Solvers for Shape Optimization

Federico Danieli

February 26, 2015

Abstract

Various methods for the solution of a shape optimization problem via pursuing diffeomorphism are discussed. As a model problem, an instance of a *Bernoulli exterior free boundary problem* is considered. The solvers described belong to the category of *Descent methods*. These methods propose different ways of evaluating both descent directions and step sizes, with various degrees of numerical accuracy. The properties of these methods are analysed and compared, paying specific attention to their computational costs and convergence rates.

1 Introduction

The aim of solving a shape optimization problem consists in finding a certain shape (or domain) Ω that is optimal in the sense that it minimizes a given cost functional $\mathcal{J}$. Problems of this sort are of major importance in engineering, and find their applications for instance in aerodynamics or in structural mechanics. Examples might be finding the profile of an airfoil that minimizes the total drag, or the shape of a beam that can best resist a given load. The value of the functional $\mathcal{J}$ is influenced by the shape itself, but it might also depend on some other variable defined on the above-mentioned domain, $u(\Omega)$. This variable is called *state variable*, and could represent a displacement, a velocity or a temperature field. In the case the value of such state is recovered via the solution of a Partial Differential Equation (PDE), or *state equation*, the problem considered is said to have a PDE constraint. Unfortunately, in the majority of applications, an analytic expression for the solution is not available; therefore, a numerical approach must be invoked.

The *Bernoulli exterior free boundary problems* represent a large set that falls within the category of PDE constrained shape optimization problems [11, 4, 13]. They appear in a range of applications, from Fluid-dynamics to Thermodynamics, and due to their characteristics are often used as models for the evaluation of numerical schemes [12]. The scheme discussed in this paper belongs to the class of *fixed-mesh* methods (see for example [7, 8]), as a counterpart to *moving-mesh* methods [9, 10]. This work is to be considered as a continuation of what has been done by R. Hiptmair and A. Paganini in [6], and it is inserted in its set-up.

The peculiarity of *fixed-mesh* methods consists in searching the optimal shape via a map from the initial domain Ω_0 to the optimal one Ω. The true unknown of the problem becomes then the map itself. As a main advantage of this approach, there is no need to update the shape step-by-step as the optimization algorithm proceeds. This may instead represent an issue in *moving-mesh* methods, since for them a way to update the mesh too must be found, which is often not a trivial task. This is due to the fact that, as the mesh is changed to take into account the modifications applied to the domain, its initial quality may not be preserved.

In the following, the mathematical description of the problem considered is given.

2 Problem definition

2.1 Exterior Bernoulli free boundary problem

Following the derivation described in [2], we consider the following overdetermined Boundary Value Problem (BVP) on a domain with an annular shape Ω_0

$$\begin{cases} -\operatorname{div}(\nabla u) = 0, & \textit{in } \Omega_0 \\ u = 1, & \textit{on } \partial\Omega_0^{in} \\ u = 0, & \textit{on } \partial\Omega_0^{out} \\ \|\nabla u\| = 1, & \textit{on } \partial\Omega_0^{out}. \end{cases} \tag{1}$$

A sketch of the domain Ω_0 can be seen in Fig.1. We fix a-priori the interior boundary $\partial\Omega_0^{out}$ and we formulate the following *exterior Bernoulli free boundary problem*: find a new domain Ω by modifying $\partial\Omega_0^{out}$ in such a way that (1) admits a solution, while preserving an annular shape.

2.2 Shape optimization problem

It can be shown that the *exterior Bernoulli free boundary problem* described in 2.1 can be reformulated as a shape optimization problem with PDE constraints [2]. In order to provide the formulation of this problem, we introduce

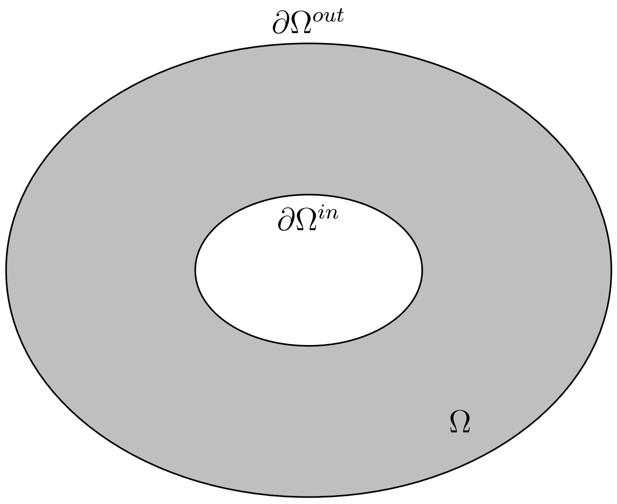

Figure 1: Sketch for the domain of the BVP (1). The internal border $\partial\Omega^{in}$ is fixed, while $\partial\Omega^{out}$ is modified to find the optimal shape.

the cost functional

$$\mathcal{J}(\Omega, u(\Omega)) = \int_{\Omega} (\nabla u \, \nabla u + 1) d\mathbf{x}, \tag{2}$$

and the PDE that acts as constraint, which is also called the *State Equation*

$$\begin{cases} -\operatorname{div}(\nabla u) = 0, & in\ \Omega_0 \\ u = 1, & on\ \partial\Omega_0^{in} \\ u = 0, & on\ \partial\Omega_0^{out}. \end{cases} \tag{3}$$

The *PDE-constrained shape optimization problem* we consider can then be formulated as follows

$$\min_{\Omega \in \mathcal{U}_{ad}} \mathcal{J}(\Omega, u(\Omega)) \ s.t. \ (3), \tag{4}$$

where $\mathcal{U}_{ad}$ denotes the set of admissible shapes.

2.3 Admissible shapes

We define the set of admissible shapes as

$$\mathcal{U}_{ad} := \{\Omega = T_{\mathcal{V}}(\Omega_0) \mid T_{\mathcal{V}} := \mathcal{I} + \mathcal{V}, \ \|\mathcal{V}\|_{C^2(\Omega_0)} \leq 1 - \epsilon, \ T_{\mathcal{V}}(\partial\Omega_0^{in}) = \partial\Omega_0^{in}\},$$

given a fixed small real number ϵ. Here, Ω_0 is the initial domain which is assumed to have Lipschitz boundary, $\mathcal{I}$ represents the identity operator, and

$T_{\mathcal{V}}$ is a map that preserves the internal boundary $\partial\Omega_0^{in}$, defined as following:

$$\begin{aligned} T_{\mathcal{V}} : \mathbb{R}^d &\to \mathbb{R}^d, \\ x &\mapsto T_{\mathcal{V}}(x) := x + \mathcal{V}(x). \end{aligned}$$

By $T_{\mathcal{V}}(\Omega_0)$ we denote the image set of the map $T_{\mathcal{V}}$ restricted to Ω_0.

We assume that the vector field

$$\mathcal{V} : \mathbb{R}^d \to \mathbb{R}^d$$

satisfies the condition $\|\mathcal{V}\|_{C^2(\Omega_0)} < 1$, so that $T_{\mathcal{V}}$ is a diffeomorphism [1, Lemma 6.13]. As a consequence of this requirement, we have that

$$\det(\mathbf{D}T_{\mathcal{V}})(x) > 0 \quad \forall x \in \mathbb{R}^d,$$

where $\mathbf{D}T_{\mathcal{V}}$ denotes the Jacobian of $T_{\mathcal{V}}$.

