SHAPE OPTIMIZATION USING THE FINITE ELEMENT METHOD ON MULTIPLE MESHES WITH NITSCHE COUPLING*

JØRGEN S. DOKKEN†, SIMON W. FUNKE †, AUGUST JOHANSSON †, AND STEPHAN SCHMIDT ‡

Abstract. An important step in shape optimization with partial differential equation constraints is to adapt the geometry during each optimization iteration. Common strategies are to employ mesh-deformation or re-meshing, where one or the other typically lacks robustness or is computationally expensive. This paper proposes a different approach, in which the computational domain is represented by multiple, independent non-matching meshes. The individual meshes can move independently, hence mesh deformation or re-meshing is entirely avoided if the geometry can be parameterized by a combination of rigid motions and scaling. For general geometry changes, we present a deformation scheme tailored to non-matching meshes. This deformation scheme is robust because the non-matching mesh interfaces are free to move, and computationally cheap because the scheme is applied only on a subset of the meshes. To solve the state and corresponding adjoint equations we use the multimesh finite element method. This method weakly enforces continuity over the non-matching mesh interfaces by using Nitsche and additional stability terms. To obtain the shape derivatives we analyze both the strong formulation (Hadamard formulation) and weak formulation (method of mappings). We demonstrate the capabilities of our approach on the optimal placement of heat emitting wires in a cable to minimize the chance of overheating, the drag minimization in Stokes flow, and the orientation of nine objects in Stokes flow.

Key words. Shape Optimization, Multimesh Finite Element Method, Hadamard representation.

AMS subject classifications. 35Q93, 49Q10, 65M85, 65N30, 68N99.

1. Introduction.

During the last two decades, there has been an increasing need to couple simulation with optimization [52]. Of particular industrial relevance are shape optimization problems, which aim to optimize the shape of an object subject to physical constraints, typically described by partial differential equations (PDEs). Examples of industrial problems that have been modeled are the drag minimization of airplanes and cars [28, 32, 38], the shape optimization of acoustic horns [44], and the optimal design of current carrying multi-cables [19]. The success of these applications is driven by efficient optimization algorithms and methods for solving PDEs. More specifically, gradient-based optimization methods have shown to converge quickly and often independently of the number of design variables. The required shape gradients are derived through shape calculus and the adjoint PDE [15, 43, 46]. The finite element method (FEM) is an efficient and flexible method for solving a wide range of PDEs. In the last decades, this method has gained popularity in both the scientific and industrial environment due to its mathematical foundation and geometrical flexibility.

A critical part in shape optimization algorithms is handling of geometry changes during each optimization iteration. For FEM based models this means that the computational mesh must be updated to a new target geometry at low cost while maintaining a high mesh quality. Mesh deformation and re-meshing are commonly used strategies to update the mesh. Mesh deformation methods often involve the solution of an auxiliary PDE. However, the mesh quality may degrade or even degenerate

*Submitted to the editors DATE.

Funding: This work was supported by the Research Council of Norway through a FRIPRO grant, project number 251237.

†Simula Research Laboratory, Lysaker, Norway (dokken@simula.no).

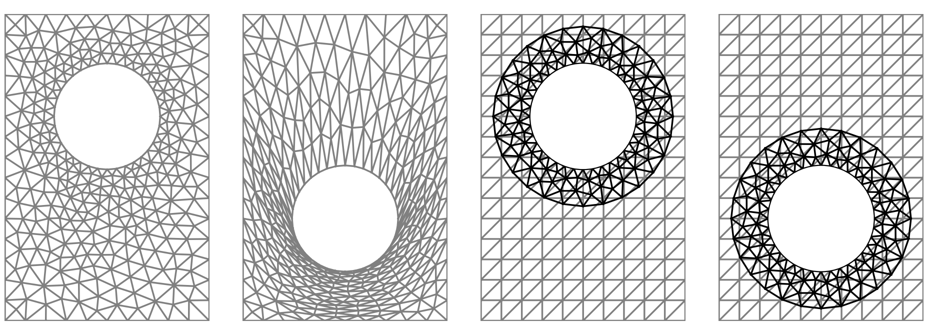
‡Universität Würzburg, Germany.

for large deformations. Several deformation schemes have therefore been proposed to handle large deformations [42, 51] at the expense of higher computational cost. In contrast, re-meshing guarantees high quality meshes for any geometrical change. However, drawbacks are that a tight coupling between the finite element model and the meshing algorithm is required, and the high computational cost of meshing algorithms [9].

