distance function (*vd*)). We observe that both quantities Δ_1^{sdf} and Δ_1^{vd} converge to 0 as the number of iterations increases. However, Δ_2^{vd} also converges to 0 with respect to the number of iterations, whereas Δ_2^{sdf} does not.



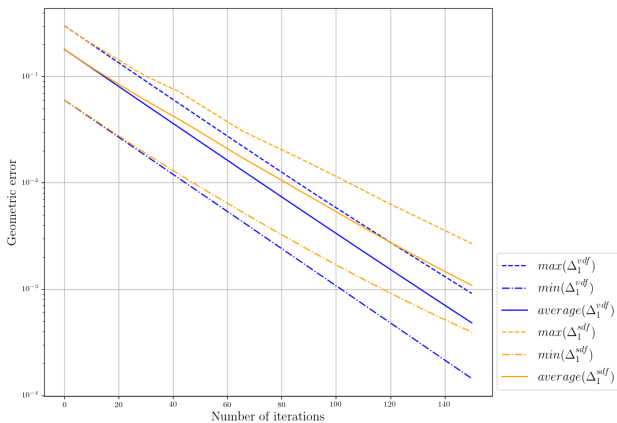
(a) Evolution of Δ_1 using both the signed distance algorithm and the vector distance algorithm.



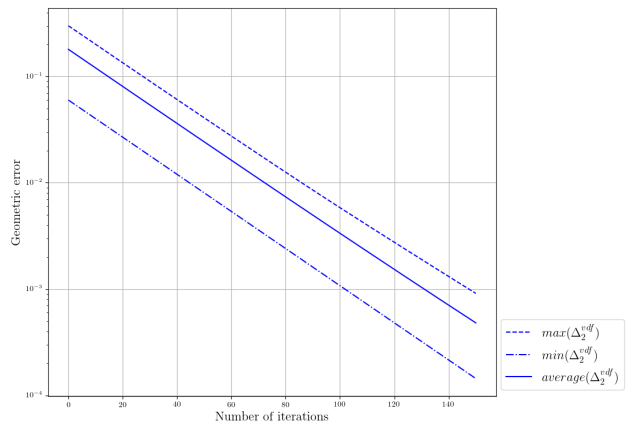
(b) Evolution of Δ_2 using the signed distance algorithm and the vector distance algorithm.

Figure 8: Evolution of Δ_1 and Δ_2 for the two algorithms for one of the samples. Using the vector distance algorithm (so that Δ_2 converges 0), we also have that Δ_1 converges to 0. This is not necessary the case when using the signed distance algorithm. We can have that Δ_1 converges to 0 without having Δ_2 converging to 0.

Figure 9 shows the average, minimum and maximum values of Δ_1 (a) and Δ_2 (b) as a function of the number of iterations of the algorithm. We observe that both quantities Δ_1^{vdf} and Δ_2^{vdf} converges exponentially to 0 with respect to the number of iterations, which is not the case of Δ_1^{sdf} . This highlights another advantage of the vector distance algorithm in comparison to the signed distance algorithm.



(a) Average, maximum and minimum error Δ_1 on a subset of 10 samples calculated using the two formulations.



(b) Average, maximum and minimum error Δ_2 on a subset of 10 samples calculated using the vector distance formulation.

Figure 9: Average, minimum and maximum geometric errors Δ_1 and Δ_2 on a subset of 10 samples.

2.5.2 AirfRANS dataset

In this second test, we consider 2D airfoils taken from the dataset in [32]. In this case, we observe that the signed distance algorithm does not always converge: this is the reason why we only present numerical results obtained with the vector function algorithm on this dataset.

We choose two samples, one as the reference and the other as the target domain. The morphing should map the lower wing surface (resp., the upper wing surface) of the airfoil Ω_0 to the lower wing surface (resp., the upper wing surface) of the airfoil Ω . We consider $N_p = 2$ target points, the two points being located at the leading edge $(0, 0)$ and the trailing edge $(1, 0)$ of the wing. At the start of the algorithm, these points coincide in the reference and the target geometries, but do not remain coincident through all the iterations. The external boundary representing the far field is fixed. The mesh of Ω_0 that is used to compute the morphing is different than the mesh provided in the

dataset. To alleviate the computational burden, we use a coarse shape-regular mesh of Ω_0 with approximately 8000 elements. Morphings that are computed on the coarse mesh can then be interpolated on the original finer meshes of the dataset (see Figure 11). Note, however, that this step may be delicate since the interpolation of the morphing may not preserve bijectivity in general, although we never encountered this issue in our numerical results. The parameters used for the simulations are $E := 0.1, \nu := 0.3, \alpha := 500, \gamma = 5, \beta_1 := 10$ and $\beta_2 := 1$. These parameters are chosen once and for all in order to calculate the morphings for all the geometries from the dataset. The value of the stopping criterion is chosen to be equal to $\epsilon = 5 \times 10^{-4}$. Convergence is obtained after 492 iterations, in about 92 seconds. The evolution of the deformation of the reference airfoil is shown in Figure 10 after 10, 25 and 492 iterations.

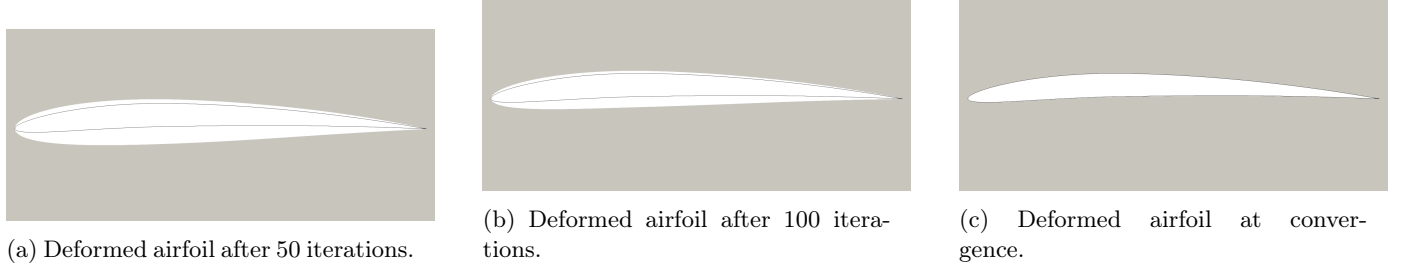


Figure 10: Evolution of the airfoil using the vector distance algorithm.

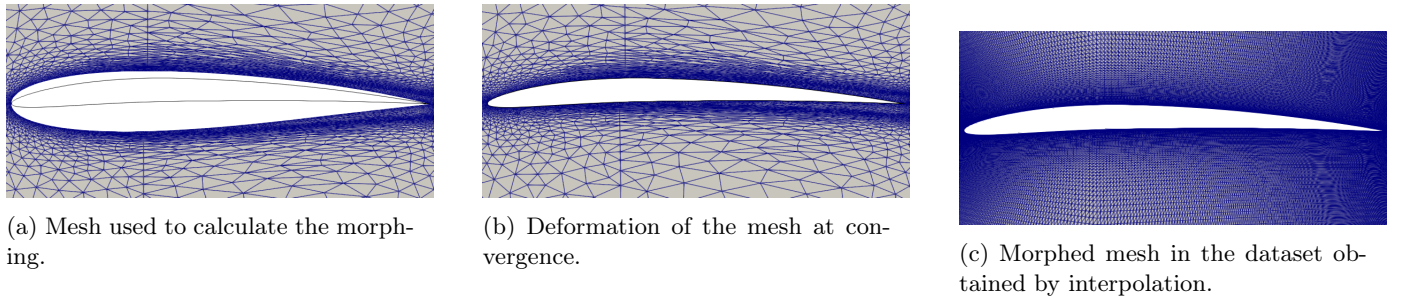


Figure 11: Morphing of the two meshes.

In Figure 12 we plot the average, minimum and maximum value of Δ_2^{vdf} as a function of the number of iterations over a set of 10 samples. As in the previous test case, we numerically observe that the vector distance algorithm converges exponentially with respect to the number of iterations.