2.4 Shape optimization in parametric form

With the change of coordinates induced by $T_{\mathcal{V}}$, we can formulate the state equation (3) on the initial domain Ω_0, as in the following

$$\begin{cases} -\operatorname{div}(\mathbf{M}_{\mathcal{V}} \nabla u) = 0, & \textit{in } \Omega_0 \\ u = 1, & \textit{on } \partial\Omega_0^{in} \\ u = 0, & \textit{on } \partial\Omega_0^{out}, \end{cases} \tag{5}$$

where

$$\mathbf{M}_{\mathcal{V}} := \mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}T_{\mathcal{V}}^{-T} \left|\det(\mathbf{D}T_{\mathcal{V}})\right|. \tag{6}$$

This can be easily seen by noting that, given the map $\tilde{x} = F(x)$, it holds

$$\int_{\Omega} \nabla u(\mathbf{x}) \nabla v(\mathbf{x})\, d\mathbf{x} = \int_{\tilde{\Omega}} (\mathbf{D}F^{-T}\tilde{\nabla}u(\tilde{\mathbf{x}}))(\mathbf{D}F^{-T}\tilde{\nabla}v(\tilde{\mathbf{x}})) \left|\det \mathbf{D}F\right| \, d\tilde{\mathbf{x}},$$

where $\tilde{\nabla}$ denotes the gradient in $\tilde{\mathbf{x}}$ coordinates. In our case, we just have to substitute $\tilde{\Omega} = \Omega_0$ and $F = T_{\mathcal{V}}$, and note that the formula above appears in the weak formulation of the problem (5).

Similarly, the functional defined in (2) can be re-cast in a form that depends on the mapping $T_{\mathcal{V}}$ or, more precisely, on $\mathcal{V}$

$$\mathcal{J}(\mathcal{V}, u(\mathcal{V})) = \int_{\Omega_0} \nabla u \mathbf{M}_{\mathcal{V}} \nabla u \, d\mathbf{x} + \int_{\Omega_0} \left|\det(\mathbf{D}T_{\mathcal{V}})\right| d\mathbf{x}. \tag{7}$$

In this way, we have dropped the dependency of $\mathcal{J}$ on the shape itself Ω, substituting it with a parametrization described by $\mathcal{V}$.

The shape optimization problem (4) can then be reformulated in the following parametric form

$$\min_{\|\mathcal{V}\|_{C^2(\Omega_0)} \leq 1-\epsilon} \mathcal{J}(\mathcal{V}, u(\mathcal{V}))\ s.t.\ (5), \tag{8}$$

with $\mathcal{V}$ becoming our so-called *control variable.*

3 First order Fréchet derivative

In order to evaluate the first derivative of the cost functional (7), we need to provide relations to express the derivatives of some quantities of interest with respect to $\mathcal{V}$, along the direction $\tilde{\mathcal{V}}$. This will be denoted as

$$\delta_{\tilde{\mathcal{V}}}(*) = \left\langle d(*), \tilde{\mathcal{V}} \right\rangle.$$

Lemma 3.1 (Fréchet derivative of $\det(\mathbf{D}T_{\mathcal{V}})$)**.**

$$\delta_{\tilde{\mathcal{V}}}(\det(\mathbf{D}T_{\mathcal{V}})) = \det(\mathbf{D}T_{\mathcal{V}})\mathrm{tr}(\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\tilde{\mathcal{V}}). \tag{9}$$

Proof. Using the definition of the mapping $T_{\mathcal{V}}$ and of the Fréchet derivative along the direction $\tilde{\mathcal{V}}$, we have

$$\delta_{\tilde{\mathcal{V}}}(\det(\mathbf{D}T_{\mathcal{V}})) = \lim_{\epsilon \to 0} \frac{\det(\mathbf{D}T_{\mathcal{V}} + \epsilon\, \mathbf{D}\tilde{\mathcal{V}}) - \det(\mathbf{D}T_{\mathcal{V}})}{\epsilon}.$$

The first term of the numerator can then be rewritten as follows (I represents the identity matrix)

$$\begin{aligned}
\det(\mathbf{D}T_{\mathcal{V}} + \epsilon\, \mathbf{D}\tilde{\mathcal{V}}) &= \det((\mathbf{D}T_{\mathcal{V}})(I + \epsilon\, \mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\tilde{\mathcal{V}})) \\
&= \det(\mathbf{D}T_{\mathcal{V}}) \det(I + \epsilon\, \mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\tilde{\mathcal{V}}) \\
&= \det(\mathbf{D}T_{\mathcal{V}})(1 + \epsilon\, \mathrm{tr}(\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\tilde{\mathcal{V}})) + \mathcal{O}(\epsilon^2).
\end{aligned}$$

If we substitute this in the definition above, we get

$$\delta_{\tilde{\mathcal{V}}}(\det(\mathbf{D}T_{\mathcal{V}})) = \lim_{\epsilon \to 0} \frac{\cancel{\det(\mathbf{D}T_{\mathcal{V}})} + \epsilon\ \det(\mathbf{D}T_{\mathcal{V}})\mathrm{tr}(\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\tilde{\mathcal{V}}) - \cancel{\det(\mathbf{D}T_{\mathcal{V}})} + \mathcal{O}(\epsilon^2)}{\epsilon}.$$

Hence, the proof.

□

Lemma 3.2 (Fréchet derivative of $\mathbf{M}_\mathcal{V}$)**.**

$$\begin{aligned}\delta_{\tilde{\mathcal{V}}}\mathbf{M}_\mathcal{V} \;=\;& \det(\mathbf{D}T_\mathcal{V})\mathbf{D}T_\mathcal{V}^{-1}\Big[\mathrm{tr}(\mathbf{D}T_\mathcal{V}^{-1}\mathbf{D}\tilde{\mathcal{V}})I \\ & - \mathbf{D}\tilde{\mathcal{V}}\mathbf{D}T_\mathcal{V}^{-1} - \mathbf{D}T_\mathcal{V}^{-T}\mathbf{D}\tilde{\mathcal{V}}^T\Big]\mathbf{D}T_\mathcal{V}^{-T}.\end{aligned} \tag{10}$$

Proof. Starting from the definition of $\mathbf{M}_\mathcal{V}$ in (6) and differentiating it brings to

$$\begin{aligned}\delta_{\tilde{\mathcal{V}}}\mathbf{M}_\mathcal{V} &= \delta_{\tilde{\mathcal{V}}}(\mathbf{D}T_\mathcal{V}^{-1}\mathbf{D}T_\mathcal{V}^{-T}\det(\mathbf{D}T_\mathcal{V})) \\ &= \delta_{\tilde{\mathcal{V}}}(\mathbf{D}T_\mathcal{V}^{-1})\mathbf{D}T_\mathcal{V}^{-T}\det(\mathbf{D}T_\mathcal{V}) \\ &\quad +\mathbf{D}T_\mathcal{V}^{-1}\delta_{\tilde{\mathcal{V}}}(\mathbf{D}T_\mathcal{V}^{-T})\det(\mathbf{D}T_\mathcal{V}) \\ &\quad +\mathbf{D}T_\mathcal{V}^{-1}\mathbf{D}T_\mathcal{V}^{-T}\delta_{\tilde{\mathcal{V}}}(\det(\mathbf{D}T_\mathcal{V})).\end{aligned}$$

We need to recover an expression for the terms $\delta_{\tilde{\mathcal{V}}}(\mathbf{D}T_\mathcal{V}^{-1})$ and $\delta_{\tilde{\mathcal{V}}}(\mathbf{D}T_\mathcal{V}^{-T})$. Starting from the first one, we use once again the definition of derivative to get

$$\delta_{\tilde{\mathcal{V}}}(\mathbf{D}T_\mathcal{V}^{-1}) = \lim_{\epsilon\to 0}\frac{(\mathbf{D}T_\mathcal{V} + \epsilon\,\mathbf{D}\tilde{\mathcal{V}})^{-1} - \mathbf{D}T_\mathcal{V}^{-1}}{\epsilon}.$$

