Second-order shape derivatives along normal trajectories, governed by Hamilton-Jacobi equations

Grégoire Allaire, Eric Cancès, Jean-Léopold Vie

Second-order shape derivatives along normal trajectories, governed by Hamilton-Jacobi equations

G. Allaire *
CMAP, UMR CNRS 7641
École Polytechnique,
Université Paris-Saclay, 91128 Palaiseau, FRANCE
(gregoire.allaire@polytechnique.fr)

E. Cancès
CERMICS, École des Ponts and INRIA,
Champs-sur-Marne, FRANCE
(cances@cermics.enpc.fr)

J.-L. Vié
CERMICS, École des Ponts,
Champs-sur-Marne, FRANCE
(viej@cermics.enpc.fr)

June 5, 2016

Dedicated to the memory of George Rozvany.

Abstract

In this paper we introduce a new variant of shape differentiation which is adapted to the deformation of shapes along their normal direction. This is typically the case in the level-set method for shape optimization where the shape evolves with a normal velocity. As all other variants of the orginal Hadamard method of shape differentiation, our approach yields the same first order derivative. However, the Hessian or second-order derivative is different and somehow simpler since only normal movements are allowed. The applications of this new Hessian formula are twofold. First, it leads to a novel extension method for the normal velocity, used in the Hamilton-Jacobi equation of front propagation. Second, as could be expected, it is at the basis of a Newton optimization algorithm which is conceptually simpler since no tangential displacements have to be considered. Numerical examples are given to illustrate the potentiality of these two applications. The key technical tool for our approach is the method of bicharacteristics for solving Hamilton-Jacobi equations. Our new idea is to differentiate the shape along these bicharacteristics (a system of two ordinary differential equations).

Key words: shape and topology optimization, level-set method, second-order shape derivative, Newton method.

1 Introduction

Differentiation with respect to a shape (an open subset of $\mathbb{R}^d$ with $d = 2$ or 3) is a key tool in shape optimization, that was first introduced by Hadamard [17]. It was then widely developed by many authors [1], [14], [18], [22], [23], [32], [35] (and references therein). There are two variants of the Hadamard method of shape differentiation. The first one, advocated by Murat and Simon [23, 32, 33], is based on a parametrization of shapes by displacement vector fields. Given a reference open set Ω of $\mathbb{R}^d$, a variation of this domain is of the type $\Omega_\theta = \big(\mathrm{Id} + \theta\big)(\Omega)$, where θ is a vector field from $\mathbb{R}^d$ into $\mathbb{R}^d$ and Id is the identity operator on $\mathbb{R}^d$. In other words, any point $x \in \Omega$ is moved to a new position $x + \theta(x) \in \Omega_\theta$. In this context, shape differentiation is defined as differentiation with respect to the vector field θ. The second approach is the so-called speed method, introduced by Zolésio and co-workers [13, 15, 35, 40], which is based on the flow of a vector field and on shapes evolving along this flow. For a given vector field $V(t, x)$, defined from $\mathbb{R}_+ \times \mathbb{R}^d$ into $\mathbb{R}^d$, consider the solution (or flow) of the ordinary differential equation

$$\left\{ \begin{array}{l} \dfrac{\partial X_V}{\partial t}(t, x) = V\big(t, X_V(t, x)\big) \quad \text{for } t > 0\,, \\ X_V(0, x) = x\,. \end{array} \right. \tag{1}$$

Then the variation of the reference domain Ω is defined, for $t \geq 0$, as $\Omega_t = X_V(t, \Omega) = \{X_V(t, x) \text{ such that } x \in \Omega\}$. In this context, shape differentiation is defined as the derivative with respect to t and it is a directional derivative in the direction of V. These two variants of the Hadamard method of shape differentiation lead to the same notion of first order derivative by identifying the vectors fields $\theta(\cdot)$ and $V(0, \cdot)$ but to slightly different second-order derivatives.

*G. A. is a member of the DEFI project at INRIA Saclay Ile-de-France. This work has partially been supported by the RODIN project (FUI AAP 13).

There are strong connections between them in the sense that results obtained with one method can be translated to similar results for the other one.

The goal of the present paper is to define a third approach of Hadamard shape differentiation by considering again a family of shapes Ω_t, evolving with time $t \geq 0$ in the direction of the normal vector $\mathbf{n}(t)$ to the boundary $\partial\Omega_t$. This new approach is especially suited to the level-set method for shape and topology optimization [2], [3], [38] where the shape is indeed advected in its normal direction by solving an eikonal or Hamilton-Jacobi equation. Following the lead of Osher and Sethian [27], a shape Ω is represented by a level-set function ϕ which, by definition, satisfies

$$\left\{ \begin{array}{ll} x \in \Omega & \text{iff } \phi(x) < 0, \\ x \in \partial\Omega & \text{iff } \phi(x) = 0, \\ x \in \mathbb{R}^d \setminus (\Omega \cup \partial\Omega) & \text{iff } \phi(x) > 0. \end{array} \right. \tag{2}$$

In other words, the boundary of Ω is given by the zero level-set $\{x \in \mathbb{R}^d \mid \phi(x) = 0\}$. Let $v(t,x)$ be a smooth function from $\mathbb{R}_+ \times \mathbb{R}^d$ into $\mathbb{R}$. Evolving the shape Ω_t with a normal velocity v is equivalent to solving the following eikonal or Hamilton-Jacobi equation

$$\left\{ \begin{array}{l} \dfrac{\partial \varphi}{\partial t}(t,x) + v(t,x)\,|\nabla_x \varphi(t,x)| = 0, \\ \varphi(0,x) = \phi(x). \end{array} \right. \tag{3}$$

For $t \geq 0$, the shape Ω_t is recovered as the set of negative values of $\varphi(t,\cdot)$, namely $\Omega_t = \{x \in \mathbb{R}^d \mid \varphi(t,x) < 0\}$. Here we give a rigorous definition of shape differentiation for such evolutions. The key ingredient is the method of bicharacteristics for solving (3) when its solution φ is smooth. It is well known [28] that smooth solutions of (3) can be computed by solving the following system of two ordinary differential equations

$$\left\{ \begin{array}{rcl} \dfrac{dx}{dt}(t) & = & \nabla_p H\big(t, x(t), p(t)\big), \\ \dfrac{dp}{dt}(t) & = & -\nabla_x H\big(t, x(t), p(t)\big), \end{array} \right. \tag{4}$$

where $H(t,x,p) = v(t,x)|p|$ is the Hamiltonian defined on $\mathbb{R}_+ \times \mathbb{R}^d \times \mathbb{R}^d$. For smooth evolutions[1], an equivalent definition of Ω_t is then $\partial\Omega_t = \{x(t) \text{ solution of (4) with } x(0) \in \partial\Omega\}$. By using (4) we can introduce a new definition of shape derivative which is different from the two previous ones (note that (4) cannot be put under the simpler form (1)). It turns out that the first order shape derivatives for all three variants are the same and it is only the second-order shape derivative which is different and simpler since it does not involve any tangential displacement of the boundary as in the two previous methods. Our main theoretical result is Theorem 3.4.

There are two main applications of the knowledge of second-order derivatives. The most obvious one is the definition of the Newton's algorithm. We explore this issue in Section 5. We are not the first ones to study second-order shape derivatives and the Newton algorithm. Let us mention the stability results in [10], [11], and the numerical algorithms in [26], [29]. Much more is even known in the context of control theory [19]. Another application of second-order derivative is presented in Section 4 for the first time, to the best of our knowledge. It turns out to be useful for the extension of a descent direction, merely known on the boundary $\partial\Omega$, to the entire space $\mathbb{R}^d$, as required for solving the Hamilton-Jacobi equation (3). The main idea is that the structure of the second-order shape derivative gives a hint to the choice of the normal derivative of the extension on $\partial\Omega$. Our numerical experiments indicate that this idea leads to better extensions in the sense that the convergence of the optimization process is improved.

This paper is organized as follows. In Section 2, we recall the two usual frameworks for computing shape derivatives, namely domain perturbation (or displacement field method) [23, 32, 33] and time moving domain (or speed method) [6, 13, 15, 35], and the respective structures of the second-order shape derivatives. Section 3 describes the proposed new setting of shape differentiation when the shape evolves in its normal direction which is the typical case in the level-set method for shape optimization. In Section 4 we take advantage of the structure of the second-order shape derivative to present new extension methods for a descent direction, from $\partial\Omega$ to $\mathbb{R}^d$. Section 5 is concerned with optimization algorithms and most notably the Newton method. In Section 6 we discuss some technical details in the computation of boundary integrals which are key ingredients in the formulas of shape derivatives. Eventually Section 7 is devoted to some 2-d numerical examples of shape and topological optimization problems. In particular we make comparisons between gradient and Newton's algorithms. These are preliminary results which are very encouraging but not definite since several important issues (3-d, CPU time, memory storage) have not yet been addressed. We give some concluding remarks and perspectives in the final Section 8. The results presented here are part of the PhD thesis [36] of the third author where much more details and numerical experiments can be found. Needless to say that topology optimization of structures is a very important field where our friend and colleague George Rozvany made essential contributions [30], [31]. This paper is dedicated to his memory.

[1] The Hamiltonian $H(t,x,p) = v(t,x)|p|$ is not smooth at $p = 0$ but it is not an issue as explained in Remark 3.3.

2 Shape differentation

We recall the basic definitions and results of the Hadamard method of shape differentiation (see [18, §5]). There are two variants of the Hadamard method. The first one, based on displacement fields, is described in Subsection 2.1 and advocated by Murat and Simon [23, 32, 33]. The second one is the speed method of Zolesio [14, 35], based on flows of ordinary differential equations, as described in Subsection 2.2. We introduce the following set of admissible shapes

$$\mathcal{O}_k = \{\Omega \subset \mathbb{R}^d \mid \Omega \text{ bounded open set of class } C^k\} \quad \text{for } k = 1, 2.$$

In the sequel, we fix a reference shape $\Omega \in \mathcal{O}_1$ with boundary $\partial\Omega$ and unit exterior normal vector $\mathbf{n}$. This reference shape will be further assumed to belong to $\mathcal{O}_2$ when second order derivatives are considered. All shape variations will be made with respect to this reference shape.

2.1 The displacement field method

Let $C^{1,\infty}(\mathbb{R}^d;\mathbb{R}^d)$ be the set of differentiable and bounded vector fields from $\mathbb{R}^d$ into $\mathbb{R}^d$, defined as $C^{1,\infty}(\mathbb{R}^d;\mathbb{R}^d) := C^1(\mathbb{R}^d;\mathbb{R}^d) \cap W^{1,\infty}(\mathbb{R}^d;\mathbb{R}^d)$, where $W^{1,\infty}(\mathbb{R}^d,\mathbb{R}^d)$ is the set of Lipschitz bounded functions of $\mathbb{R}^d$. For any $\theta \in C^{1,\infty}(\mathbb{R}^d;\mathbb{R}^d)$, let $\Omega_\theta \in \mathcal{O}_1$ be defined by

$$\Omega_\theta = \left\{ \big(\mathrm{Id} + \theta\big)(x),\, x \in \Omega \right\}.$$

Definition 2.1. *Let $E(\Omega)$ be a function from $\mathcal{O}_1$ into $\mathbb{R}$. We define*

$$\begin{array}{rccc} \mathcal{E} & : & C^{1,\infty}(\mathbb{R}^d;\mathbb{R}^d) & \to & \mathbb{R} \\ & & \theta & \mapsto & E\Big(\big(\mathrm{Id}+\theta\big)(\Omega)\Big). \end{array}$$

The function E is said to be shape-differentiable at Ω if $\mathcal{E}$ is Fréchet-differentiable at 0, that is, if there exists a continuous linear map $\mathcal{E}'(0;\cdot) : C^{1,\infty}\big(\mathbb{R}^d;\mathbb{R}^d\big) \to \mathbb{R}$ such that :

$$\mathcal{E}(\theta) - \mathcal{E}(0) - \mathcal{E}'(0;\theta) = o\big(\,\|\theta\|_{C^{1,\infty}(\mathbb{R}^d;\mathbb{R}^d)}\,\big).$$

We denote $E'(\Omega;\theta) := \mathcal{E}'\big(0;\theta\big)$.

Definition 2.2. *The function E of Definition 2.1 is said to be twice shape-differentiable at Ω if $\mathcal{E}$ is Fréchet-differentiable in a neighborhood $\mathcal{U}$ of 0 in $C^{1,\infty}\big(\mathbb{R}^d;\mathbb{R}^d\big)$ and if the first derivative $\mathcal{E}'$ defined by*

$$\begin{array}{rccc} \mathcal{E}' & : & \mathcal{U} & \to & \Big(C^{1,\infty}\big(\mathbb{R}^d;\mathbb{R}^d\big)\Big)' \\ & & \theta & \mapsto & \mathcal{E}'\big(\theta;\cdot\big), \end{array}$$

is Fréchet differentiable at 0. We denote by $\mathcal{E}''\big(0;\theta_1,\theta_2\big)$ the second Fréchet derivative at 0, θ_1 and θ_2 being respectively the first and second direction of derivation. We also denote $E''(\Omega;\theta_1,\theta_2) := \mathcal{E}''\big(0;\theta_1,\theta_2\big)$. In that case, $\mathcal{E}$ has a second-order Taylor expansion at 0 and

$$\mathcal{E}(\theta) = \mathcal{E}(0) + \mathcal{E}'(0;\theta) + \frac{1}{2}\mathcal{E}''(0;\theta,\theta) + o\big(\,\|\theta\|^2_{C^{1,\infty}(\mathbb{R}^d;\mathbb{R}^d)}\,\big).$$

Remark 2.3. *As a classical result of Calculus of Variations (see [16, §II.1.2] for example), since $C^{1,\infty}\big(\mathbb{R}^d;\mathbb{R}^d\big)$ is a linear space, the second derivative of $\mathcal{E}$ corresponds to the derivative of its first derivative. However, as explained in [33], it is not the case for $E(\Omega)$ because the variation of two successive vector fields θ_1, θ_2 is not the same as the variation obtained with the sum $\theta_1 + \theta_2$:*

$$\big(\mathrm{Id} + \theta_1\big) \circ \big(\mathrm{Id} + \theta_2\big) = \mathrm{Id} + \theta_2 + \theta_1 \circ \big(\mathrm{Id} + \theta_2\big) \neq \mathrm{Id} + \theta_2 + \theta_1. \tag{5}$$

A standard computation [33], [18] gives

$$E''(\Omega;\theta_1,\theta_2) = \Big(E'\big(\Omega;\theta_1\big)\Big)'\big(\Omega;\theta_2\big) - E'(\Omega;\nabla\theta_1\,\theta_2). \tag{6}$$

We now recall the Hadamard structure theorem for the first and second-order shape derivatives.