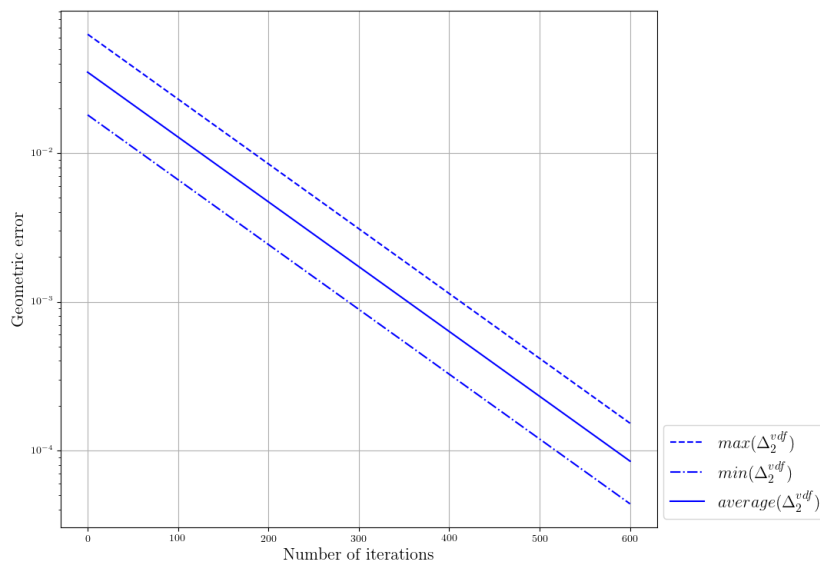


Figure 12: Average, maximum and minimum geometric errors Δ_2 in logarithmic scale on a subset of 10 samples, using the vector distance algorithm.

2.6 Extension to dimension 3

We gather here some remarks about the extension of the proposed approach in the case $d = 3$, which will be the object of a future research work. We consider a collection $\{\mathbf{P}_1, \dots, \mathbf{P}_{N_p}\} \subset \partial\Omega$ of N_p distinct points of $\partial\Omega$, a collection $\{L_1, \dots, L_{N_l}\} \subset \partial\Omega$ of disjoint, open, connected subdomains of $\partial\Omega$ with positive 1-dimensional Hausdorff measure and a collection $\{S_1, \dots, S_{N_s}\} \subset \partial\Omega$ of disjoint, open, connected subdomains of $\partial\Omega$ with positive 2-dimensional Hausdorff measure such that $\bigcup_{k_s=1}^{N_s} \overline{S_{k_s}} = \partial\Omega$. We also consider a collection $\{\mathbf{P}_1^0, \dots, \mathbf{P}_{N_p}^0\} \subset \partial\Omega_0$ of N_p distinct points of $\partial\Omega_0$, a collection $\{L_1^0, \dots, L_{N_l}^0\} \subset \partial\Omega_0$ of disjoint, open, connected subdomains of $\partial\Omega_0$ with positive 1-dimensional Hausdorff measure and a collection $\{S_1^0, \dots, S_{N_s}^0\} \subset \partial\Omega_0$ of disjoint, open, connected subdomains of $\partial\Omega_0$ with positive 2-dimensional Hausdorff measure such that $\bigcup_{k_s=1}^{N_s} \overline{S_{k_s}^0} = \partial\Omega_0$.

The extension of our approach then consists in finding a morphing $\phi \in \mathcal{T}_{\Omega_0} := \{\phi \in \mathbf{W}^{1,\infty}(\Omega_0), \phi^{-1} \in \mathbf{W}^{1,\infty}(\phi(\Omega_0)) : \phi \text{ is injective}\}$ such that

$$\phi(\Omega_0) = \Omega, \quad (23a)$$

$$\phi(\mathbf{P}_{k_p}^0) = \mathbf{P}_{k_p}, \quad \forall 1 \leq k_p \leq N_p, \quad (23b)$$

$$\phi(L_{k_l}^0) = L_{k_l}, \quad \forall 1 \leq k_l \leq N_l, \quad (23c)$$

$$\phi(S_{k_s}^0) = S_s, \quad \forall 1 \leq k_s \leq N_s. \quad (23d)$$

Our first observation is that the extension of Proposition 2 to the case $d = 3$ is not straightforward. We leave this theoretical question for future work. Assuming that the bilinear form

$$a_\phi : \mathbf{H}^1(\phi(\Omega_0)) \times \mathbf{H}^1(\phi(\Omega_0)) \ni (\mathbf{u}, \mathbf{v}) \mapsto a_\phi(\mathbf{u}, \mathbf{v}) := \int_{\phi(\Omega_0)} \sigma(\mathbf{u}) : \varepsilon(\mathbf{v}) d\mathbf{x} + \alpha \int_{\phi(\partial\Omega_0)} (\mathbf{u} \cdot \mathbf{n}_\phi)(\mathbf{v} \cdot \mathbf{n}_\phi) ds \quad (24)$$

defines an inner product on $\mathbf{H}^1(\phi(\Omega_0))$, up to some assumptions on the geometry of the domain $\phi(\Omega_0)$, the procedure described in Section 2.2 can be straightforwardly extended to design an iterative algorithm to compute a diffeomorphism $\phi \in \mathcal{T}_{\Omega_0}$ such that $\phi(\Omega_0) = \Omega$.

To extend the constrained approach proposed in Section 2.3, one would need to solve at each iteration linear elasticity problems of the form (14) where the linear form b_ϕ would be defined as the sum of three continuous real-valued linear forms b_ϕ^p , b_ϕ^l and b_ϕ^s defined on $\mathbf{H}^1(\phi(\Omega_0))$, the aim of which is to force the matching of points, lines and surfaces respectively. Appropriate definitions of these linear forms and tests of the resulting procedure on actual three-dimensional industrial test cases will be the object of future work.

3 Reduced-order modeling with geometric variability

The proposed high-fidelity morphing technique requires the resolution of a linear elasticity problem at each iteration, a process which can be time-consuming. In the context of model-order reduction with geometric variability, fast computation of this morphing is crucial for deriving efficient reduced-order models. Therefore, we introduce here a reduction technique aiming at speeding up these calculations to improve overall efficiency.

3.1 Offline phase

Given n target domains $\{\Omega_i\}_{1 \leq i \leq n}$ which compose our training set, we start by calculating the n morphings $\{\phi_i\}_{1 \leq i \leq n} \subset \mathcal{T}_{\Omega_0}$ from a fixed reference domain Ω_0 to each target domain Ω_i so that $\phi_i(\Omega_0) = \Omega_i$ for all $1 \leq i \leq n$. This can be done using either the signed distance algorithm or the vector distance algorithm presented in the previous section.

We then apply snapshot-POD (Proper Orthogonal Decomposition) on the family of displacement fields $\psi_i := \phi_i - \mathbf{Id}$ ($1 \leq i \leq n$) with respect to the $\mathbf{L}^2(\Omega_0)$ -inner product. We denote by $\lambda_1 \leq \dots \leq \lambda_n$ the n eigenvalues of the correlation matrix $\mathbf{C} := (\langle \psi_i, \psi_j \rangle_{\mathbf{L}^2(\Omega_0)})_{1 \leq i, j \leq n} \in \mathbb{R}^{n \times n}$ and by $\{\zeta_i\}_{1 \leq i \leq n} \subset \mathbf{W}^{1,\infty}(\Omega_0)$ the corresponding POD modes. For a given number of selected POD modes $r \in \mathbb{N}^*$, we introduce the mapping

$$\varphi_r : \mathbb{R}^r \ni \alpha := (\alpha_j)_{1 \leq j \leq r} \mapsto \varphi_r(\alpha) := \mathbf{Id} + \sum_{j=1}^r \alpha_j \zeta_j \in \mathbf{W}^{1,\infty}(\Omega_0). \quad (25)$$

For all $1 \leq i \leq n$, we define the vector $\alpha^i = (\alpha_j^i)_{1 \leq j \leq r} \in \mathbb{R}^r$ such that

$$\forall 1 \leq j \leq r, \alpha_j^i := \langle \phi_i - \mathbf{Id}, \zeta_j \rangle_{\mathbf{L}^2(\Omega_0)} = \langle \psi_i, \zeta_j \rangle_{\mathbf{L}^2(\Omega_0)},$$

so that $\sum_{j=1}^r \alpha_j^i \zeta_j$ is the $\mathbf{L}^2(\Omega_0)$ -orthogonal projection of $\psi_i := \phi_i - \mathbf{Id}$ onto $\text{Span}\{\zeta_1, \dots, \zeta_r\}$. Each morphing ϕ_i

can then be approximated by

$$\phi_i \approx \varphi_r(\alpha^i) := \mathbf{Id} + \sum_{j=1}^r \alpha_j^i \zeta_j = \mathbf{Id} + \sum_{j=1}^r \langle \phi_i - \mathbf{Id}, \zeta_j \rangle_{L^2(\Omega_0)} \zeta_j, \quad (26)$$

and each geometry $\Omega_i = \phi_i(\Omega_0)$ can be identified with the vector $\alpha^i \in \mathbb{R}^r$.