To overcome these limitations, we propose a shape optimization algorithm where the domain is represented by multiple, non-matching meshes, as illustrated in Figure 1. Each mesh can be freely rotated, scaled or translated at a low computational cost without impacting the mesh quality. Therefore if the goal is to optimally rotate, scale or translate objects within a larger geometry, the need for re-meshing and mesh deformation is alleviated. For arbitrary geometry changes, mesh deformation on multiple meshes is more robust than on a single mesh, since the non-matching interfaces can deform freely and hence avoid compression effects. Furthermore, re-meshing and mesh deformation is computationally cheaper on an individual mesh than on the full geometry.

We rely on the multimesh finite element method (multimesh FEM) [22] to solve PDEs on multiple non-matching meshes. This method is highly embedded in the finite element setting, as opposed to existing approaches like Chimera [11, 49, 50] and Overset methods [4, 14] and references therein.

In this paper, we present methods for solving shape optimization problems with the multimesh FEM. Specifically, we derive shape derivatives in a multimesh FEM setting using both the method of mappings [30, 37] and the Hadamard formulation [46]. We conclude that the Hadamard formulation is better suited for the multimesh FEM. In a numerical example, we investigate the discrete inconsistencies in the shape derivative introduced by the Hadamard formulation. We also propose a mesh deformation scheme, tailored to the multimesh FEM, based on the linear elasticity with Neumann boundary conditions. To the best of our knowledge, this is the first instance of a FEM with multiple overlapping meshes in the setting of shape optimization.



$T = 0$ $T = 1$ $T = 0$ $T = 1$

FIG. 1. *A comparison of a moving object described with a single mesh and with multiple meshes. To deform the single mesh, we use an Eikonal convection equation, combined with a centroidal Voronoi tessellation (CVT) smoothing [42]. The mesh quality, quantified by the minimum radius ratio decreases from* 0.75 *to* $6 \cdot 10^{-4}$*, and the mesh is degenerated. In the multimesh approach we introduce one fixed background mesh and one mesh containing the ball which can be translated arbitrarily. Here the mesh quality is not impacted by translation. The minimum radius ratio is* 0.72.

1.1. Related work. The use of multiple meshes was first used to overcome the limitations of structured meshes in finite difference and structured finite volume schemes [5, 20, 48, 55]. These many-mesh techniques (also known as Chimera or

Overset techniques) [49] allow for multiple holes and moving domains, making them particularly popular for aerodynamic applications [47] and shape optimization [25].

A recent method for generalized domain descriptions for FEM is the cut finite element method (CutFEM) [12]. This method uses a Nitsche based formulation to weakly enforce boundary conditions on non-resolved boundaries, typically described by a level-set function. CutFEM has been used for a wide range of shape and topology optimization problems, such as acoustics [8], elasticity [3, 13] and incompressible flow [54]. The multimesh FEM [22] is a generalization of the CutFEM, where the computational domain is described by an arbitrary number of overlapping non-matching meshes. The multimesh FEM has so far been explored for the Poisson and Stokes-equations [18, 23, 24], but not yet in the setting of shape optimization.

For other methods for shape optimization of complex computational domains, we refer to [6, 31, 34, 53] and the references therein.

1.2. Outline. This paper is organized as follows. Section 2 introduces the multimesh finite element method. Section 3 presents how to compute shape derivatives for problems discretized with the multimesh FEM. In Section 4 we present how to perform mesh updates on multiple meshes. Thereafter, we present several numerical examples in Section 5. Finally, we summarize and draw conclusions in Section 6.