Theorem 2.4 ([18, Theorem 5.9.2])**.** *Let $E, \mathcal{E}$ be defined as in Definition 2.1.*

1. *Take $\Omega \in \mathcal{O}_1$. Assume $\mathcal{E}$ is differentiable at 0 in $C^{1,\infty}\big(\mathbb{R}^d;\mathbb{R}^d\big)$. Then, there exists a continuous linear form $l_1 : C^1\big(\partial\Omega\big) \to \mathbb{R}$ such that, for any $\theta \in C^{1,\infty}\big(\mathbb{R}^d;\mathbb{R}^d\big)$,*

$$E'(\Omega;\theta) = \mathcal{E}'(0;\theta) = l_1\left(\theta \cdot \mathbf{n}\right).$$

2. *Take $\Omega \in \mathcal{O}_2$. Assume $\mathcal{E}$ is twice differentiable at 0 in $C^{1,\infty}\big(\mathbb{R}^d;\mathbb{R}^d\big)$. Then there exists a continuous bilinear symmetric form $l_2 : C^1\big(\partial\Omega\big) \times C^1\big(\partial\Omega\big) \to \mathbb{R}$ such that for any $\theta, \xi \in C^{2,\infty}\big(\mathbb{R}^d;\mathbb{R}^d\big)$,*

$$\left\{ \begin{array}{l} E''(\Omega;\theta,\xi) = \mathcal{E}''(0;\theta,\xi) = l_2(\theta\cdot\mathbf{n},\xi\cdot\mathbf{n}) + l_1(Z_{\theta,\xi}), \\ Z_{\theta,\xi} = \big(\,(\xi_\Gamma\cdot\nabla_\Gamma)\,\mathbf{n}\big)\cdot\theta_\Gamma - \nabla_\Gamma\,(\theta\cdot\mathbf{n})\cdot\xi_\Gamma - \nabla_\Gamma\,(\xi\cdot\mathbf{n})\cdot\theta_\Gamma. \end{array} \right. \tag{7}$$

In Theorem 2.4, for any vector field $\xi(x)$ from $\partial\Omega$ into $\mathbb{R}^d$, $\xi_\Gamma(x)$ denotes its tangential component, defined by $\xi_\Gamma = \xi - (\xi\cdot\mathbf{n})\mathbf{n}$. Furthermore, for a function $v \in C^1\big(\mathbb{R}^d;\mathbb{R}\big)$, its tangential gradient $\nabla_\Gamma v$ is defined on $\partial\Omega$ as the tangential component of its gradient, $\nabla_\Gamma v(x) = \nabla v(x) - \big(\nabla v(x)\cdot\mathbf{n}(x)\big)\mathbf{n}(x)$, for any $x \in \partial\Omega$. Explicit examples of the linear forms l_1 and l_2 will be given in Section 7.

Remark 2.5. *The vector field $Z_{\theta,\xi}$ defined by (7), and thus the second-order shape derivative $E''(\Omega;\theta,\xi)$, are symmetric in (θ,ξ). This is a consequence of the fact that $\nabla_\Gamma\mathbf{n}$ is a symmetric matrix on $\partial\Omega$ (see [18, Proposition 5.4.14, §5.9.1]). To compute $\nabla_\Gamma\mathbf{n}$, a first step is to extend the normal $\mathbf{n}$ in a neighborhood of $\partial\Omega$: the resulting value of $\nabla_\Gamma\mathbf{n}$ is independent of the choice of this extension.*

2.2 The speed method

Following the approach of Sokolowski and Zolésio [13, 15, 35], shape derivatives may also be defined as the Eulerian derivatives along time trajectories, defined as flows or solutions of ordinary differential equations.

Definition 2.6. *Let $V \in C^{1,\infty}\big(\mathbb{R}_+\times\mathbb{R}^d;\mathbb{R}^d\big)$. For $\tau > 0$ small enough and for $x \in \mathbb{R}^d$, we define the flow of the vector field V as the unique solution $X_V : [0,\tau]\times\mathbb{R}^d \to \mathbb{R}^d$ of*

$$\left\{ \begin{array}{l} \dfrac{\partial X_V}{\partial t}(t,x) = V(t, X_V(t,x)), \\ X_V(0,x) = x. \end{array} \right. \tag{8}$$

Define the set

$$X_V(t,\cdot)(\Omega) = \{X_V(t,x),\, x \in \Omega\}\,.$$

For V, W in $C^{1,\infty}\big(\mathbb{R}_+\times\mathbb{R}^d;\mathbb{R}^d\big)$ the first and second-order directional derivatives of E are defined by

$$dE\big(\Omega;V\big) := \partial_t E\big(X_V(t,\cdot)(\Omega)\big)\Big|_{t=0},$$
$$d^2E\big(\Omega;V,W\big) := \partial_t\partial_s E\big(X_V(t,\cdot)\circ X_W(s,\cdot)(\Omega)\big)\Big|_{s=t=0}.$$

Since Ω is a bounded domain, the existence time τ for (8) can be chosen to be uniform for any point x in a compact neighborhood of Ω.

Remark 2.7. *In the above definition, the velocity field V plays the same role as the vector field θ in Subsection 2.1. One can check [25, Remark 2.5] that*

$$dE\big(\Omega;V\big) = E'(\Omega;V) \quad \text{and} \quad d^2E\big(\Omega;V,W\big) = \Big(E'\big(\Omega;V\big)\Big)'\big(\Omega;W\big).$$

Contrary to Definition 2.1 where the shape derivative is a Fréchet derivative, the shape derivative of Definition 2.6 is a directional or Gâteaux derivative.

Theorem 2.8 ([18, Corollaire 5.9.3])**.** *Let $X_V(\cdot,x)$ be the solution of (8) where $V \in C^{1,\infty}\big(\mathbb{R}_+\times\mathbb{R}^d;\mathbb{R}^d\big)$. Define the function $e(t)$ from $[0,\tau]$ into $\mathbb{R}$ by*

$$e(t) = E\big(X_V(t,\cdot)(\Omega)\big) = \mathcal{E}\big(X_V(t,\cdot) - \mathrm{Id}\big). \tag{9}$$

Then e is twice differentiable on $[0,\tau]$ and

$$e'(0) = l_1\,(V\cdot\mathbf{n})\,,$$
$$e''(0) = l_2\big(V\cdot\mathbf{n}, V\cdot\mathbf{n}\big) + l_1(\tilde{Z}_{V,V}),$$

with the same linear form l_1 and bilinear form l_2, defined on $\partial\Omega$, as in Theorem 2.4, and with

$$\tilde{Z}_{V,V} = \big(\partial_t V + (V\cdot\nabla)\,V\big)\cdot\mathbf{n} + \big(\,(V_\Gamma\cdot\nabla_\Gamma)\,\mathbf{n}\big)\cdot V_\Gamma - 2\nabla_\Gamma\,(V\cdot\mathbf{n})\cdot V_\Gamma.$$

Proof. Using the chain rule we get $e'(t) = \mathcal{E}'\Big(X_V(t,\cdot) - \mathrm{Id}; V\big(t, X_V(t,\cdot)\big)\Big)$ and $e''(0) = \mathcal{E}''(0;V,V) + \mathcal{E}'\left(0; \dfrac{dV}{dt}\right)$. The expected result then follows from Theorem 2.4. □

Although Theorem 2.8 is just a corollary of Theorem 2.4, the two notions of shape derivatives differ. The first derivatives in Theorems 2.4 and 2.8 coincide but the second derivatives are different. Note however that the directional derivatives of Theorem 2.4 can be recovered from Theorem 2.8 by the special choice in (8) of the vector field

$$V(t,x) = \theta\circ\big(\mathrm{Id} + t\theta\big)^{-1}(x),$$

which corresponds to the solution $X(t,x) = \big(\mathrm{Id} + t\theta\big)(x)$.

3 Shape derivation with respect to normal evolution

In this section we introduce a new variant of shape differentiation which makes sense for domains that evolve in the direction of their normal vector, as it is the case in the level-set method [27], [9]. Note that the case of a normal evolution is a priori not covered by the flow of equation (8) since it is not always obvious that there exists a velocity V which stays parallel to the normal vector for times $t > 0$, even if it so at the initial time $t = 0$. There are simple examples of initial geometries and normal flows: indeed, take an initial disk with a radial velocity. But it is easy to construct instances of non-normal flows too. Take for example an initially flat part of the boundary (with constant unit normal n_0) with a velocity $V(x)$ which is everywhere parallel to n_0 and affine (non-constant) with respect to the tangential variable $(x - (x \cdot n_0)n_0)$. Clearly, for any later time $t > 0$, the boundary is not anymore flat since points on the initially flat part move with different velocities. Therefore, the unit normal to this part of $\partial\Omega(t)$ is no longer parallel to n_0 and thus to V. By a blow-up argument, this simple example can be extended to more general situations (at the price of some technicalities). Nevertheless, the arguments below prove the existence of at least one normal velocity V (which is not explicit in terms of the initial boundary), as stated in Remark 3.6. The main new idea here is to introduce a set of two coupled ordinary differential equations (the so-called bicharacteristics) which are equivalent to the level-set equation, in the case of smooth domains. We first recall the method of bicharacteristics for solving Hamilton-Jacobi equations, of which the level-set equation is a particular case.

3.1 Bicharacteristics method for solving Hamilton-Jacobi equations

Let $H(t, x, p) : \mathbb{R}_+ \times \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a smooth function, called a Hamiltonian in this context. For $(t, x) \in \mathbb{R}_+ \times \mathbb{R}^d$, we consider the scalar first order partial differential equation

$$\partial_t \phi(t, x) + H\big(t, x, \nabla_x \phi(t, x)\big) = 0, \tag{10}$$

with unknown $\phi : \mathbb{R}_+ \times \mathbb{R}^d \to \mathbb{R}$. Usually, (10) is complemented by initial and boundary conditions. Here, ∂_t, ∇_x and ∇_p respectively denote the derivation with respect to the first variable, the following d and the last d variables of $\mathbb{R}_+ \times \mathbb{R}^d \times \mathbb{R}^d$. We recall a classical result for solving (10).

Lemma 3.1 ([28, Lemma 5.I.2])**.** *Let $\tau > 0$ and $\mathcal{O}$ be an open set of $\mathbb{R}^d$. Assume that ϕ is a smooth solution of (10) for $(t, x) \in [0, \tau] \times \mathcal{O}$, and that $(x(t), p(t))$, defined from $\mathbb{R}_+$ into $\mathbb{R}^d \times \mathbb{R}^d$, solves*

$$\left\{ \begin{array}{rcl} \dfrac{dx}{dt}(t) & = & \nabla_p H\big(t, x(t), p(t)\big), \\ \dfrac{dp}{dt}(t) & = & -\nabla_x H\big(t, x(t), p(t)\big). \end{array} \right. \tag{11}$$

If $p(0) = \nabla_x \phi(0, x(0))$, then $p(t) = \nabla_x \phi\big(t, x(t)\big)$ as long as $(t, x(t)) \in [0, \tau] \times \mathcal{O}$. Furthermore, let $t \ni \mathbb{R}_+ \mapsto (y(t), r(t)) \in \mathbb{R} \times \mathbb{R}$ be the solution of

$$\left\{ \begin{array}{rcl} \dfrac{dy}{dt}(t) & = & -\partial_t H\big(t, x(t), p(t)\big), \\ \dfrac{dr}{dt}(t) & = & y(t) + p(t) \cdot \nabla_p H\big(t, x(t), p(t)\big), \end{array} \right. \tag{12}$$

with $y(0) = \partial_t \phi(0, x(0))$, $r(0) = \phi\big(0, x(0)\big)$. Then $y(t) = \partial_t \phi(t, x(t))$ and $r(t) = \phi\big(t, x(t)\big)$ as long as $(t, x(t)) \in [0, \tau] \times \mathcal{O}$.

Remark 3.2. *Equation* (11) *is a Hamiltonian system, called the bicharacteristics system of* (10)*, which admits global solution in time since the value of the Hamiltonian function $H\big(t, x(t), p(t)\big)$ is conserved along the flow. Lemma 3.1 roughly states that, in order to find a smooth solution of the Hamilton-Jacobi equation* (10)*, it is enough to solve the Hamiltonian system* (11)*. This construction breaks down as soon as the projection on the configuration space $\mathbb{R}^d$ of the bicharacteristics $(x(t), p(t))$ in the phase space $\mathbb{R}^d \times \mathbb{R}^d$ cross, implying that $\phi\big(t, x(t)\big)$ is multivalued. In such a case,* (10) *has no smooth solution and one should resort to the notion of viscosity solutions [9]. However, for short time τ there exists a unique smooth solution of* (10) *[28]. Note that* (12) *is decoupled from* (11) *and is just a kind of post-processing phase of the bicharacteristics method.*

For the sake of completeness, we provide a proof of Lemma 3.1.

Proof. To simplify the notation we write $\dot{z}(t)$ instead of $\dfrac{dz}{dt}(t)$. Let ϕ be a smooth solution of (10) on $[0, \tau] \times \mathcal{O}$ and define $X(t) : [0, \tau] \to \mathbb{R}^d$ the solution of the following ordinary differential equation (o.d.e.)

$$\left\{ \begin{array}{l} \dot{X}(t) = \nabla_p H\Big(t, X(t), \nabla_x \phi\big(t, X(t)\big)\Big), \\ X(0) = x(0), \end{array} \right. \tag{13}$$

which exists, at least for short times, by virtue of the Cauchy-Lipschitz theorem. Let Y, P, R be defined by

$$Y(t) = \partial_t\phi\big(t, X(t)\big), \quad P(t) = \nabla_x\phi\big(t, X(t)\big), \quad R(t) = \phi\big(t, X(t)\big).$$

We now prove that X, Y, P, R coincide with x, y, p, r. By rewriting (13) and differentiating the definitions of Y, P, R, we obtain

$$\dot{X}(t) = \nabla_p H\Big(t, X(t), P(t)\Big), \qquad \dot{Y}(t) = \partial_{tt}\phi\big(t, X(t)\big) + \partial_t\nabla_x\phi\big(t, X(t)\big)\cdot\dot{X}(t),$$
$$\dot{P}(t) = \partial_t\nabla_x\phi\big(t, X(t)\big) + \nabla_x^2\phi\big(t, X(t)\big)\dot{X}(t), \quad \dot{R}(t) = \partial_t\phi\big(t, X(t)\big) + \nabla_x\phi\big(t, X(t)\big)\cdot\dot{X}(t).$$

With the definition of Y, P and the o.d.e. for X, we already get an o.d.e. for R, similar to that for r in (12),

$$\dot{R}(t) = Y(t) + P(t)\cdot\nabla_p H\big(t, X(t), P(t)\big).$$

Differentiating the Hamilton-Jacobi equation (10) with respect to t and x respectively leads to

$$\partial_{tt}\phi + \partial_t H + \partial_t\nabla_x\phi\cdot\nabla_p H = 0, \quad \text{and} \quad \partial_t\nabla_x\phi + \nabla_x H + \nabla_x^2\phi\nabla_p H = 0. \tag{14}$$

Evaluating (14) at $\big(t, X(t)\big)$ (for small times, we are sure that $X(t)\in\mathcal{O}$), and using $\dot{X} = \nabla_p H$, we get

$$\partial_{tt}\phi + \partial_t\nabla\phi\cdot\dot{X} = -\partial_t H, \quad \text{and} \quad \partial_t\nabla\phi + \nabla_x^2\phi\dot{X} = -\nabla_x H,$$

which are exactly the o.d.e.'s, similar to those for y and p,

$$\dot{Y}(t) = -\partial_t H\Big(t, X(t), P(t)\Big), \quad \text{and} \quad \dot{P}(t) = -\nabla_x H\Big(t, X(t), P(t)\Big).$$

As a result, (x, y, p, r) and (X, Y, P, R) satisfy the same differential system with the same initial condition since $X(0) = x(0)$ and $p(0) = \nabla_x\phi(0, x(0))$ by assumption. Therefore $(x, y, p, r) = (X, Y, P, R)$ and, furthermore, from the equivalent definitions of Y, P, R we deduce

$$y(t) = \partial_t\phi\big(t, x(t)\big), \quad p(t) = \nabla_x\phi\big(t, x(t)\big), \quad r(t) = \phi\big(t, x(t)\big).$$

□

A particular case of (10), which is crucial for the level-set method, is the eikonal equation corresponding to the Hamiltonian $H(t, x, p) = v(t, x)|p|$ where v is a smooth function from $\mathbb{R}_+\times\mathbb{R}^d$ into $\mathbb{R}$. In other words, we specialize (10) to the case

$$\begin{cases} \partial_t\phi(t, x) + v(t, x)\,|\nabla_x\phi(t, x)| = 0, \\ \phi(0, x) = \phi_0(x), \end{cases} \tag{15}$$

where ϕ_0 is a smooth function, satisfying for some smooth bounded open set Ω_0,

$$\begin{cases} \phi_0(x) < 0 & \text{if } x\in\Omega_0, \\ \phi_0(x) = 0 & \text{if } x\in\partial\Omega_0, \\ \phi_0(x) > 0 & \text{if } x\in\mathbb{R}^d\setminus(\Omega_0\cup\partial\Omega_0). \end{cases}$$

The evolution of the eikonal equation defines a family of domains Ω_t :

$$\Omega_t := \big\{x\in\mathbb{R}^d \mid \phi(t, x) < 0\big\}.$$

Applying Lemma 3.1 to (15), in the case of smooth solutions, yields $\dot{p} = \nabla_x\phi$ and

$$\dot{x} = \nabla_p H = v\frac{\nabla_x\phi}{|\nabla_x\phi|} = v\,\mathbf{n} \tag{16}$$

because the derivative of the function $|p|$ is $p/|p|$ and $\nabla_x\phi/|\nabla_x\phi|$ defines a smooth extension of the unit exterior normal vector $\mathbf{n}$ to the boundary $\partial\Omega_t$. We thus recover from (16) that the level-set equation (15) corresponds to an evolution with a normal velocity v.