By re-writing the first term in the numerator as

$$\begin{aligned}(\mathbf{D}T_\mathcal{V} + \epsilon\,\mathbf{D}\tilde{\mathcal{V}})^{-1} &= ((\mathbf{D}T_\mathcal{V})(I + \epsilon\,\mathbf{D}T_\mathcal{V}^{-1}\mathbf{D}\tilde{\mathcal{V}}))^{-1} \\ &= (I + \epsilon\,\mathbf{D}T_\mathcal{V}^{-1}\mathbf{D}\tilde{\mathcal{V}})^{-1}(\mathbf{D}T_\mathcal{V})^{-1} \\ &= (I - \epsilon\,\mathbf{D}T_\mathcal{V}^{-1}\mathbf{D}\tilde{\mathcal{V}})(\mathbf{D}T_\mathcal{V})^{-1} + \mathcal{O}(\epsilon^2),\end{aligned}$$

and substituting it in the definition above, we get

$$\delta_{\tilde{\mathcal{V}}}(\mathbf{D}T_\mathcal{V}^{-1}) = \lim_{\epsilon\to 0}\frac{\cancel{\mathbf{D}T_\mathcal{V}^{-1}} - \epsilon\,\mathbf{D}T_\mathcal{V}^{-1}\mathbf{D}\tilde{\mathcal{V}}\mathbf{D}T_\mathcal{V}^{-1} - \cancel{\mathbf{D}T_\mathcal{V}^{-1}} + \mathcal{O}(\epsilon^2)}{\epsilon},$$

and finally

$$\delta_{\tilde{\mathcal{V}}}(\mathbf{D}T_\mathcal{V}^{-1}) = -\mathbf{D}T_\mathcal{V}^{-1}\mathbf{D}\tilde{\mathcal{V}}\mathbf{D}T_\mathcal{V}^{-1}. \tag{11}$$

Following a similar procedure, we recover an analogous result for the second term:

$$\delta_{\tilde{\mathcal{V}}}(\mathbf{D}T_\mathcal{V}^{-T}) = -\mathbf{D}T_\mathcal{V}^{-T}\mathbf{D}\tilde{\mathcal{V}}^T\mathbf{D}T_\mathcal{V}^{-T}. \tag{12}$$

It is possible now to use the relations (9), (11), (12) to rewrite $\delta_{\tilde{\mathcal{V}}}\mathbf{M}_\mathcal{V}$ as

$$\begin{aligned}\delta_{\tilde{\mathcal{V}}}\mathbf{M}_\mathcal{V} &= -\mathbf{D}T_\mathcal{V}^{-1}\mathbf{D}\tilde{\mathcal{V}}\mathbf{D}T_\mathcal{V}^{-1}\mathbf{D}T_\mathcal{V}^{-T}\det(\mathbf{D}T_\mathcal{V}) \\ &\quad -\mathbf{D}T_\mathcal{V}^{-1}\mathbf{D}T_\mathcal{V}^{-T}\mathbf{D}\tilde{\mathcal{V}}^T\mathbf{D}T_\mathcal{V}^{-T}\det(\mathbf{D}T_\mathcal{V}) \\ &\quad +\mathbf{D}T_\mathcal{V}^{-1}\mathbf{D}T_\mathcal{V}^{-T}\mathrm{tr}(\mathbf{D}T_\mathcal{V}^{-1}\mathbf{D}\tilde{\mathcal{V}})\det(\mathbf{D}T_\mathcal{V}),\end{aligned}$$

which, after some rearrangements, gives (10). □

Thanks to the relations proven above, we can provide an expression for the derivative of $\mathcal{J}$.

Proposition 1. *The first order Fréchet derivative of the cost functional* (7) *evaluated along the direction* $\tilde{\mathcal{V}}$ *is given by the formula*

$$\begin{aligned} \left\langle d\mathcal{J}(\mathcal{V}), \tilde{\mathcal{V}} \right\rangle &= \textstyle\int_{\Omega_0} \nabla\, u(\mathcal{V}) \delta_{\tilde{\mathcal{V}}} \mathbf{M}_{\mathcal{V}} \nabla\, u(\mathcal{V}) d\mathbf{x} \\ &\quad + \textstyle\int_{\Omega_0} \det(\mathbf{D}T_{\mathcal{V}}) \mathrm{tr}(\mathbf{D}T_{\mathcal{V}}^{-1} \mathbf{D}\tilde{\mathcal{V}}) d\mathbf{x}. \end{aligned} \tag{13}$$

Proof. We easily get

$$\begin{aligned} \left\langle d\mathcal{J}(\mathcal{V}), \tilde{\mathcal{V}} \right\rangle &= \langle \mathcal{J}_u, \delta_{\tilde{\mathcal{V}}} u \rangle + \left\langle \mathcal{J}_{\mathcal{V}}, \tilde{\mathcal{V}} \right\rangle \\ &= 2 \textstyle\int_{\Omega_0} \nabla\, u \mathbf{M}_{\mathcal{V}} \nabla\, \delta_{\tilde{\mathcal{V}}} u \; d\mathbf{x} \qquad \text{(A)} \\ &\quad + \textstyle\int_{\Omega_0} \nabla\, u \delta_{\tilde{\mathcal{V}}} \mathbf{M}_{\mathcal{V}} \nabla\, u \; d\mathbf{x} \\ &\quad + \textstyle\int_{\Omega_0} \delta_{\tilde{\mathcal{V}}} (\det(\mathbf{D}T_{\mathcal{V}}))\; d\mathbf{x}, \quad \text{(B)} \end{aligned} \tag{14}$$

where $\mathcal{J}_*$ denotes the first partial derivative of $\mathcal{J}$ with respect to the variable $*$.

We can immediately re-write term B by using (9), in order to get

$$\mathrm{B} = \int_{\Omega_0} \det(\mathbf{D}T_{\mathcal{V}}) \mathrm{tr}(\mathbf{D}T_{\mathcal{V}}^{-1} \mathbf{D}\tilde{\mathcal{V}}) d\mathbf{x}.$$

Let us focus now on term A, where it appears the sensitivity of the state u with respect to the control $\tilde{\mathcal{V}}$, *i.e.*,

$$\delta_{\tilde{\mathcal{V}}} u = \left\langle du(\mathcal{V}), \tilde{\mathcal{V}} \right\rangle.$$

We note that $\delta_{\tilde{\mathcal{V}}} u|_{\partial\Omega_0^{in}} = \delta_{\tilde{\mathcal{V}}} u|_{\partial\Omega_0^{out}} = 0$, since the value of u is fixed on the boundaries and it does not depend on $\mathcal{V}$. This implies that $\delta_{\tilde{\mathcal{V}}} u$ can be used as a test function for the weak form of (5), and hence, since u is a solution of the same equation,

$$\int_{\Omega_0} \nabla\, u \mathbf{M}_{\mathcal{V}} \nabla\, \delta_{\tilde{\mathcal{V}}} u \; d\mathbf{x} = 0.$$

Thus we can neglect the term A in (14) and retrieve the final expression (13). □

4 Second order Fréchet derivative

We denote by

$$\delta^2_{\tilde{\mathcal{W}},\tilde{\mathcal{V}}}(*) = \left\langle d^2(*)(\tilde{\mathcal{W}}), \tilde{\mathcal{V}} \right\rangle$$

the second derivative of a quantity $(*)$ evaluated along the directions $\tilde{\mathcal{W}}$ first and then $\tilde{\mathcal{V}}$.

Lemma 4.1 (Second order Fréchet derivative of $\mathbf{M}_{\mathcal{V}}$)**.**

$$\begin{aligned}\delta^2_{\tilde{\mathcal{W}},\tilde{\mathcal{V}}}\mathbf{M}_{\mathcal{V}} \;:=\;& \delta_{\tilde{\mathcal{V}}}\mathbf{M}_{\mathcal{V}}\mathrm{tr}(\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\tilde{\mathcal{W}}) - (\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\tilde{\mathcal{W}})\delta_{\tilde{\mathcal{V}}}\mathbf{M}_{\mathcal{V}} - \delta_{\tilde{\mathcal{V}}}\mathbf{M}_{\mathcal{V}}(\mathbf{D}\tilde{\mathcal{W}}^T\mathbf{D}T_{\mathcal{V}}^{-T}) \\ &+ \det(\mathbf{D}T_{\mathcal{V}})\mathbf{D}T_{\mathcal{V}}^{-1}\Big[-\mathrm{tr}(\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\tilde{\mathcal{V}}\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\tilde{\mathcal{W}})\;I \\ &+ (\mathbf{D}\tilde{\mathcal{V}}\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\tilde{\mathcal{W}}\mathbf{D}T_{\mathcal{V}}^{-1}) + (\mathbf{D}T_{\mathcal{V}}^{-T}\mathbf{D}\tilde{\mathcal{W}}^T\mathbf{D}T_{\mathcal{V}}^{-T}\mathbf{D}\tilde{\mathcal{V}}^T)\Big]\,\mathbf{D}T_{\mathcal{V}}^{-T}.\end{aligned} \tag{15}$$

