SOLVING A FREE BOUNDARY PROBLEM WITH NON-CONSTANT COEFFICIENTS

RAHEL BRÜGGER, ROBERTO CROCE, AND HELMUT HARBRECHT

Abstract. The present article is concerned with the numerical solution of a free boundary problem for an elliptic state equation with non-constant coefficients. We maximize the Dirichlet energy functional over all domains of fixed volume. The domain under consideration is represented by a level set function which is driven by the objective's shape gradient. The state is computed by the finite element method where the underlying triangulation is constructed by means of a marching cubes algorithm. We show that the combination of these tools lead to an efficient solver for general shape optimization problems.

Introduction

The development of efficient algorithms for the numerical solution of elliptic shape optimization problems is still a challenging task. In many problems, the state equation involves a differential operator with constant coefficients. In this situation, boundary integral equations are quite attractive for computing the state since only the discretization of the boundary of the domain is required. Especially, large deformations of the domains are realizable without remeshing. Hence, very efficient algorithms can be realized, especially if fast boundary element methods are applied, see [3, 11, 16, 27, 34, 38] for instance. Moreover, exterior boundary value problems are easily treatable, as for the computation of free surfaces of liquid bubbles, or drops levitating in an electromagnetic field, cf. [10, 27, 32].

The above mentioned techniques are not applicable to state equations which involve differential operators with *non-constant* coefficients. Then, volume discretization methods like finite difference methods or finite element methods have to be used. We propose here to employ the finite element method, where the underlying triangulation is constructed by a marching cubes algorithm. This ensures shape regular triangles except for the ones which intersect the domain's boundary. Thus, we apply local updating steps to improve the mesh in the neighbourhood of the boundary.

The sought optimal domain is represented by means of a level set function. More precisely, we use the so-called signed distance function. The advantage of level set functions is that they allow for topological changes of the domain. The level set function is discretized by a second order finite difference scheme. The main drawback

of level-set techniques consists in the area/volume conservation. Several techniques have been introduced to improve the area/volume loss [30, 31, 8]. However, since our algorithm approximates a steady state solution in which the dynamics is rapidly damped, we show that the area/volume loss does not play a major role in our case.

Numerical results are presented for different topological situations such as non-connected domains and domains of genus one. These results demonstrate that we succeeded in developing efficient and robust techniques for the solution of the current class of shape optimization problems.

The rest of this article is organized as follows. Section 1 is dedicated to the shape optimization problem under consideration. We introduce our model problem of maximizing the Dirichlet energy functional under a volume constraint. Especially, we derive the Hadamard representation of the associated shape gradient and proof the necessary optimality condition. Then, we introduce the level set method which we will use to represent the sought domain. In Section 3, we present the numerical scheme to compute the state function. We introduce the mesh generation algorithms and briefly recall the computation of the shape gradient by the finite element method. In Section 4, we present numerical results to demonstrate the capability of our approach. In the last section, we state concluding remarks.

1. Shape Optimization

1.1. **The model problem.** We will denote the *hold all* by $\mathbb{D} \subset \mathbb{R}^2$ and assign Υ to be the set of all bounded shapes $\Omega \subset \mathbb{D}$ which have a C^2-smooth boundary $\Gamma := \partial\Omega$. We consider the Dirichlet energy functional

$$J(\Omega) = \int_\Omega \langle \mathbf{A}\nabla u, \nabla u\rangle \,\mathrm{d}\mathbf{x} = \int_\Omega f u \,\mathrm{d}\mathbf{x}, \tag{1.1}$$

where the *state function* u solves the boundary value problem

$$\begin{aligned} -\operatorname{div}(\mathbf{A}\nabla u) &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \Gamma. \end{aligned} \tag{1.2}$$

Here, we assume that the inhomogeneity $f : \mathbb{D} \to \mathbb{R}$ and the symmetric and uniformly positive definite matrix $\mathbf{A} : \mathbb{D} \to \mathbb{R}^{2\times 2}$ are sufficiently regular.

We aim at maximizing the Dirichlet energy (1.1) over the class Υ of admissible shapes. In order to avoid degeneration, we shall impose an equality constraint on the shape's volume,

$$V(\Omega) := \int_\Omega 1 \,\mathrm{d}\mathbf{x} = V_0, \tag{1.3}$$

compare [4] for instance. Consequently, we arrive at the following shape optimization problem:

$$-J(\Omega) \to \min_{\Omega \in \Upsilon} \quad \text{subject to} \quad V(\Omega) = V_0. \tag{1.4}$$

In this article, we do not treat the question of existence and regularity of solutions. Instead, we will tacitly assume the existence of optimal shapes, being *sufficiently regular* to be members of Υ. For the existence of solutions to free boundary problems, we refer the reader to e.g. [2, 5, 13]. Results concerning the geometric form of the solutions can be found in [1] and the references therein.

1.2. **Shape calculus.** We briefly recall well known facts about shape calculus, useful for the discussion of the necessary condition and the numerical algorithms. For a general overview on shape calculus, mainly based on the perturbation of identity (Murat and Simon) or the speed method (Sokolowski and Zolésio), we refer the reader to [9, 26, 33, 35, 37], and the references therein.

For a smooth perturbation field $\mathbf{U} : \Gamma \to \mathbb{R}^2$, we define the perturbed shape Ω_ε via its boundary $\Gamma_\varepsilon = \partial\Omega_\varepsilon$ in accordance with

$$\Gamma_\varepsilon = \{\mathbf{x} + \varepsilon \mathbf{U}(\mathbf{x}) : \mathbf{x} \in \Gamma\}$$

with $\epsilon > 0$ sufficiently small [26]. This enables the definition of the *shape derivative* of the shape functional J at Ω in direction of the vector field $\mathbf{U}$ by

$$\delta J(\Omega)[\mathbf{U}] = \lim_{\varepsilon \to 0} \frac{J(\Omega_\varepsilon) - J(\Omega)}{\varepsilon}.$$

The shape functional J is shape differentiable at Ω, if the Eulerian derivative $\delta J(\Omega)[\mathbf{U}]$ exists for all directions $\mathbf{U}$ and if the mapping $\mathbf{U} \mapsto \delta J(\Omega)[\mathbf{U}]$ is linear and continuous. In particular, according to the Hadamard-Zolésio structure theorem [9, 37], it is known that the shape gradient ν can be expressed as a boundary integral of the form

$$\delta J(\Omega)[\mathbf{U}] = \int_\Gamma \langle \mathbf{V}, \mathbf{n} \rangle \, \nu \, \mathrm{d}\sigma.$$

Furthermore, we shall introduce the local shape derivative $\delta u = \delta u[\mathbf{U}]$ that describes the sensitivity of the state with respect to shape variations. It is defined pointwise by

$$\delta u(\mathbf{x}) := \lim_{\varepsilon \to 0} \frac{u_\varepsilon(\mathbf{x}) - u(\mathbf{x})}{\varepsilon}, \quad \mathbf{x} \in \Omega \cap \Omega_\varepsilon,$$

with the solution of the boundary value problem on the perturbed shape denoted by u_ε.

Theorem 1.1. *Let* $\mathbf{n}$ *denote the outward unit normal vector to the boundary* Γ *and consider a* C^2*-smooth boundary perturbation field* $\mathbf{U} : \Gamma \to \mathbb{R}^2$*. Then, the Hadamard representation of the shape gradient to the functional* (1.1) *reads*

$$\delta J(\Omega)[\mathbf{U}] = \int_\Gamma \langle \mathbf{U}, \mathbf{n} \rangle \langle \mathbf{A} \nabla u, \nabla u \rangle \, \mathrm{d}\sigma. \tag{1.5}$$

Proof. Following [37], we obtain

$$\delta J(\Omega)[\mathbf{U}] = 2 \int_\Omega \langle \mathbf{A} \nabla u, \nabla \delta u \rangle \, \mathrm{d}\mathbf{x} + \int_\Gamma \langle \mathbf{U}, \mathbf{n} \rangle \langle \mathbf{A} \nabla u, \nabla u \rangle \, \mathrm{d}\sigma.$$

Here, the local shape derivative $\delta u = \delta u[\mathbf{U}]$ satisfies

$$\begin{aligned} \operatorname{div}(\mathbf{A} \nabla \delta u) &= 0 && \text{in } \Omega, \\ \delta u &= -\langle \mathbf{U}, \mathbf{n} \rangle \frac{\partial u}{\partial \mathbf{n}} && \text{on } \Gamma. \end{aligned}$$

By applying integration by parts, we infer that the domain integral disappears

$$-\int_\Omega \langle \mathbf{A} \nabla u, \nabla \delta u \rangle \, \mathrm{d}\mathbf{x} = \int_\Omega \operatorname{div}(\mathbf{A} \nabla \delta u) u \, \mathrm{d}\mathbf{x} + \int_\Gamma \langle \mathbf{A} \nabla \delta u, \mathbf{n} \rangle u \, \mathrm{d}\sigma = 0,$$

where we inserted in the last step the state's homogeneous Dirichlet boundary condition. □

Notice that the shape gradient of the volume is given by

$$\delta V(\Omega)[\mathbf{U}] = \int_\Gamma \langle \mathbf{U}, \mathbf{n} \rangle \, \mathrm{d}\sigma. \tag{1.6}$$

1.3. **Necessary optimality condition.** The solution of the shape optimization problem (1.4) is equivalent to the solution $(\Omega^\star, \lambda^\star) \in \Upsilon \times \mathbb{R}$ of the saddle point problem

$$(\Omega^\star, \lambda^\star) = \arg \inf_{\Omega \in \Upsilon} \sup_{\lambda \in \mathbb{R}} L(\Omega, \lambda),$$

where $L(\Omega, \lambda)$ denotes the Lagrangian

$$L(\Omega, \lambda) = -J(\Omega) + \lambda \big(V(\Omega) - V_0 \big). \tag{1.7}$$

We have the following result on the necessary optimality condition.