Furthermore, if we restrict the solutions of the Hamiltonian system (11) to those initial positions $x(0)\in\partial\Omega_0$, with $p(0) = \nabla_x\phi_0(x(0))$, then the boundary of Ω_t can be obtained simply by

$$\partial\Omega_t = \{x(t) \text{ solution of (11) with } x(0)\in\partial\Omega_0\},$$

without the necessity to solve the other o.d.e.'s (12).

Remark 3.3. *Although the function $\mathbb{R}^d\ni p\mapsto|p|\in\mathbb{R}$ is not smooth at 0 (contrary to our assumption), it is not a problem for the level-set method which focuses only on the evolution of the zero level-set (the shape boundary) where $p = \nabla_x\phi$ is proportional to the normal vector to the boundary and thus does not vanish, at least for small times when the initial value $p(0)$ is uniformly bounded away from 0. In other words, the non-smooth function $|p|$ can be regularized near 0 without changing the evolution of the zero level-set.*

3.2 Derivation along normal trajectories

We now introduce our new variant of the Hadamard method of shape differentiation, based on the Hamiltonian system (11).

Theorem 3.4. *Let $E : \mathcal{O}_2 \to \mathbb{R}$ be twice differentiable at $\Omega_0 \in \mathcal{O}_2$ and $v(t,x)$ be a C^1 scalar function from $\mathbb{R}_+ \times \mathbb{R}^d$ into $\mathbb{R}$. Let ϕ_0 to be the signed distance function associated to Ω_0. For some time $\tau > 0$, let $\phi \in C^1\big([0,\tau] \times \mathbb{R}^d\big)$ be a smooth solution (see Remark 3.5) of*

$$\begin{cases} \partial_t \phi(t,x) + v(t,x)\,|\nabla_x \phi(t,x)| = 0, \\ \phi(0,x) = \phi_0(x). \end{cases} \tag{17}$$

Define $\Omega_t := \big\{x \in \mathbb{R}^d \mid \phi(t,x) < 0\big\}$ and $\epsilon(t) := E\big(\Omega_t\big)$. Then ϵ is twice differentiable at 0 and it holds

$$\epsilon'(0) = l_1\big(v(0,\cdot)\big) \quad \text{and} \quad \epsilon''(0) = l_2\big(v(0,\cdot), v(0,\cdot)\big) + l_1(\hat{Z}_{v,v}(0,\cdot)),$$

with the same linear form l_1 and bilinear form l_2, defined on $\partial\Omega$, as in Theorem 2.4, and where $\hat{Z}_{v,v} = \partial_t v + v\partial_{\mathbf{n}} v$ with $\partial_{\mathbf{n}} v = \mathbf{n} \cdot \nabla_x v$.

Remark 3.5. *As already said in Remark 3.2, (17) may not have smooth solutions for all times, even if the Hamiltonian H and the initial data are smooth. Nevertheless, there always exists a small enough time $\tau > 0$ such that there exists a unique classical (smooth) solution of (17). To obtain solutions for all times, Crandall and Lions [9] introduced the notion of viscosity solutions for Hamilton-Jacobi equations, which yields the uniqueness and global-in-time existence of such solutions.*

Remark 3.6. *Once again, the first order shape derivative in Theorem 3.4 is the same as in the two variants of the Hadamard method, discussed in Section 2. However, the second derivative is different from the previous ones in Theorems 2.4 and 2.8. The family of shape evolutions along the normal direction is somehow smaller than the previous two classes of shape evolutions since it completely eliminates the possibility of tangential displacements. Therefore, the second-order derivative is simpler in the present new setting. The formula of the second-order derivative in Theorem 3.4 coincides with that of Theorem 2.8 if one can find a vector field $V(t,x)$ such that $V(t,x) = v(t,x)\mathbf{n}(t,x)$, where $\mathbf{n}(t,x)$ is the unit normal vector to the evolving shape Ω_t. The existence of such a vector field was a priori not obvious but it is actually provided by the method of bicharacteristics in Lemma 3.1. Indeed, after solving (11), it is enough to take $V(t,x) = \nabla_p H(t,x,p(t))$, where $H(t,x,p) = v(t,x)|p|$. Note that such a velocity field V depends strongly on the initial shape Ω_0 and that it is far from being explicit.*

Proof. We apply the result of Section 3.1 with $H(t,x,p) = v(t,x)|p|$. By definition, we have $\epsilon(t) = E\big(\Omega_t\big) = \mathcal{E}\big(x(t,\cdot) - \mathrm{Id}\big)$ where $x(t,x_0)$ is the solution of (11) with $x(0) = x_0$. Arguing as in the proof of Theorem 2.8 and using Theorem 2.4, the shape derivatives are given by the chain rule

$$\begin{aligned} \epsilon'(t) &= \mathcal{E}'\Big(x(t,\cdot) - \mathrm{Id}; \dot{x}(t,\cdot)\Big), \\ \epsilon''(t) &= \mathcal{E}''\Big(x(t,\cdot) - \mathrm{Id}; \dot{x}(t,\cdot), \dot{x}(t,\cdot)\Big) + \mathcal{E}'\Big(x(t,\cdot) - \mathrm{Id}; \ddot{x}(t,\cdot)\Big). \end{aligned}$$

In (16) we already computed $\dot{x}(t,x_0) = v(t,x_0)\,\mathbf{n}(t,x_0)$ and thus $\epsilon'(0) = l_1\Big(\dot{x}(0,\cdot)\cdot\mathbf{n}\Big) = l_1\Big(v(0,\cdot)\Big)$. To obtain the second-order derivative, we remark that, by definition the tangential component of $\dot{x}(0,\cdot)$ vanishes, i.e., $\dot{x}(0,\cdot)_\Gamma = 0$ (or $\dot{x}(0,\cdot)$ is a normal vector), which implies $Z_{\dot{x}(0,\cdot),\dot{x}(0,\cdot)} = 0$ (where Z is defined in (7)) and thus

$$\mathcal{E}''\Big(x(0,\cdot) - \mathrm{Id}; \dot{x}(0,\cdot), \dot{x}(0,\cdot)\Big) = l_2\Big(\dot{x}(0,\cdot)\cdot\mathbf{n}, \dot{x}(0,\cdot)\cdot\mathbf{n}\Big),$$

so that

$$\epsilon''(0) = l_2\Big(\dot{x}(0,\cdot)\cdot\mathbf{n}, \dot{x}(0,\cdot)\cdot\mathbf{n}\Big) + l_1\Big(\ddot{x}(0,\cdot)\cdot\mathbf{n}\Big). \tag{18}$$

It remains to compute $\ddot{x}$, starting from $\dot{x}(t,\cdot) = v(t,\cdot)\,\mathbf{n}(t,\cdot)$. We first compute the time derivative of $\mathbf{n} = p/|p|$:

$$\dot{\mathbf{n}} = \frac{\dot{p}}{|p|} - p\frac{(p\cdot\dot{p})}{|p|^3}.$$

Since $\dot{p} = -\nabla_x H = -|p|\nabla_x v$, $p = \nabla\phi$ and $\mathbf{n} = \dfrac{\nabla_x\phi}{|\nabla_x\phi|}$, we get

$$\dot{\mathbf{n}} = -\nabla_x v + \frac{p}{|p|^3}|p|\,(\nabla_x v\cdot p) = -\nabla_x v + (\nabla_x v\cdot\mathbf{n})\,\mathbf{n}.$$

Here $\mathbf{n}$ is an extension in $\mathbb{R}^d$ of the unit exterior normal to $\partial\Omega_t$. If we restrict ourselves to the boundary $\partial\Omega_t$, this implies that $\dot{\mathbf{n}} = -\nabla_\Gamma v$. By using a suitable extension of $\nabla_\Gamma v$ (see [36] for details), this result is valid everywhere in $\mathbb{R}^d$. Then, differentiating $\dot{x}(t,\cdot) = v(t,\cdot)\,\mathbf{n}(t,\cdot)$ we deduce

$$\ddot{x} = \Big(\partial_t v + \nabla_x v\cdot\dot{x}\Big)\,\mathbf{n} + v\dot{\mathbf{n}} = \partial_t v\,\mathbf{n} + v\big(\nabla_x v\cdot\mathbf{n}\big)\,\mathbf{n} - v\,\nabla_\Gamma v, \tag{19}$$

which implies $\ddot{x}(0,\cdot)\cdot\mathbf{n} = \partial_t v + v\big(\nabla_x v\cdot\mathbf{n}\big)$ since $\nabla_\Gamma v\cdot\mathbf{n} = 0$. This proves the desired result. □

4 Extension of the shape derivative

In this section we discuss a first application of the formula for the second-order shape derivative, which is concerned with the problem of extending the shape derivative or the normal velocity in the eikonal equation (17) from the shape boundary to the entire space $\mathbb{R}^d$. Indeed, as stated in Theorems 2.4, 2.8 and 3.4, the first oder shape derivative is of the type $\mathcal{E}'(0;\theta) = l_1(\theta\cdot\mathbf{n})$ where l_1 is a linear form on $C^1\big(\partial\Omega\big)$. In full generality, l_1 can be represented by a first-order distribution. Nevertheless, in many practical cases, it is smoother, meaning that it can be represented by a function $j\in L^1(\partial\Omega)$. For simplicity, we assume it is the case. Thus, there exists an integrand $j(x)$, depending on the function $\mathcal{E}$ and defined only on the boundary $\partial\Omega$, such that

$$\mathcal{E}'(0;\theta) = l_1(\theta\cdot\mathbf{n}) \equiv \int_{\partial\Omega} j\,\theta\cdot\mathbf{n}\,.$$

A possible descent direction is to choose $\theta\cdot\mathbf{n} = -j$ (other choices are possible, see [3], [7], [12]). In the sequel we call v_γ a descent direction, a function from $\partial\Omega$ into $\mathbb{R}$ such that

$$l_1(v_\gamma) = \int_{\partial\Omega} j\,v_\gamma \le 0\,. \tag{20}$$

A priori, v_γ is defined only on $\partial\Omega$, while the level-set method requires to solve (17) with a velocity v_γ defined everywhere in $\mathbb{R}^d$. Therefore, one needs to extend v_γ from $\partial\Omega$ to $\mathbb{R}^d$. This issue is discussed in [3], [7], [12] and here we propose new approaches based on the following Ansatz. For small $t\ge 0$, according to Theorem 3.4, we write the second-order Taylor expansion

$$E\big(\Omega_t\big) = \epsilon(t) = \epsilon(0) + t\,l_1(v_\gamma) + \frac{1}{2}t^2\Big(l_2\big(v_\gamma,v_\gamma\big) + l_1(\partial_t v_\gamma + v_\gamma\partial_{\mathbf{n}}v_\gamma)\Big) + \mathcal{O}(t^3). \tag{21}$$

For simplicity we restrict ourselves to time independent descent direction, namely $\partial_t v_\gamma = 0$. Taking into account the bilinear form l_2 in (21) is at the basis of the Newton algorithm which shall be discussed in the next section. For the moment we focus on the other second-order term and our main idea is to build an extension of v_γ such that

$$l_1(v_\gamma\partial_{\mathbf{n}}v_\gamma) \le 0. \tag{22}$$

To enforce (22), and since v_γ is already bound to satisfy (20), we can choose a suitable normal derivative $\partial_{\mathbf{n}}v_\gamma$. We propose here two new extension methods based on this idea and we recall a third classical method which, under some variants, can be found in [3], [7], [12].

Method 4.1. *Given a descent direction v_γ on $\partial\Omega$, we compute its extension $v(t,x)$ to $\mathbb{R}^d$ as the solution of the following linear transport equation*

$$\begin{cases} \partial_t v + \mathrm{sign}(\phi)\big(\mathbf{n}\cdot\nabla_x v - 1\big) = 0 & \text{in } [0,\tau]\times\mathbb{R}^d,\\ v(t,x) = v_\gamma & \text{on } \partial\Omega,\\ v(0,x) = 0 & \text{on } \mathbb{R}^d\backslash\partial\Omega, \end{cases} \tag{23}$$

where $\mathbf{n}$ is an extension of the unit outer normal and ϕ is a level-set function for Ω. The stationary solution v of (23) satisfies

$$\mathbf{n}\cdot\nabla_x v = \partial_{\mathbf{n}}v_\gamma = 1 \text{ in } \mathbb{R}^d, \quad \text{and} \quad v = v_\gamma \text{ on } \partial\Omega.$$

Thus, we obtain $l_1(v) = l_1(v_\gamma)$ and $l_1(v\partial_{\mathbf{n}}v) = l_1(v) = l_1(v_\gamma) \le 0$.

As already said in Remark 2.5, there exists a smooth extension of the unit outer normal $\mathbf{n}$ in the neighborhood of $\partial\Omega$ [18]. A global smooth extension is easily deduced if the unit norm contraint is not enforced away from $\partial\Omega$. In numerical practice, the extension of $\mathbf{n}$ is given by $\frac{\nabla_x\phi}{|\nabla_x\phi|}$ where ϕ is the level set function (such an extension is not smooth on the skeleton of the shape Ω). Equation (23) is very much inspired from the classical re-initialization technique in the level set algorithm. In particular, the velocity being opposed on each side of $\partial\Omega$, it implies that the information from the boundary condition v_γ is carried away from $\partial\Omega$. It is not matematically clear that there exists a stationary solution of (23). In numerical practice (with our choice $\mathbf{n} = \frac{\nabla_x\phi}{|\nabla_x\phi|}$), we always find one and, in any case, since the extension is only needed locally close to $\partial\Omega$, we could stop the time evolution of (23) as soon as the characteristics associated to the velocity $\mathrm{sign}(\phi)\mathbf{n}$ have reached a certain distance away from $\partial\Omega$. The same comments apply to the following alternative extension method.

Method 4.2. *Let v_γ be a descent direction defined on $\partial\Omega$. Assume that an extension of the product jv_γ is already known in $\mathbb{R}^d$ (we keep the same notation jv_γ for the extension). We compute another extension $v(t,x)$ of v_γ as a solution of the linear transport equation*

$$\begin{cases} \partial_t v + \mathrm{sign}(\phi)\big(\mathbf{n}\cdot\nabla_x v + jv_\gamma\big) = 0 & \text{in } [0,\tau]\times\mathbb{R}^d,\\ v(t,x) = v_\gamma & \text{on } \partial\Omega,\\ v(0,x) = 0 & \text{on } \mathbb{R}^d\backslash\partial\Omega, \end{cases} \tag{24}$$

where ϕ is a level-set function for Ω. The stationary solution v of (24) satisfies

$$\mathbf{n}\cdot\nabla_x v = -jv_\gamma \text{ in } \mathbb{R}^d, \quad \text{and} \quad v = v_\gamma \text{ on } \partial\Omega.$$

Thus, we obtain $l_1(v) = l_1(v_\gamma)$ and

$$l_1(v\partial_{\mathbf{n}}v) = l_1(-jv_\gamma^2) = -\int_{\partial\Omega} j^2 v_\gamma^2 \le 0.$$

Eventually, for the sake of comparison, we recall a standard regularization and extension approach which can be found in [3], [7], [12].