Proof. We start from (10) and then differentiate again to get

$$\begin{aligned}\delta^2_{\tilde{\mathcal{W}},\tilde{\mathcal{V}}}\mathbf{M}_{\mathcal{V}} &= \delta_{\tilde{\mathcal{V}}}(\delta_{\tilde{\mathcal{W}}}\mathbf{M}_{\mathcal{V}}) \\ &= \delta_{\tilde{\mathcal{V}}}\Big(\det(\mathbf{D}T_{\mathcal{V}})\mathbf{D}T_{\mathcal{V}}^{-1}\Big[\mathrm{tr}(\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\tilde{\mathcal{W}})I \\ &\quad - \mathbf{D}\tilde{\mathcal{W}}\mathbf{D}T_{\mathcal{V}}^{-1} - \mathbf{D}T_{\mathcal{V}}^{-T}\mathbf{D}\tilde{\mathcal{W}}^T\Big]\,\mathbf{D}T_{\mathcal{V}}^{-T}\Big) \\ &= \delta_{\tilde{\mathcal{V}}}(\det(\mathbf{D}T_{\mathcal{V}})\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}T_{\mathcal{V}}^{-T})\;\mathrm{tr}(\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\tilde{\mathcal{W}}) && \text{(A)} \\ &\quad -\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\tilde{\mathcal{W}}\;\delta_{\tilde{\mathcal{V}}}(\det(\mathbf{D}T_{\mathcal{V}})\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}T_{\mathcal{V}}^{-T}) && \text{(B)} \\ &\quad -\delta_{\tilde{\mathcal{V}}}(\det(\mathbf{D}T_{\mathcal{V}})\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}T_{\mathcal{V}}^{-T})\;\mathbf{D}\tilde{\mathcal{W}}^T\mathbf{D}T_{\mathcal{V}}^{-T} && \text{(C)} \\ &\quad +(\det(\mathbf{D}T_{\mathcal{V}})\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}T_{\mathcal{V}}^{-T})\;\delta_{\tilde{\mathcal{V}}}(\mathrm{tr}(\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\tilde{\mathcal{W}})) && \text{(D)} \\ &\quad -\delta_{\tilde{\mathcal{V}}}(\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\tilde{\mathcal{W}})\;\det(\mathbf{D}T_{\mathcal{V}})\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}T_{\mathcal{V}}^{-T} && \text{(E)} \\ &\quad -\det(\mathbf{D}T_{\mathcal{V}})\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}T_{\mathcal{V}}^{-T}\;\delta_{\tilde{\mathcal{V}}}(\mathbf{D}\tilde{\mathcal{W}}^T\mathbf{D}T_{\mathcal{V}}^{-T}). && \text{(F)}\end{aligned}$$

By using the definition of $\mathbf{M}_{\mathcal{V}}$ in (6), the factors $\delta_{\tilde{\mathcal{V}}}(\det(\mathbf{D}T_{\mathcal{V}})\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}T_{\mathcal{V}}^{-T})$ that appear in the terms A, B, C can be transformed to $\delta_{\tilde{\mathcal{V}}}\mathbf{M}_{\mathcal{V}}$, thus getting

$$\begin{aligned}\mathrm{A} &= \delta_{\tilde{\mathcal{V}}}\mathbf{M}_{\mathcal{V}}\mathrm{tr}(\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\tilde{\mathcal{W}}) \\ \mathrm{B} &= -(\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\tilde{\mathcal{W}})\delta_{\tilde{\mathcal{V}}}\mathbf{M}_{\mathcal{V}} \\ \mathrm{C} &= -\delta_{\tilde{\mathcal{V}}}\mathbf{M}_{\mathcal{V}}(\mathbf{D}\tilde{\mathcal{W}}^T\mathbf{D}T_{\mathcal{V}}^{-T}).\end{aligned}$$

Moreover, the linearity of the trace operator allows to commute tr with $\delta_{\tilde{\mathcal{V}}}$ in term D. All that is left to evaluate, then, are the factors

$$\delta_{\tilde{\mathcal{V}}}(\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\tilde{\mathcal{W}}) = \delta_{\tilde{\mathcal{V}}}(\mathbf{D}T_{\mathcal{V}}^{-1})\mathbf{D}\tilde{\mathcal{W}}$$

in D and E, and

$$\delta_{\tilde{\mathcal{V}}}(\mathbf{D}\tilde{\mathcal{W}}^T\mathbf{D}T_{\mathcal{V}}^{-T}) = \mathbf{D}\tilde{\mathcal{W}}^T\delta_{\tilde{\mathcal{V}}}(\mathbf{D}T_{\mathcal{V}}^{-T})$$

in F. For these, the relations (11) and (12) are used again. After some rearrangements, expression (15) is then recovered. $\square$

We are ready to provide an expression for the second derivative of $\mathcal{J}$.

Proposition 2. *The second order Fréchet derivative of the cost functional* (7) *evaluated along the directions $\tilde{\mathcal{W}}$ and $\tilde{\mathcal{V}}$ is given by the formula*

$$\begin{aligned}\left\langle d^2 J(\mathcal{V})\tilde{\mathcal{W}}, \tilde{\mathcal{V}}\right\rangle \;=\; & 2\int_{\Omega_0} \nabla\, u \delta_{\tilde{\mathcal{V}}}\mathbf{M}_{\mathcal{V}} \nabla\, \delta_{\tilde{\mathcal{W}}} u \; d\mathbf{x} \\ & + \int_{\Omega_0} \nabla\, u \delta^2_{\tilde{\mathcal{W}},\tilde{\mathcal{V}}}\mathbf{M}_{\mathcal{V}} \nabla\, u \; d\mathbf{x} \\ & + \int_{\Omega_0} \det\,(\mathbf{D}T_{\mathcal{V}})\mathrm{tr}(\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\tilde{\mathcal{W}})\mathrm{tr}(\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\tilde{\mathcal{V}}) \; d\mathbf{x} \\ & - \int_{\Omega_0} \det(\mathbf{D}T_{\mathcal{V}})\mathrm{tr}(\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\tilde{\mathcal{W}}\mathbf{D}T_{\mathcal{V}}^{-1}\mathbf{D}\tilde{\mathcal{V}}) \; d\mathbf{x}.\end{aligned} \tag{16}$$

Proof. Starting from the definition of $\mathcal{J}$ and differentiating twice gives the general formula

$$\begin{aligned}\left\langle d^2 \mathcal{J}(\mathcal{V})\tilde{\mathcal{W}}, \tilde{\mathcal{V}}\right\rangle \;=\; & \langle \mathcal{J}_{uu}\delta_{\tilde{\mathcal{W}}}u, \delta_{\tilde{\mathcal{V}}}u\rangle + \left\langle \mathcal{J}_{u\mathcal{V}}\tilde{\mathcal{W}}, \delta_{\tilde{\mathcal{V}}}u\right\rangle & \text{(A)} \\ + \; & \left\langle \mathcal{J}_{\mathcal{V}u}\delta_{\tilde{\mathcal{W}}}u, \tilde{\mathcal{V}}\right\rangle + \left\langle \mathcal{J}_{\mathcal{V}\mathcal{V}}\tilde{\mathcal{W}}, \tilde{\mathcal{V}}\right\rangle & \text{(B)} \\ + \; & \left\langle \mathcal{J}_u, \delta^2_{\tilde{\mathcal{W}},\tilde{\mathcal{V}}}u\right\rangle, & \text{(C)}\end{aligned}$$

where $\mathcal{J}_{\star *}$ denote the second partial derivative of $\mathcal{J}$ with respect first to $\star$ and then to $*$.

The last term C can easily be simplified with the same procedure applied to the sensitivity term in the first order derivative: since the value of u is fixed at the border, $\delta^2_{\tilde{\mathcal{W}},\tilde{\mathcal{V}}}u$ vanishes at $\partial\Omega_0$, and it can be used as a test function for (5). Applying $\mathcal{J}_u$ results in the weak form of the state equation, which then shows that C $= 0$.