Theorem 1.2. *The necessary condition of the minimization problem* (1.4) *under consideration reads*

$$\langle \mathbf{A} \nabla u, \nabla u \rangle \equiv \lambda^\star \quad \text{on } \Gamma^\star.$$

Proof. Standard optimization theory implies

$$\delta_\Omega L(\Omega^\star, \lambda^\star)[\mathbf{U}] = \int_{\Gamma^\star} \langle \mathbf{U}, \mathbf{n} \rangle \big\{ -\langle \mathbf{A} \nabla u, \nabla u \rangle + \lambda^\star \big\} \, \mathrm{d}\sigma = 0$$

for all $\mathbf{U} \in C^2(\mathbb{R}^2)$. This expression immediately yields the desired necessary condition. □

To include the volume constraint into our numerical method, we will consider the quadratic penalty method. We will thus minimizing

$$J_\alpha(\Omega) := -J(\Omega) + \frac{\alpha}{2}\big(V(\Omega) - V_0\big)^2 \tag{1.8}$$

with an appropriate choice of the penalty parameter α. We mention that one usually has to increase $\alpha \to \infty$ during the optimization procedure. This will not be done in our numerical tests, since we were satisfied with the approximation of the volume constraint. As an alternative to the penalty method, one can also apply the augmented Lagrangian, see for example [12, 15].

2. Shape representation via level set functions

Although boundary representations via parametrization allow a variation of the shape, they are usually not able to deal with topological changes of the shape, for instance merging two components of the shape [7]. This is where level set methods come into play, which have firstly been introduced by Osher and Sethian in [28]. In the following, we give a short introduction to the level set method.

2.1. **Level set functions.** The level set method allows to compute the motion of a boundary under a velocity field $\mathbf{V}$ and also works for multiply-connected domains, cf. [29]. For the level set method, the boundary is described implicitly, not explicitly. Thus, the interface is described as an isocontour of the level set function, i.e. $\phi(\mathbf{x}) = a$ for some scalar $a \in \mathbb{R}$. For simplicity, we will only consider the isocontour $\phi(\mathbf{x}) = 0$.

Let $\square$ be a square containing the shape Ω such that it is surrounded by a buffer zone. Then, implicit functions or level set functions $\phi : \square \to \mathbb{R}$ define a shape Ω by the rule

$$\begin{aligned} &\phi(\mathbf{x}) > 0 \Rightarrow \mathbf{x} \in \Omega, \\ &\phi(\mathbf{x}) = 0 \Rightarrow \mathbf{x} \in \Gamma, \\ &\phi(\mathbf{x}) < 0 \Rightarrow \mathbf{x} \in \mathbb{R}^2 \setminus \overline{\Omega}. \end{aligned} \tag{2.1}$$

An important representative of level set functions is the signed distance function, which we will consider in our particular implementation. Let us first consider a distance function $d(\mathbf{x})$ defined as

$$d(\mathbf{x}) = \min_{\mathbf{x}_I \in \Gamma} \{\|\mathbf{x} - \mathbf{x}_I\|\}.$$

Thus, the evaluation at a point $\mathbf{y} \in \mathbb{R}^2$ yields the point's distance to the boundary and yields 0 for points lying on the boundary Γ. The signed distance function ϕ has

the property $|\phi(\mathbf{x})| = d(\mathbf{x})$ while its sign tells whether a point is inside or outside the shape in the same convention as in (2.1). Hence, the signed distance function has the same sign property as level set functions, but also includes the additional condition of

$$\|\nabla\phi(\mathbf{x})\| = 1$$

almost everywhere. The signed distance function has the advantage that it returns the distance of a point to the boundary, whereas a general level set function only tells whether a point is inside or outside the shape.

2.2. **Level set equation.** For solving the shape optimization problem (1.4), we need to add dynamics to the level set function. To this end, we consider the simple convection (or advection) equation

$$\frac{\partial\phi}{\partial t} + \langle \mathbf{V}, \nabla\phi\rangle = 0, \tag{2.2}$$

where $t > 0$ is the variable in which the level set function evolves [30]. We shall define a velocity field $\mathbf{V}$ in normal direction, meaning that

$$\mathbf{V} = \nu\mathbf{n} \tag{2.3}$$

with a scalar function $\nu : \square \to \mathbb{R}$. Since the relation

$$\mathbf{n} = \frac{\nabla\phi}{\|\nabla\phi\|} \quad \text{on } \{\phi = 0\} \tag{2.4}$$

holds for the normal $\mathbf{n}$, we get the level set equation for motion in normal direction (cf. [30])

$$\frac{\partial\phi}{\partial t} + \nu\|\nabla\phi\| = 0. \tag{2.5}$$

2.3. **The choice of the velocity field.** It is crucial to choose the velocity field $\nu\mathbf{n}$ suitably in the level set equation (2.5). The goal is to obtain some decrease in the shape functional under consideration and hopefully convergence to a solution of the optimization problem [7]. We first define ν on the boundary of the domain by means of the shape gradient for the quadratic penalty functional $J_\alpha(\Omega)$ defined in (1.8):

$$\nu = -\langle \mathbf{A}\nabla u, \nabla u\rangle + \alpha\big(V(\Omega) - V_0\big) \quad \text{on } \Gamma. \tag{2.6}$$

We then need to extend ν into $\square$ in order to have a contribution of it on the discrete mesh in $\square$ used for numerical computations [7]. This is done by multiplying $\nu|_\Gamma$ with the expression

$$\delta(\phi) = \begin{cases} 0, & \phi < -\varepsilon, \\ \frac{1}{2} + \frac{1}{2}\cos\left(\frac{\pi\phi}{\varepsilon}\right), & -\varepsilon \le \phi \le \varepsilon, \\ 0, & \phi > \varepsilon \end{cases} \tag{2.7}$$

leading to a smeared out velocity field $\nu\mathbf{n} = \delta(\phi)\nu|_\Gamma\mathbf{n}$. The δ-function is an adaption of an approximation to the delta distribution which is presented in [30].

2.4. **Discretization of the level set function.** The discretization of the level set function ϕ is performed in a finite difference (FD) framework. The movement of the free boundary towards the optimal shape is modelled via the transport equation (2.5) with motion in normal direction using (2.6) as according velocity strength. The transport equation employs a second order TVD Runge-Kutta scheme as time discretization and a second order ENO scheme as space discretization, compare [28, 29]. To this end, we coupled our finite element (FE) method (see the next section) with the Level Set Method Toolbox library, cf. [23, 24, 25].

In general, the transport of the level set function ϕ due to (2.5) will destroy its signed distance property. This property, however, is essential for the stable and accurate approximation of the optimal shape. There exist numerous variants for the reinitialization of the level set function; the one we employ in our implementation is essentially the one of [36]. We shall give a short review of this scheme for the sake of completeness.

Consider a given function $\phi^\star(\mathbf{x})$, whose zero level set is the actual shape from the optimization procedure, i.e., a distorted signed distance function. To generate the appropriate signed distance function with an approximately identical zero level set as $\phi^\star(\mathbf{x})$, we discretize the following pseudo-transient Hamilton-Jacobi (HJ) problem

$$\frac{\partial\phi}{\partial t} + \operatorname{sign}(\phi^\star)(\|\nabla\phi\| - 1) = 0 \tag{2.8}$$

and evolve it to steady state with the initial value $\phi(\mathbf{x}, 0) = \phi^\star(\mathbf{x})$. Note that the term $\operatorname{sign}(\phi^\star)\|\nabla\phi\|$ can be interpreted as a motion with velocity $\operatorname{sign}(\phi^\star)$ along the normal direction away from the zero level set.

The Hamilton-Jacobi reinitialization (2.8) is discretized using a second order TVD Runge-Kutta method in time and a second order ENO scheme in space on a FD mesh. Moreover, we have to smooth the sign step-function so that the CFL condition will hold. Hence, in accordance with [31], we use the following smoothing scheme for the sign-function:

$$\operatorname{sign}(\phi^\star) \approx \frac{\phi^\star}{\sqrt{(\phi^\star)^2 + \|\nabla\phi^\star\|^2(\Delta x)^2}}.$$

Since our numerical approach relies on the signed distance property in the vicinity of the free surface only, it is sufficient to compute the solution of (2.8) in an ε-neighbourhood (2.7) of the zero level set. This is also the region, where the largest deviation from the exact signed distance occurs due to the transport of the level set

function. The parameter ε must be chosen with respect to the spatial resolution, i.e., $\varepsilon \approx \mathcal{O}(\Delta x)$. We use in our simulations $\varepsilon = 6\Delta x$ for the FD mesh.

The transport of the level set function with a second order TVD Runge-Kutta scheme in time basically consists of two Euler steps in which we have to accurately track the level set function as well as the velocity strength ν. Hence, after the predictor step, we have to perform the FE-mesh generation and the computation of $\nu^{n+\frac{1}{2}}$ and further we have to include the reinitialization step after each Runge-Kutta sub-step. This means, the extended sub-steps of the Runge-Kutta time-stepping scheme now give

compute FE-mesh using ϕ^n

compute ν^n via FE-solver

predictor: $\phi^{n+\frac{1}{2}} = \phi^n + \dfrac{\Delta t}{2} L^n(\phi^n, \nu^n)$ on FD-mesh

reinitialize $\phi^{n+\frac{1}{2}}$ via HJ equation on FD-mesh

compute FE-mesh using $\phi^{n+\frac{1}{2}}$

compute $\nu^{n+\frac{1}{2}}$ via FE-solver

corrector: $\phi^\star = \phi^n + \Delta t L^{n+\frac{1}{2}}(\phi^{n+\frac{1}{2}}, \nu^{n+\frac{1}{2}})$on FD-mesh

$\phi^{n+1} \leftarrow$ reinitialize $\phi^\star$ via HJ-equation on FD-mesh

with L^n and $L^{n+\frac{1}{2}}$ being the second order ENO space operators containing the level set ϕ and the velocity strength ν at times n and $n+\frac{1}{2}$, respectively; Δt is the time step.