2. The multimesh finite element method. In this section, we give a brief introduction to the multimesh finite element method. To simplify the notation, we restrict ourselves to the case where at most two meshes may intersect at each point. We further assume that the j-th mesh will only intersect with the 0-th mesh, $j = 1, \dots, N$. More detailed information, including the case of an arbitrarily number of intersecting meshes can be found in [22] and references therein.

As a guiding example, consider the Poisson problem,

$$\begin{aligned} -\Delta T &= f && \text{in } \Omega, \\ T &= g && \text{on } \partial\Omega, \end{aligned} \tag{2.1}$$

where Ω is the problem domain, with boundary $\partial\Omega$.

We introduce a composition of Ω, such that $\Omega \subseteq \bigcup_{i=0}^{N} \hat{\Omega}_i$, where $\hat{\Omega}_i$ is defined as the i-th *predomain*. If a point $x \in \Omega$ is in multiple predomains, we associate it with the highest index i. Thus, if interpreted visually, the predomain with the higher index appears to be *on top* of the predomain with the lower index. Due to our assumptions, the j-th predomain will only overlap with the 0-th predomain for $j = 1, \dots, N$.

We define the *visible part* of $\hat{\Omega}_0$ as $\Omega_0 = \hat{\Omega}_0 \setminus \cup_{j=1}^{N} \hat{\Omega}_j$, and the *visible part* of $\hat{\Omega}_j$ as Ω_j, $j = 1, \dots, N$. We denote the boundary of the j-th visible domain as Λ_j. Note that Ω_0 is a function of the other predomains, which is crucial in the setting of shape optimization. An example composition for the domain is shown in Figure 2(a)-(c).

Using this domain composition, we can reformulate the single domain problem (2.1) into a multidomain problem. For that we define a function T_i on all visible parts Ω_i, $i = 0, \dots, N$. Then the multidomain problem is:

$$\begin{aligned} -\Delta T_i &= f && \text{in } \Omega_i, \\ T_i &= g && \text{on } \partial\Omega \cap \partial\hat{\Omega}_i, \\ T_j &= T_0 && \text{on } \Lambda_j, \\ DT_j n_j &= DT_0 n_j && \text{on } \Lambda_j, \end{aligned} \tag{2.2}$$

for $i = 0, \dots, N$ and $j = 1, \dots, N$. The normal vector n_j is pointing outwards of the domain Ω_j. The two interface conditions on Λ_j ensure sufficient smoothness of the solution across the interface.