Method 4.3. *Let v_γ be a descent direction defined on $\partial\Omega$. We compute an extension $v(x)$ of v_γ as a solution of the following variational problem on $\mathbb{R}^d$*

$$\eta^2\int_{\mathbb{R}^d}\nabla v\cdot\nabla w + \int_{\partial\Omega} vw = \int_{\partial\Omega} v_\gamma w, \quad \forall w\in H^1\big(\mathbb{R}^d\big), \tag{25}$$

where $\eta > 0$ is a small parameter (typically of the order of a few mesh cell diameters in numerical practice). In practice too, the full space $\mathbb{R}^d$ is replaced by a bounded computational domain. On the one hand, the solution v is not strictly speaking an extension of v_γ since $v_\gamma \neq v$ on $\partial\Omega$. On the other hand, the direction v is more regular than v_γ on $\partial\Omega$. In the case of the obvious descent direction $v_\gamma = -j$, one can check that the solution v of (25) remains a descent direction since

$$l_1(v) = \int_{\partial\Omega} jv = -\int_{\partial\Omega} v_\gamma v = -\Big(\eta^2\int_{\mathbb{R}^d}|\nabla v|^2 + \int_{\partial\Omega} v^2\Big) \le 0.$$

Remark 4.4. *Method 4.3 can be used to compute the extension of jv_γ which is assumed to be already known at the beginning of Method 4.2. Even though such an extension is not equal to jv_γ on $\partial\Omega$, one can show by the same argument that Method 4.2 produces again an extension satisfying $l_1(v\partial_{\mathbf{n}}v) \le 0$.*

5 Optimization method

For simplicity we consider the unconstrained minimization of an objective function $E(\Omega)$ over all possible shapes $\Omega \subset \mathbb{R}^d$. Of course, in practice there are always additional constraints, whether directly on the domain Ω (like geometrical or manufacturing constraints) or on the mechanical performances of Ω. Nevertheless, for simplicity we focus on the definition of a Newton's algorithm in the unconstrained case and we recall that Newton's method can be extended to the constrained setting [24] without theoretical difficulties, although it may be quite intricate in numerical practice. Starting from an initial shape Ω_0, we build an iterative sequence $(\Omega_p)_{p\in\mathbb{N}}$ in order to minimize the objective function $E(\Omega)$. For a given iteration number p, we denote by ϕ_p the level-set function associated to $\partial\Omega_p$. To build the new shape Ω_{p+1} from Ω_p, we solve the Hamilton-Jacobi equation

$$\begin{cases} \partial_t\phi_{p+1}(t,x) + v^p(x)\,|\nabla_x\phi_{p+1}(t,x)| = 0, \\ \phi_{p+1}(0,x) = \phi_p(x). \end{cases} \tag{26}$$

For any pseudo-time or descent parameter $t \ge 0$ we define an associated shape

$$\Omega(t,v^p) = \big\{x\in\mathbb{R}^d \mid \phi_{p+1}(t,x) < 0\big\}.$$

The new shape is defined as $\Omega_{p+1} = \Omega(t_p, v^p)$ where $t_p > 0$ is chosen in order to decrease the reduced objective function $t \mapsto E\left(\Omega(t,v^p)\right)$.

In (26) the normal velocity v^p is a descent direction (properly extended to $\mathbb{R}^d$ by one of the three methods presented in Section 4), evaluated at Ω_p with a gradient or a Newton-like method that we now describe. The linear and bilinear forms l_1 and l_2, defined in Theorem 2.4, depend on the shape Ω_p (cf. the explicit examples in Section 7). In this section, we denote them by $l_{p,1}$ and $l_{p,2}$ for the corresponding shape Ω_p.

5.1 Newton's algorithm

In view of the scaling invariance $\Omega(\frac{t}{\alpha}, \alpha v^p) = \Omega(t, v^p)$, there is no loss of generality in choosing the parameter t_p equal to 1. Then, for small enough v, the following quadratic approximation holds true

$$E\big(\Omega(1,v)\big) \simeq E(\Omega_p) + l_{p,1}(v) + \frac{1}{2}\Big(l_{p,2}(v,v) + l_{p,1}(v\partial_{\mathbf{n}}v)\Big),$$

where only the traces of v and $\partial_{\mathbf{n}}v$ on $\partial\Omega_p$ play a role. The principle of the Newton method is to minimize this quadratic approximation in order to obtain the descent direction v^p. We now observe that, on $\partial\Omega_p$, v and $\partial_{\mathbf{n}}v$ may be considered as independent variables and that $\partial_{\mathbf{n}}v$ does not enter the advection equation (26). Therefore, $\partial_{\mathbf{n}}v$ can be used only in an extension process of the shape derivative (as explained in the previous section) and it makes sense to

minimize separately in v and $\partial_{\mathbf{n}} v$ (although other choices are possible). First, we solve the following problem, referred to as the Newton problem

$$\min_{v \in C^{1,\infty}(\partial\Omega_p;\mathbb{R})} \quad l_{p,1}(v) + \frac{1}{2} l_{p,2}(v,v). \tag{27}$$

Second, knowing the optimal v^p in (27), we should minimize $l_{p,1}(v^p \partial_{\mathbf{n}} v)$ with respect to $\partial_{\mathbf{n}} v$. This linear problem does not have a solution, and this is precisely the content of the previous section. Note that, if we impose a trust-region constraint (say $\|\partial_{\mathbf{n}} v\|_{L^2(\partial\Omega_p;\mathbb{R})} \leq m$), then this linear problem has a solution $\partial_{\mathbf{n}} v$ which is proportional to jv and we are back to Method 4.2 for extending the descent direction away from $\partial\Omega$. Another possibility could be to first extend v by Method 4.3 and then consider $\partial_{\mathbf{n}} v$ to be the image of the extended v by the Dirichlet-to-Neumann map. It results in a quadratic approximation featuring only v. It is not clear how it can be implemented in practice but, at least, it shows that there are different ways of extracting useful information from the second-order approximation of the objective function.

There is a priori no reason for the Newton problem (27) to have a finite minimum. Indeed, the Hessian operator $l_{p,2}$ may have zero or negative eigenvalues. To be convinced, consider the examples of Section 7 where the Hessian depends on the normal vector $\mathbf{n}$ but also on the shape curvature $\mathcal{H}$ and has thus no clear sign. In such a case, the minimal value of (27) could be $-\infty$. For this reason, instead of solving the Newton problem (27) we consider a trust-region variant (see for example [24, Section 6.4], [39])

$$\min_{\substack{v \in C^{1,\infty}(\partial\Omega_p;\mathbb{R}),\\ \|v\|_{L^\infty(\partial\Omega_p;\mathbb{R})} \leq v_M,}} \quad l_{p,1}(v) + \frac{1}{2} l_{p,2}(v,v)\,, \tag{28}$$

for some finite positive bound v_M. Such a bound implies that, at least after finite dimensional discretization, (28) admits a minimimizer. If the solution v^p to (28) satisfies a strict bound $\|v^p\|_{L^\infty(\partial\Omega;\mathbb{R})} < v_M$, then it is also a solution of (27) and it satisfies the optimality condition or Newton equation

$$l_{p,2}(v^p, w) = -l_{p,1}(w) \quad \forall w \in C^{1,\infty}(\partial\Omega_p;\mathbb{R})\,.$$

After having found a good descent direction v^p, we also look for a good descent step t_p to ensure an efficient decrease of the reduced objective function $t \mapsto E\big(\Omega(t, v^p)\big)$, or equivalently $t \mapsto E\big(\Omega(1, tv^p)\big)$, due to the scaling invariance. Ideally we should choose t_p as the global minimizer of the reduced objective function, but repeated evaluations of the objective function are way too costly. Therefore, our strategy is a trade-off which ensures a substantial decrease of the objective function, at an affordable computing price. We use a well-known backtracking procedure [24, Procedure 3.1, Chapter 3] :

Algorithm 5.1. *Choose $t_p > 0$, c_1, c_2 in $]0,1[$.*

1. *Set $t_p := c_1 t_p$.*
2. *If $E\big(\Omega(1, t_p v^p)\big) \leq E\big(\Omega_p\big) + c_2\, t_p\, l_{p,1}(v^p)$ stop. Else go to step 1.*

The following algorithm will be referred to as the Newton algorithm in the sequel.

Algorithm 5.2. *Newton algorithm*

1. ***Initialisation*** *Set $p = 0$. Choose an initial domain Ω_0, two convergence thresholds $\varepsilon > 0$, $\eta > 0$, and two coefficients c_0, c_1 in $]0,1[$.*
2. ***Newton direction*** *Compute the solution v^p of (28).*
3. ***Extension*** *Compute an extension of v^p with one of the three methods detailed in Section 4.*
4. ***Time Step*** *Compute a time step t_p with Algorithm 5.1.*
5. ***Update*** *Set $p = p + 1$. If $\int_{\partial\Omega_p} |v^p_\gamma|^2 \leq \varepsilon$ or $t_p \leq \eta$ stop. Else return to step 2.*

Remark 5.3. *Since the Hessian operator $l_{p,2}$ may have negative eigenvalues, the extension v^p of the solution to (28) may fail to be be a descent direction,* i.e *we may have $l_{p,1}(v^p) > 0$. When it occurs, we take its opposite in order to ensure that $l_{p,1}(v^p) \leq 0$.*

5.2 Gradient algorithm

Gradient algorithms are based on a first order Taylor expansion for sufficiently small v

$$E\big(\Omega(1,v)\big) \simeq E(\Omega_p) + l_{p,1}(v).$$

When $l_{p,1}(v) = \int_{\partial\Omega} j_p v$, it yields a descent direction which, up to a positive multiplicative factor, is $v^p = -j_p$. Then, one look for a descent step $t_p > 0$ such that the objective function decreases. This is the usual gradient-type method. However, in order to make a fair comparison with the previous Newton algorithm which is coupled to a trust-region method, we change this usual algorithm and couple it again with a trust-region method. More precisely, we compute a descent direction as the solution of

$$\min_{\substack{v \in C^{1,\infty}(\partial\Omega_p;\mathbb{R}),\\ \|v\|_{L^\infty(\partial\Omega_p;\mathbb{R})} \leq v_M}} \quad l_{p,1}(v) + \frac{1}{2}\int_{\partial\Omega_p} v^2. \tag{29}$$

Comparing with (28), we keep the same L^∞ bound and we replace the Hessian operator by the simpler $L^2(\partial\Omega)$ scalar product. Note that this quadratic term is not really necessary to make (29) well-posed but it helps for making a fair comparison with (28). In particular, if v^p is a solution to (29) which does not saturate the bound, $\|v^p\|_{L^\infty(\partial\Omega_p;\mathbb{R})} < v_M$, then the optimality condition of (29) yields that $v^p = -j_p$ as usual, when $l_{p,1}(v) = \int_{\partial\Omega} j_p v$.

The main reason for choosing this approach for the gradient agorithm is to ease the comparison with the Newton method. Indeed, the only difference between (28) and (29) is the replacement of the Hessian operator by the identity operator on $\partial\Omega$.

The following algorithm will be referred to as the gradient algorithm in the sequel.

Algorithm 5.4. *Gradient algorithm*

1. ***Initialisation*** *Set $p = 0$. Choose an initial domain Ω_0, two convergence thresholds $\varepsilon > 0$, $\eta > 0$, and two coefficients c_0, c_1 in $]0,1[$.*
2. ***Gradient direction*** *Compute the solution v^p of (29).*
3. ***Extension*** *Compute an extension of v^p with one of the three methods detailed in Section 4.*
4. ***Time Step*** *Compute a time step t_p with Algorithm 5.1.*
5. ***Update*** *Set $p = p + 1$. If $\int_{\partial\Omega_p} |v^p|^2 \leq \varepsilon$ or $t_p \leq \eta$ stop. Else return to step 2.*

6 Computing boundary integrals

Both gradient and Newton algorithms are based on shape derivatives which are written in terms of boundary integrals. It is therefore crucial to compute them accurately for a maximal efficiency of optimization. We first recall the standard procedure in the level-set method for structural optimization which is based on the so-called approximate Dirac mass for the boundary. Then we introduce a better approximation using a piecewise linear reconstruction of the boundary, in the spirit of earlier results of Min and Gibou [20, 21].

6.1 Approximation with a Dirac mass function

In the level-set framework, let s_Ω be the sign function of the level-set, *i.e*

$$\begin{cases} s_\Omega(x) = -1 & \text{if } x \in \Omega, \\ s_\Omega(x) = 0 & \text{if } x \in \partial\Omega, \\ s_\Omega(x) = 1 & \text{if } x \in \mathbb{R}^d \setminus (\Omega \cup \partial\Omega). \end{cases} \tag{30}$$

The derivative (in the sense of distributions) of s_Ω is $\nabla s_\Omega = 2\delta_{\partial\Omega}\mathbf{n}$, where $\delta_{\partial\Omega}$ is the Dirac mass carried by $\partial\Omega$. In other words, for any smooth function $\psi \in C_c^\infty(\mathbb{R}^d;\mathbb{R}^d)$

$$\langle \nabla s_\Omega, \psi \rangle = 2\int_{\partial\Omega} \psi \cdot \mathbf{n}\,,$$

where $\langle\cdot,\cdot\rangle$ denotes the duality product between C_c^∞-functions and distributions (a simple integral on $\mathbb{R}^d$ if the distribution is just a function). Introducing a small regularization parameter $\varepsilon > 0$, we compute an approximation of the sign function by

$$s_{\Omega,\varepsilon} = \frac{d_\Omega}{\sqrt{d_\Omega^2 + \varepsilon^2}},$$

where d_Ω is the signed distance function to $\partial\Omega$. Therefore $\delta_{\partial\Omega}$ is approximated by

$$\delta_{\partial\Omega,\varepsilon} = \frac{1}{2}|\nabla s_{\Omega,\varepsilon}|,$$

and a boundary integral is computed as

$$\int_{\partial\Omega} f \approx \int_{\mathbb{R}^d} \delta_{\partial\Omega,\varepsilon} f.$$

Finally, using finite differences and standard quadrature formulas makes this last approximation fully discrete (see [3] for details).

6.2 Approximation with a linear interpolation method

The previous Dirac mass approach is very simple to implement in practice but may well not be accurate enough in some cases. We therefore introduce another approach to evaluate boundary integrals. The main idea is to reconstruct a piecewise linear approximation of the boundary and compute exactly the boundary integral for a linear approximation of the integrand. There has been many works on improving the computation of surface integrals in the context of the level-set method [5], [4], [20], [34] Here we follow the lead of Min and Gibou [20, 21] and we compute boundary integrals as in a finite element method, building a mass matrix.

In numerical practice we work in 2-d with a regular square mesh $\mathcal{Q}_h$ of the computational domain in which all admissible shapes are included (extension to the 3-d case is conceptually simple although more cumbersome than in 2-d). Each rectangular cell is divided in four triangles separated by its diagonals. Let $\mathcal{T}_h$ be the resulting triangular mesh. The level-set function ϕ is discretized by finite differences on the square mesh. To obtain its $\mathbb{P}_1$ finite element interpolation on the triangular mesh $\mathcal{T}_h$ we simply assign to the additional node at the center of each square the average of the values at the four corner nodes. Once we have a $\mathbb{P}_1$ interpolation ϕ_h of ϕ on the triangular mesh, it is standard to obtain a discretization Ω_h of Ω which is defined by $\Omega_h = \{x \in \mathbb{R}^d \,|\, \phi_h(x) < 0\}$. By construction, the boundary of Ω_h is piecewise linear (more precisely it is the continuous union of line segments in each triangle). To compute an approximation of an integral on $\partial\Omega$, we compute an exact integral on $\partial\Omega_h$ of an interpolation of the integrand.