Since the reinitialization equation and the transport equation are explicitly discretized, we have to control the time step so that the information is not tracked further than one grid-cell to preventing numerical instabilities. In case of the reinitialization equation, we know that the normal velocities have an absolute value ≤ 1, i.e., using a CFL number of $C = 0.5$ is sufficient for providing numerical stability, thus $\Delta t = 0.5\Delta x$ on the FD mesh. In case of the transport equation, we have to take care of the velocity profile along the free boundary. Hence, we use an adaptive time-step control in which a CFL number of $C = 0.5$ is imposed and, in addition, the maximum velocity is taken into account which gives

$$\Delta t = \frac{1}{2\sqrt{2}} \frac{\Delta x}{\|\nu\|_\infty}$$

to ensure numerical stability for the transport on the FD mesh.

3. Numerical method to compute the state

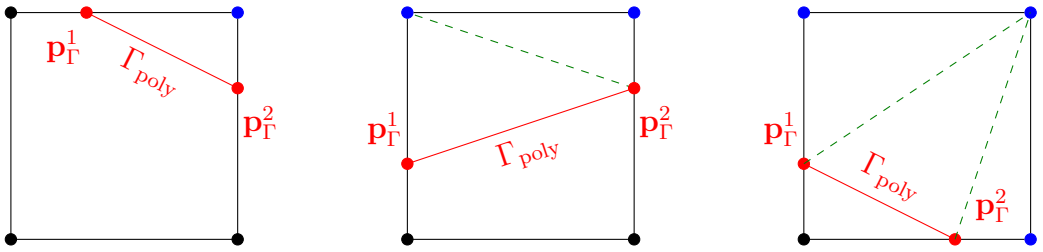
For solving the state equation (1.2) and for computing the shape gradient (1.5), which enters the definition of the velocity field via (2.6), we will use the finite element method. To that end, we implement a mesh generator based on the idea of marching cubes, which has firstly been proposed by Lorensen and Cline in [22].

3.1. **Marching cubes algorithm.** We create a regular grid of quadratic cells of step size h. By using the sign of the level set method, each vertex is classified as being inside or outside of the domain. For each edge of the cells which has one endpoint inside the domain and one endpoint outside of the shape, the point on the shape's boundary is determined by applying a bisection algorithm. Connecting all the boundary points yields a polygonal approximation Γ_{poly} of the boundary Γ of the zero level set.

For every boundary cell, i.e., cells which have both, inner and outer vertices, one of the situations seen in Figure 1 applies provided that the cell size h has been chosen small enough:

- one vertex lies inside and three vertices lie outside the zero level set,
- two vertices lie inside and two vertices lie outside the zero level set,
- three vertices lie inside and one vertex lies outside the zero level set.

Other cases are just rotations of the three cases in the figure.



(A) 1 inner / 3 outer points (B) 2 inner / 2 outer points (C) 1 outer / 3 inner points

FIGURE 1. Different cases for boundary cells, where blue vertices indicate inner points and black vertices indicate outer points. The dashed green lines present the subdivision into triangles.

A triangulation $\mathcal{T}_h = \{T_k\}_k$ is finally derived as follows. All interior cells are divided into two triangles. The boundary cells are divided as follows into triangles. In the first case (Figure 1a), no action is needed, since the inner vertex, indicated in blue, $\mathbf{p}_\Gamma^1$ and $\mathbf{p}_\Gamma^2$ already form a triangle. In the second case (Figure 1b), the quadrilateral formed by the two inner vertices, $\mathbf{p}_\Gamma^1$ and $\mathbf{p}_\Gamma^2$ is divided into two triangles using the

shorter one of the two diagonals as a new edge. In the last case (Figure 1c), the pentagon formed by the three inner vertices, $\mathbf{p}_\Gamma^1$ and $\mathbf{p}_\Gamma^2$ needs to be divided into three triangles. To this end, the shortest of the five diagonals of the pentagon is taken as an edge leading to a triangle and a quadrilateral. The quadrilateral is then subdivided into two triangles as in the second case. We refer to Figure 2 for an illustration of the output of the marching cubes algorithm.

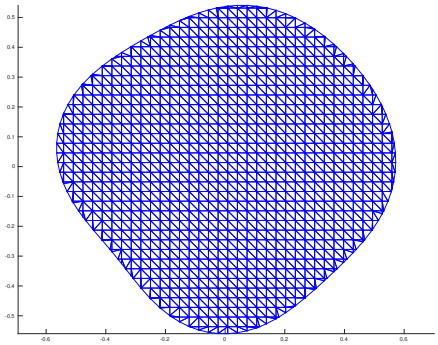


FIGURE 2. Output of the marching cubes algorithm.

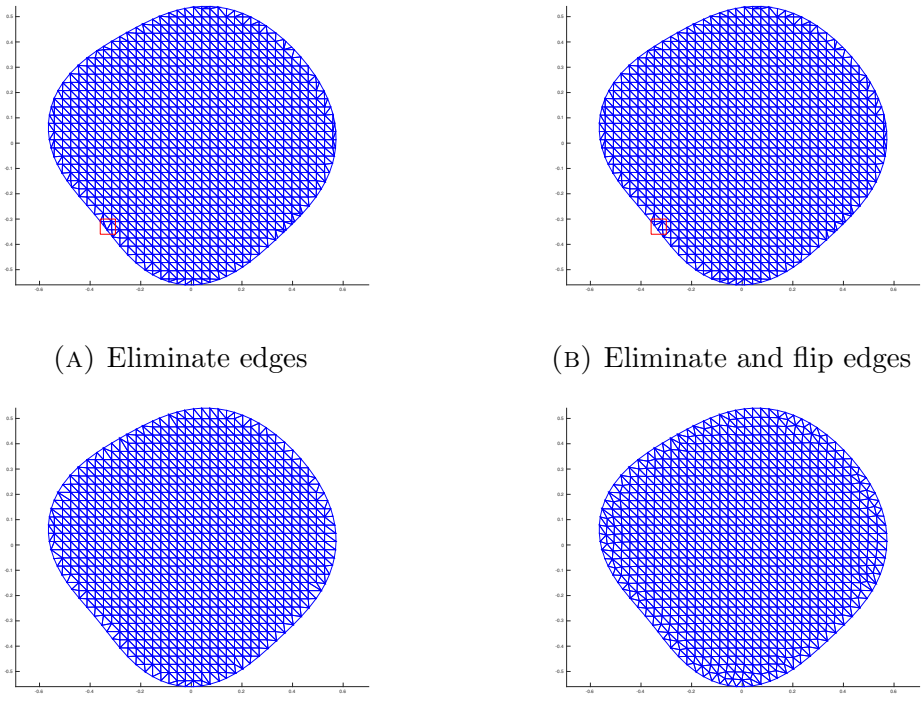
3.2. **Improvement of the triangulation.** Since triangulations created with the marching cubes algorithm may look bad, i.e. having short edges compared to the size h of the cells or acute angles, the triangulation $\mathcal{T}_h$ should be improved. Note that only triangles near the boundary need to be modified, as the triangulation is fine in the interior. We differ the procedures *edge flip*, *edge elimination*, and *vertex relocation* (see below for a precise description), which are applied in the following order:

(1) eliminate edges with small residua
(2) flip edges
(3) eliminate edges with small residua
(4) vertex relocation.

In our experiments, this yields fairly good meshes, compare Figure 3, where the original mesh was the one of Figure 2.

3.2.1. *Edge flip.* For a given edge in the interior of the domain, we consider the two associated elements T_1 and T_2. Together, these two elements form a quadrilateral and the edge is one of its two diagonals (see Figure 4a). The mesh might improve if we take the other diagonal in the quadrilateral as the edge, yielding two different triangles T_3 and T_4. The edge flip is accepted if the total fitness increases, i.e. if

$$\operatorname{fit}(T_1)\operatorname{fit}(T_2) < \operatorname{fit}(T_3)\operatorname{fit}(T_4).$$



(A) Eliminate edges

(B) Eliminate and flip edges

(C) Eliminate, flip and eliminate edges

(D) Eliminate, flip, eliminate edges and relocate vertices

FIGURE 3. The combination of the different mesh improvements. The red square indicates where an edge is flipped.

Here, the *fitness* of a triangle T is defined by

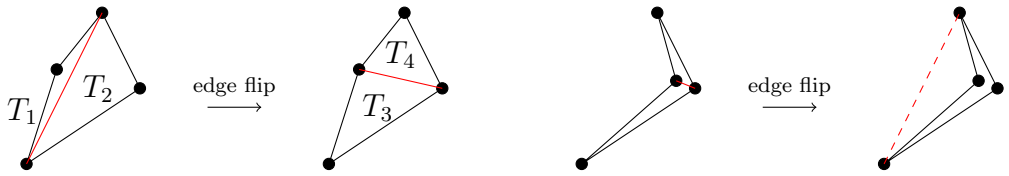
$$\text{fit}(T) = \frac{4\sqrt{3}|T|}{a^2 + b^2 + c^2},$$

where a, b, and c are the side lengths of the triangle under consideration. An optimal triangle, which is an equilateral triangle, has fitness 1, all other triangles have a fitness between 0 and 1.

We mention that the edge flip can only be applied in case of two triangles which form a strictly convex quadrilateral, otherwise the newly constructed triangles overlap (see Figure 4b).