Term A too reduces to 0. Considering the constraint induced by (5), *i.e.*,

$$e(\mathcal{V}, u(\mathcal{V})) = -\,\mathrm{div}(\mathbf{M}_{\mathcal{V}} \nabla\, u) = 0,$$

and by differentiating it with respect to $\mathcal{V}$ along the direction $\tilde{\mathcal{W}}$, we recover the expression [1]

$$e_u \delta_{\tilde{\mathcal{W}}}u + e_{\mathcal{V}}\tilde{\mathcal{W}} = -\,\mathrm{div}\,(\mathbf{M}_{\mathcal{V}} \nabla\, \delta_{\tilde{\mathcal{W}}}u) - \mathrm{div}\,(\delta_{\tilde{\mathcal{W}}}\mathbf{M}_{\mathcal{V}} \nabla\, u) = 0. \tag{17}$$

Together with the usual boundary condition on $\delta_{\tilde{\mathcal{W}}}u$, (17) can be written as the following PDE, also named *sensitivity equation*

$$\begin{cases} -\,\mathrm{div}\,(\mathbf{M}_{\mathcal{V}} \nabla\, \delta_{\tilde{\mathcal{W}}}u) - \mathrm{div}\,(\delta_{\tilde{\mathcal{W}}}\mathbf{M}_{\mathcal{V}} \nabla\, u) = 0 & \textit{in } \Omega_0 \\ \delta_{\tilde{\mathcal{W}}}u = 0 & \textit{on } \partial\Omega_0. \end{cases} \tag{18}$$

[1] This result is also a consequence of the implicit function theorem [15]. See for example [3, Par. 1.4.2].

It can be easily seen that, apart from a factor of 2, term

$$\mathrm{A} = 2\int_{\Omega_0} \left[\nabla\, \delta_{\tilde{\mathcal{W}}} u \mathbf{M}_{\mathcal{V}}\, \nabla\, \delta_{\tilde{\mathcal{V}}} u + \nabla\, u \delta_{\tilde{\mathcal{W}}} \mathbf{M}_{\mathcal{V}}\, \nabla\, \delta_{\tilde{\mathcal{V}}} u\right]\, d\mathbf{x},$$

corresponds to the weak formulation of (18), where $\delta_{\tilde{\mathcal{V}}} u$ has been used as a test function. Since $\delta_{\tilde{\mathcal{W}}} u$ must satisfy (18), then, A = 0.

This further simplifies the expression for the second order Fréchet derivative of $\mathcal{J}$, which is now reduced to the sole term B. Using again the relations (9), (11) and (12), the proof is concluded.

□

5 Descent method

In order to solve the shape optimization problem (8), a Descent method has been set up [3, chap. 2.2]. The basic form of the algorithm is analogous to the one presented in [6] and is described in Algorithm 1.

Algorithm 1: Generic descent method

```
1  Set parameters $\gamma \in (0,1)$, $\epsilon > 0$, $it_{max}$;
2  Start from an initial design $\Omega_0$;
3  Initialize $\tilde{\mathcal{V}} = 0, \tilde{\mathcal{V}}^{temp} = 0$;
4  for $k$=0 to $it_{max}$ do
5      Evaluate solution $u(\tilde{\mathcal{V}}^{temp})$ of (5);
6      Evaluate cost functional at current position:
       $\mathcal{J}^{new} = \mathcal{J}(\tilde{\mathcal{V}}^{temp}, u(\tilde{\mathcal{V}}^{temp}))$;
7      if $(k > 0)$ and $\left(\mathcal{J}^{new} - \mathcal{J}^{old} > \gamma\tilde{\sigma}^{old}\left\langle d\mathcal{J}(\tilde{\mathcal{V}}), \tilde{\mathcal{V}}^{new}\right\rangle\right)$ then
8          Halve step size: $\tilde{\sigma}^{new} = \tilde{\sigma}^{old}/2$;
9      else
10         Set $\tilde{\mathcal{V}} = \tilde{\mathcal{V}}^{temp}$, $\mathcal{J}^{old} = \mathcal{J}^{new}$;
11         Evaluate gradient at current position: $d\mathcal{J}(\tilde{\mathcal{V}})$;
12         Find a descent direction: $\tilde{\mathcal{V}}^{new}$;
13         Find a step size: $\tilde{\sigma}^{new}$;
14     Set $\tilde{\sigma}^{old} = \tilde{\sigma}^{new}$;
15     Update map: $\tilde{\mathcal{V}}^{temp} = \tilde{\mathcal{V}} + \tilde{\sigma}^{new}\tilde{\mathcal{V}}^{new}$;
16     while $(\min(\det(\mathbf{D}T_{\tilde{\mathcal{V}}^{temp}})(\Omega_0)) < \epsilon)$ do
17         Halve step size: $\tilde{\sigma}^{new} = \tilde{\sigma}^{old}/2$;
18         Set $\tilde{\sigma}^{old} = \tilde{\sigma}^{new}$;
19         Update map: $\tilde{\mathcal{V}}^{temp} = \tilde{\mathcal{V}} + \tilde{\sigma}^{new}\tilde{\mathcal{V}}^{new}$;
```

The condition $\min(\det(\mathbf{D}T_{\mathcal{V}^{temp}})(\Omega_0)) > \epsilon$ must be fulfilled for the map $T_{\mathcal{V}}$ to be a diffeomorphism. The condition that requires to halve the step size in case $\mathcal{J}^{new} - \mathcal{J}^{old} > \gamma\tilde{\sigma}^{old}\left\langle d\mathcal{J}(\tilde{\mathcal{V}}), \tilde{\mathcal{V}}^{new}\right\rangle$, is known as *Armijo rule* [3, Par.2.2.1.1], and ensures that the chosen step sizes are admissible.

The methods used to recover the descent direction $\tilde{\mathcal{V}}^{new}$ and the step size $\tilde{\sigma}^{new}$ determine the algorithm. In this work, three different approaches have been used, whose definition is given next.

5.1 Steepest Descent

As a first approximation, a representative of the gradient $d\mathcal{J}(\tilde{\mathcal{V}})$ is chosen as a descent direction, and the new step size $\tilde{\sigma}^{new}$ depends on the one at the previous iteration. In this study, an initial step size of $\bar{\sigma} = 0.3$ is used. Lines 12 and 13 in Algorithm 1 are then modified as in Algorithm 2.

Algorithm 2: Steepest Descent: modifications to Algorithm 1

```
12a Recover $H^1$ representative of $d\mathcal{J}(\tilde{\mathcal{V}})$, *i.e.*,
    $\nabla \mathcal{J}_{H^1} = \underset{\|\mathcal{V}\|_{H^1(\Omega_0)}=1}{\operatorname{argmin}} \left\langle d\mathcal{J}(\tilde{\mathcal{V}}), \mathcal{V}\right\rangle$;
12b Set $\tilde{\mathcal{V}}^{new} = -\nabla \mathcal{J}_{H^1}$;
13  Set the new step size as: $\tilde{\sigma}^{new} = 2\tilde{\sigma}^{old}$;
```

The choice of the H^1 representative of the gradient is somewhat arbitrary, but indeed it shows a better regularity than their Euclidean or L^2 counterparts.

5.2 Employing second order derivative

This approach aims to recover a better approximation of the cost functional (7) via Taylor expansion by employing information regarding its second order directional derivative (16). This allows us to choose an optimal step size at each iteration by solving a (trivial) minimization problem. In this case, step 13 in Algorithm 2 is replaced as in Algorithm 3.

We point out that in order to recover the full second derivative it is necessary to solve the sensitivity equation (18), whose right-hand side depends on the direction $\tilde{\mathcal{V}}^k$. As an alternative, one can neglect the term of (16) that depends on the sensitivity $\delta_{\tilde{\mathcal{W}}}u$, and employ just partial information of the second derivative, thus avoiding the need to solve and additional PDE.