To be more specific, let us consider the approximate computation of a quadratic integral

$$I = \int_{\partial\Omega} v\, w,$$

where $v(x)$ and $w(x)$ are two functions which have v_h and w_h as $\mathbb{P}_1$ interpolation on $\mathcal{T}_h$. Our approximation of I is

$$I_h = \int_{\partial\Omega_h} v_h\, w_h,$$

where the last integral is evaluated exactly (it amounts to integrate a quadratic function on a line segment). In truth, representing v_h and w_h as vectors in the finite-dimensional space of finite elements, I_h is a bilinear form of these two vectors which is represented by a matrix. It is this (mass) matrix which is computed. This numerical method for computing boundary integrals is more precise than the approximation with a Dirac function. It is more time consuming but the additional burden is completely negligible in front of the CPU time required for the finite element analysis. The same process of discretizing $\partial\Omega$ in $\partial\Omega_h$ works also for computing bulk integrals on Ω. We refer to [36] for more details.

7 Numerical Examples

7.1 Setting of the problem

We start by describing our model in linearized elasticity. Let $\Omega \in \mathbb{R}^2$ be a smooth open bounded set, filled with a homogeneous isotropic material, having Hooke's tensor A, and Lamé's coefficients λ, μ. For a symmetric tensor ζ, we have $A\zeta = 2\mu\zeta + \lambda \mathrm{Tr}(\zeta)\mathrm{Id}$. We consider a partition of the boundary of Ω as

$$\partial\Omega = \Gamma \cup \Gamma_N \cup \Gamma_D\,,$$

where Γ_N and Γ_D remain unchanged, whereas Γ is the part to be optimized. We introduce a working domain $\mathcal{D}$ containing all admissible shapes. For $g \in H^1(\mathcal{D};\mathbb{R}^d)$ such that $g = 0$ on Γ, we consider the following boundary problem, with unknown u :

$$\left\{\begin{array}{rcll} -\mathrm{div}\big(A\varepsilon(u)\big) & = & 0 & \text{in } \Omega, \\ u & = & 0 & \text{on } \Gamma_D, \\ A\varepsilon(u)\mathbf{n} & = & g & \text{on } \Gamma_N, \\ A\varepsilon(u)\mathbf{n} & = & 0 & \text{on } \Gamma, \end{array}\right. \tag{31}$$

with the strain tensor $\varepsilon(u) = \frac{1}{2}\Big(\nabla u + (\nabla u)^T\Big)$. We introduce the Hilbert space

$$\mathcal{V} = \left\{u \in H^1(\Omega;\mathbb{R}^d) \mid u = 0 \text{ on } \Gamma_D\right\}.$$

The variationnal formulation of (31) is

$$\begin{cases} \text{find } u \in \mathcal{V} \text{ such that,} \\ \forall \varphi \in \mathcal{V},\ \displaystyle\int_\Omega A\varepsilon(u) : \varepsilon(\varphi) = \int_{\Gamma_N} g \cdot \varphi. \end{cases} \tag{32}$$

Our aim is to minimize an objective function subject to a volume constraint. A first common choice is the compliance defined as

$$J_1(\Omega) = \int_\Omega A\varepsilon(u) : \varepsilon(u) = \int_{\Gamma_N} g \cdot u.$$

It is also common to choose a least square error with a target displacement u_0

$$J_2(\Omega) = \sqrt{\int_\Omega k|u - u_0|^2},$$

where $k \in L^\infty(\mathcal{D})$ and $u_0 \in L^2(\mathcal{D};\mathbb{R}^d)$. In the following numerical examples for J_2, we shall take $u_0 = 0$ and $k(x)$ will be the characteristic function of a thin neighborhood of Γ_N. The volume is defined as

$$V(\Omega) = \int_\Omega 1\, dx.$$

For simplicity in the sequel, we shall fix the Lagrange multiplier $\Lambda > 0$ for the volume constraint and minimize, without any constraint, a Lagrangian L, defined by

$$L(\Omega) = J(\Omega) + \Lambda V(\Omega),$$

where J is either J_1 or J_2. Optimizing without constraints the Lagrangian L allows us to focus exclusively on the comparison between the gradient and the Newton algorithms.

7.2 Shape derivatives and implementation issues

When the shape Ω and the boundary load g are sufficiently smooth, the objective function and the volume are shape differentiable [18], [3]. We recall without proofs these shape derivatives.

Proposition 7.1. *Assume that Ω is of class C^2. Then, the volume $V(\Omega)$ is twice shape differentiable. For $\theta, \xi \in C^{1,\infty}\big(\mathbb{R}^d;\mathbb{R}^d\big)$, the maps l_1, l_2 (defined in Theorem 2.4) corresponding to the derivatives of $V(\Omega)$ are defined by*

$$l_1(\theta) = \int_{\partial\Omega} (\theta \cdot \mathbf{n}) \quad \text{and} \quad l_2(\theta,\xi) = \int_{\partial\Omega} \mathcal{H}\,(\theta \cdot \mathbf{n})\,(\xi \cdot \mathbf{n}),$$

where $\mathcal{H} = \operatorname{div}_\Gamma \mathbf{n}$ is the mean curvature of the boundary.

Proposition 7.2. *Assume that Ω is of class C^2 and that Γ_D and Γ_N are fixed. Then, the compliance J_1 is twice shape differentiable. For $\theta, \xi \in C^{1,\infty}\big(\mathbb{R}^d;\mathbb{R}^d\big)$, the maps l_1, l_2 corresponding to the derivatives of J_1 are defined by*

$$\begin{aligned} l_1(\theta) &= -\int_\Gamma (\theta \cdot \mathbf{n})\, A\varepsilon(u) : \varepsilon(u), \\ l_2(\theta,\xi) &= 2\int_\Omega A\varepsilon(u'_\theta) : \varepsilon(u'_\xi) - \int_\Gamma (\theta \cdot \mathbf{n})\,(\xi \cdot \mathbf{n}) \left(\mathcal{H} A\varepsilon(u) : \varepsilon(u) + \partial_{\mathbf{n}}\Big(A\varepsilon(u) : \varepsilon(u)\Big)\right), \end{aligned}$$

where $\mathcal{H}$ is the mean curvature on the boundary and u'_θ is the solution of

$$\begin{cases} -\operatorname{div}\big(A\varepsilon(u'_\theta)\big) &= 0 & \text{in } \Omega, \\ u'_\theta &= 0 & \text{on } \Gamma_D, \\ A\varepsilon(u'_\theta)\mathbf{n} &= 0 & \text{on } \Gamma_N, \\ A\varepsilon(u'_\theta)\mathbf{n} &= \operatorname{div}_\Gamma\big(\theta \cdot \mathbf{n}\, A\varepsilon(u)\big) & \text{on } \Gamma. \end{cases} \tag{33}$$

For the least square criteria, we denote $\tilde{J}_2(\Omega) = (J_2(\Omega))^2 = \int_\Omega k|u - u_0|^2$. In order to ease the reading, we give the derivatives of $\tilde{J}_2$ instead of J_2.

Proposition 7.3. *Assume that the domain Ω is of class C^2. The least square criteria $\tilde{J}_2$ is twice shape differentiable. For $\theta, \xi \in C^{1,\infty}\big(\mathbb{R}^d;\mathbb{R}^d\big)$, the map l_1 corresponding to the first shape derivative of $\tilde{J}_2$ is defined by*

$$l_1(\theta) = \int_\Gamma (\theta\cdot\mathbf{n})\left(\frac{C_0}{2}k|u-u_0|^2 + A\varepsilon(u):\varepsilon(p)\right),$$

where $C_0 = \Big(\int_\Omega k|u-u_0|^2\Big)^{-\frac{1}{2}}$ and the adjoint state p is assumed to be smooth, defined as the solution to

$$\left\{\begin{array}{rcll} -\mathrm{div}\big(A\varepsilon(p)\big) &=& -C_0k(u-u_0) & \text{in } \Omega,\\ p &=& 0 & \text{on } \Gamma_D,\\ A\varepsilon(p)\mathbf{n} &=& 0 & \text{on } \Gamma_N,\\ A\varepsilon(p)\mathbf{n} &=& 0 & \text{on } \Gamma. \end{array}\right. \tag{34}$$

The map l_2 corresponding to the second-order shape derivative of $\tilde{J}_2$ is defined by

$$\begin{aligned} l_2(\theta,\xi) = \frac{2}{C_0}\int_\Gamma (\xi\cdot\mathbf{n})\,A\varepsilon(u'_\theta):\varepsilon(p) + (\theta\cdot\mathbf{n})\,A\varepsilon(u'_\xi):\varepsilon(p) + 2\int_\Omega k\,u'_\theta\cdot u'_\xi\\ + 2\int_\Gamma k(u-u_0)\Big((\theta\cdot\mathbf{n})\,u'_\xi + (\xi\cdot\mathbf{n})\,u'_\theta\Big)\\ + \frac{2}{C_0}\int_\Gamma (\theta\cdot\mathbf{n})\,(\xi\cdot\mathbf{n})\Big(\partial_{\mathbf{n}}\mathcal{J} + \mathcal{H}\mathcal{J}\Big), \end{aligned}$$

where $\mathcal{H}$ is the mean curvature on the boundary, u'_θ is the smooth solution to (33) and

$$\mathcal{J} = \frac{C_0}{2}k|u-u_0|^2 + A\varepsilon(u):\varepsilon(p)\,.$$

We now discuss some implementation issues. We use a single regular square mesh $\mathcal{Q}_h$ for both the Hamilton-Jacobi equation and the linearized elasticity systems, like (31), (33), (34). The Hamilton-Jacobi equation is solved by a second-order finite difference scheme and the elasticity equations by $\mathbb{Q}_1$ finite elements with an ersatz material approach for the void region (see [3] for details). These algorithms are implemented in Scilab [8] with some routines (concerning the evaluation of the objective function and their gradients) in C.

We use the gradient algorithm 5.4 or the Newton algorithm 5.2, and the corresponding quadratic problems (28), (29) are solved by using Ipopt [37]. The numerical parameters ε, η for both algorithms 5.4 and 5.2 are set to $\varepsilon = \eta = 10^{-5}$. Even if we can observe a decrease on the L^2-norm of the descent direction, it appears that it is always the criteria on the descent step that stops the algorithms. The descent direction is a priori defined only on $\partial\Omega$, and we need to extend it to the entire domain $\mathcal{D}$ in order to solve the Hamilton-Jacobi equation (26). We compare the three extension methods proposed in Section 4.

The most tricky part is the computation of the matrix discretizing the Hessian operator, and more precisely the bilinear form l_2. Note that the discretization of the linear form l_1 (or the first order derivative) is standard [3]. A first remark concerns the assumed smoothness of the shape Ω for deriving the above formulas. In numerical practice, the optimal structures may exhibit corners, i.e., are not smooth. In such a case the mean curvature $\mathcal{H} = \mathrm{div}_\Gamma\,\mathbf{n}$ is not well defined. To mitigate this effect, we truncate the value of the discretized version of $\mathcal{H}$ by a maximum (absolute) value of the order of $1/h$ where h is the mesh size. A second remark deals with the computation of normal derivatives of $A\varepsilon(u):\varepsilon(u)$ or $\mathcal{J}$ in the formulas for l_2. When u and p are $\mathbb{Q}_1$, we need to interpolate their gradients as $\mathbb{Q}_1$ finite elements before computing these normal derivatives. The third and most important comment is about the matrix arising from the discretization of l_2. It is a full matrix of size proportional to the number of nodes belonging to a mesh cell cut by the boundary $\partial\Omega$. Therefore its storage is a real issue, at least for 3-d problems, but we did not yet experience it in 3-d. A possible cure could be to store only elements which are above a threshold, in absolute value. Furthermore, the evaluation of the Hessian matrix requires the knowledge of the shape derivatives of the displacement, namely u'_θ and u'_ξ. For a given $\theta \in \mathbb{P}_1\big(\mathcal{T}_h;\mathbb{R}^2\big)$, computing u'_θ corresponds to solving the linear system with the stiffness matrix and the right-hand side parametrized by θ in (33). In 2-d, the stiffness matrix is already factorized for the computation of u. There are as many different right-hand sides as the number of nodes in the vicinity of the boundary (this number is usually much smaller than the number of cells in the mesh). This computation of the Hessian is not too expensive if the stiffness matrix is factorized and stored, as is easily the case in 2-D. However, in 3-d, when iterative solvers are used, the computational cost may be outrageous. For example in 2-d, and without any performance optimization (using Scilab), for a mesh of size 120×60 (as used in the next section) the computation of the Hessian is of the order of $5s$ in CPU time, whereas the computation of the gradient is of the order of $100ms$. We plan to study incomplete Hessian assembly or approximate Newton methods for 3-d effectiveness.

7.3 Arch

As a first test case, we consider an arch under boundary conditions described by Figure 1. We consider a working domain of size 2×1 with a square mesh of size 120×60. For both the compliance and the least square criteria, the

applied force is a point load $(0, -1)$ applied at the middle of the bottom boundary. The Young's modulus and the Poisson coefficient are respectively $E = 1.0$, $\nu = 0.3$. In the definition of the Lagrangian, the Lagrange multiplier is $\Lambda = 20$ for the initialization with holes and $\Lambda = 8$ for the initialization without holes.

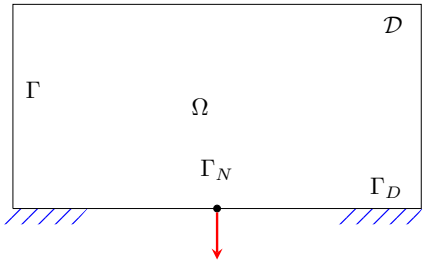

Figure 1: Boundary condition for the arch.

7.3.1 Least square criteria - Initialization with holes

In this example we consider the least square error with the target displacement $(0, 0)$ on the boundary Γ_N. Since both components of the displacement are minimized on Γ_N this least square objective function is different from compliance. The initial shape is the one of Figure 2

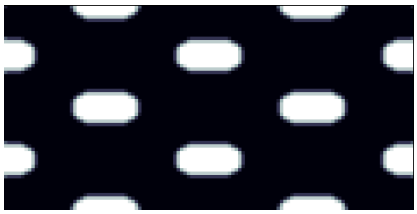

Figure 2: Initial shape for the arch.

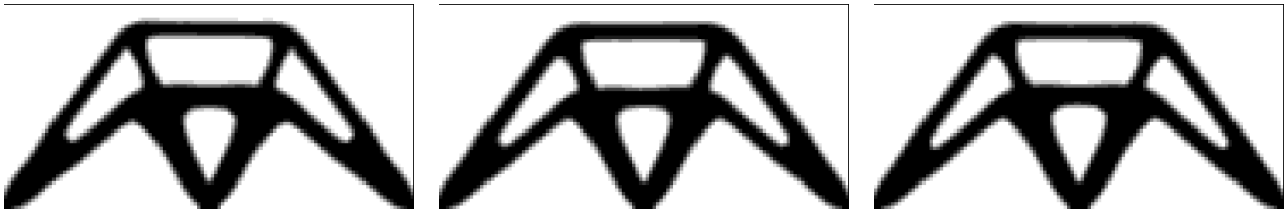

Figure 3: Final domains for the gradient algorithm on problem of Section 7.3.1. From left to right, gradient with extension 4.3, gradient with extension 4.1, gradient with extension 4.2.