In general, we choose the value $1 \leq r \leq n$ in one of the following two ways:

- (i) For a prescribed tolerance criterion $\delta^{\text{POD}} > 0$, we choose r as the smallest positive integer such that

$$1 - \frac{\sum_{j=1}^r \lambda_j}{\sum_{j=1}^n \lambda_j} \leq \delta^{\text{POD}}.$$

This approach aims at controlling the accuracy of the reconstruction of the morphings using the first r POD modes in $L^2(\Omega_0)$ -norm.

- (ii) When dealing with variable geometries, however, we may want to control the maximum geometrical error between $\varphi_r(\alpha^i)(\Omega_0)$ and Ω_i . To this end, given a tolerance $\delta^{\text{geo}} > 0$, we choose r as the smallest integer such that

$$\max_{1 \leq i \leq n} \Delta_2(\varphi_r(\alpha^i), \Omega_0, \Omega_i) < \delta^{\text{geo}}, \quad (27)$$

where the quantity Δ_2 is defined in (21).

Note that, in addition to one of the two selection criteria highlighted above, we also ensure that r is large enough so that, for all $1 \leq i \leq n$, the POD approximation $\varphi_r(\alpha^i)$ is a diffeomorphism from Ω_0 onto $\varphi_r(\alpha^i)(\Omega_0)$.

Remark 5 (Geometry vs. vector α). *The correspondence between a geometry Ω_i and a vector $\alpha^i \in \mathbb{R}^r$ is not necessarily unique: for a geometry Ω_i such that $\Delta_2(\varphi_r(\alpha^i), \Omega_0, \Omega_i) < \delta^{\text{geo}}$, we may find another vector $\bar{\alpha}$ such that $\Delta_2(\varphi_r(\bar{\alpha}), \Omega_0, \Omega_i) < \delta^{\text{geo}}$ as well, without having $\phi_i = \varphi_r(\bar{\alpha}^i)$ up to the error on the POD. In other terms, we can find multiple morphings mapping Ω_0 on Ω_i in the affine space $\mathbf{Id} + \text{Span}\{\zeta_1, \dots, \zeta_r\}$.*

3.2 Online phase

Given a new geometry $\tilde{\Omega} \subset \mathbb{R}^d$ that is a domain of \mathbb{R}^d , we search for a morphing $\tilde{\phi} \in \mathcal{T}_{\Omega_0}$ in the affine space $\mathbf{Id} + \text{Span}\{\zeta_i\}_{1 \leq i \leq r}$, so that $\tilde{\phi}(\Omega_0)$ is close to $\tilde{\Omega}$ with respect to the criterion Δ_1 or Δ_2 . More precisely, the morphing $\tilde{\phi}$ will be computed as $\tilde{\phi} = \varphi_r(\tilde{\alpha})$ for some $\tilde{\alpha} \in \mathbb{R}^r$.

To present the online procedure in some detail, we first introduce some notation. On the one hand, for all $\alpha \in \mathbb{R}^r$, we define $\mathcal{J}_g(\alpha) := J_g(\varphi_r(\alpha))$ with J_g defined in (4). We have $\nabla \mathcal{J}_g(\alpha) = DJ_g(\varphi_r(\alpha))(\nabla \varphi_r(\alpha))$, so that, using (7), we obtain, for all $\alpha = (\alpha_j)_{1 \leq j \leq r} \in \mathbb{R}^r$ and all $1 \leq j \leq r$,

$$\nabla \mathcal{J}_g(\alpha)_j = \frac{\partial \mathcal{J}_g(\alpha)}{\partial \alpha_j} = \int_{\varphi_r(\alpha)(\partial \Omega_0)} g(\mathbf{x}) (\zeta_j \circ \varphi_r^{-1}(\alpha) \cdot \mathbf{n}_{\varphi_r(\alpha)}(\mathbf{x})) ds. \quad (28)$$

On the other hand, we introduce the functional

$$\mathcal{I} : \mathbb{R}^r \ni \alpha \mapsto \mathcal{I}(\alpha) := (\mathcal{I}_j(\alpha))_{1 \leq j \leq r} \in \mathbb{R}^r, \quad (29)$$

such that, for all $1 \leq j \leq r$ and all $\alpha \in \mathbb{R}^r$,

$$\begin{aligned} \mathcal{I}_j(\alpha) &:= b_{\varphi_r(\alpha)}^p(\zeta_j \circ \varphi_r^{-1}(\alpha)) + b_{\varphi_r(\alpha)}^l(\zeta_j \circ \varphi_r^{-1}(\alpha)) \\ &= \beta_1 \sum_{k_p=1}^{N_p} \int_{N(\varphi_r(\alpha)(\mathbf{P}_{k_p}^0))} (\mathbf{P}_{k_p} - \varphi_r(\alpha)(\mathbf{P}_{k_p}^0)) \cdot \zeta_j \circ \varphi_r^{-1}(\alpha)(\mathbf{x}) ds \\ &\quad + \beta_2 \int_{\varphi_r(\alpha)(\partial \Omega_0)} \left(\mathbf{D}_{\varphi_r(\alpha)}^{\partial \tilde{\Omega}} \cdot \mathbf{n}_{\varphi_r(\alpha)} \right) (\zeta_j \circ \varphi_r^{-1}(\alpha) \cdot \mathbf{n}_{\varphi_r(\alpha)}(\mathbf{x})) ds, \end{aligned} \quad (30)$$

where $b_{\varphi_r(\alpha)}^p$ and $b_{\varphi_r(\alpha)}^l$ are defined in (15) and (18), respectively.

The online procedure we propose to compute $\tilde{\alpha}$ is an iterative procedure which we now describe.

3.2.1 Initialization using the vector distance function

In line with the high-fidelity construction of the morphing, we could initialize the online iterative procedure so that $\phi^{(0)} = \varphi_r(\tilde{\alpha}^{(0)}) = \mathbf{Id}$. This corresponds to $\tilde{\alpha}^{(0)} = 0_{\mathbb{R}^r}$. However, this approach was observed to yield results which were not satisfactory, neither from an accuracy nor from an efficiency point of view. The initialization procedure we propose here remedies these shortcomings. It builds on the observation that if the new geometry $\tilde{\Omega}$ is close to one of the geometries belonging to the training set, one should be able to use this information to initialize the algorithm with a solution near an optimal solution. The idea is to rely on the construction of an appropriate regression model. More precisely, suppose that we have (or we can determine) a quantity that defines each geometry in the dataset. We can then build a regression metamodel that, for a given geometry $\tilde{\Omega}$, takes as input that quantity and produces as output the morphing coordinates $\tilde{\alpha} \in \mathbb{R}^r$.

We propose to use the vector distance function defined in (17) by proceeding as follows:

1. In the offline phase:

(a) For each geometry Ω_i , calculate the function $\mathbf{D}_{\mathbf{Id}}^{\partial\Omega_i}$ such that

$$\mathbf{D}_{\mathbf{Id}}^{\partial\Omega_i} : \partial\Omega_0 \ni \mathbf{x} \mapsto \mathbf{D}_{\mathbf{Id}}^{\partial\Omega_i}(\mathbf{x}) := \sum_{k_l=1}^{N_l} (\Pi_{L_{k_l}^i}(\mathbf{x}) - \mathbf{x}) 1_{L_{k_l}^0}(\mathbf{x}) \in \mathbb{R}^2, \quad (31)$$

where $\{L_{k_l}^i\}_{1 \leq k_l \leq N_l}$ is the set of curves partitioning the boundary of Ω_i .

(b) Compute the POD of the family of functions $\{\mathbf{D}_{\mathbf{Id}}^{\partial\Omega_i}\}_{1 \leq i \leq n}$ in $L^2(\partial\Omega_0)$ and denote by $(\theta_j)_{1 \leq j \leq n}$ the corresponding set of POD modes. Fix some $q \in \mathbb{N}^*$ and for all $1 \leq i \leq n$, compute $d^i = (d_j^i)_{1 \leq j \leq q} \in \mathbb{R}^q$ as

$$\forall 1 \leq j \leq q, \quad d_j^i = \left\langle \mathbf{D}_{\mathbf{Id}}^{\partial\Omega_i}, \theta_j \right\rangle_{L^2(\partial\Omega_0)}.$$

(c) Train a regression model that takes as input the vector $d^i \in \mathbb{R}^q$ and as output the generalized coordinates $\alpha^i \in \mathbb{R}^r$ of the morphing ϕ_i . Denote by $\mathcal{R} : \mathbb{R}^q \rightarrow \mathbb{R}^r$ the corresponding regression model.