3.2.2. *Elimination of edges.* Edges leading to acute triangles or triangles with short edges, as shown in Figure 5, can be eliminated in order to improve the triangulation. An edge at the boundary is eliminated by substituting it by the edge's midpoint $\mathbf{p}$, which is then moved onto the boundary again by means of the signed distance function:

$$\mathbf{q} = \mathbf{p} - \phi(\mathbf{p}) \frac{\nabla\phi(\mathbf{p})}{\|\nabla\phi(\mathbf{p})\|}.$$



(A) admissible and accepted (B) inadmissible

FIGURE 4. Edge flip

In case of an interior edge, we do not just replace the edge by its midpoint. We use a more sophisticated choice for the new vertex, which yields much better results in general.

FIGURE 5. Elimination of an edge.

We shall explain the elimination of the edge with endpoints $\mathbf{p}_1$ and $\mathbf{p}_2$. For all edges, which connect a vertex either with the vertex $\mathbf{p}_1$ or with the vertex $\mathbf{p}_2$, we consider the following least squares problem. We like to find the point $\mathbf{q} = (q_x, q_y)$, which has minimal distance to all the lines which underly the edges under consideration (see Figure 5). Thus, in Figure 6, we would like to find $\mathbf{q}$ such that the lengths of the green lines become minimal.

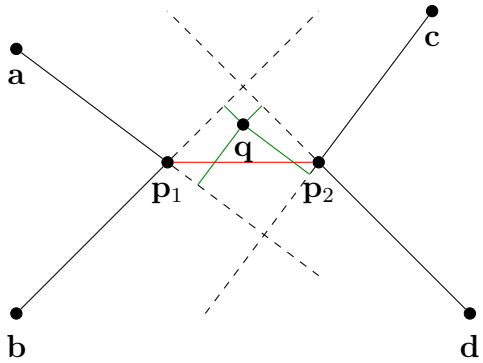


FIGURE 6. Illustration of the least squares problem for interior edges.

Let us now formulate the least squares problem by firstly considering the line going through the vertices $\mathbf{a} = (a_x, a_y)$ and $\mathbf{p}_1 = (p_x, p_y)$. This line is described by the

equation

$$(a_y - p_y)x + (p_x - a_x)y + a_x p_y - p_x a_y = 0.$$

The signed distance of a point (q_x, q_y) to this line is

$$\text{dist} = \frac{(a_y - p_y)q_x + (p_x - a_x)q_y + a_x p_y - p_x a_y}{\sqrt{(a_y - p_y)^2 + (p_x - a_x)^2}}.$$

Splitting this expression in terms with q_x and q_y and in terms without q_x and q_y leads to the first line of a matrix $\mathbf{M}$ and a right hand side $\mathbf{b}$, respectively, of a least squares problem. Repeating this procedure for all the other vertices leads to the following overdetermined system $\mathbf{Mq} = \mathbf{b}$ of linear equations which can be solved in the mean square sense:

$$\begin{bmatrix} \frac{a_y - p_y}{\sqrt{(a_y-p_y)^2+(p_x-a_x)^2}} & \frac{p_x - a_x}{\sqrt{(a_y-p_y)^2+(p_x-a_x)^2}} \\ \vdots & \vdots \end{bmatrix} \begin{bmatrix} q_x \\ q_y \end{bmatrix} = \begin{bmatrix} \frac{p_x a_y - a_x p_y}{\sqrt{(a_y-p_y)^2+(p_x-a_x)^2}} \\ \vdots \end{bmatrix}.$$

To avoid intersections among the new elements or reversion of the orientation of the vertices in an element, an edge elimination will only be accepted if for all elements, which have either $\mathbf{p}_1$ or $\mathbf{p}_2$ as a vertex, this vertex and the new point $\mathbf{q}$ lie on the same side of the supporting line.

In practice, only edges with a small residuum

$$\text{res} = \|\mathbf{Mq} - \mathbf{b}\|$$

are taken into account for a possible elimination. We thus sort the residua by size and considers only the first N smallest residua for a suitable constant N.

3.2.3. *Vertex relocation.* For an inner vertex $\mathbf{p}$, we consider all elements T_k which have $\mathbf{p}$ as their vertex (see Figure 7). For each such element, the barycentre is calculated by

$$\mathbf{c}_k = \frac{1}{3}(\mathbf{p}_1 + \mathbf{p}_2 + \mathbf{p}_3),$$

where $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$ denote the element's vertices. Moreover, the area $|T_k|$ of every element T_k is computed. Then, the barycentre $\mathbf{q}$ of the patch $\omega_{\mathbf{p}} = \bigcup_{\mathbf{p}\in T_k} T_k$ is given by

$$\mathbf{q} = \sum_{\mathbf{p}\in T_k} \frac{|T_k|}{|\omega_{\mathbf{p}}|}\mathbf{c}_k.$$

The vertex $\mathbf{p}$ gets replaced by $\mathbf{q}$ and one continues to the next inner point executing the same procedure. By moving every point into the barycentre of the patch, hopefully a fitter trangulation is generated.

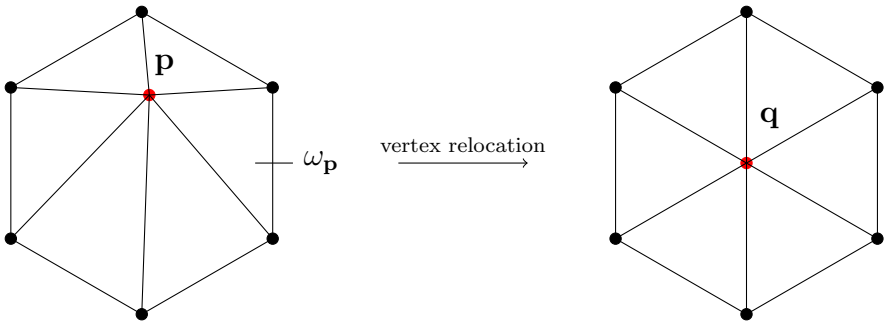


FIGURE 7. Move a vertex into the barycentre of the patch.

3.3. **Finite element discretization.** Having the triangulation $\mathcal{T}_h$ at hand, we can use the finite element method to solve the state equation (1.2), which is necessary for computing the velocity field (2.6). To that end, we discretize the variational formulation

$$\text{seek } u \in H^1_0(\Omega) \text{ such that } \int_\Omega \langle \mathbf{A}\nabla u, \nabla v\rangle \,\mathrm{d}\mathbf{x} = \int_\Omega f v \,\mathrm{d}\mathbf{x} \text{ for all } v \in H^1_0(\Omega)$$

in the finite element space

$$V_h = \{v \in C(\Omega) : v|_T \in \mathcal{P}_1 \text{ for all } T \in \mathcal{T}_h\}$$

of globally continuous, piecewise linear ansatz functions on the triangulation $\mathcal{T}_h$. Denoting the basis in V_h by $\{\varphi_k\}_k$ and making the ansatz $u_h = \sum_k u_k \varphi_k$, we arrive at the system of linear equations

$$\mathbf{A}_h \mathbf{u}_h = \mathbf{f}_h,$$

where

$$\mathbf{A}_h = \left[\int_\Omega \langle \mathbf{A}\nabla\varphi_\ell, \nabla\varphi_k\rangle \,\mathrm{d}\mathbf{x}\right]_{k,\ell}, \quad \mathbf{u}_h = [u_k]_k, \quad \mathbf{f}_h = \left[\int_\Omega f\varphi_k \,\mathrm{d}\mathbf{x}\right]_k.$$

This system of linear equations can be solved in nearly linear time when using nested dissection, see e.g. [6, 14, 19, 21] and the references therein.

Recall that for determining the velocity field $\nu\mathbf{n}$ we need to compute

$$\nu|_\Gamma = -\langle \mathbf{A}\nabla u, \nabla u\rangle + \alpha\big(V(\Omega) - V_0\big)$$

at every vertex of the boundary of the finite element mesh. To this end, we compute ∇u_h for every element which contains an edge on the boundary. Since every vertex at the boundary is the endpoint of two adjacent boundary edges, we have thus two values for $\langle \mathbf{A}\nabla u_h, \nabla u_h\rangle$ for the same boundary vertex. We then take the mean of these two values to obtain the value at the boundary point. For example, for the red boundary point in Figure 8, the two yellow elements are taken into account.

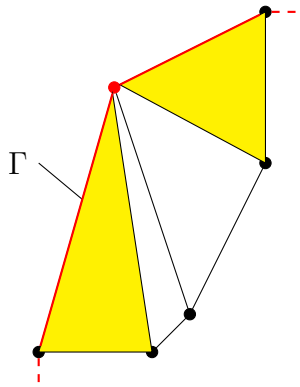


FIGURE 8. The two yellow triangles are considered in the computation of $\nu|_\Gamma$ at the red boundary vertex.

The volume $V(\Omega)$, also appearing in the definition of the velocity field, can be computed by using Gauss' theorem:

$$\int_\Omega 1 \, \mathrm{d}\mathbf{x} = \int_\Omega \frac{1}{2} \operatorname{div} \mathbf{x} \, \mathrm{d}\mathbf{x} = \int_\Gamma \frac{1}{2} \langle \mathbf{x}, \mathbf{n} \rangle \, \mathrm{d}\mathbf{x}.$$

The boundary integral can easily be evaluated by means of the trapezoidal rule. In particular, this is cheaper than summing up the area $|T_k|$ of all triangles.

