ESAIM: COCV 25 (2019) 3
https://doi.org/10.1051/cocv/2017014
ESAIM: Control, Optimisation and Calculus of Variations
www.esaim-cocv.org

SHAPE OPTIMIZATION *VIA* A LEVELSET AND A GAUSS-NEWTON METHOD

Jérôme Fehrenbach[1] and Frédéric de Gournay[2,*]

Abstract. In the context of shape optimization *via* level-set methods, we propose a general framework for a Gauss-Newton method to optimize quadratic functionals. Our approach provides a natural extension of the shape derivative as a vector field defined in the whole working domain. We implement and discuss this method in two cases: first a least-square error minimization reminiscent of the Electrical Impedance Tomography problem, and second the compliance problem with volume constraints.

Mathematics Subject Classification. 49Q10, 49M15, 49M29, 74P05, 74P10, 74P20.

Received June 20, 2016. Accepted February 18, 2017.

1. Introduction

Shape optimization adresses the problem of finding an optimal domain Ω among a set of admissible domains, with respect to a given criterion. We are interested in the case where the objective function depends on the solution u_Ω of a partial differential equation.

Different approaches and numerical implementations have been proposed since the 1960s and it is not possible to credit all the contributions here. The difference in the methods lie

- In the various ways of describing a shape: parametrization of the shape see *e.g.* [9], implicit representation using level-sets [23], numerical relaxation of the characteristic function to have values in the interval $[0, 1]$ [7], relaxation of the continuous problem by homogenization method [3].
- In the way the optimal domain is estimated. We will use Hadamard's boundary sensitivity method [18]. Other methods include topological sensitivity analysis [17, 29], which estimates the variation of the criterion after nucleation of infinitesimal holes, standard parameter sensitivity analysis in the case of relaxation since then the unknown is a real-valued function, and lamination parameters estimation in the case of homogenization.

The classical contributions in shape optimization mentioned above proposed to minimize the objective using first order minimization methods. In the search of improving the convergence and to assess the stability of the solution, other works addressed shape optimization problems using second order methods. The second variation of a cost functional w.r.t. the domain was derived first by Simon [28]. A necessary optimality condition is that the first derivative cancels, but it is not sufficient and the coercivity of the Hessian needs to be accounted for.

Keywords and phrases. Shape optimization, shape derivative, second order method, level-set method.

[1] Institut de Mathématiques de Toulouse, UMR CNRS 5219, Université de Toulouse, France
[2] INSA de Toulouse, Institut de Mathématiques de Toulouse, UMR CNRS 5219, Université de Toulouse, France.
*Corresponding author: `frederic@degournay.fr`

Detailed studies including regularity estimates showed that if the domain is $C^{2,\alpha}$ and the perturbation is a $C^{2,\alpha}$ diffeomorphism then the Hessian is coercive at the minimum, but with respect to a weaker norm [11, 12]. The detection of obstacles for the conductivity equation using second order shape optimization was proposed in a series of work [1, 2, 15] under various hypotheses regarding the obtacle: perfectly insulating, perfectly conducting, having a different conductivity. Another series of work adressed electromagnetic shaping, and the explicit computation of the Hessian that is affordable using a boundary integral method to estimate the solutions of the equation [22, 24].

More recently Kasumba and Kunisch [20] proposed to compute the shape Hessian without computing the sensitivity of the state, which allows to weaken the regularity hypotheses. In [4] the authors propose a new method of shape parametrization that allows to define a second order derivative that is directed along the normal to the boundary of Ω. A clever implementation of this method provides convergence in less iterations than the first order descent, although the explicit computation of the Hessian requires a heavy computational load.

We propose in the present work to adress shape optimization using Gauss-Newton method, which is an approximate second order method dedicated to the minimization of quadratic cost functions of a variable θ that are written

$$\frac{1}{2}\|F(\theta)\|^2.$$

It is only "approximate" second order since the Hessian is not computed (the second derivatives are not computed), and it is approximated by first order derivatives. The strategy that we adopt can be sketched as follows. We define a scalar product on the set Θ of vector fields defined on Ω and our method is a Gauss-Newton method relatively to this scalar product. Therefore the vector fields that we consider are already defined on the whole domain Ω. The descent direction d is obtained by solving the Gauss-Newton update equation:

$$\mathrm{d}F^\star \mathrm{d}F\,.d = -\mathrm{d}F^\star F.$$

In the present work, the domain is represented by a level-set and a key ingredient is a local remeshing operation and a re-initialization of the distance function on non-cartesian grid using Dapogny-Frey's algorithm [13]. We treat the case a generic elliptic partial differential equation and present two applications where the objective can be expressed as a quadratic cost function: L^2-fit and compliance for the linear elasticity equation.

The present paper is organized as follows. In Section 2 we define the setting of the problem and the notation that will be used in the sequel. In Section 3 we recall classical facts regarding Gauss-Newton method, and emphasize the differences with the gradient method. In Section 4 we provide the sensitivity of the direct state u_Ω with respect to a variation of the shape, and direct and adjoint derivatives of the objective function (in the L^2 fit case and in the compliance case). Section 5 details the numerical methods that were implemented, and Section 6 contains the presentation and discussion of the results.

2. Notation and settings

2.1. Setting of the problem

Consider a second order elliptic system of s coupled PDEs set on a domain $\Omega \subset \mathbb{R}^d$ whose boundary $\partial\Omega$ is partitionned into a Dirichlet part $\Gamma_D(\Omega)$ and a Neumann part $\Gamma_N(\Omega)$. Denote

$$H^1_D(\Omega) = \{u \in L^2(\Omega) \text{ s.t. } \nabla u \in L^2(\Omega), u^i = 0 \qquad \text{on} \qquad \Gamma_D(\Omega), \forall i \in S_D\},$$

where $S_D \subset [1, s]$ is the subset of indices on which the Dirichlet boundary conditions are applied. Let A be a fourth order tensor of size $d \times d \times s \times s$ such that a Poincaré inequality holds for some constant $C > 0$:

$$\int_\Omega A\nabla u \colon \nabla u \geq C \int_\Omega u \cdot u \quad \forall u \in H^1_D(\Omega).$$

A necessary condition for this Poincaré inequality to hold is that both S_D and $\Gamma_D(\Omega)$ are non-empty. In order to clarify the notation, we write in coordinates

$$A\nabla u\colon \nabla\phi = \sum_{ijkl} A^{ij}_{kl} \frac{\partial u^k}{\partial x_i}\frac{\partial \phi^l}{\partial x_j}$$

In this case, for any $f \in L^2(\Omega)$, there exists a unique u_Ω minimizer of

$$\min_{u\in H^1_D(\Omega)} \int_\Omega A\nabla u\colon \nabla u - 2\int_\Omega f\cdot u.$$

When A is symmetric, the minimizer u_Ω is a solution to the corresponding Euler-Lagrange equation, or so-called variational formulation:

$$\int_\Omega A\nabla u_\Omega\colon \nabla\phi = \int_\Omega f\cdot\phi, \quad \forall \phi\in H^1_D(\Omega).$$

The strong form of the above equation is:

$$\begin{cases} -\mathrm{div}(A\nabla u_\Omega) = f & \text{in} & \Omega & \\ u^i_\Omega = 0 & \text{on} & \Gamma_D(\Omega) & \forall i\in S_D \\ A\nabla u\colon e^j\otimes n = 0 & \text{on} & \Gamma_D(\Omega) & \forall j\notin S_D \\ A\nabla u_\Omega\colon n = 0 & \text{on} & \Gamma_N(\Omega) & \end{cases}, \tag{2.1}$$

where n is the outer normal of the domain Ω.

For instance, the choice $s = d$ and

$$A^{ij}_{kl} = \lambda\delta^i_k\delta^j_l + \mu\left(\delta^{ij}\delta_{kl} + \delta^i_l\delta^j_k\right),$$

where δ is the standard Kronecker's symbol, leads to Hooke's law which describes isotropic linear elasticity, and the choice $A^{ij}_{kl} = \delta^{ij}\delta_{kl}$ leads to the s dimensional Laplacian. The goal of the present article is to propose a new algorithm designed to optimize w.r.t. the domain Ω a quadratic functional of the state u_Ω, and to test this algorithm in two cases where the state solves a linear elasticity equation. We address first the minimization of the least square error (L^2 fit) to a target state u_G on a subdomain ω:

$$\mathcal{J}_{ls}(\Omega) = \int_\omega |u_\Omega - u_G|^2.$$

In our second toy model we optimize the compliance defined by

$$\mathcal{J}_c(\Omega) = \int_\Omega A\nabla u_\Omega : \nabla u_\Omega.$$

Note that the ingredients of our method can in principle be applied to other quadratic cost functions.

2.2. Differentiable structure

It is well known that the set of domains is not a vector space and does not have a differentiable structure. Following the approach of Murat−Simon [21], there exists a manifold structure on the sets of domains that can be attained by diffeomorphisms from an original domain. To make things clear, denote $\Omega\subset\mathbb{R}^d$ the working domain, and for any $\theta\in W^{1,\infty}(\mathbb{R}^d)$ (which is a Banach space) we consider the domain $\Omega_\theta = (Id+\theta)(\Omega)$. The domain Ω_θ is defined in a neighbourhood of $\theta = 0$, namely if $\|\theta\|_{1,\infty} < 1$ then $Id+\theta$ is a diffeomorphism. The "geometric shape derivative" is the differential of the functional of interest with respect to θ. In practical examples, we might require that some region of the original domain Ω is non-optimisable, hence that θ is equal to zero on this non-optimisable subset denoted as $\omega_{\text{n.o.}}$. Moreover, in order to show differentiability, the Dirichlet part of Ω_θ has to be the image of $\Gamma_D(\Omega)$ under the diffeomorphism $Id+\theta$, in other words $\Gamma_D(\Omega_\theta) = (Id+\theta)(\Gamma_D(\Omega))$. In the applications considered here the Dirichlet part is defined as $\Gamma_D(\Omega_\theta) = \partial\Omega_\theta \cap \Gamma_D^0$, where Γ_D^0 is part of the data. The boundary part Γ_D^0 must therefore be invariant under the diffeomorphism $Id+\theta$. For this reason we enforce $\theta\cdot n = 0$ on Γ_D^0 and consider the following space

$$\Theta = \left\{\theta \in W^{1,\infty}(\mathbb{R}^d) \quad \text{such that} \quad \theta = 0 \quad \text{on} \quad \omega_{\text{n.o.}} \quad \text{and} \quad \theta\cdot n = 0 \quad \text{on} \quad \Gamma_D^0\right\}.$$

From now on, functionals that depend on Ω, will be (locally) interpreted as functionals on Θ, which is a Banach space so that it makes sense to differentiate w.r.to $\theta \in \Theta$. Otherwise specified, differentiation is taken at the origin $\theta = 0$.

Since shape optimization is an optimization problem set on a manifold, the design of second order methods is more involved because a choice has to be made in order to compute the transport of a tangent plane to another one. The advantage of the Gauss-Newton method is that it requires only first order information and is known to be able to achieve local quadratic convergency rate.