2. In the online phase:

(a) For a new geometry $\tilde{\Omega}$, calculate the vector distance $\mathbf{D}_{\mathbf{Id}}^{\partial\tilde{\Omega}}$, then project on the low-dimensional representation to obtain the corresponding vector $\tilde{d} = (\tilde{d}_j)_{1 \leq j \leq q} \in \mathbb{R}^q$ such that

$$\forall 1 \leq j \leq q, \quad \tilde{d}_j = \left\langle \mathbf{D}_{\mathbf{Id}}^{\partial\tilde{\Omega}}, \theta_j \right\rangle_{L^2(\partial\Omega_0)}$$

(b) Define $\tilde{\alpha}^{(0)} = \mathcal{R}(\tilde{d})$.

The regression model used in this work is the Gaussian process regression (GPR) [33].

We make the following observations:

1. For a geometry $\tilde{\Omega}$, taking $\mathbf{D}_{\mathbf{Id}}^{\partial\tilde{\Omega}}$ as input does not mean that the output of the metamodel will map each point $\mathbf{x} \in \partial\Omega_0$ to its projection onto $\partial\tilde{\Omega}$. Here, the vector distance is used only to measure in some way the deviation of each Ω_i from $\partial\Omega_0$. Another possibility could have been to consider the gradient of the signed function $d_{\tilde{\Omega}}$.
2. Another choice to initialize the optimization algorithm could consider the full geometry, which is not the case with the vector distance function. The problem is that defining a quantity that is representative and unique for each geometry, and its boundary, is not straightforward, especially as we suppose that the geometries are non-parameterized. Another issue is that evaluating such a quantity for every new geometry should be sufficiently fast for an efficient evaluation of the ROM.

We emphasize that the above approach is not devised as a means of directly predicting the morphing coefficients without using the iterative algorithm (to be presented in the next section). Indeed, this would lead to two main drawbacks. Firstly, the output of the metamodel does not generally precisely satisfy $\varphi_r(\tilde{\alpha})(\Omega_0) = \tilde{\Omega}$. Secondly, it is possible for two different geometries to yield identical inputs \tilde{d} , resulting in the same coefficients $\tilde{\alpha}$ from the regression model. In conclusion, the above approach merely serves as a means of predicting an initialization $\tilde{\alpha}^{(0)}$ for the online optimization algorithm, that is (hopefully) sufficiently close to the optimal solution. Thus, even if two distinct geometries share the same initialization during the online phase, they will not produce identical morphings after solving the optimization problem.

3.2.2 Online iterative algorithm

To find the final reduced coordinates $\tilde{\alpha} \in \mathbb{R}^r$ for the new geometry $\tilde{\Omega}$, we use an iterative algorithm which consists in updating at each iteration m the vector $\tilde{\alpha}^{(m)} \in \mathbb{R}^r$ as

$$\tilde{\alpha}^{(m+1)} = \tilde{\alpha}^{(m)} - \gamma^{(m)} \mathcal{I}(\tilde{\alpha}^{(m)}), \quad (32)$$

for some $\gamma^{(m)} > 0$ starting from the initial value $\tilde{\alpha}^{(0)} \in \mathbb{R}^r$ obtained from the initialization procedure described in the previous section. In practice, the value $\gamma^{(m)}$ is always chosen to be equal to some constant value $\gamma > 0$ for all iterations $m \in \mathbb{N}$. In cases where one only wishes to match domains (and not necessarily points and/or lines), it is also possible to design an iterative algorithm with the updating formula

$$\tilde{\alpha}^{(m+1)} = \tilde{\alpha}^{(m)} - \gamma^{(m)} \nabla \mathcal{J}_g(\tilde{\alpha}^{(m)}). \quad (33)$$

Remark 6 (Elasticity-based update). *Another possibility could have been to use an inner product associated with the elasticity bilinear $a_{\varphi_r(\alpha)}$ defined in (8). We have $a_{\varphi_r(\alpha)}(\varphi_r(\mathbf{u}), \varphi_r(\mathbf{v})) = \langle M(\alpha)\mathbf{u}, \mathbf{v} \rangle_{\mathbf{L}^2(\mathbb{R}^r)}$ with the stiffness matrix $M(\alpha) := (a_{\varphi_r(\alpha)}(\zeta_i, \zeta_j))_{1 \leq i, j \leq r} \in \mathbb{R}^{r \times r}$. The iterative algorithm then becomes*

$$\alpha^{(m+1)} = \alpha^{(m)} - \gamma^{(m)} M^{-1}(\alpha^{(m)}) \mathcal{I}(\alpha^{(m)}). \quad (34)$$

While this inner product actually introduces physical information to deform the mesh, it requires determining, at each iteration, the stiffness matrix $M(\alpha^{(m)})$, which boils down to calculating $\frac{r(r+1)}{2}$ volume integrals. This can be quite costly. The advantage of the approach relying on (32) (or (33)) is that we only need to compute surface integrals, instead of computing volume integrals and solving a linear elasticity system at each iteration. Thus, the computational efficiency is much higher than with the approach relying on (34).

3.2.3 Stopping criterion and out-of-distribution geometry

Fix an error tolerance $\delta^{\text{geo}} > 0$. Then, for every new geometry $\tilde{\Omega}$ considered in the online phase, the iterative procedure described in Section 3.2.2 is carried out until the following stopping criterion is met:

$$\Delta_2(\varphi_r(\tilde{\alpha}^{(m)}), \Omega_0, \tilde{\Omega}) < \delta^{\text{geo}}.$$

We also choose a value $M_{\text{max}} \in \mathbb{N}^*$ corresponding to a maximum number of iterations and an a priori error threshold $\delta_{\nabla} > 0$. If the above stopping criterion is not reached after M_{max} iterations, we evaluate $\eta := \|\mathcal{I}(\tilde{\alpha}^{(M_{\text{max}})})\|$ if (32) is used (or $\eta := \|\nabla \mathcal{J}_g(\tilde{\alpha}^{(M_{\text{max}})})\|$ if (33) is used) and proceed as follows:

1. If $\eta \geq \delta_{\nabla}$, the iterative algorithm did not reach convergence. Depending on the required precision and the cost per iteration, we may allow here to increase the number of iterations M_{max} .
2. On the other hand, if $\eta < \delta_{\nabla}$, this means that the target domain $\tilde{\Omega}$ cannot be well-approximated in the form $\phi(\Omega_0)$ for some morphism ϕ computed as an element of $\mathbf{Id} + \text{Span}\{\zeta_i, 1 \leq i \leq r\}$. One practical way to choose the value of δ_{∇} is to define it as $\delta_{\nabla} := \frac{1}{n} \sum_{i=1}^n \|\mathcal{I}(\alpha^i)\|$ (or $\delta_{\nabla} := \frac{1}{n} \sum_{i=1}^n \|\nabla \mathcal{J}_g(\alpha^i)\|$). The geometry $\tilde{\Omega}$ is then classified as being out-of-distribution (ood) and the following two steps are performed:
 - (a) Increase the number of modes r ; this allows for more flexibility in finding a reduced morphing $\tilde{\phi}$ such that $\tilde{\phi}(\Omega_0)$ is close to $\tilde{\Omega}$.
 - (b) Or, use the high-fidelity routine to compute a high-fidelity map $\tilde{\phi} : \Omega_0 \rightarrow \tilde{\Omega}$, possibly initialized with $\tilde{\phi}^{(0)} = \varphi_r(\tilde{\alpha}^{(M_{\text{max}})})$, update $r := r + 1$ and define $\zeta_{r+1} := \tilde{\phi}$.