4. Numerical results

In this section, we will investigate the behaviour of our shape optimization algorithm. We choose the diffusion matrix

$$\mathbf{A}(x, y) = \begin{bmatrix} 4 + 2.75 \sin(10x) & -1 \\ -1 & 2 + \sin(3x) \end{bmatrix}$$

and the inhomogeneity

$$f(x, y) = 2(1 - 3x^2)(1 - 3y^2)$$

as the data of the state equation (1.2). Moreover, we consider the volume constraint $V_0 := 0.2$. We consider five test cases, using initial shapes with different topology but always the correct overall area $V(\Omega) = 0.2$, which all lead finally to the same potato shaped optimum. However, the evolution of the optimization steps towards this result is substantially different for each test case and gives a clear insight into the way how the algorithm finds its minimum.

Identical parameters for all test cases are the penalization parameter used for imposing the volume constraint which is $\alpha = 10$ as well as the following level set parameters. The ε smoothing region for the Dirac functional (2.7) is set to $\varepsilon = 6\Delta x$ while the number of iterations for the reinitialization (2.8) is set to 12.

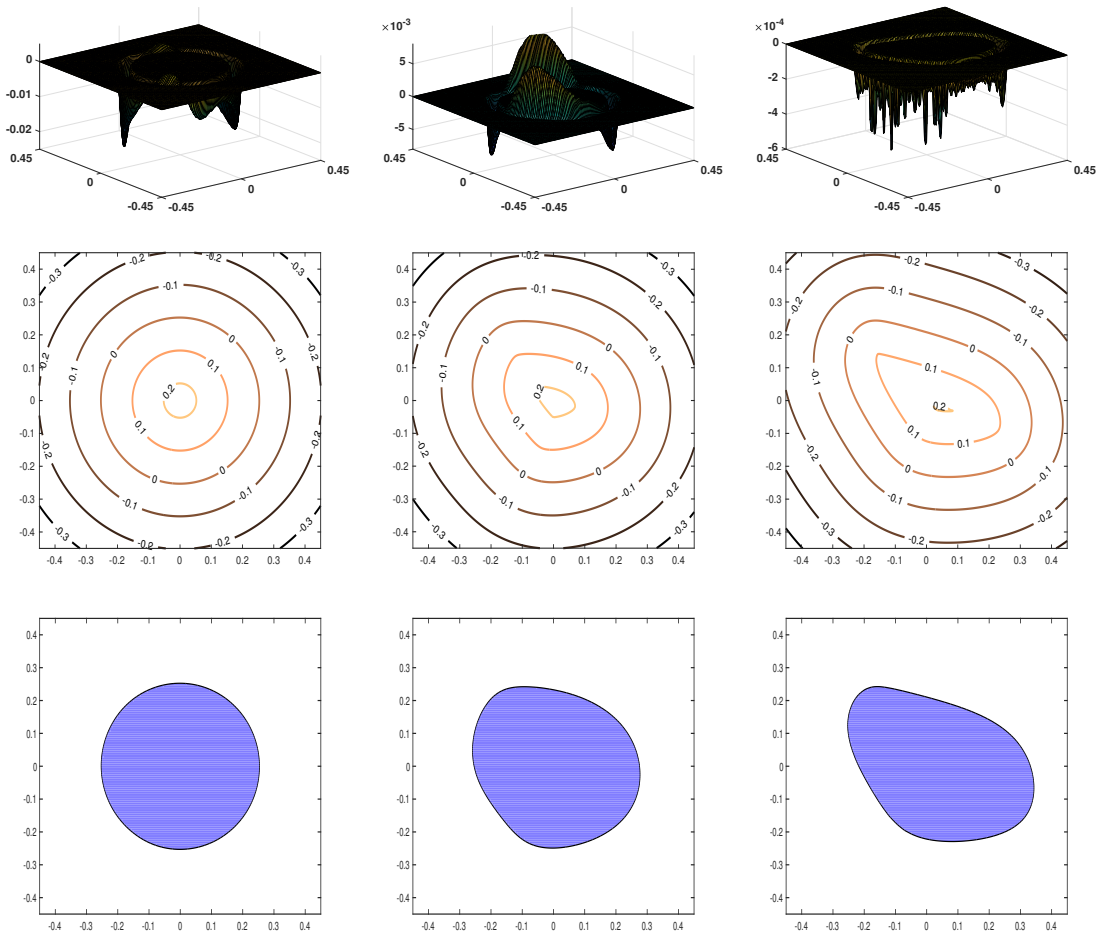


FIGURE 9. Scalar field of normal velocities (upper line) and level sets (middle line) and actual shapes after 1 and 40 and 1000 iterations.

4.1. **One circle.** The first test case concerns a circle with center $(0, 0)$ and radius $r = \sqrt{0.2/\pi}$. The size of the computational domain is $[-0.45, 0.45] \times [-0.45, 0.45]$, which is subdivided into 159^2 quadrangular grid-cells for the level set equation. We use a uniform mesh of the same grid size for the FEM solver away from the boundary and boundary fitted triangles defined by the zero level set near to the boundary.

Figure 9 shows the magnitude of the normal velocities and the level sets of the signed distance function as well as the actual shapes after 1 and 20 and 1000 iterations. Here, the level sets of the signed distance function are in each iteration parallel to each other which means that the signed distance property is well conserved after each optimization iteration. In particular, the zero level sets have always a smooth boundary and, hence, no artificial effects lead to an unrobust boundary evolution during optimization.

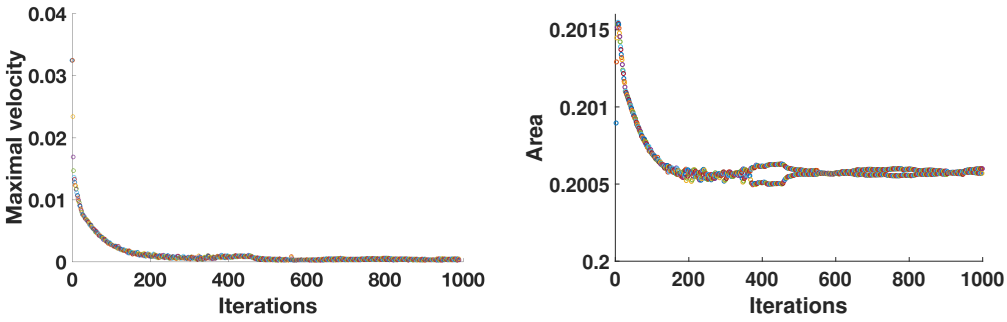


FIGURE 10. Maximal velocity and according mass after each iteration for the "one circle" test case as initial condition.

The left plot of Figure 10 shows the decay of the maximal velocity in each iteration step. When the velocity reaches the order of the discretization error, it does not decay anymore. Instead, the sign of the velocity alternates around the current boundary and, hence, the shape does not change fundamentally, which means that the algorithm found its approximate optimum. The right plot of Figure 10 shows how the area of the initial circle is conserved during the optimization procedure. Here, we observe that the initial circle area 0.2 increases up to the value $A \approx 0.2015$, then decays towards the value $A \approx 0.2006$, and finally alternates around this value until the steady state solution, i.e., the optimum shape, is reached.

4.2. **Ring with small hole.** The second test case concerns an annulus with outer radius $r = \sqrt{0.225/\pi}$ and with inner radius $r = \sqrt{0.025/\pi}$. The domain of computation has the size $[-0.45, 0.45] \times [-0.45, 0.45]$ and the center of the initial ring is set to $(0, 0)$. The grid sizes for the level set method and the finite element method are the same as in the first example. The interesting observation in this test is that the last line in Figure 11 shows how the hole of the ring starts to move towards the outer boundary of the ring. While the optimization algorithm tracks the hole, it does also continuously reduce its area. Finally, before reaching the boundary it completely disappears and then, as next step, the shape of the uniform area is optimized via the free boundary velocities leading to the potato shaped optimum known from test case one.

The second line of Figure 11 shows that the level sets of the signed distance function are parallel to each other in each iteration, i.e., the signed distance property is well conserved in each iteration. Again, the zero level sets have a smooth boundary for each iteration.

Figure 12 shows on the left the maximal velocity in each iteration step. Here, we observe that the optimization algorithm takes into account a rising velocity for

the free boundary as long as the ring-hole is shrinking. From the point in time where the hole of the ring disappears, the velocity rapidly decreases and finally stays on the order of the mesh size when the optimal shape is found. Note that the algorithm predicts a rapid reduction of the boundary velocity when the hole of the ring disappears. Hence, it works into that direction by moving the hole towards the outer boundary of the ring as well as by reducing its area. According to the right plot of Figure 12, the initial area of $A = 0.2$ increases simultaneously with the growing velocity up to the value of $A \approx 0.204$ and then finds its steady state.