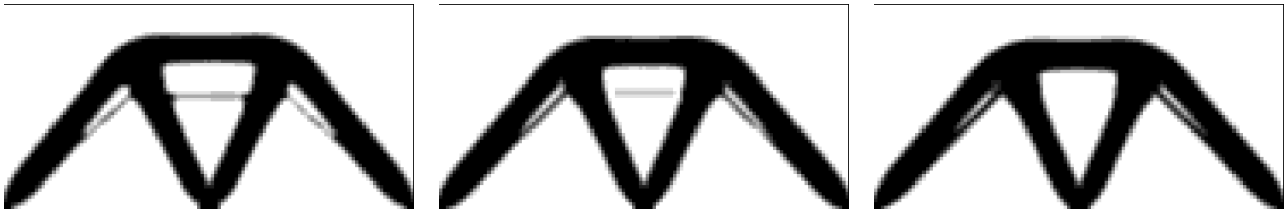

Figure 4: Final domains for Newton's algorithm on problem of Section 7.3.1. From left to right, Newton with extension 4.3, Newton with extension 4.1, Newton with extension 4.2.

We denote Ω_∞ the final shape for each method, and $\Omega^\star$ the best of all six final shapes (which are very similar). We keep this notation in the sequel. The final value of the Lagrangian and the number of iterations needed to get convergence are reported in Figure 5. Due to the line search, each iteration may need several finite element computations for the evaluation of the criteria. Therefore, we also give the number of finite element computations needed to get convergence.

	Extension 4.3			Extension 4.1			Extension 4.2		
	Lagrangian	Iterations	FE	Lagrangian	Iterations	FE	Lagrangian	Iterations	FE
gradient	26.991224	92	1006	26.894358	105	1032	26.868040	110	1235
Newton	26.454090	38	128	26.173114	43	121	26.125447	40	150

Figure 5: Number of iterations, of finite element computations and value of $L(\Omega_\infty)$ for each algorithm for the arch problem of Section 7.3.1.

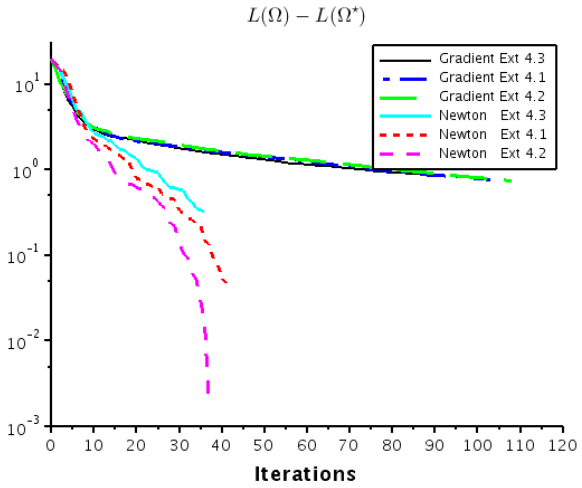

Figure 6: Convergence of $L(\Omega) - L(\Omega^\star)$ for problem of Section 7.3.1

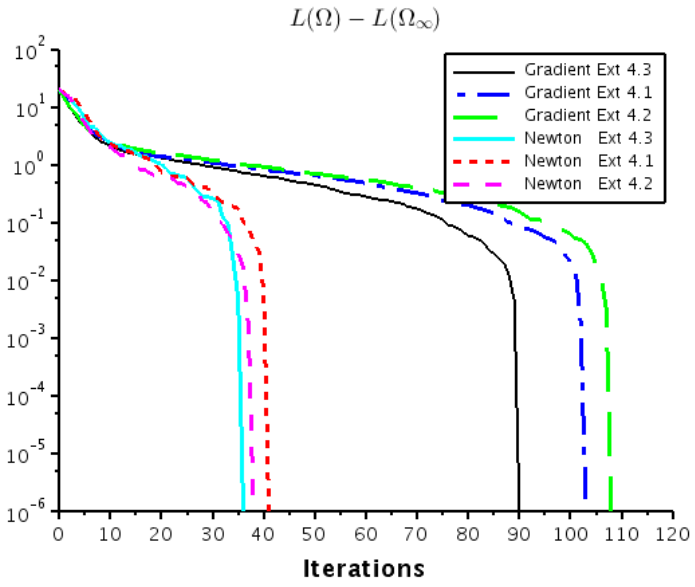

Figure 7: Convergence of $L(\Omega) - L(\Omega_\infty)$ for problem of Section 7.3.1.

In this first example one can observe that the best shape is obtained with the Newton method and the extension 4.2. Looking at Figure 7 we see that there are three methods that converge with fewer iterations, but Figure 6 shows that their optimal objective values are worse. Comparing the curves of the Newton method, we can also conclude that the improvement of the second order method relies on the computation of the Hessian but also on the choice of the extension which has a great influence on the result.

Of course, the final convergence of $L(\Omega)$ to $L(\Omega_\infty)$ is very fast but it is a lure since, by definition, $L(\Omega_\infty)$ is the limit value of $L(\Omega)$. Nevertheless, the convergence histories of Figures 6 and 7 are good measures of convergence speed for the early iterations. Plotting the residual of the derivative (some norm of the normal velocity v^p in the

advection equation (26), for example) is deceptive and not a good convergence indicator [36]. The reason (already explained in [3]) is that our optimization algorithms rely on continuous derivatives and, as is well known, derivation and discretization do not always commute.

7.3.2 Least square criteria - Initialization without holes

Secondly we consider the same example but the initialization differs since it has no holes

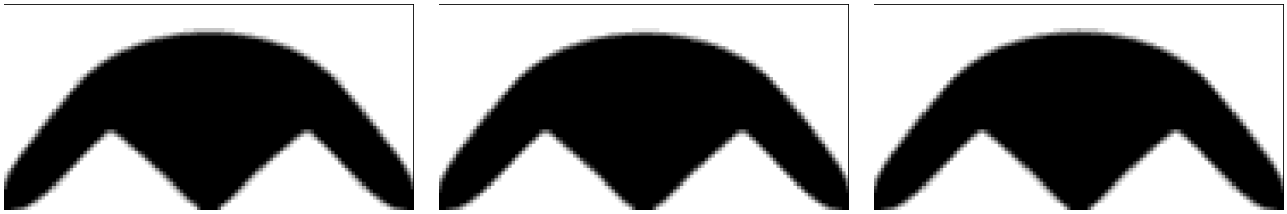

Figure 8: Final domains for the gradient algorithm on problem of Section 7.3.2. From left to right, gradient with extension 4.3, gradient with extension 4.1, gradient with extension 4.2.

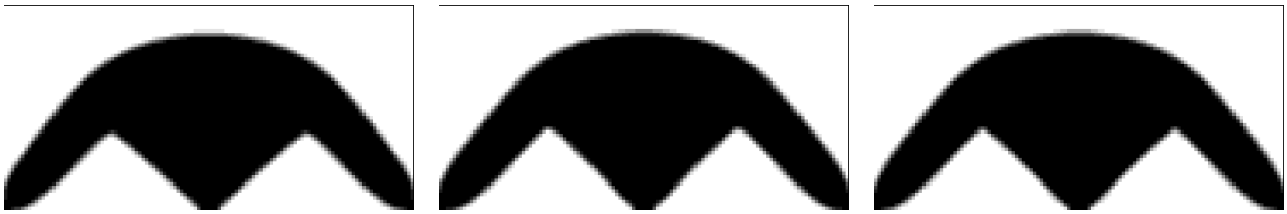

Figure 9: Final domains for Newton's algorithm on problem of Section 7.3.2. From left to right, Newton with extension 4.3, Newton with extension 4.1, Newton with extension 4.2.

The final value of the Lagrangian and the number of iterations needed to get convergence are transcripted in Figure 10.

	Extension 4.3			Extension 4.1			Extension 4.2		
	Lagrangian	Iterations	FE	Lagrangian	Iterations	FE	Lagrangian	Iterations	FE
gradient	10.641893	55	353	10.627132	100	847	10.643709	59	415
Newton	10.654021	50	162	10.623230	36	99	10.651637	41	156

Figure 10: Number of iterations, of finite element computations and value of $L(\Omega_\infty)$ for each method for problem of Section 7.3.2.

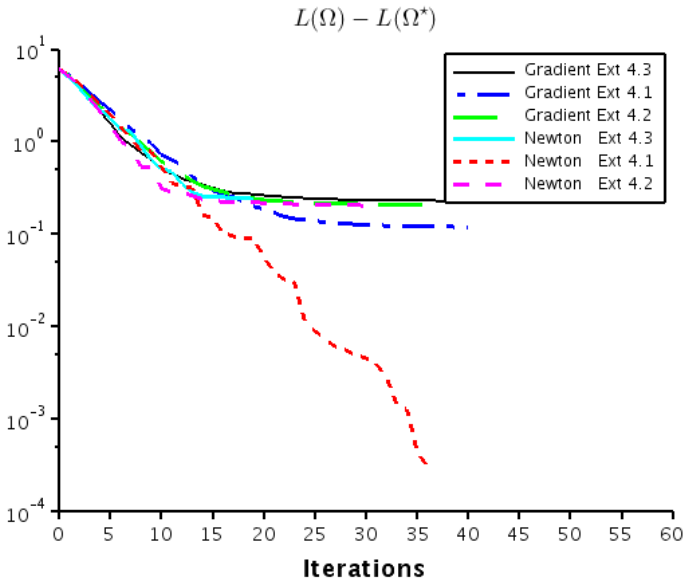

Figure 11: Convergence of $L(\Omega) - L(\Omega^\star)$ on problem of Section 7.3.2.

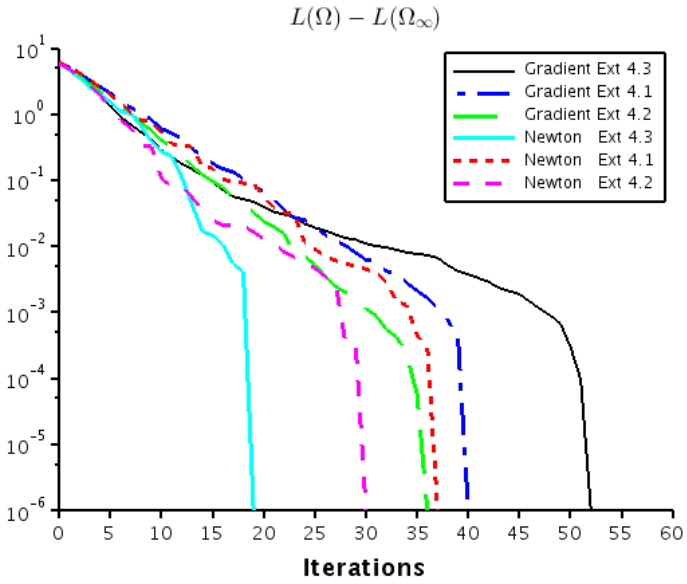

Figure 12: Convergence of $L(\Omega) - L(\Omega_\infty)$ on problem of Section 7.3.2.

In this example the main advantage of the Newton method consists in converging nearly to the best of all shapes in fewer iterations than with only a gradient method. With the Newton method, the convergence is also obtained with fewer finite elements computations.

7.3.3 Compliance - Initialization without holes

Next we consider the compliance for the arch, for an initialization without holes.

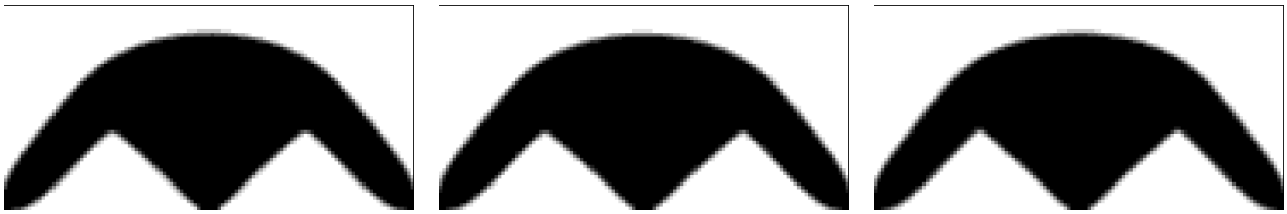

Figure 13: Final domains for the gradient algorithm on problem of Section 7.3.3. From left to right, gradient with extension 4.3, gradient with extension 4.1, gradient with extension 4.2.

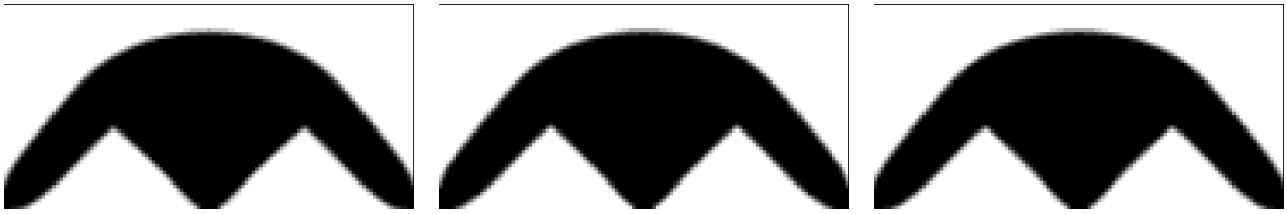

Figure 14: Final domains for Newton's algorithm on problem of Section 7.3.3. From left to right, Newton with extension 4.3, Newton with extension 4.1, Newton with extension 4.2.

In Figure 15 we can see the final values of the Lagrangian and the number of iterations needed.

	Extension 4.3			Extension 4.1			Extension 4.2		
	Lagrangian	Iterations	FE	Lagrangian	Iterations	FE	Lagrangian	Iterations	FE
gradient	18.210883	48	466	18.137610	35	225	18.227131	28	191
Newton	18.225898	22	96	17.970161	31	76	18.192965	24	132

Figure 15: Number of iterations, of finite element computations and value of $L(\Omega_\infty)$ for each method for problem of Section 7.3.3.

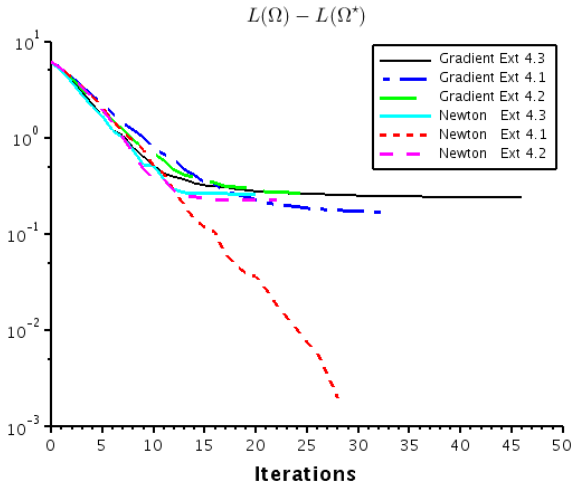

Figure 16: Convergence of $L(\Omega) - L(\Omega^\star)$ on problem of Section 7.3.3

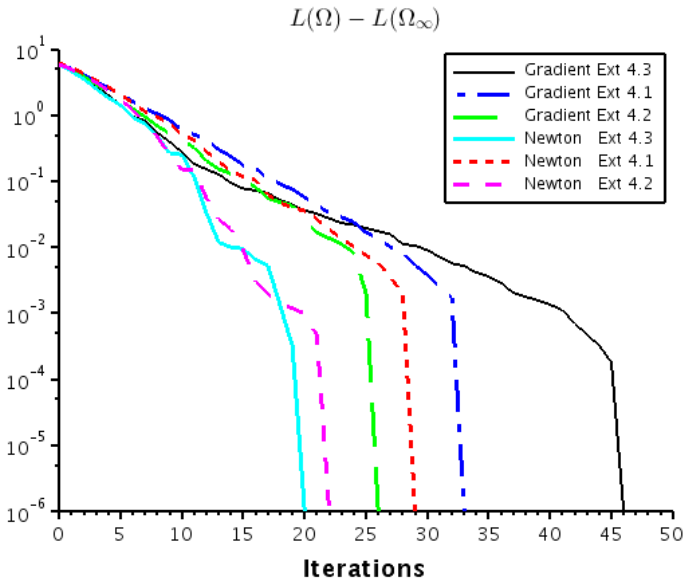

Figure 17: Convergence of $L(\Omega) - L(\Omega_\infty)$ on problem of Section 7.3.3.