3.3 Overall workflow

The following tables summarize our offline and online workflows:

Data: Training set of domains $\{\Omega_i\}_{1 \leq i \leq n}$
Input: Reference domain Ω_0 , tolerance $\delta^{\text{geo}} > 0$, step size $\gamma > 0$
for $i \leftarrow 1$ **to** n **do**
 Calculate $\mathbf{D}_{\text{Id}}^{\partial\Omega_i}$;
 Initialize $\phi_i^{(0)} = \text{Id}$;
 $m \leftarrow 0$;
 repeat
 Solve for $\mathbf{u}_{\Omega_i}^{(m)}$;
 Update $\phi_i^{(m+1)} \leftarrow \phi_i^{(m)} + \gamma \mathbf{u}_{\Omega_i}^{(m)} \circ \phi_i^{(m)}$;
 Calculate $\mathbf{D}_{\phi_i^{(m+1)}}^{\partial\Omega_i}$;
 $m \leftarrow m + 1$;
 until $\Delta_2(\phi_i^{(m)}, \Omega_0, \Omega_i) < \epsilon$;
 $\phi_i \leftarrow \phi_i^{(m)}$;
end
POD: $\{\phi_i\}_{1 \leq i \leq n} \rightarrow \{\zeta_j\}_{1 \leq j \leq r}, \{\alpha^i\}_{1 \leq i \leq n}$ with tolerance δ^{geo} ;
SVD: $\{\mathbf{D}_{\text{Id}}^{\partial\Omega_i}\}_{1 \leq i \leq n} \rightarrow \{d^i\}_{1 \leq i \leq n}$;
Train GPR : $\{d^i\}_{1 \leq i \leq n}, \{\alpha^i\}_{1 \leq i \leq n} \rightarrow \mathcal{R}$;
Determine : δ_{∇} ;

Algorithm 1: Offline workflow

Data: Reduced-order basis $\{\zeta_j\}_{1 \leq j \leq r}$, bounds: $\delta^{\text{geo}}, \delta_{\nabla}$
Input: Reference domain Ω_0 , target domain $\tilde{\Omega}$, maximum number of iterations M_{max} , step size $\gamma > 0$
Output: Generalized coordinates $\tilde{\alpha}$
 Calculate $\mathbf{D}_{\text{Id}}^{\partial\tilde{\Omega}}$;
 Project $\mathbf{D}_{\text{Id}}^{\partial\tilde{\Omega}}$ to obtain \tilde{d} ;
 Use GPR to obtain $\tilde{\alpha}^{(0)}$;
 $m \leftarrow 0$;
while $m \leq M_{\text{max}}$ **do**
 $\tilde{\alpha}^{(m+1)} \leftarrow \tilde{\alpha}^{(m)} - \gamma \mathcal{I}(\tilde{\alpha}^{(m+1)})$;
 Calculate $\mathbf{D}_{\varphi_r(\tilde{\alpha}^{(m+1)})}^{\partial\tilde{\Omega}}$;
 if $\Delta_2(\varphi_r(\tilde{\alpha}^{(m)}), \Omega_0, \tilde{\Omega}) < \delta^{\text{geo}}$ **is true then**
 | Terminate the loop;
 end
 $m \leftarrow m + 1$;
 if $m = M_{\text{max}}$ **and** $\Delta_2(\varphi_r(\tilde{\alpha}^{(M_{\text{max}})}), \Omega_0, \tilde{\Omega}) > \delta^{\text{geo}}$ **then**
 | **if** $\|\mathcal{I}(\tilde{\alpha}^{(M_{\text{max}})})\|_2 \geq \delta_{\nabla}$ **then**
 | Increase M_{max} ;
 | **end**
 | **else**
 | Increase r or perform offline routine for $\tilde{\Omega}$;
 | **end**
 end
end

Algorithm 2: Online Workflow

3.4 Complexity

The cost of one iteration in the offline phase comprises the assembly of the stiffness matrix associated with the bilinear form a_ϕ defined in (8), the computation of the matching term (the signed distance function (6) or the vector distance function (17)), the assembly of the right-hand-side vector corresponding to the linear form ($\widetilde{D\mathcal{J}}_g(\phi)$ in (7) or \tilde{b}_ϕ^p and \tilde{b}_ϕ^l in (15) and (18)), and finally the resolution of the resulting sparse linear system to obtain the coordinates of the displacement field in the finite element basis.

The cost of one iteration in the online phase comprises computing the matching term (the signed distance function

(6) or the vector distance function (17)), and the evaluation of $\nabla \mathcal{J}_g(\alpha)$ or $\mathcal{I}(\alpha)$ for $\alpha \in \mathbb{R}^r$, which corresponds to computing r integrals as in (28) or (30).

We denote by \mathcal{N} the number of nodes of \mathcal{M}_0 , the mesh of Ω_0 that is used in the computation. The number of nodes on $\partial\Omega_0$ depends on the dimension of the problem, and for 2D elements, is of the order of $O(\sqrt{\mathcal{N}})$. We also denote by p the number of nodes used to discretize the boundary of the target domain.

	Offline	Online
Matrix assembly	$O(\mathcal{N})$	-
Matching term computation	$O(\log(p)\sqrt{\mathcal{N}})$	$O(\log(p)\sqrt{\mathcal{N}})$
Computation of the gradient	$O(\sqrt{\mathcal{N}})$	$O(r\sqrt{\mathcal{N}})$
Linear system resolution (dense matrix)	$O(\mathcal{N}^3)$	-

Table 1: Cost of one iteration in the offline and online phases.

In Table 1, we report the complexity per iteration for the offline and online phases. The complexity of the computation of the matching term is shown for the vector distance function using KD-tree to determine the closet point. The complexity is actually larger for the signed distance as we need to calculate also the sign for each node.

For the general case of dense matrices, the complexity of solving a linear system by a direct method is of order $O(\mathcal{N}^3)$. For sparse matrices such as the ones encountered here, the complexity depends on the algorithm used and the sparsity of the matrix. In our implementation, we used the LU decomposition for sparse matrices to solve the linear systems. Usually, this step is the most expensive one in the offline phase.

The efficiency of the online phase results from the fact that we do not need to solve any linear system. Another important aspect which speeds up the computations in the online phase is the initialization step described in Section 3.2.1. Because we initialize the iterative procedure close to the solution, the number of iterations needed to achieve convergence is significantly smaller than the number of iterations needed in the offline phase. Additional speed-up can be gained also from using parallel implementation to calculate the r integrals in (28) or (30).

3.5 Numerical results

In this section, we present numerical results to illustrate the performance of the above offline/online algorithm.

3.5.1 Tensile2D dataset

Offline phase: we adopt the same notation as in Section 2.5.1. The size of the training set is $n = 500$. For all $1 \leq i \leq n$, we define $\Omega_i = [-1, 1]^2 \setminus B(R_i)$, with $R_i = 0.2 + 0.6 \times \frac{i}{n}$. We define the reference domain $\Omega_0 := \Omega_{250}$. This is the same reference domain as the one used in Section 2.5.1. We start by calculating the morphings $\{\phi_i\}_{1 \leq i \leq n}$ using the vector distance algorithm. We emphasize that the parameterization is not used in the construction of the morphings.

Next, we choose $\delta^{\text{geo}} := 5 \times 10^{-4}$, which is smaller than the size of an element. Employing POD with the criterion (27) leads to $r = 5$ modes. Finally, we train a Gaussian process regression (GPR) that takes as input the SVD coordinates of the vector distance function with $q = 5$, and gives as output the generalized morphing coordinates to initialize the online optimization problem.

Online phase: The testing set is composed of $n_{\text{test}} := 200$ geometries $\{\tilde{\Omega}_j\}_{1 \leq j \leq n_{\text{test}}}$ which have the same form as the training set, that is, $\tilde{\Omega}_j = [-1, 1]^2 \setminus B(\tilde{R}_j)$ for some (supposedly unknown) radius \tilde{R}_j . All the radii \tilde{R}_j are different from those of the training set. For each $\tilde{\Omega}_j$, we use the vector distance function in the regression model to predict the initial iteration to the online optimization problem. In Table 2, we report the quantities

$$\begin{aligned} \delta_{\text{avg}}^{\text{geo}}(r, N) &:= \frac{1}{N} \sum_{i=1}^N \Delta_2(\varphi_r(\alpha^i)(\Omega_0), \tilde{\Omega}_i), \\ \delta_{\text{max}}^{\text{geo}}(r, N) &:= \max_{1 \leq i \leq N} \{\Delta_2(\varphi_r(\alpha^i)(\Omega_0), \tilde{\Omega}_i)\}, \\ \delta_{\text{min}}^{\text{geo}}(r, N) &:= \min_{1 \leq i \leq N} \{\Delta_2(\varphi_r(\alpha^i)(\Omega_0), \tilde{\Omega}_i)\}, \end{aligned}$$

for $r = 5$, $N = n_{\text{test}}$ and $\delta^{\text{geo}} = 5 \times 10^{-4}$. As we observe in Table 2, for all the samples in the test set, the initialized solution here is an optimal one that satisfies the stopping criterion, so the online iterative procedure is not used. Thus, the cost of each morphing calculation is only one evaluation of the vector distance function and one evaluation of the GPR which drastically cuts down the cost of morphing computation. In Table 3, we report the ratio of the average (resp., maximum) time needed to compute the high-fidelity morphing (offline) over the average (resp., maximum) time needed to compute the reduced-order morphing (online) using our implementations. We observe that the reduced-order model we propose is about 270 times faster than the high-fidelity one. The steps required to construct the online

phase model are morphing computation, morphing POD, vector distance function POD and GPR training. The time required to construct the online phase model is dominated by the morphing computation.