4.3. **Ring with big hole.** Having the impression of the second test case in mind where the algorithm predicts a rapid reduction of the boundary velocity when working into the direction of moving the hole towards the outer boundary of the ring as well as of reducing its area, we are curious to see the behaviour of a ring containing a hole with twice as big radius. Hence, we shall modify the previous test

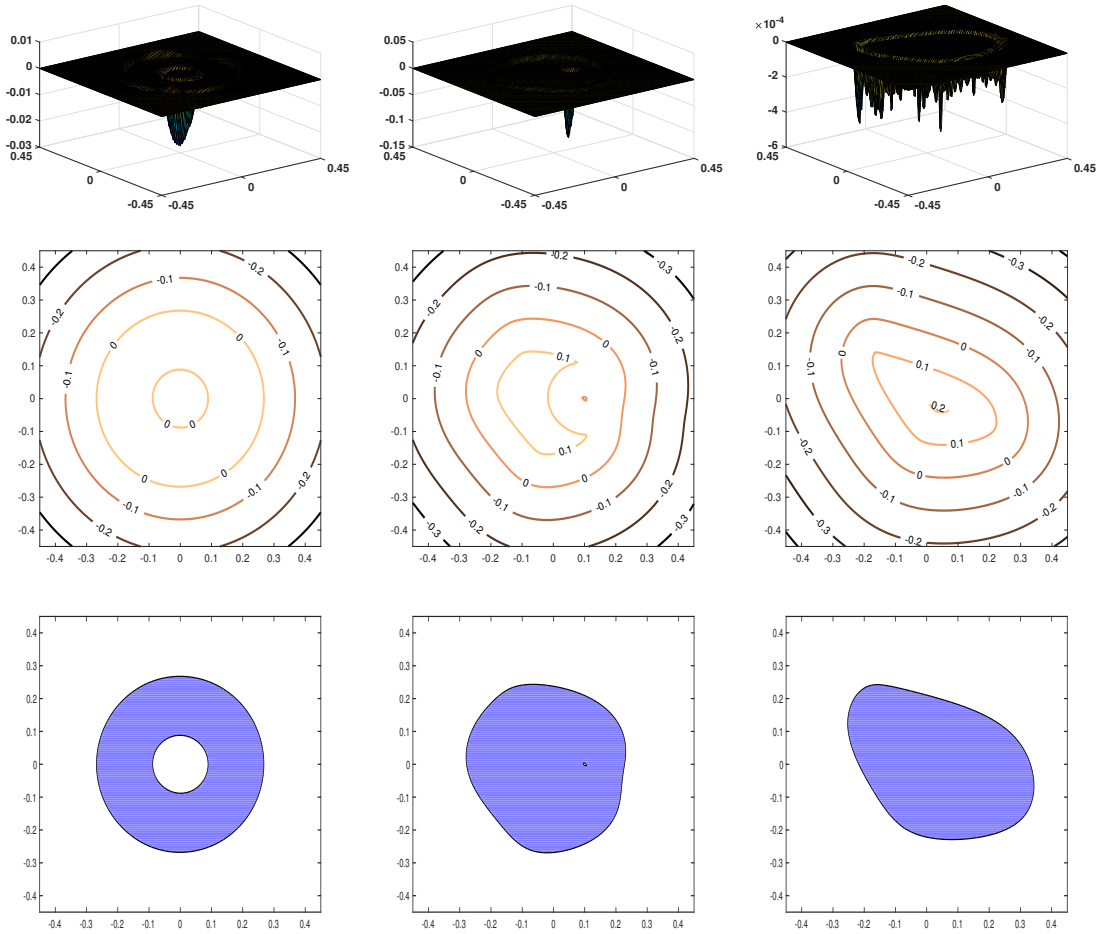


FIGURE 11. Scalar field of normal velocities (upper line) and level sets (middle line) and actual shapes after 1 and 95 and 1000 iterations.

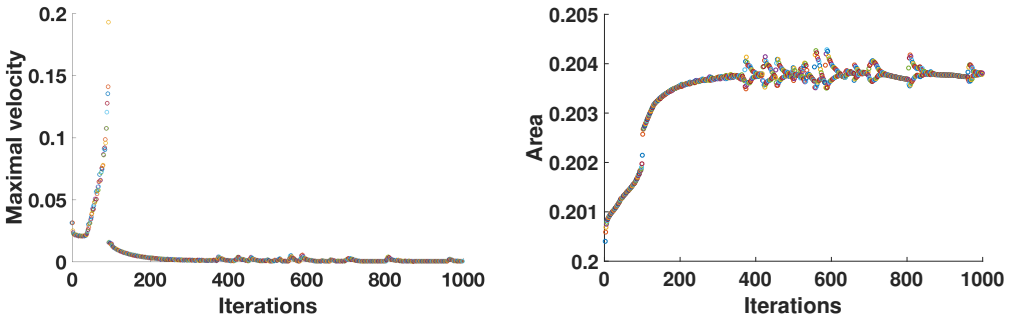


FIGURE 12. Maximal velocity and according mass after each iteration for the "ring with small hole" test case as initial condition.

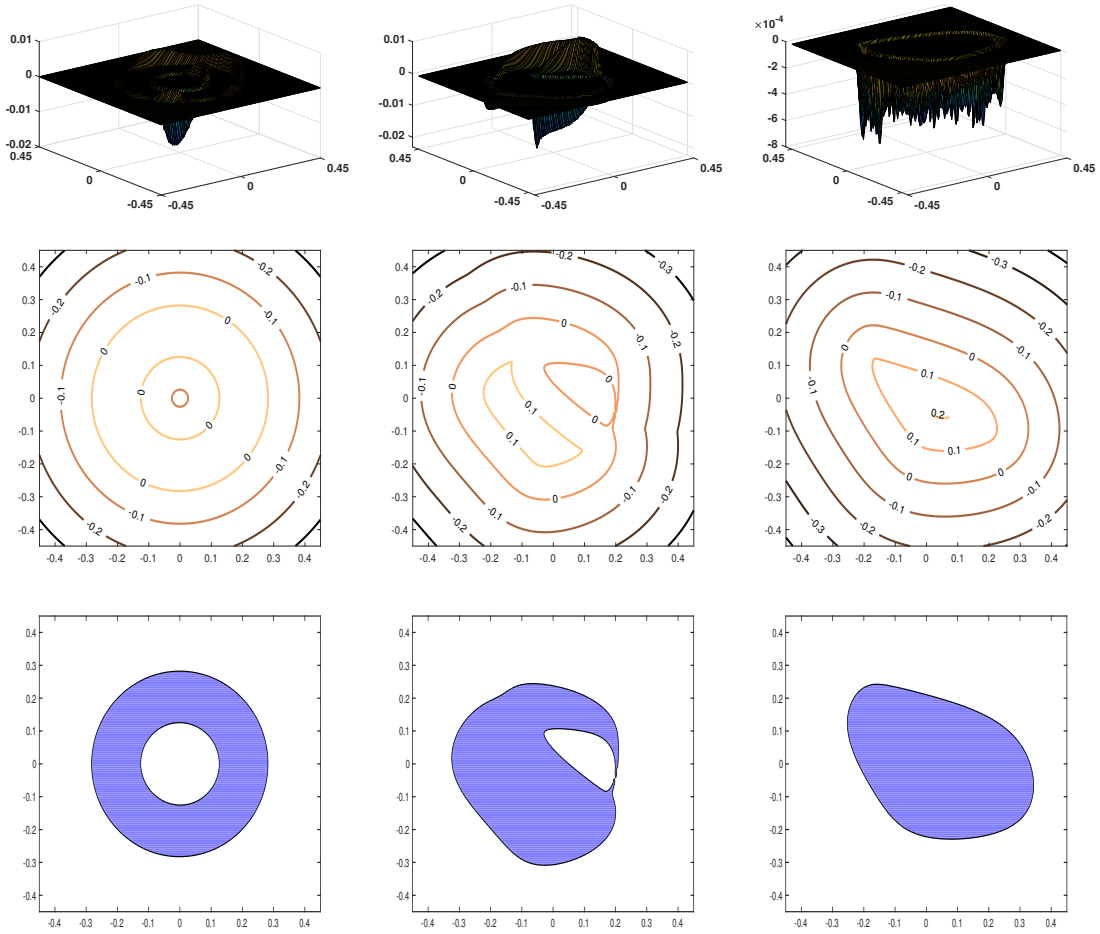


FIGURE 13. Scalar field of normal velocities (upper line) and level sets (middle line) and actual shapes after 1 and 125 and 1000 iterations.

case by increasing the hole in the ring domain: we choose a ring with outer radius $r = \sqrt{0.25/\pi}$ and with inner radius $r = \sqrt{0.05/\pi}$. All the other settings are the same as before.

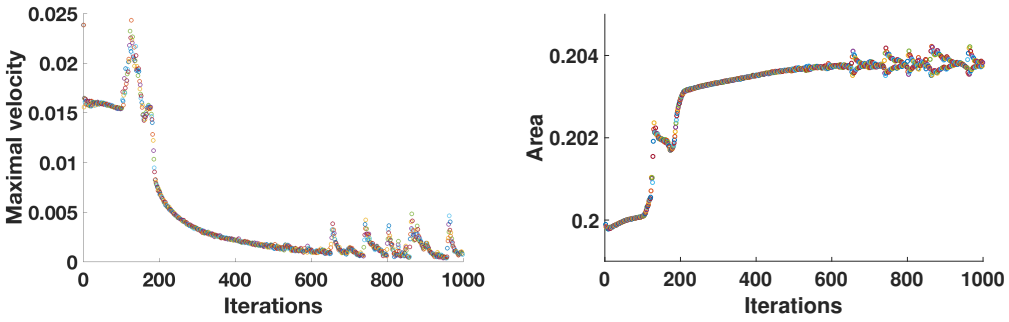


FIGURE 14. Maximal velocity and according mass after each iteration for the "ring with big hole" test case as initial condition.

The last line of Figure 13 shows the evolution of the optimization algorithm for this case. Again, we clearly observe that the hole starts to move towards the outer boundary of the ring. But now, during the course of the iteration, the area of the hole does not completely disappear. Instead, it reaches the outer boundary of the ring and hence undergoes a topological change, so that finally we get one bulk of surface area without holes. As next step, the shape of the uniform area converges towards the potato shaped optimum known from the test cases before where the boundary velocities appear to be minimal.

Figure 14 shows on the left that the optimization process for the ring with a big hole as initial condition takes a limited rising free boundary velocity into account with the aim to finally find an optimal shape, leading to minimal velocities for the steady state. In Figure 14 on the right we see the evolution of the area during the optimization process. Again, the initial area increases simultaneously with the velocity for the first 300 iterations, in which the velocity is strong enough, too. The area grows up to the value of $A \approx 0.204$ where it becomes stationary.