3. Gauss-Newton method

3.1. Gradient descent *vs.* Gauss-Newton descent

The main contribution of the present work is to design a Gauss-Newton method in order to minimize $\mathcal{J}$ over the admissible sets Ω. We recall that the Gauss-Newton method is defined in the general case of a differentiable function $F : X \to Y$, where X and Y are Hilbert spaces. It aims at minimizing the quadratic functional

$$\mathcal{J}(x) = \frac{1}{2}\|F(x)\|_Y^2$$

by minimizing at iteration k the local quadratic approximation:

$$\min_h \frac{1}{2}||F(x_k) + \mathrm{d}F(x_k).h||^2$$

One iteration of the Gauss-Newton algorithm is then defined as follows:

- Compute $\mathrm{d}F$ the differential of F and $\mathrm{d}F^\star$ its adjoint. The adjoint is defined *via* the Hilbertian structures of X and Y as:
$$\langle \mathrm{d}F^\star.u, v\rangle_X = \langle u, \mathrm{d}F.v\rangle_Y \quad \forall u \in Y, v \in X$$
- Implement an iterative minimisation algorithm based on the following update rule
$$x_{k+1} = x_k + d_k,$$
with d_k the direction of descent given by
$$(\mathrm{d}F^\star \mathrm{d}F)d_k = -\mathrm{d}F^\star F. \tag{3.1}$$

We recall here for comparison that the gradient method to minimize a functional $\mathcal{J} : X \to \mathbb{R}$ is defined by

- Implement an iterative minimisation algorithm based on the following update rule
$$x_{k+1} = x_k + s_k d_k,$$
with s_k an admissible step given by a linesearch method (see *e.g.* [31]) and d_k the direction of descent given by
$$d_k = -\nabla\mathcal{J}. \tag{3.2}$$

In the quadratic case when $\mathcal{J}(x) = \dfrac{1}{2}\|F(x)\|_Y^2$ with $F : X \to Y$ then $\nabla\mathcal{J} = \mathrm{d}F^\star F$, and equation (3.2) reads

$$d_k = -\mathrm{d}F^\star F.$$

Let us emphasize the differences between the gradient descent and Gauss-Newton descent:

(1) Gauss-Newton method does not require the computation of a step since the optimal step length is 1 close to the minimum.
(2) On the other hand, the price to pay is the solution of the linear system (3.1). However note that the solution of this equation does not require the computation and storage of the matrix $\mathrm{d}F^\star\mathrm{d}F$: in large dimension it is convenient to implement the product of a given vector by the matrix $\mathrm{d}F^\star\mathrm{d}F$. The equation (3.1) is then solved using an iterative method like conjugate gradient. An approximate resolution is sufficient and can be obtained by performing a handfull products by $\mathrm{d}F^\star\mathrm{d}F$. This strategy is called *matrix-free Gauss-Newton methods* [16, 26].
(3) Gauss-Newton method is known to have a faster convergence than the gradient method in a neighbourhood of a minimum (local quadratic convergence under suitable hypotheses). An heuristic explanation of this fact (see [30]) is that the descent direction in both methods is orthogonal to the level-sets of the functional $\mathcal{J}$ but for different scalar products. In the case of the gradient descent it is the original scalar product in the space X, whereas in the Gauss-Newton case it is an adapted scalar product that makes locally (around a minimum) the levels-sets circles and not ellipses. It is well known that the lines orthogonal to a circle point towards the center, which is not the case for ellipses.

3.2. Hilbertian structure

In this section we describe the Hilbertian structure Θ on the set of variables. In order to work out the Gauss-Newton method, the space with respect to which the differentiation is performed has to be endowed with a scalar product. We choose the following scalar product:

$$\langle\theta,\eta\rangle_\Theta = \int_{\mathbb{R}^d} \alpha\theta\cdot\eta + \nabla\theta : \nabla\eta, \tag{3.3}$$

with $\alpha > 0$ ($\alpha = 10$ in the tests presented in Sect. 6). Let us emphasize the fact that the space Θ is not an Hilbert space, since the corresponding scalar product is not complete for the norm $\|\bullet\|_{W^{1,\infty}(\Omega)}$. A natural choice would be to work in the completion of Θ with respect to this scalar product but this approach is also doomed since the functional would in generality fail to be differentiable. We choose not to work with a scalar product defined on a smaller subset of Θ which completion is included in Θ (*e.g.* take a scalar product that involves products of derivatives of degree 3 in dimension 2) because it would lead to unwanted numerical regularization.

Note. In the standard shape optimization using gradient descent, the space Θ should also be endowed with a scalar product. In practice, it follows from Hadamard's theorem and Riesz representation theorem that the differential of a cost function $\mathcal{J}$ can (if it exists) be expressed as an integral over the boundary $\partial\Omega$ of a quantity g multiplied by the normal trace $\theta\cdot n$, that is:

$$\mathrm{d}\mathcal{J}.\theta = \int_{\partial\Omega} g(\theta\cdot n) \tag{3.4}$$

The so-called 'gradient' is the quantity g which amounts to use as a scalar product on the set Θ the L^2 product of the normal trace on the boundary. This does not either define a Hilbert space structure on Θ, but is reasonnably effective. In the present work we study the effectiveness of another abuse described above. Note also that previous work [14] already used the scalar product defined by equation (3.3) on the space Θ.

4. Direct and adjoint derivatives

4.1. Derivative of the solution

In this section, we recall the formula of the shape derivative of the solution, and sketch the main steps of the complete proof.

It is well known that the solution of the PDE under consideration admits a geometric shape derivative [18]. Denote $u_{\Omega_\theta} \in H^1_D(\Omega_\theta)$ the solution to the elliptic PDE and denote $u_\theta = u_{\Omega_\theta} \circ (Id + \theta)$, the pull back of u_{Ω_θ} on Ω. As long as $\Gamma_D(\Omega_\theta) = (Id+\theta)\circ\Gamma_D(\Omega)$, it is a standard result to show that u_θ belongs to the space $H^1_D(\Omega)$.

Proposition 4.1 (Derivative of the state [6, 19, 21])**.** *The maping $\theta \mapsto u_\theta$ is differentiable from $W^{1,\infty}(\mathbb{R}^d)$ to $H^1_D(\Omega)$ around the value $\theta = 0$. If $(\mathrm{d}u.\theta) \in H^1_D(\Omega)$ denotes the value of this differential at the point 0 in the direction θ, it is the solution to the following equation in variational form:*

$$\int_\Omega A\nabla(\mathrm{d}u.\theta) \colon \nabla\phi = \int_\Omega C(\theta)\nabla u_\Omega \colon \nabla\phi + \int_\Omega (\nabla f \cdot \theta + div(\theta) f)\phi \quad \forall\phi \in H^1_D(\Omega),$$

whith the notation $(\nabla f \cdot \theta)^i = \sum_m \partial_m f^i \theta^m$, and where we define:

$$C(\theta)^{ij}_{kl} = \sum_m \left(-\frac{\partial\theta^m}{\partial x_m} A^{ij}_{kl} + \frac{\partial\theta^m}{\partial x_i} A^{mj}_{kl} + \frac{\partial\theta^m}{\partial x_j} A^{im}_{kl} \right).$$

The proof of this proposition is detailed in the references [6, 19, 21], we only sketch the main ideas here. The variational formulation of the equation reads

$$\int_{\Omega_\theta} A\nabla u_{\Omega_\theta} \colon \nabla\psi = \int_{\Omega_\theta} f\psi \quad \forall\psi \in H^1_D(\Omega_\theta).$$

We perform a change of variable with $\Omega_\theta = T(\Omega)$, $T = Id + \theta$, to obtain

$$\int_\Omega A(\nabla u_{\Omega_\theta}) \circ T \colon (\nabla\psi) \circ T |\det \nabla T| = \int_\Omega f \circ T\, \psi \circ T |\det \nabla T| \quad \forall\psi \in H^1_D(\Omega_\theta).$$

We denote $\phi = \psi \circ T$ so that when ψ describes $H^1_D(\Omega_\theta)$, then ϕ describes $H^1_D(\Omega)$ and we recall the notation $u_\theta = u_{\Omega_\theta} \circ T$. According to Leibniz' rule:

$$(\nabla u_{\Omega_\theta}) \circ T = \nabla u_\theta (\nabla T)^{-1} \circ T, \quad \nabla\psi \circ T = \nabla\phi(\nabla T)^{-1} \circ T,$$

and finally we use the implicit function theorem which allows a first order expansion in terms of θ which yields the corresponding result.

4.2. Derivative of the L^2 fit

In the rest of the paper we consider the case where $s = d$ and A is the linear elasticity operator, although the ideas presented can be applied to any elliptic operator.

We address in this section the optimization of the domain with respect to the L^2 fit to a given objective. Let us consider the following quadratic cost function:

$$\mathcal{J}(\theta) = \frac{1}{2}\int_\omega (u_{\Omega_\theta} - u_G)^2,$$

where the target state $u_G \in L^2(\mathbb{R}^2)$ and the subdomain ω is not optimizable (in other words $\theta = 0$ on ω). In pratical implementation, we will use a neighbourhood of the boundary as observation domain ω, hence this problem is close to Electrical Impedance Tomography [8, 10] and bears the same kind of instability.

A change of variable with $T = Id + \theta$ shows that

$$\mathcal{J}(\theta) = \frac{1}{2}\int_\omega |\mathrm{det}\nabla T|(u_\theta - u_G \circ T)^2 = \frac{1}{2}\int_\omega (u_\theta - u_G)^2,$$

since θ vanishes on ω. We introduce the space $\mathcal{M} = L^2(\omega)$ with its standard scalar product and the functional

$$\begin{aligned} F : \mathit{\Theta} &\longrightarrow \mathcal{M} \\ \theta &\longmapsto F(\theta) = u_\theta - u_G. \end{aligned}$$

Then $\mathcal{J}$ is a quadratic cost function written as

$$\mathcal{J}(\theta) = \frac{1}{2}\|F(\theta)\|^2_{\mathcal{M}}.$$

Proposition 4.2 (Direct derivative for the L^2 fit)**.** *The mapping $F : \mathit{\Theta} \to \mathcal{M}$ is differentiable at the point $\theta = 0$. Moreover, its differential $\mathrm{d}F.\theta$ at point 0 in direction θ is equal to*

$$\mathrm{d}F.\theta = (\mathrm{d}u.\theta)\mathbf{1}_\omega,$$

where the derivative $\mathrm{d}u.\theta$ of the state was given in Proposition 4.1.