$\delta_{\text{avg}}^{\text{geo}}(r, n_{\text{test}})$	$\delta_{\text{max}}^{\text{geo}}(r, n_{\text{test}})$	$\delta_{\text{min}}^{\text{geo}}(r, n_{\text{test}})$
1.5×10^{-4}	3.6×10^{-4}	5.2×10^{-7}

Table 2: Average, maximum, and minimum values of the criterion Δ_2 for all the samples from the dataset in the online phase after the initialization.

Ratio of average time (offline/online)	267.1
Ratio of maximum time (offline/online)	269.6

Table 3: Ratio of average and maximum time to compute the morphing in the offline and online phases for the Tensile2D dataset.

3.5.2 AirfRANS

Offline phase. For this test, we use all the 1000 airfoils in the AirfRANS dataset. The number of samples in the training set is $n := 800$. We use the same reference geometry, mesh and physical parameters as in Section 2.5.2. After calculating each morphing, we apply the POD on the displacement fields to obtain the principal modes of the displacements. Finally, we train the GPR to use it to predict an initial iteration for the samples in the testing set.

Online phase. The testing set is composed of the remaining $n_{\text{test}} := 200$ samples. Here, the initialization of the morphing is not sufficient enough to satisfy our criterion on the error, so we also use the online optimization strategy.

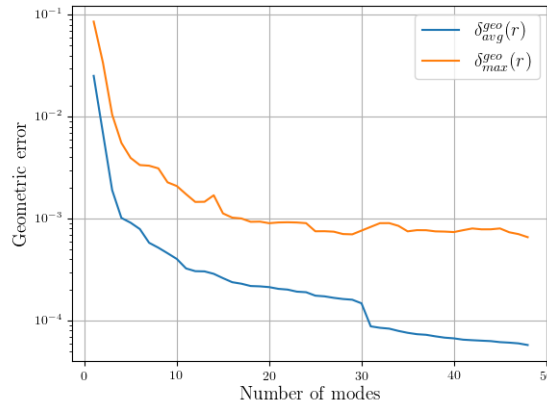


Figure 13: Evolution of the geometrical errors $\delta_{\text{avg}}^{\text{geo}}(r, n)$ and $\delta_{\text{max}}^{\text{geo}}(r, n)$ (in logarithmic scale) for the training set as a function of the number of modes r . The errors are not equal to zero due to the POD truncation error.

In Figure 13, we report both the average and maximum geometrical errors in the training set as a function of the number of modes. As expected, both errors tend to zero as we add more modes for morphing reconstruction. Note, however, that the convergence process is not monotone. This is due to the fact that additional modes actually can have the effect of better approaching the morphing field ϕ_i , and not necessarily minimizing the geometrical error. Obviously, taking all the modes yields of zero error between ϕ_i and $\varphi_r(\alpha^i)$, and, as a result, zero geometrical error. We test the morphing strategy for $r \in \{12, 16, 20, 24, 28, 32, 48\}$. For each value, we re-initialize $\tilde{\alpha}^{(0)}$ in \mathbb{R}^r for each sample in the testing set and perform the optimization in \mathbb{R}^r . For all the tests on the values of r , we fix the same geometric tolerance $\delta^{\text{geo}} = 1.5 \times 10^{-4}$. In Figure 14, we show the number of samples for which convergence is achieved, *i.e.* satisfying $\Delta_2(\varphi_r(\tilde{\alpha}^{(m)})(\Omega_0), \tilde{\Omega}) < \delta^{\text{geo}}$, as a function of the number of iterations. Here, zero iteration means that the initialized solution is sufficiently close so that further optimization is not needed.

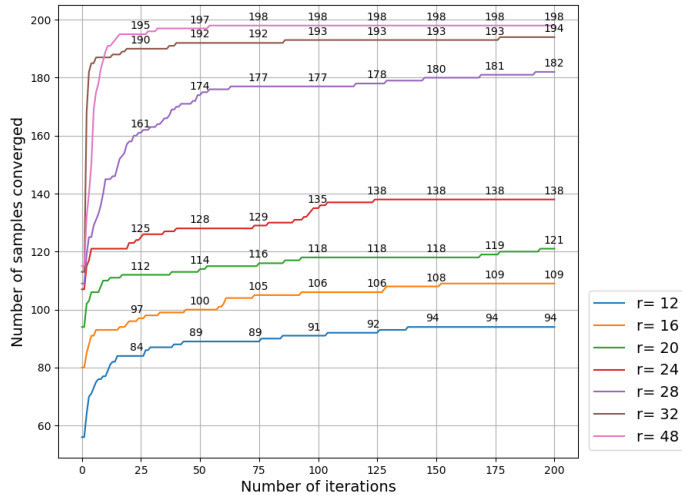
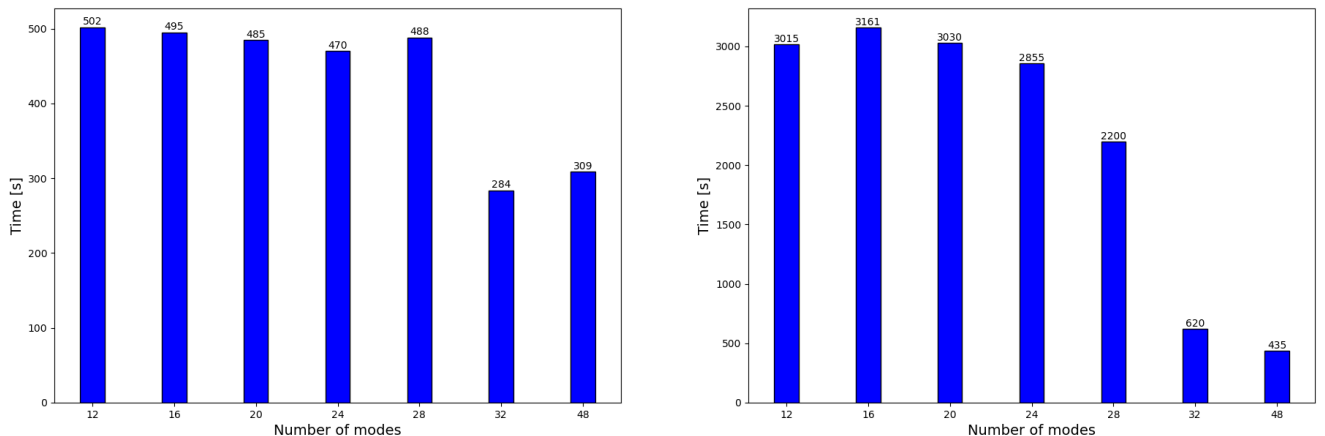


Figure 14: Number of samples that converged in Example 3.5.2 as a function of the number of iterations for different values of r .

When using more modes, the cost per iteration is higher but the convergence is achieved in fewer iterations, so that the overall cost to convergence is actually significantly lower. In Figure 15, we report the time needed to compute all the morphings in the test set for the different values of r . As we can see, for the more modes allows for faster convergence.



(a) Time to converge with $M_{\max} = 25$ iterations.

(b) Time to converge with $M_{\max} = 200$ iterations.

Figure 15: Overall time needed to compute all the morphings for different values of r and for the maximum number of iterations M_{\max} .

4 Learning scalar outputs from simulations

In this section, we show numerical results to illustrate how the above morphing strategy can be exploited to build regression models to predict scalar outputs from physical simulations under non-parameterized geometrical variability. This approach is physics-agnostic, that is, the physical equations do not play a role in the process.

4.1 Methodology

Let $\{\Omega_i\}_{1 \leq i \leq n}$ to be a collection of different geometries. Each geometry is equipped with a (non-geometrical) parameter $\mu_i \in \mathcal{P}$ where $\mathcal{P} \subset \mathbb{R}^p$ is a set of parameter values that is used to perform the physical simulations. The parameters can be boundary conditions, material properties and so on; however, we emphasize here that the parametrization of the geometries is not known. In this context, the objective is to determine the outputs of interest of the physical

problem, which consist of:

1. The physical fields $U_i := (u_{i,j})_{1 \leq j \leq n_{\text{fields}}}$ with $u_{i,j} : \Omega_i \rightarrow \mathbb{R}$ for all $1 \leq i \leq n$ and $1 \leq j \leq n_{\text{fields}}$ with $n_{\text{fields}} \in \mathbb{N}^*$. These fields are usually solutions to a set of partial differential equations. For example, depending on the problem, these can be stress, deformation, velocity, pressure, etc...
2. The scalar outputs $W_i := (w_{i,j})_{1 \leq j \leq n_{\text{scalars}}}$ for all $1 \leq i \leq n$ and $1 \leq j \leq n_{\text{scalars}}$ with $n_{\text{scalars}} \in \mathbb{N}^*$. Examples of scalar quantities of interest are the drag and lift coefficients.