All three the variants described have been tested for efficiency and computational cost. The results are discussed in sections 7 and 8.

```
Algorithm 3: Higher order information: modifications to Algorithm 2
13a Evaluate first derivative of the cost functional along $\tilde{\mathcal{V}}^{new}$:
    $b = \left\langle d\mathcal{J}, \tilde{\mathcal{V}}^{new} \right\rangle$;
13b Evaluate second derivative of the cost functional along $\tilde{\mathcal{V}}^{new}$:
    $a = \left\langle d^2\mathcal{J}(\tilde{\mathcal{V}}^{new}), \tilde{\mathcal{V}}^{new} \right\rangle$;
13c Find the step size $\tilde{\sigma}^{new}$ by solving: $\tilde{\sigma}^{new} = \operatorname*{argmin}_{\sigma} \ \frac{1}{2}\sigma^2 a + \sigma b$;
```

6 Discretization aspects

In order to recover a numerical solution of the problem (8), discrete approximations for both the state variable u and the control variable $\mathcal{V}$ need to be given. The details are described in the following.

State A Finite-Elements approach [17, Chap.4] is used to provide a discretization of the state variable $u_h \approx u$. To do this, a set of piecewise linear Lagrangian basis functions ϕ_i is constructed on a triangular grid on the domain Ω_0. These functions are defined such that $\phi_i(x_j) = \delta_{i,j}$ for each node x_j of the mesh, where δ_{ij} is the Kronecker delta. Any finite-element function can then be expressed as a linear combination of these basis functions,

$$u_h = \sum_{i=1}^{n} u_i \phi_i, \tag{19}$$

where n is the number of mesh nodes (which corresponds to the number of basis functions).

Boundary value problems like (5) or (18) can be approximated via a Galerkin method (again, [17, Chap.4]), which in our case gives rise to a linear system in the form $A\mathbf{u} = \mathbf{b}$, with $(\mathbf{u})_i = u_i$, the coefficients in (19). For example, if we consider (5), we have

$$A_{ij} = \int_{\Omega_0} \nabla\, \phi_j \mathbf{M}_{\mathcal{V}} \nabla\, \phi_i \; d\mathbf{x}, \tag{20}$$

while $\mathbf{b} = 0$ (before boundary conditions are applied). A is also called *stiffness matrix.* For (18), we have instead $(\mathbf{b})_i = -\int_{\Omega_0} \nabla\, u \mathbf{M}_{\mathcal{V}} \nabla\, \phi_i \; d\mathbf{x}$.

Control A similar approach is used also to discretize the control variable $\mathcal{V}$. We introduce a uniform Cartesian grid that covers the domain Ω_0, and on it we define a set of m cubic B-splines basis functions, ψ_i [16, Chap.7.6]. We consider discrete vector fields $\mathcal{V}_h$ in the form

$$\mathcal{V}_h = \sum_{i=1}^{m} \psi_i \sum_{d=1}^{2} \nu_{i,d} \mathbf{e_d}, \tag{21}$$

with $\mathbf{e_d}$ being the basis vectors of $\mathbb{R}^2$. We point out that cubic B-splines basis functions have a compact support that spans only $4 * 4 = 16$ grid cells: this property has some remarkable effects on the algorithms for the evaluation of the derivatives of $\mathcal{J}$, as explained in section 7.

7 Cost analysis

The most relevant functions of the algorithm are those responsible of solving the PDEs (5) and (18), and those evaluating the first (13) and second (16) directional derivatives of the cost functional $\mathcal{J}$.

PDE solver For solving the State and Sensitivity equations, functions from the library LehrFEM have been used in order to assemble the matrix related to the problem at hand, as described in section 6. Build-in Matlab functions for the solution of the resulting linear systems are employed to retrieve the approximations u_h and $\delta_{\tilde{\mathcal{V}}} u_h$.

The effort required to solve the systems varies depending on the properties of matrix (20). A is symmetric and sparse, because $\mathbf{M}_{\mathcal{V}}$ is symmetric and the support of any ϕ_i is bounded. Therefore, the order of the computational cost for solving (5) and (18) lies in between $\mathcal{O}(b^2 n)$, for a banded symmetric linear system with bandwidth b [18], and $\mathcal{O}(n^{\frac{3}{2}})$ for a generic sparse symmetric matrix [19], given n Finite-Elements basis functions.

The solution u of (5) is used both for retrieving the value of the functional (7) at the current iteration and for building up the operator $d\mathcal{J}$. The solution $\delta_{\tilde{\mathcal{W}}} u$ of (18), is used instead to recover the value of $d^2\mathcal{J}$ along the desired descent direction, in the case where full information of the second directional derivative is employed.

Derivatives of $\mathcal{J}$ In order to retrieve a representative of the operator $d\mathcal{J}$, we need to evaluate (13) for every basis vector field $\psi_i \mathbf{e_d}$. This requires an effort that grows linearly with the number of triangles that compose the mesh discretizing the domain Ω_0, since for any of those the integrand must be evaluated on a fixed number of quadrature points [20]. On the other hand, the dependency on the number m of B-splines basis functions is not so trivial. In our Matlab code, in order to exploit vectorization, the evaluation of $d\mathcal{J}$ leads to the construction of a matrix in the form

$$B_{kj} = \sum_{q=1}^{n_Q} \sum_{i=1}^{n_B} \omega_q \nabla \phi_i(\mathbf{x_{q,k}}) u_i \delta_{\psi_j} \mathbf{M}_{\mathcal{V}}(\mathbf{x_{q,k}}) \nabla \phi_i(\mathbf{x_{q,k}}) u_i, \qquad (22)$$

where ω_q is the Gaussian weight for the quadrature point q, n_Q is the number of quadrature points per triangle, $\mathbf{x_{q,k}}$ is the position of the quadrature point q in the triangle k, and n_B is the number of FEM basis functions with support on a triangle (in our case, where piecewise linear ϕ_i are used, $n_B = 3$ for every triangle k). B is then an $n \times m$ matrix, but since the support of ψ_i is limited (as described in section 6), it is also sparse. In fact, for each row of (22) (meaning, for each triangle k), the number of non-zero elements equals the number of splines whose support contains at least one of the quadrature points of k. The complexity of the algorithm directly depends on the number of non-zero elements of B. This in its turn depends on the mutual arrangement of the quadrature points and the support of the splines basis functions. The more the quadrature points in each element are close to each other, the higher is the chance of them all residing in the support of the same spline basis functions, hence the fewer non-zero elements (22) has. Vice-versa, if the quadrature points are spread apart in the triangle, then increasing the number of splines basis functions ends up in increasing the number of non-zero elements of (22). An example of this is shown in Fig.2.

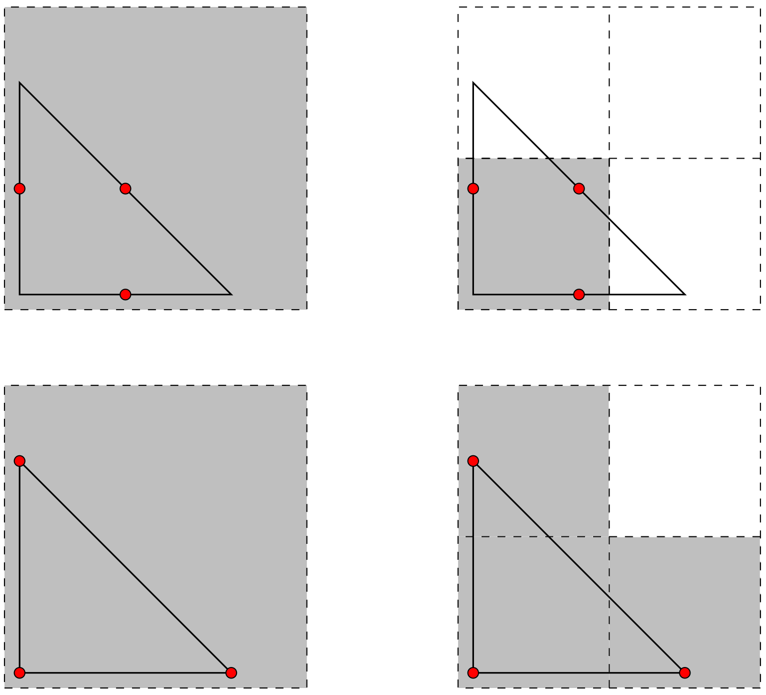
Figure 2: Depending on how the quadrature points (red dots) are spread apart in the triangle, refining the B-splines grid (squared dashed line) might increase the amount of B-splines basis functions ψ_i whose support contains at least one quadrature point. In the case described in the top row, refining the mesh (moving from left to right) does not change the number of cells that contain a quadrature point (gray squares), but it does in the case below.