Proposition 4.3 (Adjoint derivative for the L^2 fit)**.** *Let $z \in L^2(\mathit{\Omega})$. If we denote $\mathrm{d}F^\star$ the adjoint of $\mathrm{d}F$, then $\mathrm{d}F^\star$ solves*

$$\forall\theta \in \mathit{\Theta}, \quad \langle \mathrm{d}F^\star z|\theta\rangle_{\mathit{\Theta}} = \int_{\mathit{\Omega}} C(\theta)\nabla u_{\mathit{\Omega}}.\nabla p + \int_{\mathit{\Omega}} (\nabla f\cdot\theta + \mathrm{div}(\theta)f)p$$

where the adjoint state p solves

$$\int_{\mathit{\Omega}} A\nabla\phi.\nabla p = -\int_\omega \phi z \qquad \forall\phi \in H^1_D(\mathit{\Omega}).$$

Proof. The differentiation of F follows from the differentiation of u_θ. The computation of $\mathrm{d}F^\star$ follows from the computation of $\mathrm{d}F$ and the use of the adjoint p in order to get rid of the term $\mathrm{d}u.\theta$ in equation (4.1) below:

$$\begin{aligned} \langle \mathrm{d}F^\star z|\theta\rangle_{\mathit{\Theta}} &= \int_\omega z\mathrm{d}F.\theta = \int_{\mathit{\Omega}} A\nabla(\mathrm{d}u.\theta).\nabla p \qquad (4.1)\\ &= \int_{\mathit{\Omega}} C(\theta)\nabla u_{\mathit{\Omega}}.\nabla p + \int_{\mathit{\Omega}} (\nabla f\cdot\theta + \mathrm{div}(\theta)f)p \end{aligned}$$ □

4.3. Case of multiple loads for the L^2 fit

In order to stabilize the problem to solve, we use multiple loads and hence multiple target states. More precisely, let $f^1, \ldots, f^m$ be m different loads. The target state u^i_G is obtained by solving the state equation (2.1) with the i-th source term f^i on the true (to be recovered) domain. The multiple target cost function to be minimized is then

$$\mathcal{J}_{\mathrm{multi}}(\theta) = \frac{1}{2}\sum_{i=1}^m \int_\omega (u^i_\theta - u^i_G)^2, \qquad (4.2)$$

where u^i_θ is obtained by solving equation (2.1) on the domain $\mathit{\Omega}_\theta$ with source term f^i, and ω still denotes the observation domain.

If we denote

$$\begin{aligned} F^i : \mathit{\Theta} &\longrightarrow \mathcal{M} \\ \theta &\longmapsto F^i(\theta) = u^i - u^i_G, \end{aligned}$$

then the Gauss-Newton update equation (3.1) is replaced by the following multiple loading update equation:

$$\left(\sum_{i=1}^{m} \mathrm{d}F^{i\star}\mathrm{d}F^{i}\right)\eta = -\sum_{i=1}^{m}\mathrm{d}F^{i\star}F^{i}, \tag{4.3}$$

where the product of a given vector of Θ, resp. of $L^2(\Omega)$, by the operator $\mathrm{d}F^i$, resp. $\mathrm{d}F^{i\star}$, is given by Proposition 4.2, resp. Proposition 4.3.

4.4. Derivative of the compliance

Let us consider the following cost function composed of two terms: the compliance plus a Lagrange multiplier for the volume:

$$\mathcal{L}(\theta) = \frac{1}{2}\int_{\Omega_\theta} A\nabla u_{\Omega_\theta} \colon \nabla u_{\Omega_\theta} + \frac{\lambda}{2}\int_{\Omega_\theta} 1. \tag{4.4}$$

A change of variable with $T = Id + \theta$ shows that

$$\mathcal{L}(\theta) = \frac{1}{2}\int_{\Omega} |\mathrm{det}\nabla T| A(\nabla u_\theta \nabla T^{-1}) \colon (\nabla u_\theta \nabla T^{-1}) + \frac{\lambda}{2}\int_{\Omega} |\mathrm{det}\nabla T|.$$

We consider the space

$$\mathcal{M} = \left\{ \begin{pmatrix} m \\ v \end{pmatrix} \text{ s.t. } \quad m_i^k \in L^2(\Omega), v \in L^2(\Omega) \text{ for } 1 \le i \le d, 1 \le k \le s \right\}$$

endowed with the scalar product

$$\left\langle \begin{pmatrix} m \\ v \end{pmatrix}, \begin{pmatrix} \tilde{m} \\ \tilde{v} \end{pmatrix} \right\rangle_{\mathcal{M}} = \int_\Omega Am \colon \tilde{m} + \lambda \int_\Omega v \cdot \tilde{v}.$$

Let F be the mapping from $W^{1,\infty}(\mathbb{R}^d)$ to $\mathcal{M}$ defined by

$$F(\theta) = \begin{pmatrix} \sqrt{|\mathrm{det}\nabla T|}\nabla u_\theta \nabla T^{-1} \\ \sqrt{|\mathrm{det}\nabla T|} \end{pmatrix}.$$

The cost function that we adress reads:

$$\mathcal{L}(\theta) = \frac{1}{2}\|F(\theta)\|^2_{\mathcal{M}}.$$

Proposition 4.4 (Direct derivative for the compliance)**.** *For $\theta \in \Theta$ we have*

$$\mathrm{d}F.\theta = \begin{pmatrix} \frac{1}{2}\mathrm{div}(\theta)\nabla u_\Omega - \nabla u_\Omega \nabla\theta + \nabla(\mathrm{d}u.\theta) \\ \frac{1}{2}\mathrm{div}(\theta) \end{pmatrix}.$$

Proof. It follows from the chain rule. □

Proposition 4.5 (Adjoint derivative for the compliance)**.** *For any $z = \begin{pmatrix} m \\ v \end{pmatrix} \in \mathcal{M}$, we have*

$$\begin{aligned}\langle \mathrm{d}F^\star z, \theta\rangle_\Theta = \int_\Omega Am \colon \left(\frac{1}{2}\mathrm{div}(\theta)\nabla u - \nabla u \nabla\theta\right) + \lambda\int_\Omega \frac{1}{2}\mathrm{div}(\theta) v \\ + \int_\Omega C(\theta)\nabla u_\Omega \colon \nabla p + \int_\Omega (\nabla f \cdot \theta + \mathrm{div}(\theta) f)p\end{aligned}$$

where the adjoint state p solves:

$$\int_\Omega A\nabla\phi \colon \nabla p = \int_\Omega Am \colon \nabla\phi \qquad \forall \phi \in H^1_D(\Omega).$$

Proof. We first use the calculation of $\mathrm{d}F$ and the definition of $\mathrm{d}F^\star$. Then, as usual, an adjoint is used to remove the term in $\mathrm{d}u.\theta$ in the resulting equation.

$$\begin{aligned}\langle \mathrm{d}F^\star z, \theta\rangle_\Theta = \langle z, \mathrm{d}F.\theta\rangle_{\mathcal{M}} &= \int_\Omega Am\colon \left(\frac{1}{2}\mathrm{div}(\theta)\nabla u - \nabla u \nabla\theta + \nabla(\mathrm{d}u.\theta)\right) + \lambda\int_\Omega \frac{1}{2}\mathrm{div}(\theta)v\\ &= \int_\Omega Am\colon \left(\frac{1}{2}\mathrm{div}(\theta)\nabla u - \nabla u\nabla\theta\right) + \lambda\int_\Omega\frac{1}{2}\mathrm{div}(\theta)v\\ &\quad + \int_\Omega C(\theta)\nabla u_\Omega \colon \nabla p + \int_\Omega (\nabla f\cdot\theta + \mathrm{div}(\theta)f)p \qquad \square\end{aligned}$$

5. Implementation

5.1. Finite elements methods

The PDEs are solved using first order Lagrangian (P1) finite elements. The tests presented in Section 6 were implemented in Python using GetFem++ finite elements library. The tests were run on a laptop equipped with an Intel®-core i7, 2.9Ghz.

5.2. Level set method

Consider a working box D meshed with simplexes. The domain Ω is represented and modified *via* a level-set method with adapted mesh. The level-set is described by a time dependent $P1$ levelset function $\Phi(t)$. At time t, the domain $\Omega(t)$ is the set $\{x, \Phi(x,t) < 0\}$. At each iteration (interpreted as discrete values of the time), the exact boundary of $\partial\Omega(t)$ is computed, at the cell level it consists merely of cutting the simplexes of the boundary by hyperplanes. The cells of D that are cut by the zero-level set of Φ are re-meshed and an exact mesh of Ω and its complementary Ω^c is computed. This mesh is called the "adapted mesh". A new levelset function $\tilde{\Phi}$ is then computed on the adapted mesh and is set to be equal to the distance function with respect to the boundary with first a Fast-Marching method [5,27] and then the Dapogny-Frey algorithm [13]. The PDE is then solved on the adapted mesh and the direction of descent $\theta^\star$ is a $P1$ finite element function of the adapted mesh. Once a direction of descent $\theta^\star$ is computed, the level-set $\tilde{\Phi}$ is evolved during a time interval T by an exact caracteristic scheme on the mesh of Ω according to the equation:

$$\partial_t\tilde{\Phi} - \theta^\star\nabla\tilde{\Phi} = 0,$$

the time step T of this equation is the equivalent of the step of the descent method. The final $\tilde{\Phi}$ is then projected on the original mesh of D, which yields the levelset Φ and the algorithm is ready for the next rinse and repeat phase.

Note that if the Gauss-Newton algorithm naturally yields a direction of descent which is a $P1$ vector-field on the whole domain, the standard gradient method only yields a direction of descent g which is a scalar field on the boundary of Ω, see equation (3.4). In the examples presented here, the PDE is solved using a P1 finite element method, and g is a P0 scalar field. In order to compute $\theta^\star$ from g, our strategy is to first project g on the boundary only in order to obtain a $P1$ scalar field on the boundary, then to extend it harmonically inside the domain and then to multiply it by the extended normal defined as $\frac{\nabla\Phi}{\|\nabla\Phi\|}$ in order to obtain a $P1$ vector field.

5.3. Solving for the descent direction

Let us now detail how to compute the solution η of

$$\mathrm{d}F^\star\mathrm{d}F\,.\eta = -\mathrm{d}F^\star F(0). \tag{5.1}$$

The equation (5.1) is solved using GMRES [25]. For this it is necessary to compute the following quantities:

– $\mathrm{d}F^\star F(0)$: it is the classical shape gradient excepted that the scalar product is not the usual one. Its value is given by Proposition 4.3 (for the L^2 fit case) or by Proposition 4.5 (for the compliance case). Note that it is the gradient w.r.t. the scalar product given in equation (3.3).
– given $\eta \in W^{1,\infty}$ we must compute $\mathrm{d}F^\star \mathrm{d}F\,\eta$. This is done in two steps: $z = \mathrm{d}F\,\eta$ is the direct derivative which is given by Proposition 4.2 (for the L^2 fit case) or Proposition 4.4 (for the compliance case), and $\mathrm{d}F^\star z$ is the adjoint derivative given by Proposition 4.3 or by Proposition 4.5.

5.4. Gauss-Newton algorithm

As a summary the Gauss-Newton shape optimization method is implemented using the algorithms below. The total number of iterations is k, and the number of convergent iteration is l, the difference $k-l$ is the number of iterations spent reducing the step. In practice, the step is never reduced as can be seen in Table 1.