Here, we restrict ourselves to the prediction of scalars outputs. Ongoing work aims at proposing a more sophisticated and efficient approach, built on the concepts introduced here, to realize the prediction of physical fields.

Given the set of input pairs $(\Omega_i, \mu_i)_{1 \leq i \leq n}$, and outputs $(W_i)_{1 \leq i \leq n}$, calculated using a high-fidelity model, our goal is to learn a mapping \mathcal{W} which maps a pair (Ω, μ) , where Ω is a subdomain of \mathbb{R}^d and $\mu \in \mathcal{P}$ is a parameter value, to the corresponding output $W \in \mathbb{R}^{n_{\text{scalars}}}$ so that $W = \mathcal{W}(\Omega, \mu)$.

Because the geometries are not parameterized, the only available information that represents each geometry is its mesh \mathcal{M}_i . However, the learning task on meshes can be quite challenging owing to the high number of degrees of freedom that should be taken as input. To deal with large meshes, solutions using deep neural network architectures were the most popular of machine learning techniques [34, 35]. Furthermore, recent advances rely on graph neural networks [36] as they can overcome the limitation of having graph input with different numbers of nodes [37]. In [38], the authors propose a method that does not rely on neural network architecture, and uses Gaussian process regression model based on the sliced Wasserstein-Weisfeiler-Lehman kernel between graphs to deal with variable geometry to predict scalar outputs. Instead, we propose here to consider the offline/online morphing technique described above. We proceed as follows:

1. In the offline phase, given the input pairs $(\Omega_i, \mu_i)_{1 \leq i \leq n}$ and the outputs $(W_i)_{1 \leq i \leq n}$:
 - (a) Choose a reference domain Ω_0 and calculate the morphings $\phi_i : \Omega_0 \rightarrow \Omega_i$ (Section 2).
 - (b) Apply the snapshot-POD on $(\phi_i)_{1 \leq i \leq n}$, and calculate the generalized coordinates $\alpha^i \in \mathbb{R}^r$ for each geometry (Section 3).
 - (c) Train the regression model \mathcal{W} :

$$\mathbb{R}^r \times \mathcal{P} \ni (\alpha, \mu) \mapsto \mathcal{W}(\alpha, \mu) \in \mathbb{R}^{n_{\text{scalars}}}. \quad (35)$$

Notice that, each geometry is parameterized by the coordinates of the POD modes of the displacement field $\phi_i - \text{Id}$.

2. In the online phase, given a new pair $(\tilde{\Omega}, \tilde{\mu})$:
 - (a) Calculate the vector $\tilde{\alpha} \in \mathbb{R}^r$ that corresponds to the morphing $\varphi_r(\tilde{\alpha}) \in \mathcal{T}_{\Omega_0}$ corresponding to the target domain $\tilde{\Omega}$ (Section 3).
 - (b) Use the regression model to obtain the scalar outputs $\tilde{W} = \mathcal{W}(\tilde{\alpha}, \tilde{\mu})$.

We use a Gaussian process regression for the learning task. For the training phase, we employ anisotropic Matern-5/2 kernels and zero mean-functions for the priors, and the training is done using the GPy package [39].

The proposed strategy becomes very similar to the MMGP method from [18]. The two main differences are: 1) the morphing algorithm used here is more versatile and is not tailored to specific cases; 2) the increased efficiency of the procedure consisting of the offline-online separation of the morphing algorithm. Moreover, the present method computes morphings (and thus displacement fields) from the reference domain, eliminating the need for some finite element interpolation of the displacement fields to a common support in order to apply the snapshot-POD as in MMGP (where morphings are computed towards the reference domain).

4.2 AirFRANS: drag coefficient prediction

We apply the above methodology to the AirFRANS dataset. In addition to a mesh of the NACA profile, each sample in the AirFRANS dataset has two scalars as input: the inlet velocity v_0 and the angle of attack θ_0 . The outputs of the physical simulation are the velocity, pressure and dynamic viscosity fields, as well as the drag and lift coefficients. The outputs are obtained using a 2D incompressible RANS model.

We focus our attention here on learning the drag coefficient C_d from the inputs $\mu = (v_0, \theta_0)$ and the geometry Ω . The Gaussian process regression model \mathcal{W} takes as input the morphing generalized coordinates α^i and the physical parameters $\mu_i = (v_0^i, \theta_0^i)$, and gives as output the drag C_d^i . To see the effect of the number of modes and the stopping criterion on the precision of the prediction, we perform the following tests:

1. Test 1: we compute the morphings online using r modes for $r \in \{12, 16, 20, 24, 28, 32, 36, 40, 44, 48\}$ (including the initial solution prediction). We use the calculated coordinates to predict C_d . We use the same stopping geometrical criterion $\delta^{\text{geo}} := 1.5 \times 10^{-4}$ for the different values of r . We stop the optimization after $M_{\text{max}} = 200$ iterations if the algorithm does not converge.
2. Test 2: we compute the morphings using $r' = 48$ modes, but we take only the first r coordinates to perform the prediction, with $r \in \{12, 16, 20, 24, 28, 32, 36, 40, 44, 48\}$. In this case, each morphing $\varphi_{r'}(\alpha^i)$ is calculated once, but the number of used components of α^i changes. We use the same tolerance δ^{geo} and maximum number of iterations M_{max} as test 1.
3. Test 3: similar to Test 2, but with $r' = 64$ modes.
4. Test 4: as in Test 2, we take $r' = 48$ modes. But we change the stopping criterion: we perform a fixed number of iterations for all the samples regardless of the geometrical error. We choose here to perform 25 iterations for all samples.

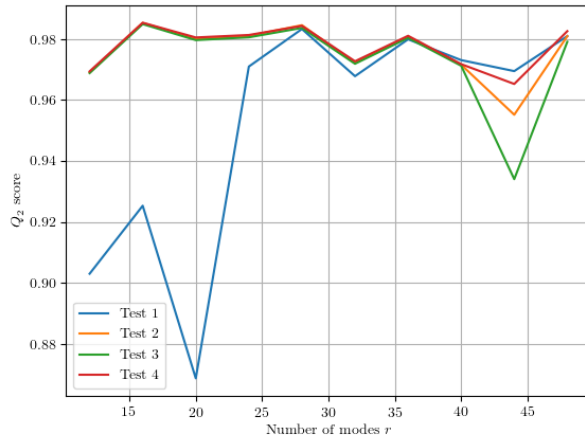


Figure 16: Q^2 scores (see (36)) for different values of r .

We notice that for different values of r , the regression model \mathcal{W}_r changes (we use the subscript r to indicate this). However, the model does not change for a given value of r over the different tests. All the models \mathcal{W}_r are trained once in the offline phase and used for the different tests.

To evaluate the performance of the method to predict the drag coefficient C_d , we evaluate, for each test and for each value of r , the Q^2 -score defined as:

$$Q^2 := 1 - \frac{\sum_{i=1}^{n_{\text{test}}} (y_i - f_i)^2}{\sum_{i=1}^{n_{\text{test}}} (y_i - \bar{y})^2} \quad (36)$$

with y_i the true values of the drag C_d , f_i the predicted values using the model \mathcal{W} , and $\bar{y} := \frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} y_i$ the mean. In

the best-case scenario, the score is $Q^2 = 1$, which means that the model predicted correctly all the true values.

In Figure 16, we present the various Q^2 scores. The main observation is that using more modes (r' modes) to calculate the morphing can be beneficial when using models that take r ($r < r'$) mode coefficients as inputs. For instance, calculating the morphing with 48 modes but utilizing only the first 16 components of α to predict the values of C_d with \mathcal{W}_{16} yields a superior Q^2 score compared to calculating the morphing using only 16 modes and using the obtained coordinates \mathcal{W}_{16} to predict C_d . Thus, employing more modes to calculate the morphings enhances the quality of the coefficients for the prediction.

From the conducted tests, our best Q^2 score is obtained for Test 4 when using 16 modes for the prediction, with $Q^2 = 0.9853$. A similar result is obtained for Test 2 using also 16 modes for the prediction, with $Q^2 = 0.9852$. In comparison with the results shown in [18], both results surpassed the scores obtained using, for the same dataset, MMGP ($Q^2 = 0.9831$), a graph convolutional neural network GCNN [36] ($Q^2 = 0.9596$), and MeshGraphNets (MGN) [37] ($Q^2 = 0.9743$).