Fig.4 describes the effect that the disposition of the quadrature points has on the sparsity of (22). There, the change in the amount of non-zero elements for different spline grid refinement levels is reported for various quadrature schemes. A sketch of the disposition of the quadrature points for each scheme considered is shown in Fig.3.

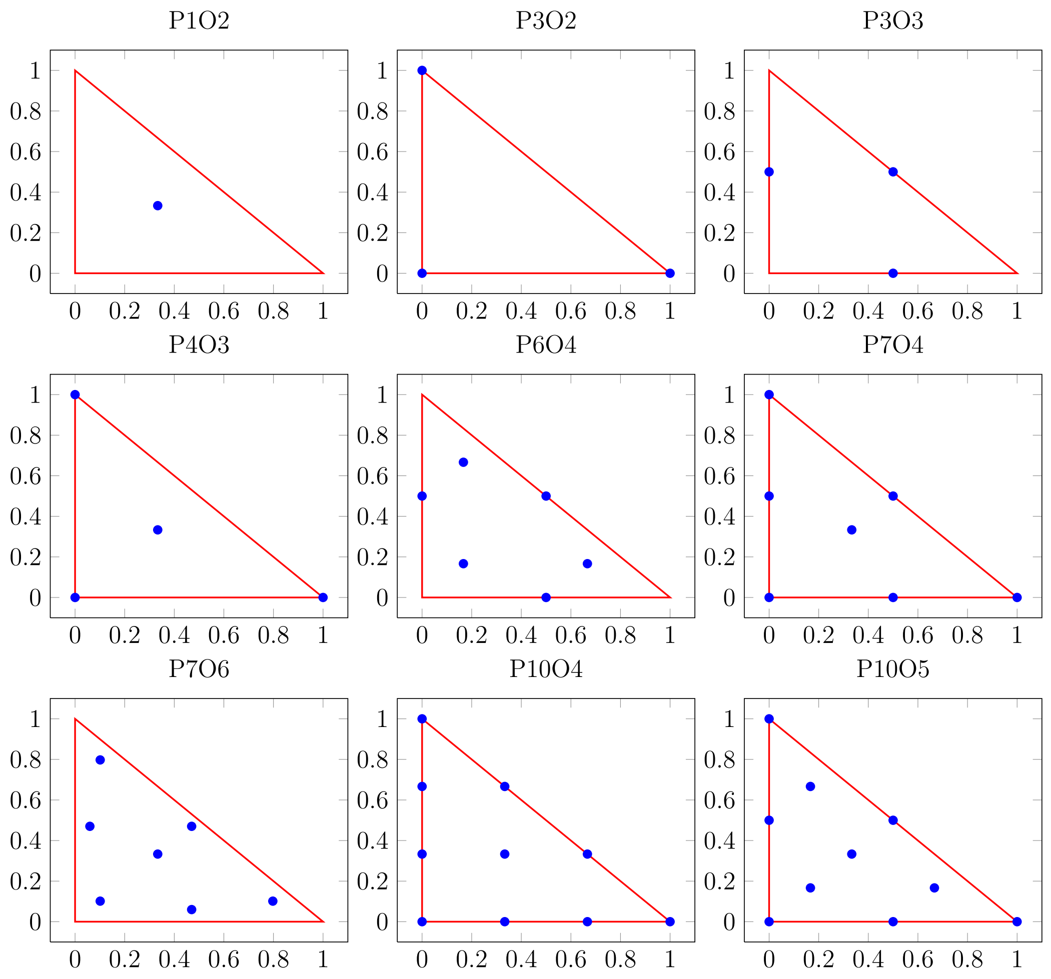

Figure 3: Arrangement of quadrature points inside a reference triangle for various quadrature schemes. The schemes are labelled as as "PnOm", according to the LehrFEM library manual: here, n stands for the number of points used, while m defines the order of accuracy.

It is noticeable how, in the case a simple one-point quadrature scheme is used (P1O2), the sparsity of the matrix (22) is not affected by an increase in the amount of splines basis functions. That is due to the fact that, given a fixed point, the number of basis functions whose support includes that point is fixed, and only depends on the order of splines used[2].

[2]As pointed out in section 6, the support of each basis function covers $4 * 4 = 16$ cells. This also means that, for any given point belonging to a specific cell, then 16 basis functions have a support that contains it. The number of non-zero elements of (22), in this case, is in fact $198 * 16 = 3168$, since 198 elements were used.

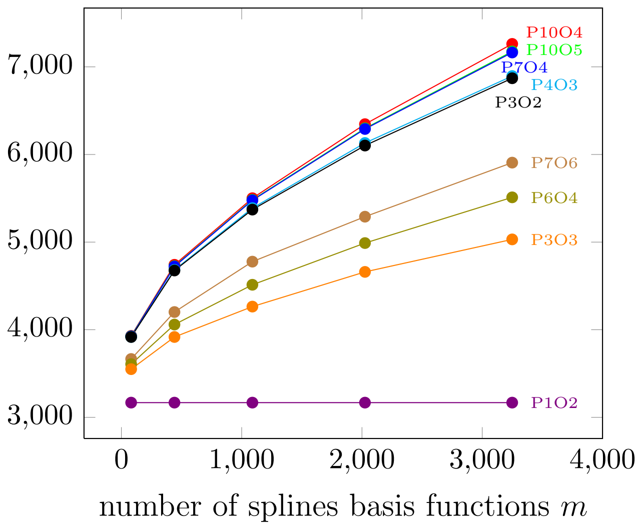

Figure 4: Number of non-zero elements of (22) for different splines refinement levels, using various quadrature schemes. The number of elements is kept fixed to $n = 198$. Please refer to Fig.3 for a description of the quadrature points positions.

On the other hand, the algorithm for the evaluation of the second order derivative exhibits a different behaviour. Analogously to what has been described before, an approximation of the integrand in (16) must in fact be recovered for each quadrature point in every triangle. In particular, this requires the computation of terms like $\mathbf{D}\tilde{\mathcal{V}}_h \approx \mathbf{D}\tilde{\mathcal{V}}$. However, we evaluate the second derivative along a fixed direction. Thus, for each quadrature point, $\mathbf{D}\tilde{\mathcal{V}}_h$ can be computed as a linear combination of the derivatives of the basis vector fields $\mathbf{D}(\psi_i \mathbf{e_d})$. This allows us to get rid of the dependency of the computational cost of the algorithm on the actual number of B-splines m, because the number of those whose support intersect a quadrature point is constant. This is shown in Fig.5, where an averaged timing of the algorithm is provided, for different numbers of finite elements n and splines m used.

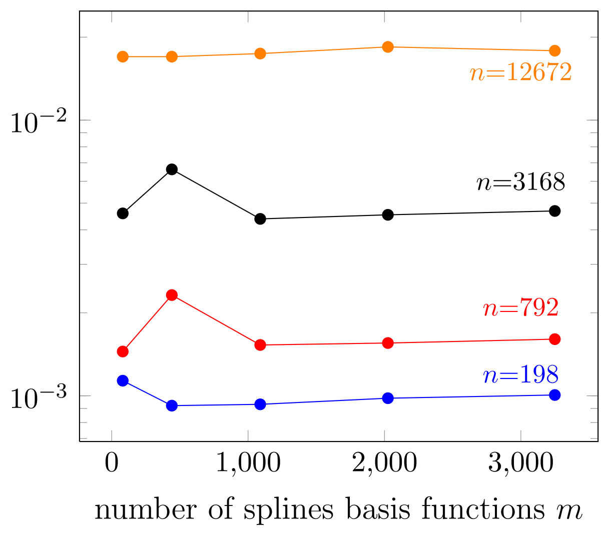

Figure 5: Average execution times of the function responsible for the evaluation of the second order directional derivative (16). Experiment conducted for different number of elements n and splines m. Apart from some noise for the small mesh sizes, the execution time keeps around the same level even though the number of splines basis functions used increases, as expected from the analysis conducted. The quadrature scheme used is P3O3.