Algorithm : Shape optimization using Gauss-Newton

```
Data: $\Omega_0$ = initial domain
$N$ = maximum number of iterations,
Result: $\Omega^l$ = optimal domain after $l$ iterations
$k = 0, l = 0$
compute the direct states, evaluate the objective function
While $k \leq N$
    $l = l + 1$,$k = k + 1$
    Compute the direction of descent of Gauss-Newton.
    Compute the distance function on the domain $\Omega^{k-1}$.
    The domain $\Omega^k$ is obtained by advecting $\Omega^{k-1}$.
    Mesh the domain $\Omega^k$.
    Compute the direct states, evaluate the objective function.
    If the objective function increases:
        Send a warning message, $k = k + 1$.
        Go back to previous convergent iteration.
        Continue with a smaller step.
```

6. Numerical results

In all the tests presented here the PDE into consideration is the standard linear elasticity equation with Young modulus $E = 1$ and Poisson ratio $\nu = 0.3$. The holes are numerically considered as ersatz material (Young modulus of 10^{-3}), this allows to smoothly handle island of material disconnected from the Dirichlet boundary condition and to stabilize the problem if the remeshing operation yields a triangulation close to the boundary with a bad conditionning.

In order to document more completely the results presented in this section the authors propose the movies of shape evolution on their webpages:
https://www.math.univ-toulouse.fr/~{}fehren/videos.html, http://degournay.fr/frederic/gn.

6.1. First test case: L^2 fit to retrieve a hole

The first test into consideration consists in retrieving a perforation in a $[-1,1] \times [-1,1]$ square knowing the measurement of the displacement in an annulus along the boundary. The domain is subjected to Dirichlet boundary conditions on $x = \pm 1$ and Neumann boundary conditions on $y = \pm 1$. The objective function is the L^2 fit in the multiple loads case $\mathcal{J}_{\text{multi}}$ defined in equation (4.2). The goal states $u_G^k, k = 1..m$ are synthetized

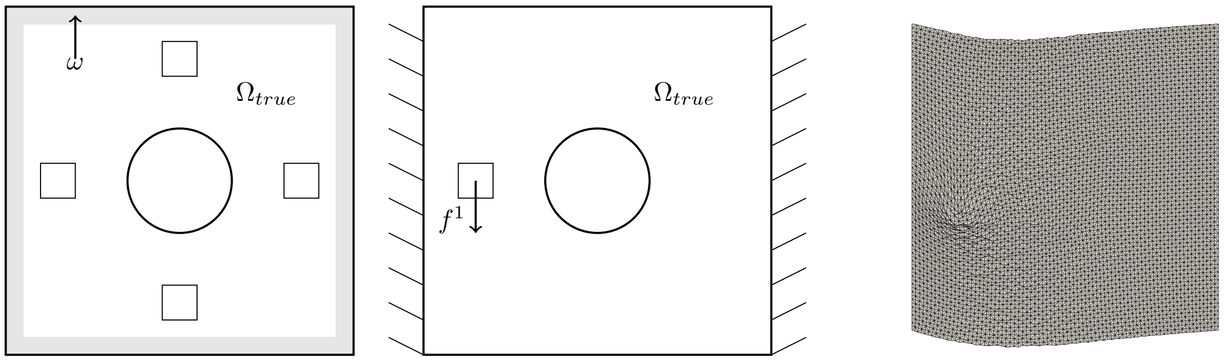


FIGURE 1. True domain Ω_{true}, observation domain ω, and location of the four different source terms (*left*); condition for the first source term f^1 (*middle*); solution u_G^1 for the first source term (*right*).

with known source terms f^k on a target set Ω_{true} perforated by the disk of center $(0,0)$ and of radius 0.3. The set ω on which the L^2 fit is performed is the neigbourhood of the boundary of Ω defined by

$$\omega = \Omega \cap \{(x,y) \text{ s.t.} |x| > 0.9 \text{ or } |y| > 0.9\}.$$

The mesh on which optimization is performed is a triangular decomposition of a 80×80 square mesh, whereas the data is synthetized on a 200×200 mesh and interpolated on the working mesh. The loads are $P1$ finite element functions computed on the synthetic mesh interpolated on the working mesh and fed to the optimization algorithm.

Figure 1 (*left*) presents the target domain Ω_{true}, and the subdomain ω where the measurements are used in the L^2 fit. There are $m = 4$ source terms which are vertical loads of magnitude 1 on their support. The loads are supported on the four squares of length 0.2 whose centers are $(-0.7, 0)$, $(0.7, 0)$, $(0, -0.7)$, $(0, +0.7)$, see Figure 1 (*left*) for the location of the loads and Figure 1 (*middle*) for the illustration of one load. Since the data has symmetry w.r.t. the two lines $x = 0$ and $y = 0$, so should have the corresponding solutions. Although symmetry is lost in the optimization process, it is a validation of the algorithm to find a symmetric final shape. In order to compare with existing methods we have also implemented the gradient descent (described in Sect. 5.2) with line search. The displacement for the first load is shown in Figure 1 (*right*).

In order to document the behavior of Algorithm 1 and the reference algorithm (gradient) under various conditions we have studied the following cases:

- *Good starting point.* The initial hole is a disk of center $(0,0)$ and of radius $\sqrt{3/10}$. Figure 2 presents different steps of Gauss-Newton and gradient methods.
- *Bad starting point.* The initialisation is farther away from the exact solution. The initialisation and the results are presented in Figure 3.
- *Noisy case.* The data is perturbed by multiplicative noise (10% additive noise with a Gaussian distribution). The results are presented in Figure 4.

The convergence history (evolution of the objective function) for the different cases is shown in Figure 5 and the computational cost is presented in Tables 1 and 2. Note that the convergence history shows that Gauss-Newton method is more efficient in terms of convergence history versus iteration. The computation of the descent direction for Gauss-Newton is computationally more expensive than the computation of the descent direction for the gradient method. This observation however has to be ponderated by two remarks:

- First, the choice of the initial step for the gradient method is costly. In the tests presented here we performed a manual tuning of the initial step. The Gauss-Newton method on the contrary has an optimal step of 1 (this

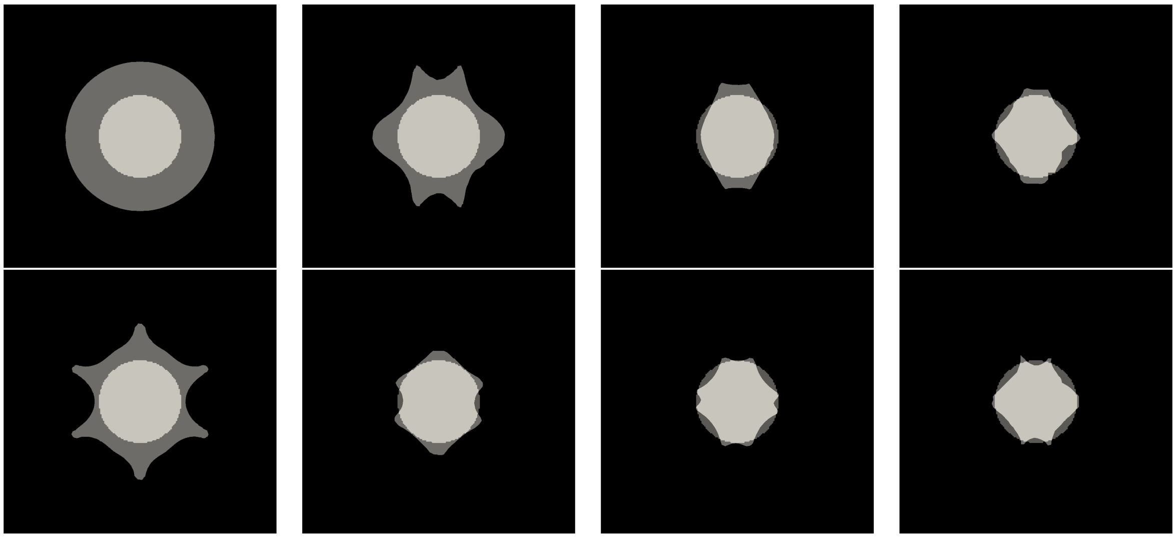


FIGURE 2. Good starting point case, the shape is in black, the hole in gray. The target hole is highlighted. *Top*: iterations 0,1,3 and 27 of the gradient algorithm. *Bottom*: iterations 1, 3, 5, 16 of the Gauss-Newton algorithm.

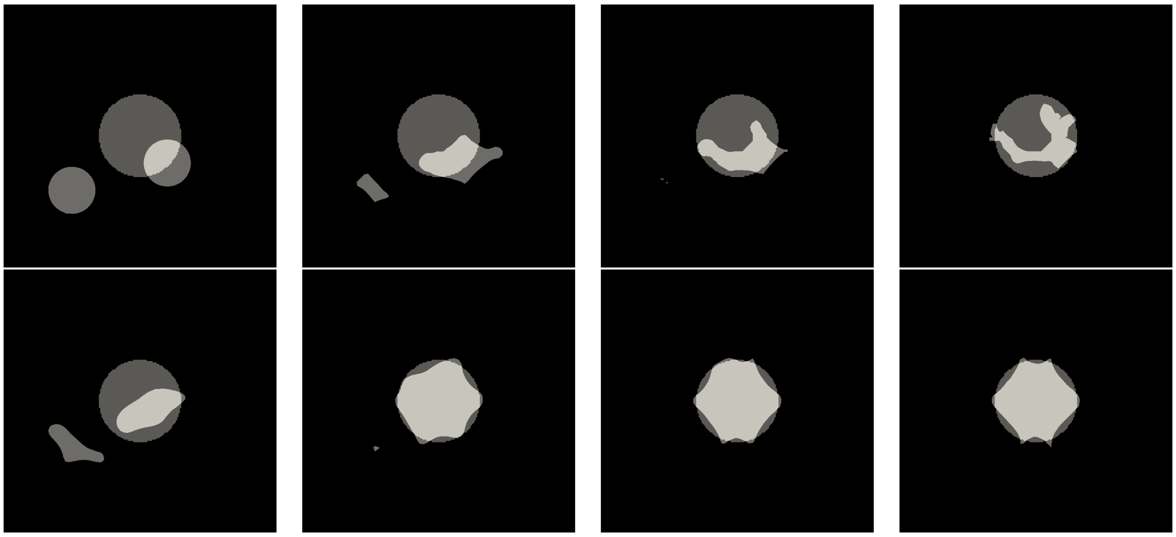


FIGURE 3. Bad starting point case, the shape is in black, the hole in gray. The target hole is highlighted. *Top*: iterations 0,4,8 and 25 of the gradient algorithm. *Bottom*: iterations 1, 4, 6, 9 of the Gauss-Newton algorithm.

is asymptotically exact close to the solution and is still acceptable all over the iterations in our experiments) and does not require any tuning. At each iteration the line search for the gradient method requires a remeshing operation and an advection of the level set, which are computationally expensive operations.