5 Conclusion

We presented a new method to construct morphings between geometries that share the same topology. The technique is suitable to model-order reduction with non-parameterized geometries, as it does not suppose any knowledge of a parameterization of the geometries. In the offline phase, morphings are constructed using elastic deformations from a reference domain to a target domain. In the online phase, morphings are obtained as the solution to a low-dimensional fixed-point iteration problem, which can be solved more efficiently in a reduced-order model framework. For both the offline and online phases, we provided numerical examples in 2D to show the performance of the proposed method. Moreover, we illustrated how the computed morphings can be used to predict scalar quantities in physical problems using Gaussian process regression models. The geometry is taken into consideration in the prediction by the coordinates of the POD modes of the displacement field, an approach similar to MMGP. However, the present approach proves to be generic as the morphing strategy is not case-dependent, and is more efficient owing to the offline/online separation of the morphing algorithm.

Acknowledgements

Funded/Co-funded by the European Union (ERC, HighLEAP, 101077204). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References

- [1] A. Quarteroni, A. Manzoni, and F. Negri, *Reduced Basis Methods for Partial Differential Equations: An Introduction*. Springer, 2015, vol. 92.
- [2] J. S. Hesthaven, G. Rozza, and B. Stamm, *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*. Springer, 2016, vol. 590.
- [3] G. Berkooz, P. Holmes, and J. L. Lumley, “The proper orthogonal decomposition in the analysis of turbulent flows,” *Annual Review of Fluid Mechanics*, vol. 25, no. 1, pp. 539–575, 1993.
- [4] A. Chatterjee, “An introduction to the proper orthogonal decomposition,” *Current Science*, pp. 808–817, 2000.
- [5] D. Ryckelynck, “Hyper-reduction of mechanical models involving internal variables,” *International Journal for Numerical Methods in Engineering*, vol. 77, no. 1, pp. 75–89, 2009.
- [6] G. Welper, “Transformed snapshot interpolation with high resolution transforms,” *SIAM Journal on Scientific Computing*, vol. 42, no. 4, pp. A2037–A2061, 2020.
- [7] N. Cagniard, Y. Maday, and B. Stamm, “Model order reduction for problems with large convection effects,” *Contributions to partial differential equations and applications*, pp. 131–150, 2019.
- [8] G. Rozza, D. B. P. Huynh, and A. T. Patera, “Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations: Application to transport and continuum mechanics,” *Archives of Computational Methods in Engineering*, vol. 15, no. 3, pp. 229–275, 2008.
- [9] T. W. Sederberg and S. R. Parry, “Free-form deformation of solid geometric models,” in *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, 1986, pp. 151–160.
- [10] A. De Boer, M. S. Van der Schoot, and H. Bijl, “Mesh deformation based on radial basis function interpolation,” *Computers & Structures*, vol. 85, no. 11–14, pp. 784–795, 2007.
- [11] T. J. Baker, “Mesh movement and metamorphosis,” *Engineering with Computers*, vol. 18, no. 3, pp. 188–198, 2002.
- [12] S. M. Shontz and S. A. Vavasis, “A robust solution procedure for hyperelastic solids with large boundary deformation,” *Engineering with Computers*, vol. 28, pp. 135–147, 2012.
- [13] B. Froehle and P.-O. Persson, “Nonlinear elasticity for mesh deformation with high-order discontinuous galerkin methods for the navier-stokes equations on deforming domains,” in *Spectral and High Order Methods for Partial Differential Equations ICOSAHOM 2014: Selected Papers from the ICOSAHOM Conference, June 23–27, 2014, Salt Lake City, Utah, USA*. Springer, 2015, pp. 73–85.
- [14] A. Manzoni and F. Negri, “Efficient reduction of pdes defined on domains with variable shape,” *Model Reduction of Parametrized Systems*, pp. 183–199, 2017.
- [15] F. Salmoiraghi, A. Scardigli, H. Telib, and G. Rozza, “Free-form deformation, mesh morphing and reduced-order methods: Enablers for efficient aerodynamic shape optimisation,” *International Journal of Computational Fluid Dynamics*, vol. 32, no. 4–5, pp. 233–247, 2018.
- [16] N. Demo, M. Tezzele, A. Mola, and G. Rozza, “Hull shape design optimization with parameter space and model reductions, and self-learning mesh morphing,” *Journal of Marine Science and Engineering*, vol. 9, no. 2, p. 185, 2021.
- [17] C. Lehrenfeld and S. Rave, “Mass conservative reduced order modeling of a free boundary osmotic cell swelling problem,” *Advances in Computational Mathematics*, vol. 45, no. 5, pp. 2215–2239, 2019.
- [18] F. Casenave, B. Staber, and X. Roynard, “MMGP: a Mesh Morphing Gaussian Process-based machine learning method for regression of physical problems under nonparametrized geometrical variability,” *Advances in Neural Information Processing Systems*, vol. 36, 2024.
- [19] D. Ye, V. Krzhizhanovskaya, and A. G. Hoekstra, “Data-driven reduced-order modelling for blood flow simulations with geometry-informed snapshots,” *Journal of Computational Physics*, vol. 497, p. 112639, 2024.
- [20] T. Taddei, “A registration method for model order reduction: Data compression and geometry reduction,” *SIAM Journal on Scientific Computing*, vol. 42, no. 2, pp. A997–A1027, 2020.

-
- [21] —, “An optimization-based registration approach to geometry reduction,” *arXiv Preprint arXiv:2211.10275*, 2022.
- [22] —, “Compositional maps for registration in complex geometries,” *arXiv Preprint arXiv:2308.15307*, 2023.
- [23] A. Iollo and T. Taddei, “Mapping of coherent structures in parameterized flows by learning optimal transportation with gaussian models,” *Journal of Computational Physics*, vol. 471, p. 111671, 2022.
- [24] S. Cucchiara, A. Iollo, T. Taddei, and H. Telib, “Model order reduction by convex displacement interpolation,” *Journal of Computational Physics*, p. 113230, 2024.
- [25] M. F. Beg, M. I. Miller, A. Trouvé, and L. Younes, “Computing large deformation metric mappings via geodesic flows of diffeomorphisms,” *International Journal of Computer Vision*, vol. 61, pp. 139–157, 2005.
- [26] F. Galarce, D. Lombardi, and O. Mula, “State estimation with model reduction and shape variability. application to biomedical problems,” *SIAM Journal on Scientific Computing*, vol. 44, no. 3, pp. B805–B833, 2022.
- [27] M. De Buhan, C. Dapogny, P. Frey, and C. Nardoni, “An optimization method for elastic shape matching,” *Comptes Rendus Mathématique*, vol. 354, no. 8, pp. 783–787, 2016.
- [28] G. Allaire, *Conception Optimale de Structures*. Springer, 2007, vol. 58.
- [29] G. Allaire, F. Jouve, and A.-M. Toader, “Structural optimization using sensitivity analysis and a level-set method,” *Journal of Computational Physics*, vol. 194, no. 1, pp. 363–393, 2004.
- [30] “Muscat,” <https://gitlab.com/drti/muscat>, since 2023 (accessed 28 August 2024).
- [31] F. Casenave, X. Roynard, and B. Staber, “Tensile2d: 2D Quasistatic Non-linear Structural Mechanics Solutions, under Geometrical Variations,” Nov. 2023. [Online]. Available: <https://doi.org/10.5281/zenodo.10124594>
- [32] F. Bonnet, J. Mazari, P. Cinnella, and P. Gallinari, “Airfrans: High fidelity computational fluid dynamics dataset for approximating reynolds-averaged navier–stokes solutions,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 23 463–23 478, 2022.
- [33] C. K. Williams and C. E. Rasmussen, *Gaussian Processes for Machine Learning*. MIT Press, 2006.
- [34] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, “Machine learning for fluid mechanics,” *Annual Review of Fluid Mechanics*, vol. 52, pp. 477–508, 2020.
- [35] J. Willard, X. Jia, S. Xu, M. Steinbach, and V. Kumar, “Integrating scientific knowledge with machine learning for engineering and environmental systems,” *ACM Computing Surveys*, vol. 55, no. 4, pp. 1–37, 2022.
- [36] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, “The graph neural network model,” *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008.
- [37] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. W. Battaglia, “Learning mesh-based simulation with graph networks,” *arXiv Preprint arXiv:2010.03409*, 2020.
- [38] R. C. Perez, S. Da Veiga, J. Garnier, and B. Staber, “Gaussian process regression with sliced wasserstein weisfeiler-lehman graph kernels,” *arXiv Preprint arXiv:2402.03838*, 2024.
- [39] GPy, “GPy: A gaussian process framework in python,” <http://github.com/SheffieldML/GPy>, since 2012.