The extension 4.3 leads to the best shapes for both Newton and gradient method, but it has a great cost in terms of iterations. Excepting this two results, the best shape is obtained by the Newton method with extension 4.2. In this case there is not much difference - for the Newton method - between the extensions 4.3 and 4.2.

7.3.4 Compliance - Initialization with holes

We continue to minimize compliance but with an initialization with holes as in Figure 2. In this example, we take the opportunity to report, not only the total number of iterations and finite element analyses, but also the total number of linear system solves. In the case of the gradient algorithm, each finite element analysis corresponds to one and only one linear system to solve (for compliance minimization there is no adjoint equation). However, for Newton's algorithm, at each iteration we compute the Hessian matrix which requires to solve as many equations (33) as there are degrees of freedom on the boundary $\partial\Omega$. Solving (33) amounts to solve a linear system with the same rigidity matrix as the state equation (31). Therefore, if the rigidity matrix is factorized once per iteration, the additional cost of the Newton's algorithm is just a large number of back-and-forth substitutions. Factorizing the rigidity matrix, and thus minimizing the overhead of Newton's algorithm is possible in 2-d but is clearly a problem in 3-d (see the Conclusion for possible remedies). We recall that, for the moment, we do not try to minimize the CPU and memory cost of Newton's algorithm.

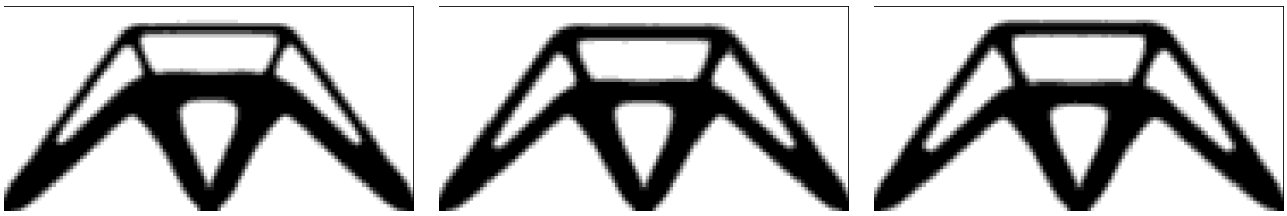

Figure 18: Final domains for the gradient algorithm on problem of Section 7.3.4. From left to right, gradient with extension 4.3, gradient with extension 4.1, gradient with extension 4.2.

Figure 19: Final domains for Newton's algorithm on problem of Section 7.3.4. From left to right, Newton with extension 4.3, Newton with extension 4.1, Newton with extension 4.2.

		Gradient	Newton
Extension 4.3	Lagrangian	26.60038339	26.61049365
	Iterations	81	29
	FE	771	90
	linear solves	771	41061
Extension 4.1	Lagrangian	26.72446997	26.59986314
	Iterations	78	36
	FE	645	96
	linear solves	645	51976
Extension 4.2	Lagrangian	26.83716099	26.43690880
	Iterations	68	32
	FE	608	109
	linear solves	608	47308

Figure 20: Number of iterations, of finite element analyses, of linear system solves and value of $L(\Omega_\infty)$ for each method for problem of Section 7.3.4.

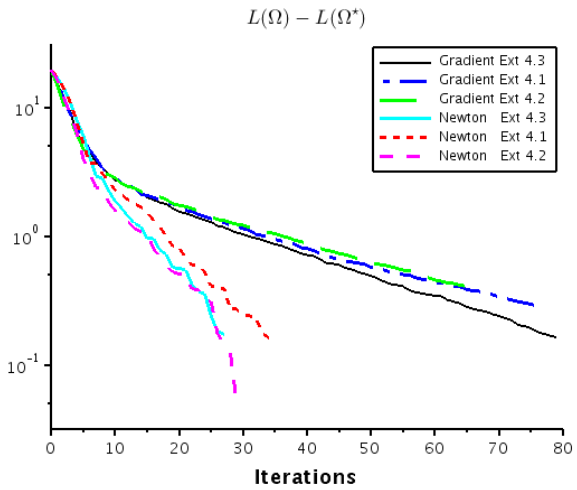

Figure 21: Convergence of $L(\Omega) - L(\Omega^\star)$ for problem of Section 7.3.4

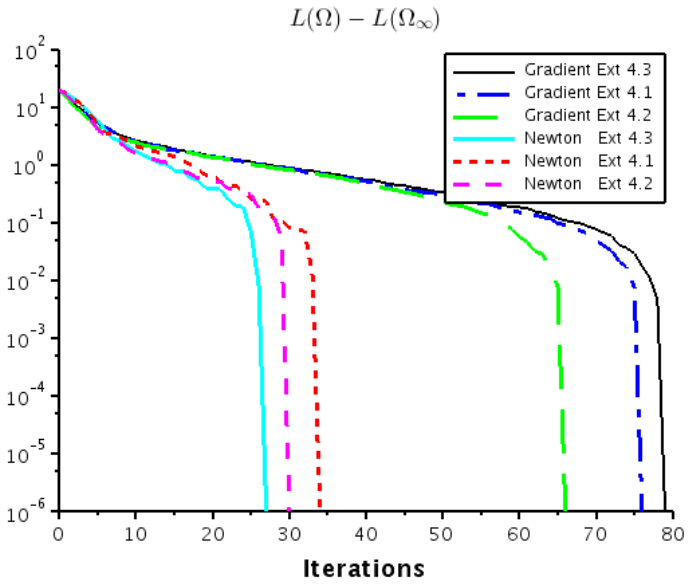

Figure 22: Convergence of $L(\Omega) - L(\Omega_\infty)$ for problem of Section 7.3.4.

7.3.5 Compliance - Initialization with holes and a finer 200x100 mesh

To assess the mesh dependency, or scalability, of Newton's algorithm, compared to the gradient algorithm, we run the same test case as in the previous Section 7.3.4, but on a finer mesh of size 200x100 (while the previous one was 120x60). Starting from the same initialization, the final domains are the same (but with different values of the Lagrangian).

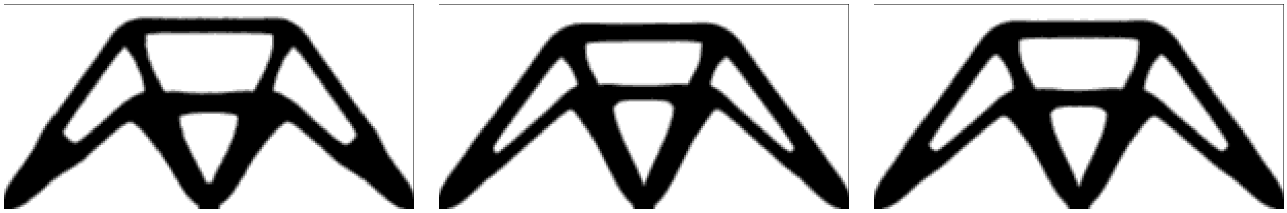

Figure 23: Final domains for the gradient algorithm and the fine mesh of Section 7.3.5. From left to right, gradient with extension 4.3, gradient with extension 4.1, gradient with extension 4.2.

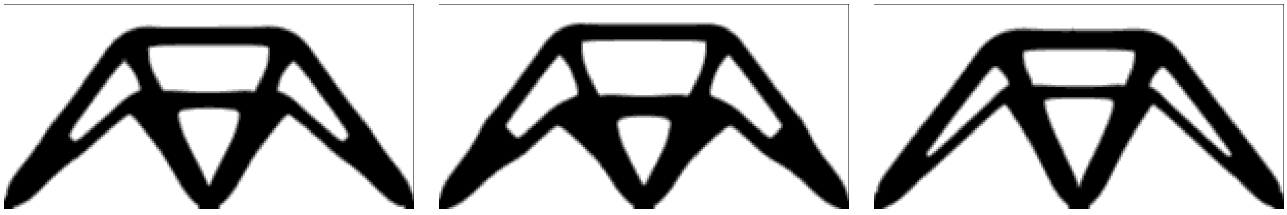

Figure 24: Final domains for Newton's algorithm and the fine mesh of Section 7.3.5. From left to right, Newton with extension 4.3, Newton with extension 4.1, Newton with extension 4.2.

		Gradient	Newton
Extension 4.3	Lagrangian	28.72140629	28.43652421
	Iterations	65	51
	FE	496	122
	linear solves	496	125950
Extension 4.1	Lagrangian	28.19959181	28.81666153
	Iterations	138	54
	FE	1123	130
	linear solves	1123	131798
Extension 4.2	Lagrangian	28.41156703	28.05832997
	Iterations	99	66
	FE	835	168
	linear solves	835	171261

Figure 25: Number of iterations, of finite element analyses, of linear system solves and value of $L(\Omega_\infty)$ for each method on the fine mesh of Section 7.3.5.

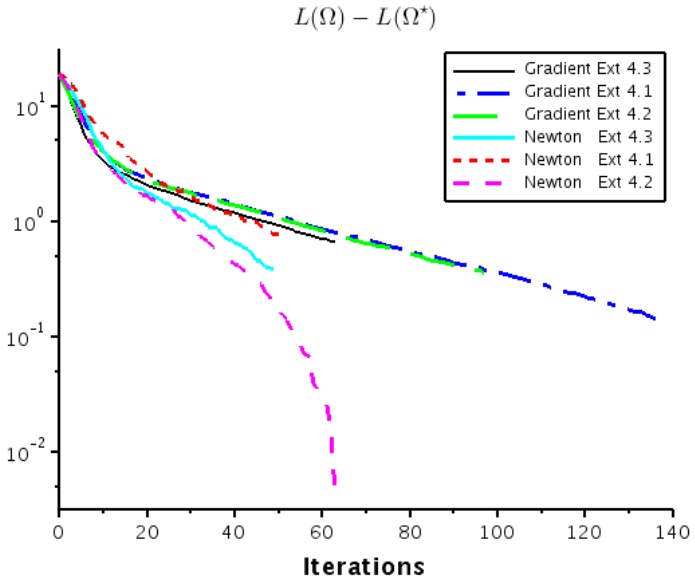

Figure 26: Convergence of $L(\Omega) - L(\Omega^\star)$ for the fine mesh of Section 7.3.5.

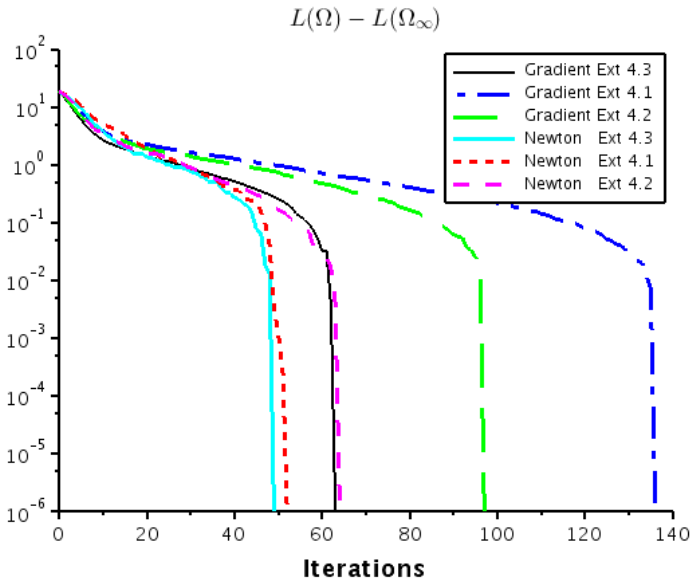

Figure 27: Convergence of $L(\Omega) - L(\Omega_\infty)$ for the fine mesh of Section 7.3.5.

Compared to the coarser mesh of the previous Section 7.3.4, the number of iterations and finite element analyses has roughly doubled (except for the exceptional case of the gradient algorithm with extension 4.3, the convergence of which has prematurately stopped). Since the number of cells has doubled in each space direction too, the number of degrees of freedom on the boundary $\partial\Omega$ has roughly doubled. Therefore it is not a surprise that the number of linear system solves for Newton's algorithm has approximately quadrupled. In any case, it is clear that the cost of Newton's algorithm may turn out to be prohibitive for fine meshes, motivating the development of inexact (but cheaper) Newton strategies.

7.4 Cantilever

Our second and last test case is a cantilever. The working domain and the mesh have the same size than in the case of the arch, i.e., 2×1 and 120×60. For both the compliance and the least square criteria, the applied force is a point load $(0, -0.1)$ applied at the middle of the right boundary. The Young's modulus and the Poisson coefficient are still respectively $E = 1.0$, $\nu = 0.3$.

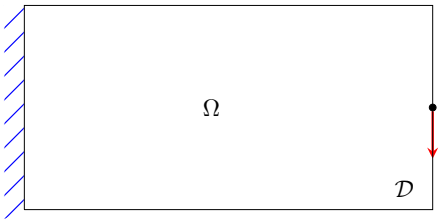

Figure 28: Cantilever

7.4.1 Least square criteria - Initialization with Holes

We first consider the least square error on a cantilever for an initialization with holes (see Figure 29). The criteria is similar to the case of the arch, since we take the same target displacement, i.e $(0, 0)$ on Γ_N. In the definition of the Lagrangian, the Lagrange multiplier is $\Lambda = 16$.

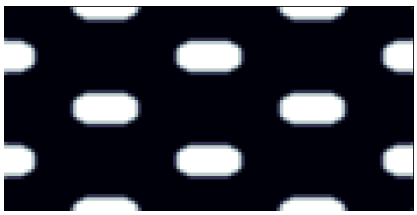

Figure 29: Initial shape for the cantilever.

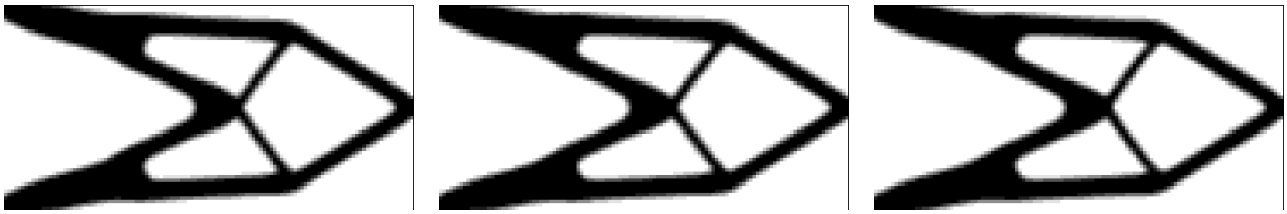

Figure 30: Final domains for the gradient algorithm on problem of Section 7.4.1. From left to right, gradient with extension 4.3, gradient with extension 4.1, gradient with extension 4.2.

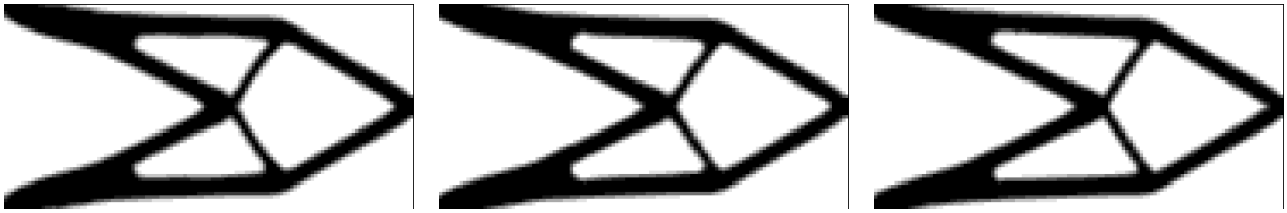

Figure 31: Final domains for Newton's algorithm on problem of Section 7.4.1. From left to right, Newton with extension 4.3, Newton with extension 4.1, Newton with extension 4.2.