- The linear elasticity systems to be inverted for the Gauss-Newton method have the same matrix througout one iteration. The computation and storage of a LU decomposition of the stiffness matrix greatly decreases the total time spent in one Gauss-Newton iteration (compared to the number of systems solves multiplied

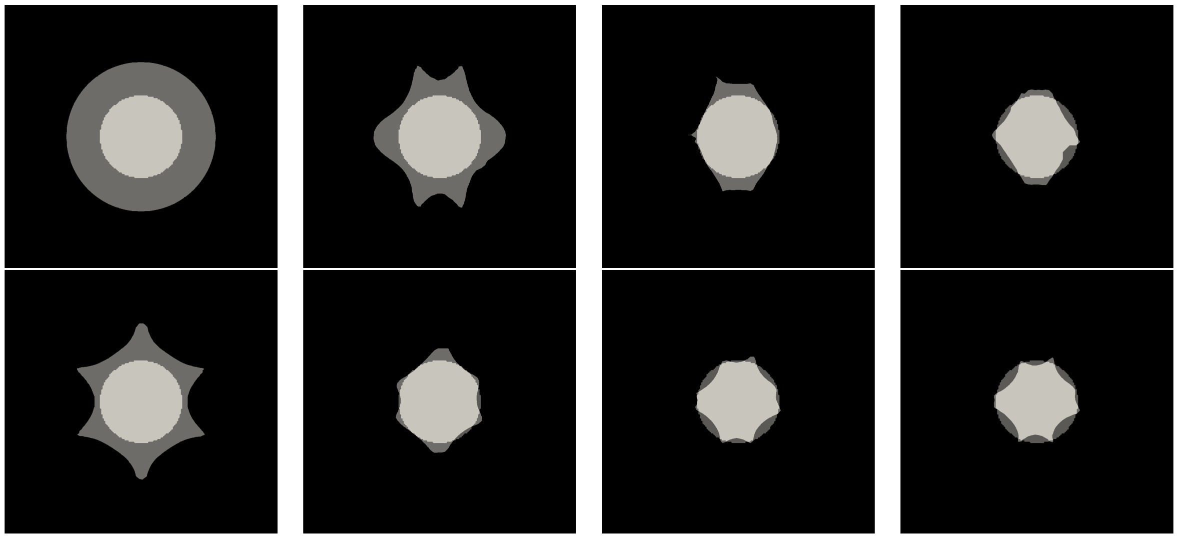


FIGURE 4. Noisy case, the shape is in black, the hole in gray. The target hole is highlighted. *Top*: iterations 0,1,3 and 17 of the gradient algorithm. *Bottom*: iterations 1, 3, 5 and 6 of the Gauss-Newton algorithm.

by the cost of solution of one single system). Note also that the computation of the descent direction is done by an iterative algorithm. We choosed to stop the GMRES algorithm after 10 iterations. In practice the remainder in the GMRES algorithm after 10 iterations is of order 2%.

These remarks are documented in Table 2, where it appears that cost of the Gauss-Newton method is dominated by the assembly of stiffness matrices or right-hand sides (required in the computation of direct and adjoint derivatives), while the cost of the gradient method is dominated by the advection and distance function calculation. As an overall balance the conclusion is that in this test case the computational cost of Gauss-Newton method is of the same order as the computational cost for the gradient descent, and provide a better optimization of the objective.

Finally, we give statistics on the noisy case, the test with noise has been run 80 times, and the Gauss-Newton method converges in 7.46 iterations in average with a standard deviation of 1.35 iterations. The Gradient algorithm converges in 21.8 iterations in average with a standard deviation of 4.35 iterations. The running time is proportional to the Table 1. The average final value of the objective function is 5.916e-7 (resp 5.924e-7) with standard deviation 1.317e-08 (resp. 1.301e-08) for the Gauss-Newton (resp. Gradient) method. The final

TABLE 1. Computational load for the various cases of the experiment: number of convergent iterations; total number of iterations including the line search in the case of the gradient method (these are the same for Gauss-Newton method); the total time; final value of the objective function $\mathcal{J}_{\text{multi}}$ defined in equation (4.2).

	Convergent iterations	Total Iterations	Time	Final objective
Gradient (Good starting point)	27	44	114s	7.46e-9
Gauss-Newton (Good starting point)	16	16	136s	4.046e-9
Gradient (Bad starting point)	25	41	111s	1.89e-8
Gauss-Newton(Bad starting point)	9	9	79s	4.02e-9
Gradient (Noisy case)	19	46	100s	5.96e-7
Gauss-Newton (Noisy case)	6	6	58s	5.92e-7

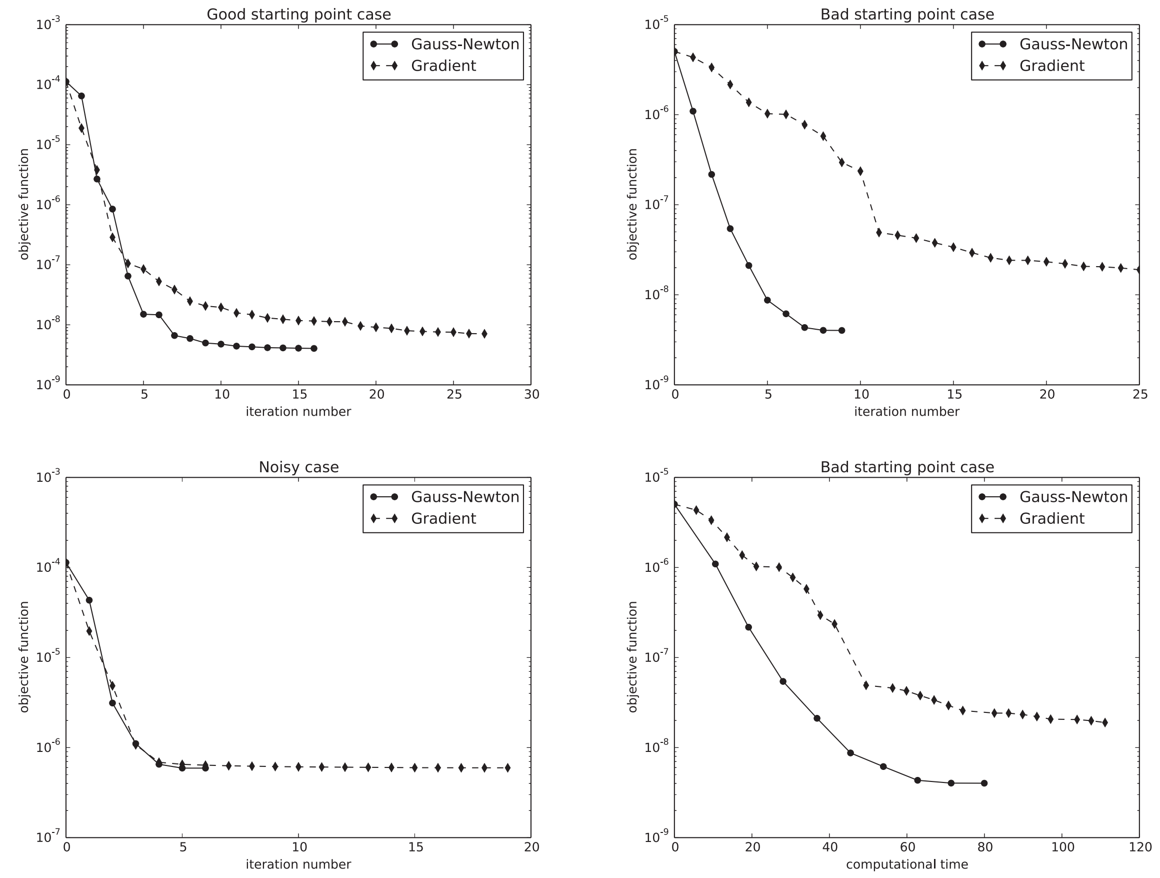


Figure 5. The convergence history in the different cases for the hole problem. The cases are (*from left to right and top to bottom*): the good starting point (initial shape is a circle with different radius), the bad starting point (initial shape is composed of two circles) and the noisy data. The last test is the bad starting point with the elapsed computational time as axis instead of iteration number.

Table 2. Detailed computational load for the Gauss-Newton and Gradient algorithm, both with the good, bad initialization and the noisy case. For each line the number of calls of the functions and the total time spent in the function is displayed. Some output and control functions are not displayed, hence the total time does not exactly sum up to the results of Table 1.

	Good starting point		Bad starting point		Noisy case	
	Gradient	Gauss-Newton	Gradient	Gauss-Newton	Gradient	Gauss-Newton
Matrix or RHS assembly	378 (25.1s)	1782 (87.4s)	352 (24.7s)	1005 (51s)	358(27.0s)	672(38.3s)
LU factorization	45 (3.6s)	33 (1.9s)	42 (3.4s)	19 (1.2s)	47(3.8s)	13(0.8s)
Linear system resolution	288 (1.4s)	1732(8.4s)	268(1.3s)	976 (4.8s)	264(1.3s)	652(3.4s)
Remeshing operations	45 (8.5s)	17 (3.3s)	42 (8.7s)	10 (1.9s)	47(9.1s)	7(1.4s)
Advection	45 (36.1s)	17 (16.7s)	42 (34.2s)	10 (9.6s)	47(26.1s)	7(6.6s)
Distance Function	27 (17.4s)	16 (10.6s)	25 (16.6s)	9(5.9s)	19(12.5s)	6(4.2s)

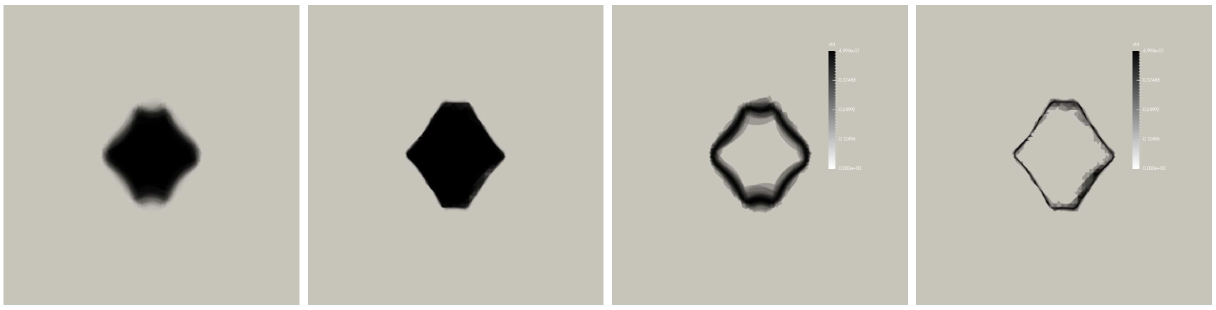


FIGURE 6. *From left to right*: average shape for the Gauss-Newton and the Gradient Method and standard deviation for the Gauss-Newton and the Gradient Method for the noisy case with 80 runs. The scale of the standard deviation is between 0. and 0.48.

TABLE 3. Computational load for the second test case: number of convergent iterations; total number of iterations including the line search in the case of the gradient method; total time; final value of the objective function $\mathcal{J}_{\text{multi}}$ defined in equation (4.2).