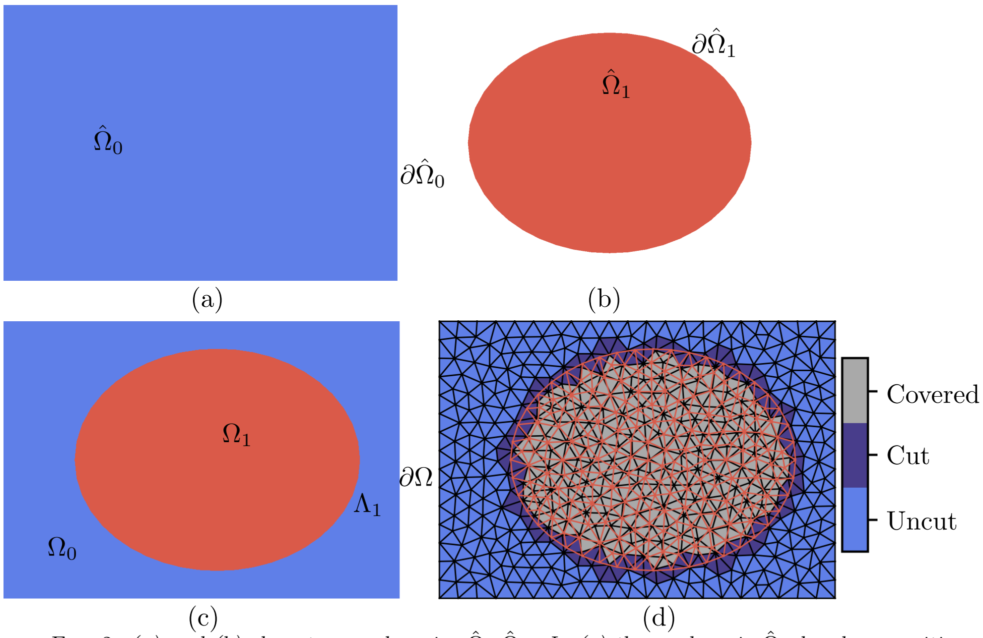


FIG. 2. *(a) and (b) show two predomains $\hat{\Omega}_0$,$\hat{\Omega}_1$. In (c) the predomain $\hat{\Omega}_1$ has been positioned on top of predomain $\hat{\Omega}_0$. The picture shows the resulting visible domains. In (d) we introduced the premeshes $\hat{\mathcal{K}}_{h,0}$ (black) and $\hat{\mathcal{K}}_{h,1}$ (red) of the predomains. The cell-types of the background mesh are visualized.*

Next, we discretize the computational domain. For that, we create a *premesh* $\hat{\mathcal{K}}_{h,i}$ of each predomain $\hat{\Omega}_i$, and denote its maximum cell diameter h_i. The elements of $\hat{\mathcal{K}}_{h,i}$ can be categorized as *uncut*, *cut* and *covered* elements. Uncut elements are the fully visible elements, cut elements are the partially visible elements, and covered elements are the hidden elements. The i-th *active mesh* $\mathcal{K}_{h,i}$ consists of all cut and uncut elements of $\hat{\mathcal{K}}_{h,i}$. We define the cut domain Ω_i^{cut} as the union of all cut elements. Note that $\Omega_N^{cut} = \emptyset$. The i-th *overlap* is defined as $\mathcal{O}_i := \Omega_i^{cut} \setminus \Omega_i$, $i = 0, \dots, N$. This is the hidden part of the active mesh. We define the visible part of the cut cells as $\mathcal{C}_i := \Omega_i^{cut} \setminus \mathcal{O}_i$. Figure 2(d) shows an example of premeshes and the classification of the cells on the background mesh.

2.1. The variational form for the multimesh finite element method. We can now formulate the multimesh variational formulation of problem (2.2). Let $V_{h,i}$, $i = 0, \dots, N$, be a continuous piece-wise polynomial finite element space on the active mesh $\mathcal{K}_{h,i}$. We define $V_h := \bigoplus_{i=0}^N V_{h,i}$. Let V_h^g denote the corresponding function space that satisfy the boundary condition. The multimesh finite element formulation for the Poisson problem is: Find $T = (T_0, \dots, T_N) \in V_h^g$ such that

$$a(T, v) + a_{IP}(T, v) + a_O(T, v) - l(v) = 0 \quad \forall v \in V_h^0, \tag{2.3}$$

where $v = (v_0, \dots, v_N)$. The volume terms for each visible domain are

$$a(T, v) := \sum_{i=0}^{N} \int_{\Omega_i} (\nabla T_i, \nabla v_i)\, \mathrm{d}x, \quad l(v) := \sum_{i=0}^{N} \int_{\Omega_i} (f, v_i)\, \mathrm{d}x. \tag{2.4}$$

Here $(\cdot,\cdot)$ denotes the Euclidean inner product. The symmetric interior penalty terms enforce the interface conditions of (2.2) weakly using a Nitsche method [33]:

$$a_{IP}(T,v) := \sum_{j=1}^{N} \int_{\Lambda_j} -(\langle DT\rangle n_j, [\![v]\!]) - ([\![T]\!], \langle Dv\rangle n_j) + \frac{\beta_0}{\langle h\rangle}([\![T]\!],[\![v]\!]) \,\mathrm{d}S, \tag{2.5}$$

where $\langle\psi\rangle = \frac{1}{2}(\psi_j + \psi_0)$ denotes the average, $[\![\psi]\!] = \psi_j - \psi_0$ denotes the jump, and $\beta_0 > 0$ is a sufficiently large penalty parameter. The overlap stability term is

$$a_O(T,v) := \sum_{i=0}^{N} \int_{\mathcal{O}_i} \beta_1([\![\nabla T]\!],[\![\nabla v]\!]) \,\mathrm{d}x, \tag{2.6}$$

where $\beta_1 > 0$ is needed to obtain a stable system even in cases where the mesh intersections become arbitrarily small.