	Extension 4.3			Extension 4.1			Extension 4.2		
	Lagrangian	Iterations	FE	Lagrangian	Iterations	FE	Lagrangian	Iterations	FE
gradient	19.740827	32	290	19.718258	33	269	19.755948	33	309
Newton	19.656573	23	82	19.665630	22	80	19.571101	22	95

Figure 32: Number of iterations, of finite element computations and value of $L(\Omega_\infty)$ for each method for the cantilever problem of Section 7.4.1.

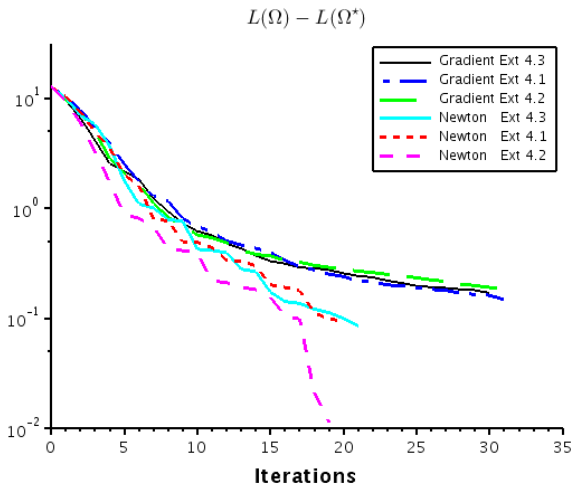

Figure 33: Convergence of $L(\Omega) - L(\Omega^\star)$ on problem of Section 7.4.1

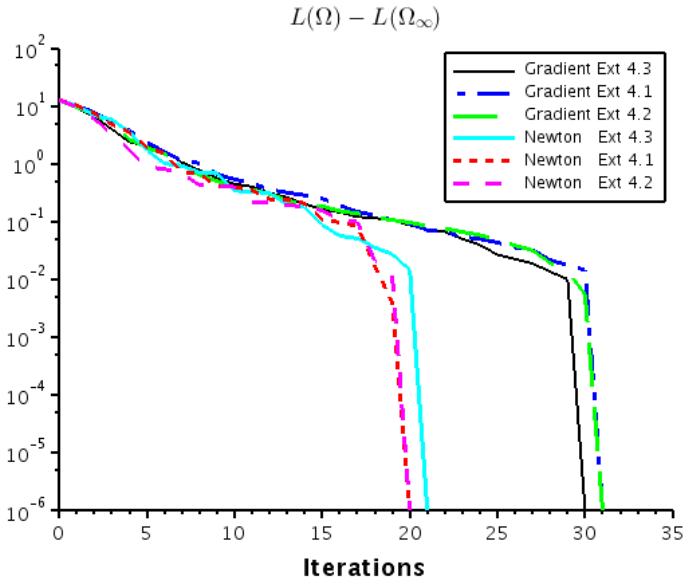

Figure 34: Convergence of $L(\Omega) - L(\Omega_\infty)$ on problem of Section 7.4.1.

The best shape is obtained with the gradient algorithm and extension 4.1, at the price of many iterations (compared to the other algorithms). The second best shape is obtained with Newton's algorithm and extension 4.2 with three times less iterations.

7.4.2 Compliance - Initialization with Holes

Now we consider the compliance for the cantilever with an initialization with holes (see Figure 29). In the definition of the Lagrangian, the Lagrange multiplier is $\Lambda = 1$.

Figure 35: Final domains for the gradient algorithm on problem of Section 7.4.2. From left to right, gradient with extension 4.3, gradient with extension 4.1, gradient with extension 4.2.

Figure 36: Final domains for Newton's algorithm on problem of Section 7.4.2. From left to right, Newton with extension 4.3, Newton with extension 4.1, Newton with extension 4.2.

	Extension 4.3			Extension 4.1			Extension 4.2		
	Lagrangian	Iterations	FE	Lagrangian	Iterations	FE	Lagrangian	Iterations	FE
gradient	1.571077	23	195	1.570056	34	276	1.572323	30	283
Newton	1.565558	21	84	1.574471	13	53	1.566995	11	54

Figure 37: Number of iterations, of finite element computations and value of $L(\Omega_\infty)$ for each method for problem of Section 7.4.2.

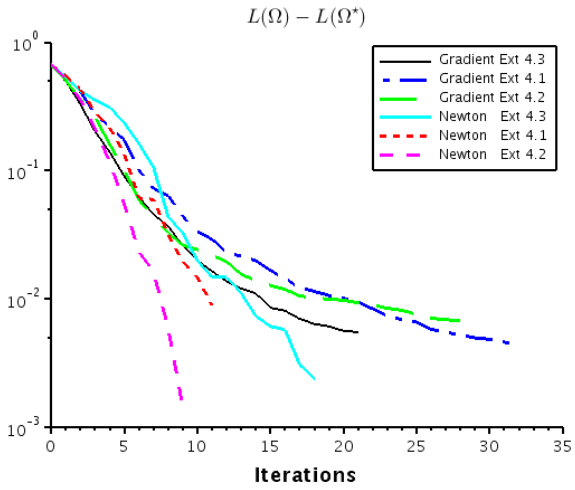

Figure 38: Convergence of $L(\Omega) - L(\Omega^\star)$ on problem of Section 7.4.2

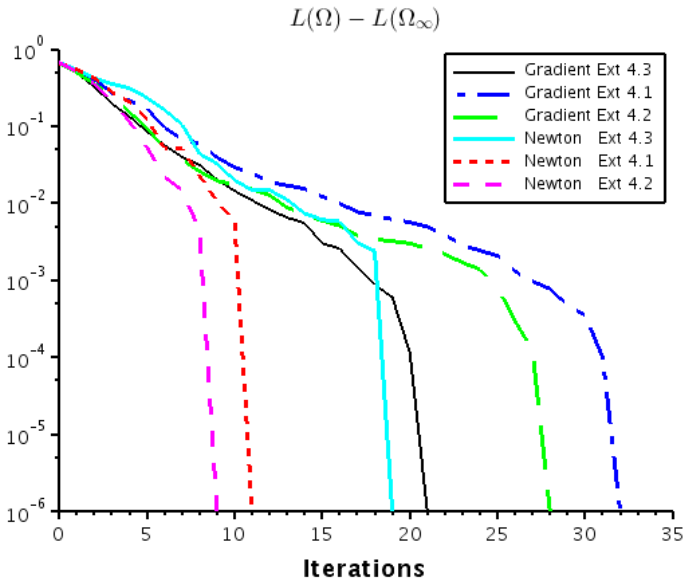

Figure 39: Convergence of $L(\Omega) - L(\Omega_\infty)$ on problem of Section 7.4.2.

Here, the main advantage of Newton's algorithm combined with extension 4.2 is to lower the numbers of iterations and of finite element computations. Compared to the gradient algorithm, the convergence is obtained with at least two times less finite elements computation, which is the most expensive part of the computation.

8 Conclusion

In this work we presented a new approach for the second order shape derivative, adapted to the level-set setting. The resulting formula for the Hessian is simpler since it does not involve tangential components of the shape displacement (contrary to the previous variants of the Hadamard method of shape variations). As a consequence the Newton's algorithm is simpler to implement. This new formula gives also a clear indication on how to extend a descent direction from the boundary $\partial\Omega$ to the whole computational domain $\mathcal{D}$.

Our first numerical examples, presented here, are very encouraging in showing that a good combination of a second order algorithm and a suitable extension may significantly improve the optimization process by reducing the number of iterations. However it remains a lot of work to do in order to confirm this appraisal. In particular, we did not pay too much attention to the CPU time and rather counted only the iteration numbers (both from optimization and from finite element analysis). We plan to investigate more carefully the topic. Several potential ideas, in order to minimize the CPU time, are as follows. It is customary to apply quasi or inexact Newton algorithms instead of plain Newton. In this line of thoughts, we think of approximating the Hessian instead of computing it exactly. Since the time consuming part of the computation of the Hessian amounts in solving linear systems with the stiffness matrix (to obtain the shape derivatives of the displacement, namely u'_θ), it could be worthwhile to approximate its inverse (by a few diagonals) and then to keep the structure of the global Hessian.

Of course, we need to test our approach on 3-d examples, where the cost of computing the Hessian would be way more important. Even in 2-d, we intend to perform more various examples and to vary the mesh size. Since the Newton method is a priori known to be efficient close to the optimum, at the end of the optimization, it could be interesting to try combine, in a first step, the gradient algorithm, and in a second step, the Newton's algorithm, when approaching convergence. Another key issue is to test our approach in the case of constrained optimization problems. In such a case, one has to compute the Hessian of the Lagrangian [24]. However, our idea from Section 4 for extending the descent direction is not completely clear in view of the constraints. In any case, we plan to address those issues in a forthcoming paper or in [36].

References

[1] G. Allaire. *Conception optimale de structures*, volume 58 of *Mathématiques & Applications (Berlin) [Mathematics & Applications]*. Springer-Verlag, Berlin, 2007.

[2] G. Allaire, F. Jouve, and A.-M. Toader. A level-set method for shape optimization. *C. R. Math. Acad. Sci. Paris*, 334(12):1125–1130, 2002.

[3] G. Allaire, F. Jouve, and A-M. Toader. Structural optimization using sensitivity analysis and a level-set method. *Journal of Computational Physics*, 194(1):363–393, February 2004.

[4] A.-K. Tornberg B. Engquist and R. Tsai. Discretization of Dirac delta functions in level-set methods. *Journal of Computational Physics*, 207(1):28 – 51, 2005.

[5] J. Th. Beale. A proof that a discrete delta function is second-order accurate. *Journal of Computational Physics*, 227(4):2195 – 2197, 2008.

[6] D. Bucur and J-P. Zolésio. Anatomy of the shape Hessian via Lie brackets. *Annali di Matematica pura ed applicata*, IV(CLIX):315–339, 1997.

[7] M. Burger. A framework for the construction of level-set methods for shape optimization and reconstruction. *Interfaces and Free Boundaries*, 5:301–329, 2003.

[8] S. L. Campbell, J.-Ph. Chancelier, and R. Nikoukhah. *Modeling and simulation in Scilab/Scicos*. Springer, New York, 2006.

[9] M. G. Crandall and P.-L. Lions. Viscosity solutions of Hamilton-Jacobi equations. *Trans. Amer. Math. Soc.*, 277:1–42, 1983.

[10] M. Dambrine. On variations of the shape Hessian and sufficient conditions for the stability of critical shapes. *RACSAM. Rev. R. Acad. Cienc. Exactas Fís. Nat. Ser. A Mat.*, 96(1):95–121, 2002.

[11] M. Dambrine and M. Pierre. About stability of equilibrium shapes. *Mathematical Modeling and Numerical Analysis*, 34(4):811–834, 3 2000.

[12] F. de Gournay. Velocity extension for the level-set method and multiple eigenvalues in shape optimization. *SIAM Journal on Control and Optimization*, 45(1):343–367, 2006.

[13] M-C. Delfour and J-P. Zolesio. Velocity method and Lagrangian formulation for the computation of the shape Hessian. *SIAM Journal on Control and Optimization*, 29(6):1414–1442, 1991.

[14] M. C. Delfour and J.-P. Zolésio. *Shapes and geometries*, volume 4 of *Advances in Design and Control*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2001. Analysis, differential calculus, and optimization.

[15] M-C. Delfour and J-P. Zolésio. Anatomy of the shape Hessian. *Annali di Matematica pura ed applicata*, CLIX(IV):315–339, 1991.

[16] M. Giaquinta and S. Hildebrandt. *Calculus of Variations I*, volume 310 of *Grundlehren der mathematischen Wissenschaften*. Springer Berlin Heidelberg, 2004.

[17] J. Hadamard. Mémoire sur le problème d'analyse relatif à l'équilibre des plaques élastiques encastrées. In *Oeuvres de J. Hadamard*, volume II, pages 515–641. C.N.R.S., 1907.

[18] A. Henrot and M. Pierre. *Variation et optimisation de formes*, volume 48. Springer, 2005.

[19] H. Kasumba and K. Kunisch. On computation of the shape Hessian of the cost functional without shape sensitivity of the state variable. *J. Optim. Theory Appl.*, 162(3):779–804, 2014.

[20] C. Min and F. Gibou. Geometric integration over irregular domains with application to level-set methods. *Journal of Computational Physics*, 226(2):1432 – 1443, 2007.

[21] C. Min and F. Gibou. Robust second-order accurate discretizations of the multi-dimensional Heaviside and Dirac delta functions. *Journal of Computational Physics*, 227(22):9686 – 9695, 2008.

[22] B. Mohammadi and O. Pironneau. *Applied shape optimization for fluids*. Numerical Mathematics and Scientific Computation. Oxford University Press, Oxford, second edition, 2010.

[23] F. Murat and J. Simon. Sur le contrôle par un domaine géométrique. Pré-publication du laboratoire d'analyse numérique, no 76015, Université Paris VI, 1976.

[24] J. Nocedal and S-J. Wright. *Numerical Optimization*. Springer, 1999.

[25] A. Novruzi and M. Pierre. Structure of shape derivatives. *Journal of Evolution Equations*, 2:365–383, 2002.

[26] A. Novruzi and J. R. Roche. Newton's method in shape optimisation : a three-dimensional case. *BIT*, 40(1):102–120, 2000.

[27] S. Osher and J-A. Sethian. Fronts propagating with curvature-dependent speed : Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79(1):12 – 49, 1988.

[28] J. Rauch. *Hyperbolic partial differential equations and geometrics optics*, volume 133 of *Graduate Studies in Mathematics*. American Mathematical Society, 2012.

[29] J. R. Roche. Adaptive Newton-like method for shape optimization. *Control Cybernet.*, 34(1):363–377, 2005.

[30] G. Rozvany. Aims, scope, methods, history and unified terminology of computer-aided topology optimization in structural mechanics. *Struct. Multidiscip. Optim.*, 21(2):90–108, 2001.

[31] G. Rozvany. A critical review of established methods of structural topology optimization. *Struct. Multidiscip. Optim.*, 37(3):1–38, 2009.

[32] J. Simon. Differentiation with respect to the domain in boundary value problems. *Numer. Funct. Anal. Optim.*, 2:649–687, 1980.

[33] J. Simon. Second variation for domain optimization problems. *Control and estimation of distributed parameter systems, F. Kappel, K. Kunish et W. Schappacher éd., International Series of Numerical Mathematics*, 91:361–378, 1989.

[34] P. Smereka. The numerical approximation of a delta function with application to level-set methods. *Journal of Computational Physics*, 211(1):77 – 90, 2006.

[35] J. Sokolowski and J-P. Zolesio. *Introduction to Shape Optimization*, volume 16 of *Springer Series in Computational Mathematics*. Springer Berlin Heidelberg, 1992.

[36] J.-L. Vié. *Second-order methods for shape optimization with the level-set method.* PhD thesis, Université Paris Est, 2016.

[37] A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[38] M. Y. Wang, X. Wang, and D. Guo. A level-set method for structural topology optimization. *Comput. Methods Appl. Mech. Engrg.*, 192(1-2):227–246, 2003.

[39] Y.-X. Yuan. A review of trust region algorithms for optimization. volume 99, pages 271–282. International Congress on Industrial and Applied Mathematics, Oxford University Press, 2000.

[40] J.-P. Zolésio. The material derivative (or speed) method for shape optimization. In *Optimization of distributed parameter structures, Vol. II (Iowa City, Iowa, 1980)*, volume 50 of *NATO Adv. Study Inst. Ser. E: Appl. Sci.*, pages 1089–1151. Nijhoff, The Hague, 1981.