	Convergent iterations	Total iterations	Time	Optimal value
Gradient	22	38	98s	4.89e-8
Gauss-Newton	9	9	80s	3.58e-9

shapes in average and the standard deviation are shown in Figure 6 for the two methods. We can observe that the Gauss-Newton method achieves a smaller value of the cost function in average but that the optimal shape standard deviation is more important. As a result, the average shape appears to be more blurred in the Gauss-Newton method. Since there is no regularization term in the fitting problem, it is expected that the optimal shapes account for the noise and hence their average be blurred. The behavior of the Gradient method can be interpreted as if the algorithm is attracted to a local minimum. This might explain the lack of blurring of the average shape obtained by the Gradient method.

6.2. Second test case: L^2 fit to retrieve a hole with a complex geometry

This test is the same as the previous one except that the target shape to be retrieved is the union of three disks respectively of center $(0,0)$, $(-0.3,0.2)$, $(0.2,0.2)$ and of radii 0.3, $\sqrt{3/100}$, and $\sqrt{4/100}$. The shape of this hole is similar to a mouse (see Fig. 7 top left). The different iterations and the target shape are described in Figure 7, whereas convegence history is shown in Figure 8.

The conclusion of the second test case is that a more precise solution is obtained with Gauss-Newton method, for a computational load of the same order as the gradient method.

6.3. Third test case: compliance of the cantilever

This section adresses the standard test case of optimizing the compliance of the cantilever. The working domain is a 2×1 rectangle meshed with 160×80 cells. The applied force is a vertical load applied on the middle of the right boundary, and the left boundary is clamped. The initial shape is shown in Figure 11 (iteration 0 of the algorithm). We compared the following strategies to optimize the compliance $\mathcal{L}$ given by equation (4.4):

- *Gradient*: the standard gradient method described in Section 5.2,
- *Gauss-Newton*: the method described in Algorithm 2,
- *Gauss-Newton late start*: after 10 iterations of the gradient algorithm the current domain is used as initial guess for a Gauss-Newton method,

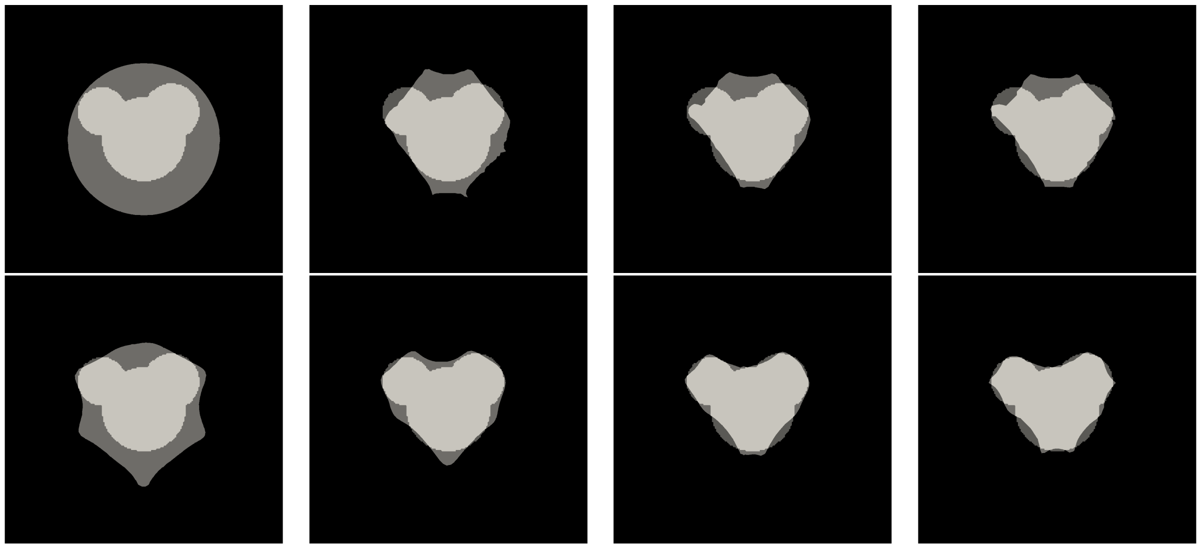


FIGURE 7. The second test case (mouse), the shape is in black, the hole in gray. The target hole is highlighted. *Top*: iterations 0,3,8 and 22 of the gradient algorithm. *Bottom*: iterations 1, 3, 5, 9 of the Gauss-Newton algorithm.

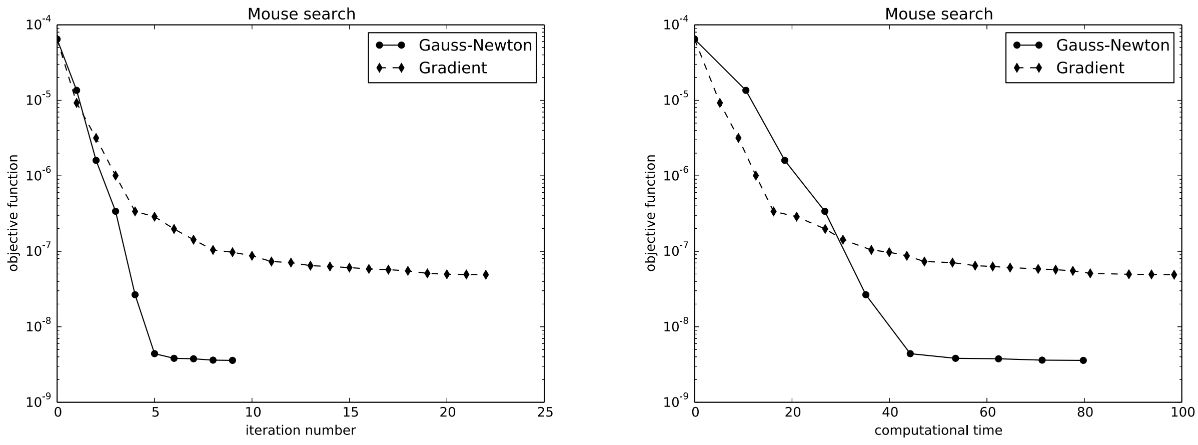


FIGURE 8. The convergence history for the second test case, objective function *vs.* iteration (*left*) and *vs.* computational time in seconds (*right*).

- H^1*-gradient*: as we noted in Section 5.2, the classical gradient method for shape optimization uses as a scalar product in the space of vector fields the L^2 norm of the normal trace on the boundary. The H^1*-gradient* algorithm uses the scalar product defined by (3.3) to compute the descent direction. It is thus the right-hand side $-\mathrm{d}F^\star F(0)$ of equation (5.1).

A selected choice of the shapes obtained during the iterations are displayed in Figure 10 for the Gauss-Newton algorithm, in Figure 11 for the Gradient algorithm, in Figure 12 for the Gauss-Newton algorithm with late start, and in Figure 13 for the H^1-gradient algorithm. The convergence history of these algorithm is shown in Figure 14. Finally Table 4 provides details about the computational load and the objective function for the different algorithms.

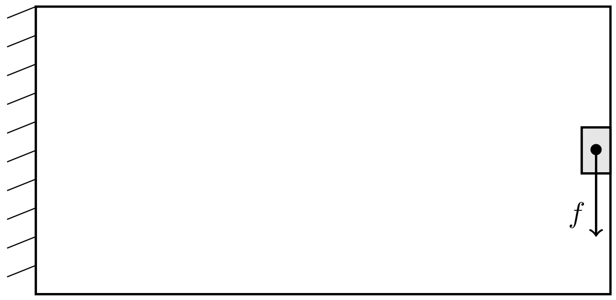


FIGURE 9. The cantilever test case.

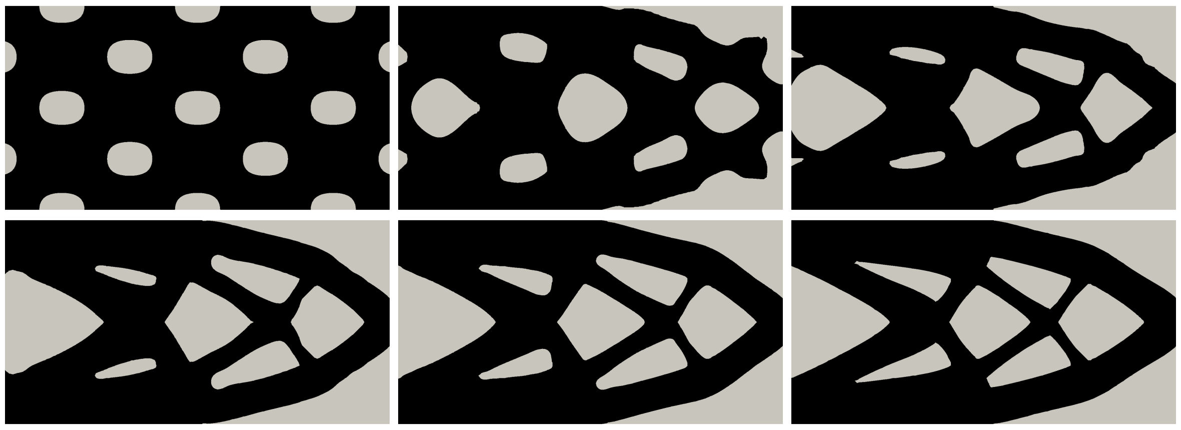


FIGURE 10. The Gauss-Newton algorithm for the compliance of the cantilever, iterations 0, 1, 2, 3, 4 and 15 of the algorithm are shown.

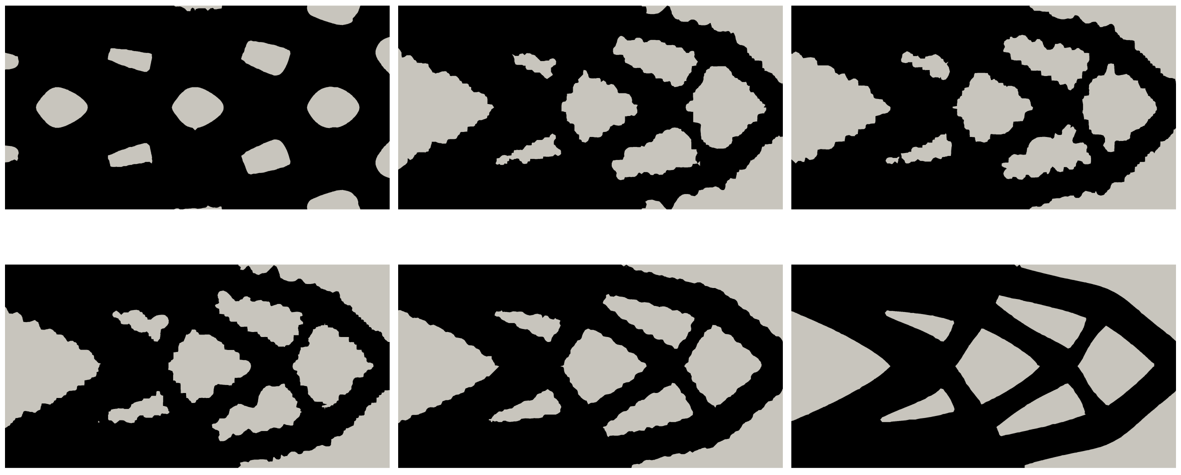


FIGURE 11. The Gradient algorithm for the compliance of the cantilever, iterations 5, 10, 11, 12, 19 and 25 of the algorithm are shown.