4.4. **Two circles in short distance.** Knowing the behaviour of the optimization algorithm for the two rings, we came up with the question how the optimization algorithm will behave for two neighbouring circles as initial condition. The circles are of the same radius $r = \sqrt{0.1/\pi}$ and their centers have a short distance of $d = 0.44$ to each other, so that the overall area of both circles is $A = 0.2$ as in the test cases before. The dimension of the computational domain is $[-0.6, 0.6] \times [-0.45, 0.45]$ and the according grid resolution is 212×159 quadrangular grid-cells.

The first column of Figure 16 shows the according scalar values for the normal velocities and the second column shows the evolution towards the optimal shape for the two circles in short distance to each other. We see that both circles start to move towards each other while one circle is growing and the other is shrinking.

However, the shrinking circle will not disappear completely before it merges with its neighbour and becomes a uniform bulk quantity again. Then, the shape of the uniform area starts to deform towards the optimal shape.

The maximal velocity in each iteration is found in Figure 15 on the left. We observe that the optimization algorithm starts with a soft monotone decay in where the circles approach each other. But when the circles start to merge, the maximal free boundary velocity suddenly rises by a factor of 4.5 and then slowly decays and finally converges towards zero up to the order of the discretization error. As in the cases before, we observe on the right of Figure 15 that the initial area $A = 0.2$ increases simultaneously with the velocity for the first 300 iterations and then stagnates around $A \approx 0.204$ since the optimal shape is reached.

4.5. **Two circles in long distance.** The final test case concerns the two circles of radius $r = \sqrt{0.1/\pi}$ but now with a longer distance of $d = 0.6$ between their centers. All the other settings like the computational domain and its grid resolution are the same as in the previous test case.

We observe in this last numerical test that both circles start to approach each other. However, during the course of the iteration, the left circle is growing in area while the right one is shrinking, see Figure 17. Because of the longer initial distance between the circles compared to the test case before, this leads into a unification process without merging of the circles, i.e., the right circle loses area until it completely vanishes while the left one doubles its area instead. Thereafter, the free boundary velocity starts to deform the unified area into the well known optimal potato shape.

This behaviour is also found in the left plot of Figure 18, which is somehow looking like the mirrored maximal free boundary velocities in the left plot of Figure 15. After a decrease of the maximal velocity during the first 100 iterations, instead of suddenly jumping to a maximum value, the maximal velocity starts to continuously

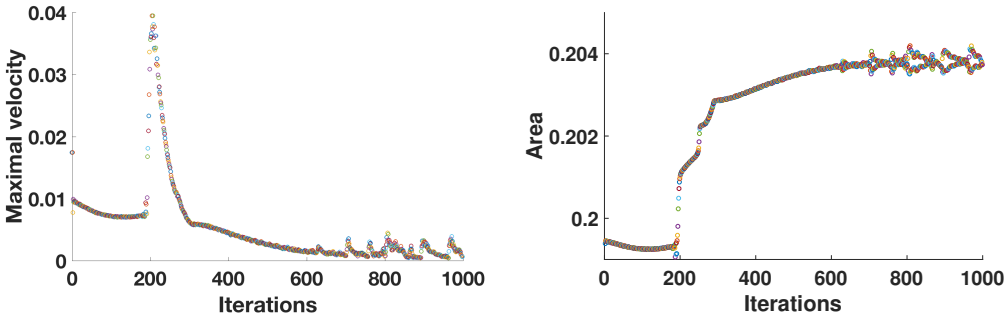


FIGURE 15. Maximal velocity and according mass after each iteration for the "two circles in short distance" test case as initial condition.

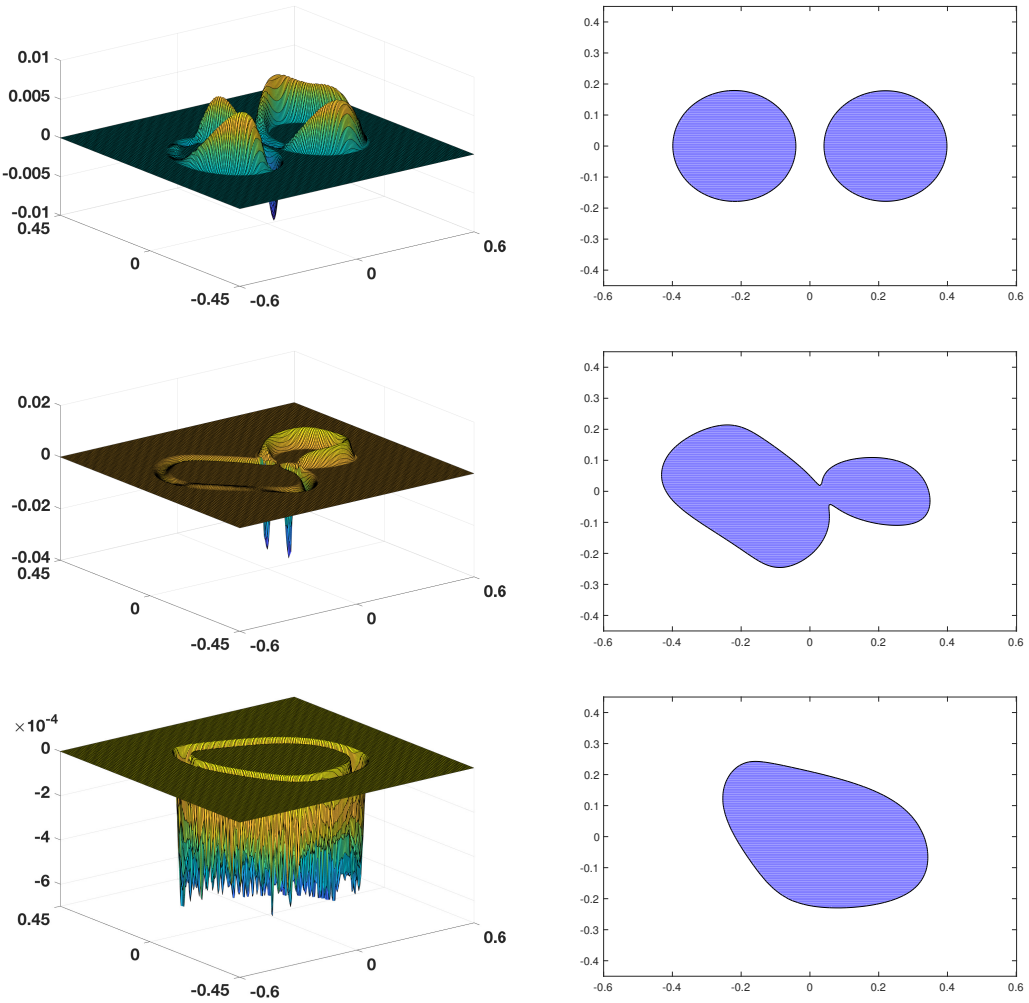


FIGURE 16. Scalar field of normal velocities (left) and actual shapes (right) after 1 and 200 and 1000 iterations.

grow as long as the right circle is shrinking. From the point on in which the area of the right circle completely vanishes, also the maximal boundary velocity suddenly decreases by a factor of three and keeps on deforming the single bulk area towards the optimal potato shape.

Opposite to the cases before, we observe in the right plot of Figure 18 that the initial area $A = 0.2$ initially slightly decreases as long as the velocity decreases. Then after 200 iterations the velocity increases simultaneously with the area. The area grows up to the value of $A \approx 0.2005$ and then finds its steady state in which the final optimal shape is approximately reached. Indeed, also for this test case, we get the same potato shaped optimum as in the test cases before. This leads us to the conclusion that this shape is a very robust solution, at least under several different initial conditions.