8 Convergence rates

As additional information is used in the algorithm, one would expect the convergence rates to improve accordingly, meaning that the algorithm employing the full evaluation of (16) behaves better than the one that makes use of its linear part only, which again is better than the simple Steepest Descent algorithm. A plot of the convergence results can be seen in Fig.6. Here, the three methods described in Sec.5 have been tested, and the evolution of their respective normalized error is reported.

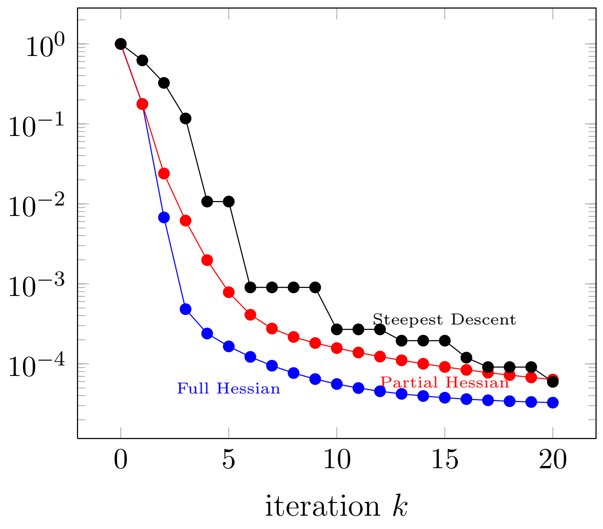

Figure 6: Convergence history of normalized error of cost functional, $\left|\frac{\mathcal{J}^k - \mathcal{J}_{min}}{\mathcal{J}^0}\right|$, for the three different methods. $\mathcal{J}_{min}$ is the value of the cost functional at the exact solution, $\mathcal{J}^k$ is $\mathcal{J}$ evaluated at iteration k.

As expected, the algorithm that employs full information from the second order derivative converges much faster than the others, followed by the one that neglects the non-linear part of (16). Also, for the Steepest Descent method, Armijo rule is often applied, as represented by the "flat" lines in the graph. In our experiments, this does not happen with the other two algorithms.

Given the short amount of iterations before saturation is reached, it is not simple to assess the actual order of convergence of each method. During the first iterations, the algorithm that employs full second order information shows some steepening in the convergence curve slope, which might indicate superlinear convergence. It is not the case for the one that uses only partial information, which tends to preserve the same slope until saturation. The behaviour of the Steepest Descent algorithm is instead widely influenced by the application of Armijo rule.

9 Further Work

More complex algorithms for the solution of the problem considered (8) can be investigated. For example, if the whole operator $d^2\mathcal{J}$ is available, then a Newton method can be applied in order to retrieve a more accurate descent direction [3, Chap.2]. Such methods look for the minimum of a second order approximation of (7) by solving the equation

$$\left\langle d^2\mathcal{J}(\mathcal{V}^k), s^k \right\rangle = -d\mathcal{J}(\mathcal{V}^k), \tag{23}$$

and use this to update the solution to (8),

$$\mathcal{V}^{k+1} = \mathcal{V}^k + s^k.$$

However, as the dimension of the problem increases, storing the whole matrix representing the approximation of the operator $d^2\mathcal{J}$ could let some memory issues arise. Moreover, first experiments in this direction show that $d^2\mathcal{J}$ present some ill-conditioning, which makes it necessary to apply some stabilization techniques in order to correctly retrieve a solution to (23).

10 Conclusion

This work is inserted in the set-up of [6]. It aims to expand it by comparing the properties of various Descent Methods applied to the optimization algorithm used to solve (8). In order to access more refined numerical schemes, a formula for the second order derivative of the cost functional (16) is derived. The methods considered use various degree of information from both first and second order derivatives, in order to retrieve appropriate descent directions and step sizes for the algorithm to proceed. Both theoretical and empirical computational costs have been investigated, directing particular attention to their dependency on the number of finite elements and splines basis functions employed. By the analysis of those, it emerges that the choice of employing the full second order directional derivative results in higher accuracy at a relatively small cost. Evaluating (16), in fact, does not increase the degree of complexity necessary for (13), but it does provide a much faster convergence of the error with respect to a basic Steepest Descent algorithm.

References

[1] G. Allaire. *Conception optimale de structures*, vol. 58 of Mathématiques & Applications (Berlin) [Mathematics & Applications], Springer-Verlag, Berlin, 2007. With the collaboration of Marc Schoenauer (INRIA) in the writing of Chapter 8.

[2] K. Eppler, H. Harbrecht. Efficient treatment of stationary free boundary problems. *Applied Numerical Mathematics*, Vol. 56, pp. 13261339. Elsevier, 2006.

[3] M. Hinze, R. Pinnau, M. Ulbrich, S. Ulbrich. *Optimization with PDE constraints.* Springer, New York, 2009.

[4] M. Flucher. Bernoullis Free-boundary Problem *Variational problems with concentration.* Birkhuser Mathematics, 1999.

[5] R. Hiptmair, A. Paganini, S. Sargheini. Comparison of approximate shape gradients. *BIT Numer Math DOI 10.1007/s10543-014-0515-z*, July 2014.

[6] R. Hiptmair, A. Paganini. Shape optimization by pursuing diffeomorphisms. *ETH SIAM, Research Report No. 2014-27*, December 2014.

[7] F. Ballarin, A. Manzoni, G. Rozza, and S. Salsa. Shape optimization by free-form deformation: existence results and numerical solution for Stokes flows. *J. Sci. Comput., 60*, December 2013.

[8] N.H. Kim, Y. Chang. Eulerian shape design sensitivity analysis and optimization with a fixed grid. *Comput. Methods Appl. Mech. Engrg. 194 3291-3314*, December 2004.

[9] Z Liu, J.G. Korvink. Structural Shape Optimization Using Moving Mesh Method. *Excerpt from the Proceedings of the COMSOL Users Conference 2007 Grenoble*, 2007.

[10] M.L. Staten, S.J. Owen, S.M. Shontz, A.G. Salinger, and T.S. Coffey. A Comparison of Mesh Morphing Methods for 3D Shape Optimization.

[11] M. Flucher, M. Rumpf. Bernoulli's free-boundary problems, qualitative theory and numerical approximation . *Journal fr die reine und angewandte Mathematik (1997). Vol: 486, pp. 165-204*, 1997.

[12] K. Eppler and H. Harbrecht. Shape Optimization for Free Boundary Problems -Analysis and Numerics. *Deutsche Forschungsgemeinschaft, Priority Program 1253, Optimization with Partial Differential Equations*, February 2010.

[13] J.I. Daz, J.F. Padial, J.M. Rakotoson. On some Bernoulli free boundary type problems for general elliptic operators. The Royal Society of Edinburgh, October 2007.

[14] H. Schwerdtfeger. *Introduction to linear algebra and the theory of matrices.* Noordhoff, 1950

[15] S. Kumagai. An implicit function theorem: Comment. *Journal of Optimization Theory and Applications 31 (2): pp. 285-288*, June 1980

[16] A. Quarteroni, R. Sacco, F. Salieri. *Modellistica Numerica per Problemi Differenziali.* Springer, 2012

[17] A. Quarteroni. *Matematica Numerica.* Springer, 2008.

[18] D. Zhang, E. Polizzi. Linear scaling techniques for first-principle calculations of large nanowire devices. *NSTI-Nanotech 2008, www.nsti.org, ISBN 978-1-4200-8503-7 Vol. 1.* 2008

[19] J.H. Reif. Efficient approximate solution of sparse linear systems. *Computers and Mathematics with applications, Vol. 36, No. 9, pp. 37-58.* November 1998

[20] F. Hussain, M.S. Karima, R. Ahamada. Appropriate Gaussian quadrature formulae for triangles. *International Journal of Applied Mathematics and Computation, Vol. 4, pp. 24-38.* 2012