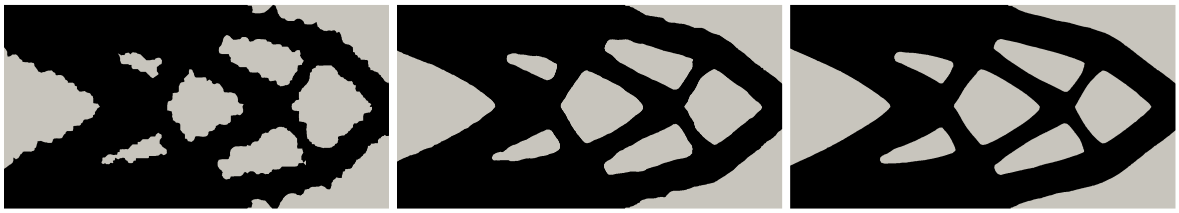


FIGURE 12. The Gauss-Newton algorithm with late start for the compliance of the cantilever, iterations 10, 11, 12 of the algorithm are shown.

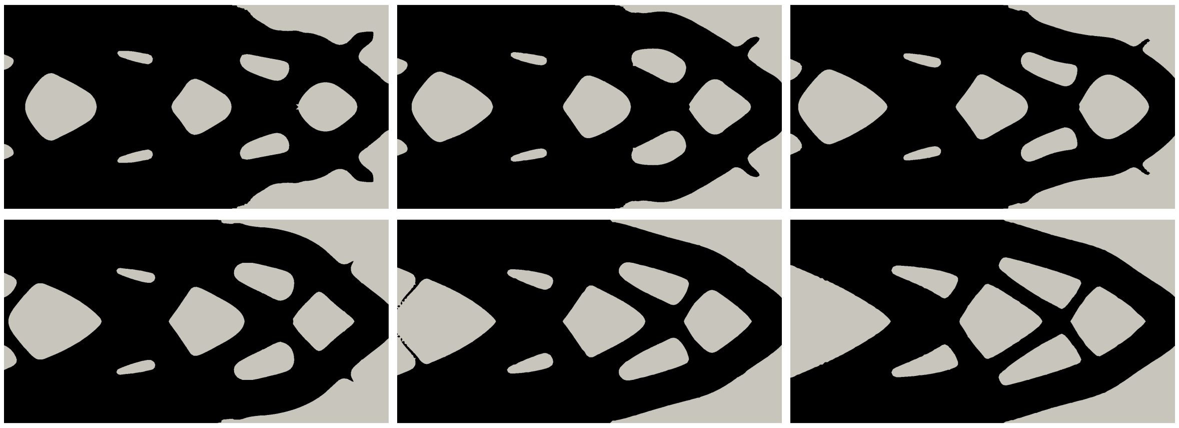


FIGURE 13. The H^1-gradient algorithm for the compliance of the cantilever, iterations 5, 10, 11, 12, 19 and 25 of the algorithm are shown.

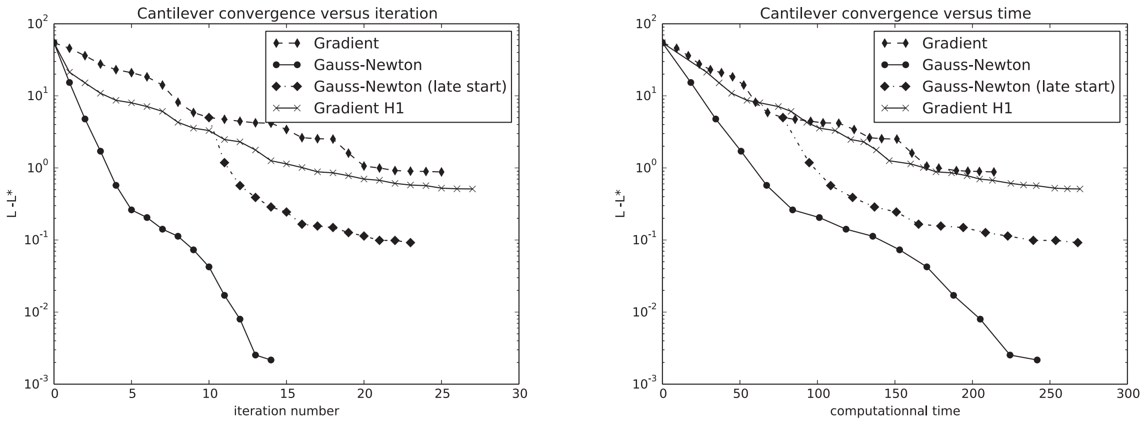


FIGURE 14. The convergence history for the cantilever problem. *Left*: objective function. *Right*: difference (in log scale) of the objective function with $L^\star$ the minimal value of the objective function (obtained by the Gauss-Newton algorithm).

Table 4. Computational load for cantilever case: number of convergent iterations; total number of iterations including the line search; total time; final value of the objective function $\mathcal{L}$ defined in equation (4.4).

	Convergent iterations	Total iterations	Time	Optimal value
Gradient	25	36	214s	184.432
Gauss-Newton	15	15	262s	183.546
Gradient-H1	27	48	269s	184.057

The convergence history and the carefull study of the intermediate shapes during iterations shows the efficiency of the Gauss-Newton algorithm in finding shapes compared to the other algorithms. Three importants points have to be discussed in view of this numerical test: the step choice, the (local) oscillations of the gradient algorithm, and the (global) oscillations of the H^1-gradient algorithm.

(1) As already discussed in Section 3.1 the step has to be carefully chosen when working with the gradient algorithm, whereas in Gauss-Newton algorithm the step is 1. For the Gradient algorithm we chose manually the best initial step that ensures the quickest convergency of the method. In the case of the cantilever with this initialization, two different phases of the algorithm can be observed: first the algorithm removes un-necessary material in the middle of the left boundary and on the two top-right and bottom-right corners (See the iteration 1 of the Gauss-Newton algorithm, Figure 10 and the iterations 5 to 10 of the Gradient algorithm, Fig. 11), then the algorithm spends some time enhancing the shape of the holes before arriving to its final shape. The important thing is that the optimal step has to be different for those two phases, hence the Gradient algorithm has to go through a slow phase of step adjustement, in our experiment the step has to go from 2.5×10^{-3} to 1×10^{-4}. Gauss-Newton algorithm does not have this issue, its step is constant throughout iterations. If we increase the step of the Gradient algorithm, this will lead to a first phase that is quicker but a longer phase of step adjustement, hence an overall slower algorithm.

(2) The oscillations in the shape of Figure 11 are not common in shape optimization by the gradient method, and one could wonder where they exactly come from.

The reason for these oscillations in our implementation is that in order to perform accurate computation the mesh is cut at each iteration by the level-set. A classical merging method is then applied when points are too close. At the level of the domain, this merging method will contribute to removing small triangles (smaller than 1/10 of the grid size), one of these missing triangles can be seen on iteration 5 of Figure 11 (bottom part of the central hole). The response of the algorithm is to compensate these missing triangles by expanding the level-set at this point, thus creating oscillations. Note that the oscillations are created during the first phase (the algorithm is busy removing parts in the middle of the left boundary and on the top-left and bottom-left corners), so that the algorithm does not really care about these oscillations beeing created. This first phase is completed at iteration 10 of the gradient algorithm, the algorithm then tackles the issue of the oscillations. Hence, the behavior observed by the gradient algorithm in iterations 10 to 12 in Figure 11 is not that of a bad numerical method, but that of a gradient algorithm trying to smooth the boundary of the shape, but without the correct step. Indeed one can see that the oscillations from one iteration to the next one are inverted. This is where Gauss-Newton algorithm shows its efficiency: if we start a Gauss-Newton algorithm at iteration 10 of the Gradient algorithm, in one iteration (see Fig. 12) the oscillations of the shape are killed and the algorithm goes back to optimizing the shape of the holes. In the literature, this effect is avoided since either the velocity is regularized or the level-set is diffused, these strategies prevent the oscillations to appear. This is exactly the strategy followed by the H^1-gradient algorithm that regularises the descent direction, since the computation of $\mathrm{d}F^\star$ contains a relevement by the scalar product of Θ which is regularizing since it contains first order derivative in its definition. We note also that the Gauss-Newton algorithm (see Fig. 10) does not even expand these small oscillations.

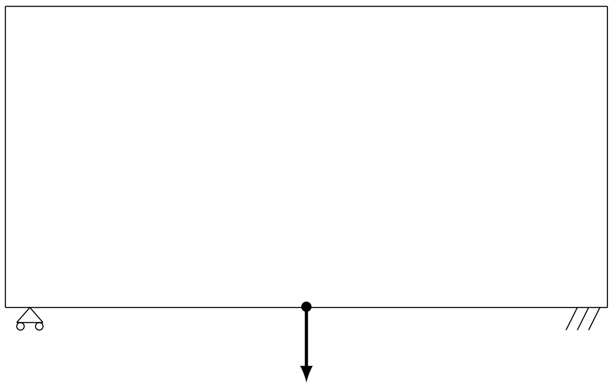
Table 5. Computational load for the sliding arch case: number of convergent iterations; total number of iterations including the line search; total time; final value of the objective function $\mathcal{L}$ defined in equation (4.4).

	Convergent iterations	Total iterations	Time	Optimal value
Gradient	44	66	215s	1.654
Gauss-Newton	10	12	102s	1.622
H^1-Gradient	47	65	238s	1.632

(3) The H^1 gradient method shows oscillations of the shape that are global oscillations and not mere oscillations at the scale of the mesh (see iterations 10,11,12 of Fig. 13). Such oscillations are typical of the gradient method when dealing with important variation of the gradient in one direction, in other words steep valleys where the method does not provide a next iterate that is located exactly at the bottom of the valley.

6.4. Fourth test case: Compliance of the sliding arch

The last test case is a sliding arch which is a shape included in a 2×1 rectangle meshed with 120×60 cells subjected to a vertical load at the middle of the bottom boundary. The vertical displacement is set to zero on the bottom left and bottom right corners. Constraints on the horizontal displacement are only given on one corner (hence the “sliding” in the name of the test case).



The initialization and the final shape obtained by the three algorithms in consideration are shown in Figure 15. Finally, Figure 16 shows the convergence history of the objective function and Table 5 some information regarding the optimization algorithms: iteration count, elapsed time and final objective value. This test-case is known to be a little more difficult than the cantilever case, one can observe here that the H^1-gradient algorithm is slower than the standard Gradient algorithm. The reason is that the regularization procedure (included implicitly in the computation of $\mathrm{d}F^\star$) diffuses the value of the gradient and that the scalar product of Θ is numerically a worst choice than the L^2 scalar product of the normal component on the boundary. Note that in this case again, the Gauss-Newton algorithm behaves better than the two other algorithms, not only in terms of iteration number, of final value of the objective function but also in terms of overall computational time.