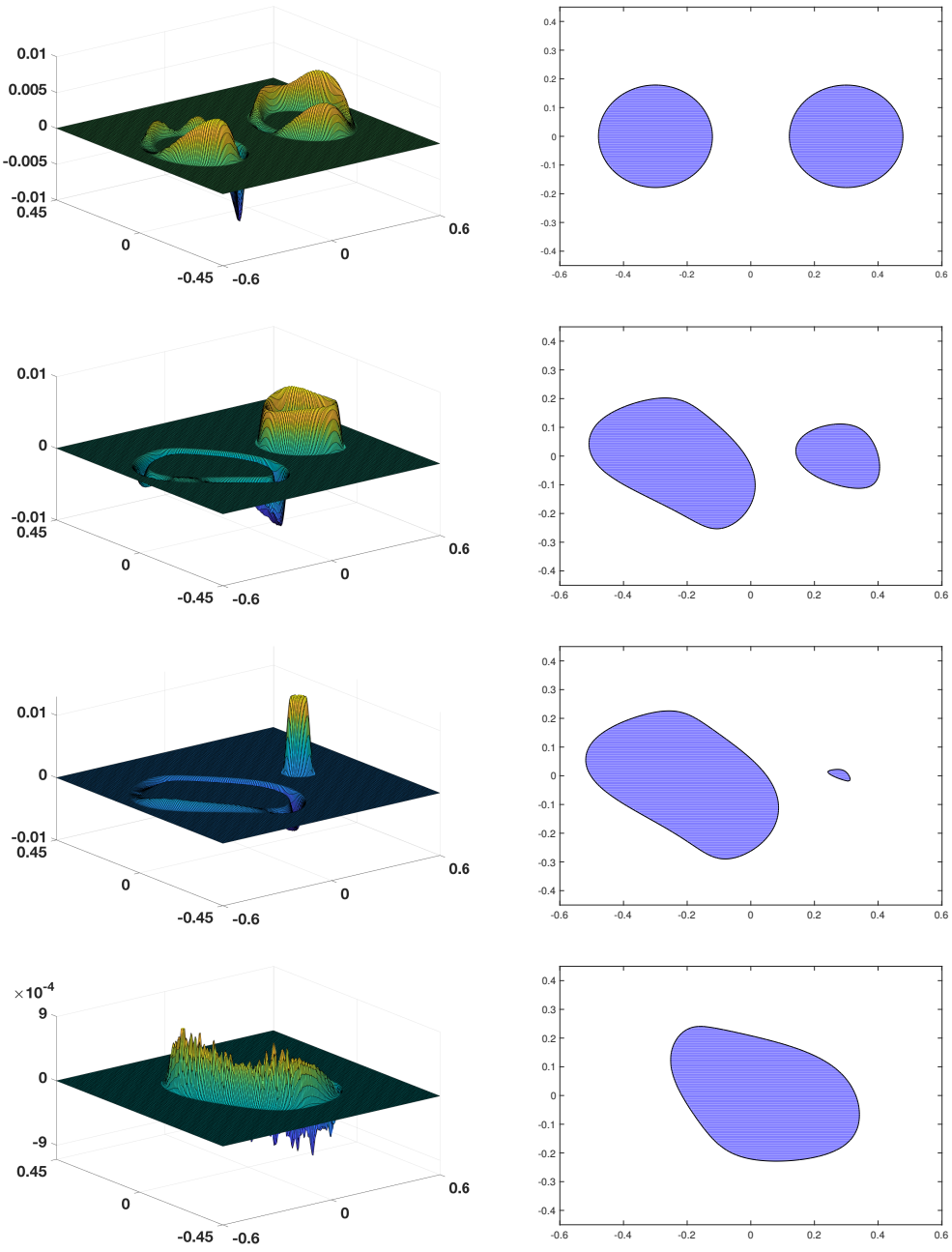


FIGURE 17. Scalar field of normal velocities (left) and actual shapes (right) after 1 and 190 and 300 and 1000 iterations.

5. Concluding remarks

In the present article, we solved a free boundary problem for an elliptic partial differential operator with non-constant coefficients. The sought domain is represented by a level set function, which is discretized by a second-order scheme for the Hamilton-Jacobi reinitialization as well as for the transport equation. We applied a marching

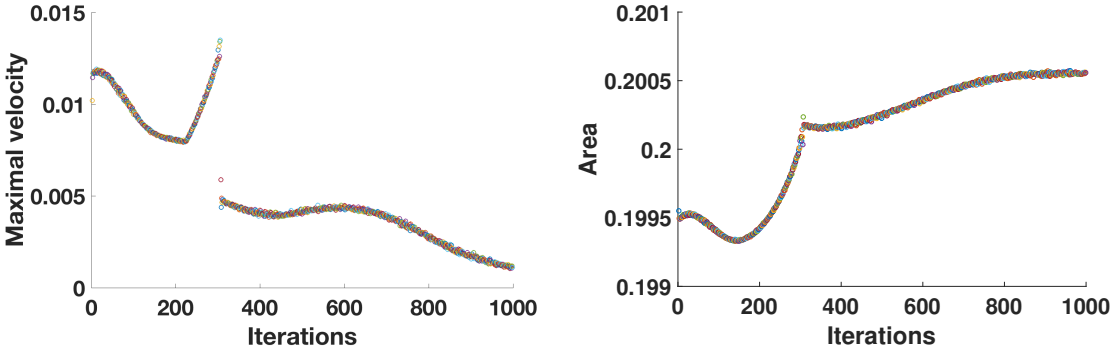


FIGURE 18. Maximal velocity and according mass after each iteration for the "two circles in long distance" test case as initial condition.

cubes algorithm to construct suitable triangulations for the finite element method. We demonstrated by numerical experiments that we succeeded in developing an efficient algorithm for the numerical solution of advanced shape optimization problems.

Acknowledgement. The authors like to thank Monica Bugeanu and Peter Zaspel very much for fruitful discussions and their valuable input to the present research.

REFERENCES

[1] A. Acker. On the geometric form of Bernoulli configurations. *Math. Meth. Appl. Sci.*, 10:1–14, 1988.

[2] H.W. Alt and L.A. Caffarelli. Existence and regularity for a minimum problem with free boundary. *J. Reine Angew. Math.*, 325:105–144, 1981.

[3] K. Bandara, F. Cirak, G. Of, O. Steinbach, and J. Zapletal. Boundary element based multiresolution shape optimisation in electrostatics. J. Comput. Phys., 297:584–598, 2015.

[4] S. Belov and N. Fujii. Symmetry and sufficient condition of optimality in a domain optimization problem. *Control Cybern.*, 26:45–56, 1997.

[5] A. Beurling. On free boundary problems for the Laplace equation. *Seminars on Analytic functions, Institute for Advanced Study, Princeton, NJ*, 1:248–263, 1957.

[6] I. Brainman and S. Toledo. Nested-dissection orderings for sparse LU with partial pivoting. *SIAM J. Matrix Anal. Appl.*, 23:998–1012, 2002.

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

[8] R. Croce, M.A. Schweitzer, and M. Griebel. Numerical simulation of bubble and droplet-deformation by a level set approach with surface tension in three dimensions. *Int. J. Numer. Meth. Fluids*, 62(9):963-993, 2009.

[9] M. Delfour and J.-P. Zolésio. *Shapes and Geometries*. SIAM, Philadelphia, 2001.

[10] K. Eppler and H. Harbrecht. Exterior Electromagnetic Shaping using Wavelet BEM. *Math. Meth. Appl. Sci.*, 28:387–405, 2005.

[11] K. Eppler and H. Harbrecht. Efficient treatment of stationary free boundary problems. *Appl. Numer. Math.*, 56:1326–1339, 2006.

[12] R. Fletcher. *Practical Methods for Optimization.* Volumes 1&2, Wiley, New York, 1980.

[13] M. Flucher and M. Rumpf. Bernoulli's free-boundary problem, qualitative theory and numerical approximation. *J. Reine Angew. Math.*, 486:165–204, 1997.

[14] A. George. Nested dissection of a regular finite element mesh. *SIAM J. Numer. Anal.*, 10:345–363, 1973.

[15] C. Grossmann and J. Terno. *Numerik der Optimierung.* Teubner, Stuttgart, 1993.

[16] H. Harbrecht and J. Tausch. On the numerical solution of a shape optimization problem for the heat equation. *SIAM J. Sci. Comput.*, 35(1):A104–A121, 2013.

[17] J. Haslinger and P. Neitaanmäki. *Finite Element Approximation for Optimal Shape, Material and Topology Design*, 2nd edition. Wiley, Chichester, 1996.

[18] J. Haslinger und R.A.E. Mäkinen. *Introduction to Shape Optimization. Theory, Approximation and Computation.* Advances in Design and Control, vol. 7, SIAM Philadelphia, 2003.

[19] B. Hendrickson and E. Rothberg. Improving the run time and quality of nested dissection ordering. *SIAM J. Sci. Comput.*, 20:468–489, 1998.

[20] K. Kunisch and G. Peichl. Shape optimization for mixed boundary value problems on an embedding domain method. *Dyn. Contin. Discrete Impulsive Syst.*, 4:439–478, 1998.

[21] R.J. Lipton, D.J. Rose, and R.E. Tarjan. Generalized nested dissection. *SIAM J. Numer. Anal.*, 16:346–358, 1979.

[22] W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, 1987.

[23] I.M. Mitchell. The flexible, extensible and efficient toolbox of level set methods. *J. Sci. Comput.*, 35(2–3):300–329, 2008.

[24] I.M. Mitchell. A toolbox of level set methods (version 1.1). Department of Computer Science, University of British Columbia, Vancouver, Canada, Tech. Rep. TR-2007-11, June 2007. `http://www.cs.ubc.ca/~mitchell/ToolboxLS/toolboxLS.pdf`

[25] I.M. Mitchell and J.A. Templeton. A toolbox of Hamilton-Jacobi solvers for analysis of nondeterministic continuous and hybrid systems. In *Hybrid Systems: Computation and Control*, edited by M. Morari and L. Thiele. Lect. Notes Comput. Sci. 3414, Springer, Berlin, pages 480–494, 2005.

[26] F. Murat and J. Simon. Étude de problèmes d'optimal design. In *Optimization Techniques, Modeling and Optimization in the Service of Man*, edited by J. Céa. Lect. Notes Comput. Sci. 41, Springer, Berlin, pages 54–62, 1976.

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

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

[29] S. Osher and R. Fedkiw. Level set methods: An overview and some recent results. *J. Comput. Phys.*, 169(2):463–502, 2001.

[30] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, New York, 2003.

[31] D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang, A PDE-based fast local level set method. *J. Comput. Phys.*, 155:410–438, 1999.

[32] M. Pierre and J.-R. Roche Numerical Simulation of tridimensional electromagnetic shaping of liquid metals. *Numer. Math.*, 65: 203–217, 1993.

[33] O. Pironneau. *Optimal shape design for elliptic systems*. Springer, New York, 1983.

[34] J.-R. Roche and J. Sokolowski. Numerical methods for shape identification problems. *Control Cybern.*, 25:867–894, 1996.

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

[36] M. Sussman, P. Smereka, and S. Osher. A level set approach for computing solutions to incompressible two-phase flow. *J. Comput. Phys.*, 114:146–159, 1994.

[37] J. Sokolowski and J.-P. Zolésio. *Introduction to Shape Optimization*. Springer, Berlin, 1992.

[38] A.D. Zacharopoulos, S.R. Arridge, O. Dorn, V. Kolehmainen and J. Sikora. Three-dimensional reconstruction of shape and piecewise constant region values for optical tomography using spherical harmonic parametrization and a boundary element method. *Inverse Problems*, 22:1509–1532, 2006.

Rahel Brügger, Roberto Croce, Helmut Harbrecht, Departement Mathematik und Informatik, Universität Basel, Spiegelgasse 1, 4051 Basel, Schweiz.

E-mail address: {ra.bruegger,roberto.croce,helmut.harbrecht}@unibas.ch