7. Conclusions

We presented in this work a framework that allows to implement a Gauss-Newton method in shape optimisation problems, which is adapted to the case when the objective function is quadratic. It is an approximate second order method where no Hessian computation is required. This Gauss-Newton method requires the definition

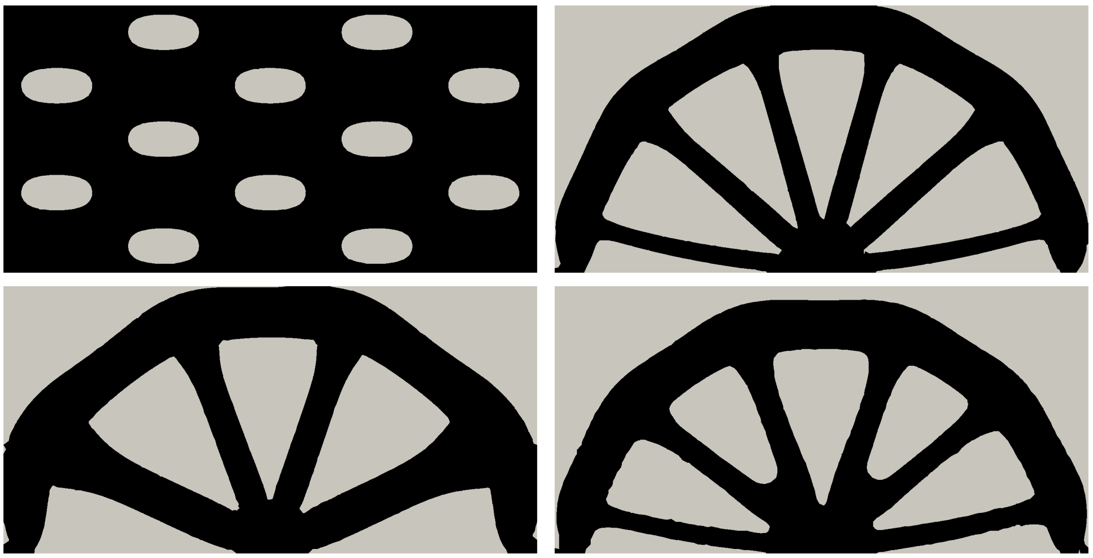


FIGURE 15. Initialization and final shape for the sliding arch problem and, respectively, the Gauss-Newton algorithm, the gradient algorithm and the H^1-gradient algorithm.

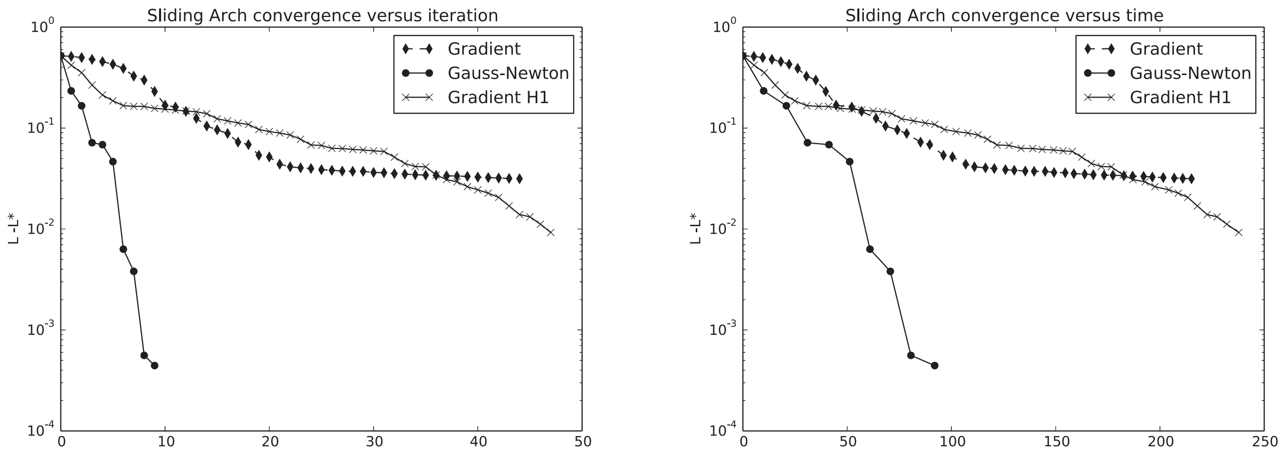


FIGURE 16. The convergence history for the sliding arch. We plot the log scale of the difference of the objective function L with the minimal objective function $L^\star$ (obtained for Gauss-Newton). On the left, the absciss is the iteration number and on the right, it is the computational time.

of a scalar product in the space of vector fields, and although we did not obtain a Hilbert space the formulas for the descent direction proved to be effective. The fact that the estimated vector field is global (and not only defined by its normal trace on the boundary) provides a natural regularization although it seems at first sight that unnecessary information is added in view of Hadamard's structure theorem. The toy test cases that are presented show that shape optimization using Gauss-Newton method provides more efficient shapes than the gradient method (since the objective function is smaller), and in the same time the computational load is of the same order of magnitude. Indeed Gauss-Newton method is less costly in total iteration number but more costly in computational time per iteration In the cases in consideration, the cost of the Gauss-Newton method in computational time per iteration is reduced by using complete LU factorization. It is questionnable that the trade-off between the gain in total iteration number and computational time per iteration would still favor the Gauss-Newton method for more difficult problems. In one hand, if the size or dimension of the problem rises to a point where LU factorization is not allowed in memory terms, one has to shift back to iterative methods that drastically augment the computational time of the Gauss-Newton method. On the other hand, more difficult quadratic problems lead to a huge augmentation of computational time of the solution of the forward problem, hence a reduction of the total iteration time has an important impact on total computational time. It would be interesting to challenge the Gauss-Newton algorithm in such problems. Future work will adress more realistic 3D examples, and constrained shape optimization. For the latter we will investigate the efficiency of projected descent or quadratic programming.

Acknowledgements. As a final note, the authors wants to thank Allaire, Dapogny, Jouve and Vié for their always fruitfull and enjoyable discussions.

References

[1] L. Afraites, M. Dambrine, K. Eppler and D. Kateb, Detecting perfectly insulated obstacles by shape optimization techniques of order two. *Discrete Contin. Dyn. Systems Series B* **8** (2007) 389.

[2] L. Afraites, M. Dambrine and D. Kateb, On second order shape optimization methods for electrical impedance tomography. *SIAM J. Control Optimiz.* **47** (2008) 1556–1590.

[3] G. Allaire, Shape optimization by the homogenization method. Vol. 146. Springer Science & Business Media (2012).

[4] G. Allaire, E. Cancès and J.-L. Vié, Second-order shape derivatives along normal trajectories, governed by Hamilton-Jacobi equations. *Calcolo* **49** (2012) 193–219.

[5] G. Allaire, F. Jouve and A.-M. Toader, Structural optimization using sensitivity analysis and a level-set method. *J. Comput. Phys.* **194** (2004) 363–393.

[6] G. Allaire and M. Schoenauer, Conception optimale de structures. Vol. 58. Springer (2007).

[7] M.Ph. Bendsøe and N. Kikuchi, Generating optimal topologies in structural design using a homogenization method. *Comput. Methods Appl. Mech. Eng.* **71** (1988) 197–224.

[8] L. Borcea, Electrical impedance tomography. *Inverse Problems* **18** (2002) R99.

[9] V. Braibant and C. Fleury, Shape optimal design using B-splines. *Comput. Methods Appl. Mech. Eng.* **44** (1984) 247–267.

[10] A.P. Calderón, On an inverse boundary value problem. *Comput. Appl. Math.* **25** (2006) 133–138.

[11] M. Dambrine, On variations of the shape Hessian and sufficient conditions for the stability of critical shapes. *Real Acad. Ciencias Exactas, Fis. Nat. Serie A: Mat.* **96** (2002) 95–122.

[12] M. Dambrine and M. Pierre, About stability of equilibrium shapes. *ESAIM: M2AN* **34** (2000) 811–834.

[13] Ch. Dapogny and P. Frey, Computation of the signed distance function to a discrete contour on adapted triangulation. *Calcolo* **49** (2012) 193–219.

[14] F. de Gournay, Velocity extension for the level-set method and multiple eigenvalues in shape optimization. *SIAM J. Control Optimiz.* **45** (2006) 343–367.

[15] K. Eppler and Helmut Harbrecht, A regularized Newton method in electrical impedance tomography using shape Hessian information. *Control Cibern.* **34** (2005) 203.

[16] J. Fehrenbach, M. Masmoudi, R. Souchon and Ph. Trompette, Detection of small inclusions by elastography. *Inverse Problems* **22** (2006) 1055.

[17] S. Garreau, Ph. Guillaume and M. Masmoudi, The topological asymptotic for PDE systems: the elasticity case. *SIAM J. Control Optimiz.* **39** (2001) 1756–1778.

[18] J. Hadamard, Sur le problème d'analyse relatif à l'équilibre des plaques élastiques encastrées. Mémoire couronné en 1907 par l'académie: Prix vaillant, mémoires présentés par divers savants à l'académie des sciences **33** (1908). In vol. 2 of Oeuvres de Jacques Hadamard (1907) 515–629.

[19] A. Henrot and M. Pierre, Variation et optimisation de forme. Springer (2005).

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

[21] F. Murat and J. Simon, Sur le contrôle optimal par un domaine géométrique. Université Pierre et Marie Curie (Paris VI), Publication du Laboratoire d'Analyse Numérique (1976).

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

[23] S. Osher and J.A. Sethian, Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.* **79** (1988) 12–49.

[24] J.R. Roche, Adaptative Newton-like method for shape optimization. *Control and Cybernetics* **34** (2005) 363–377.

[25] Y. Saad and M.H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.* **7** (1986) 856–869.

[26] M. Schweiger, S.R. Arridge and I. Nissilä. Gauss–Newton method for image reconstruction in diffuse optical tomography. *Physics in medicine and biology* **50** (2005) 2365.

[27] J.A. Sethian, Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science, volume 3. Cambridge University Press (1999).

[28] J. Simon, Second variations for domain optimization problems. Control and estimation of distributed parameter systems, edited by F. Kappel, K. Kunish et W. Schappacher. *Int. Ser. Numer. Math.* **91** (1989) 361–378.

[29] J. Sokolowski and A. Zochowski, Topological derivatives for elliptic problems. *Inverse Problems* **15** (1999) 123.

[30] A. Tarantola, Inverse problem theory and methods for model parameter estimation. SIAM (2005).

[31] Ph. Wolfe, Convergence conditions for ascent methods. *SIAM Rev.* **11** (1969) 226–235.