This variational form is stable and well conditioned [22]. Since the interfaces Λ_j is not aligned with the meshes, custom quadrature rules are needed to perform the volume and interface integrals that appear in the formulation. We refer to [22] for details. A multimesh variational form for a Stokes equations can be found in [24].

2.2. Creation of holes with the multimesh FEM. It is often useful to embed obstacles in the computational domain. In the multimesh FEM this can be achieved by changing the status of visible elements to covered elements. This is exemplified in Figure 3. Since the covered cells are never removed from the mesh, the placement of holes can easily be changed. This is very convenient for shape optimization problems.

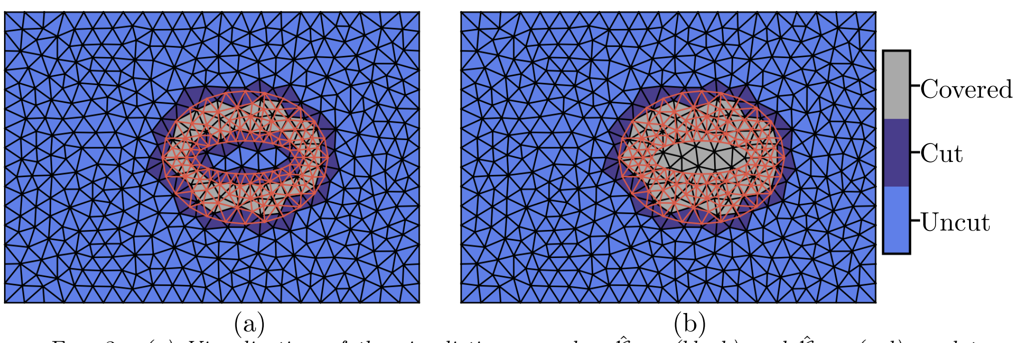


Fig. 3. *(a) Visualization of the simplistic premeshes $\hat{\mathcal{K}}_{h,0}$ (black) and $\hat{\mathcal{K}}_{h,1}$ (red) used to represent a channel with an obstacle. The initial uncut, cut and covered elements of $\hat{\mathcal{K}}_{h,0}$ are shown. (b) The element types after introducing a hole in the domain by setting all elements in $\hat{\mathcal{K}}_{h,0}$ that are cut or covered by the obstacle on $\hat{\Omega}_0$ to being covered. The boundary of the obstacle is now a physical boundary of $\hat{\mathcal{K}}_{h,1}$.*

3. Shape calculus for the multimesh finite element method. In this section, we derive the shape derivative for optimization problems constrained by multimesh models. We start by considering the necessary prerequisites for computing shape derivatives in general, and then derive the specific shape derivatives for multimesh problems. Given a domain Ω, we assume that we have the following shape optimization problem

$$\min_{\Omega} J(u,\Omega), \tag{3.1}$$

subject to

$$F(u,v)=0 \quad \forall v, \tag{3.2}$$

where $F(u,v)$ is the residual of the variational formulation of a PDE. The state u and test-function v are in the respective space of the variational PDE problem. They are assumed smooth enough for the shape differentiation to hold. In our application examples, we typically have u,v in $H^1(\Omega)$ with respective boundary conditions. We assume that (3.2) yields a unique solution u for any given domain Ω. We define the reduced functional $\hat{J}(\Omega) := J(u(\Omega),\Omega)$, and the perturbed domain as

$$\Omega(\epsilon)[s] := L_\epsilon[s](\Omega) = \{L_\epsilon[s](x) : x \in \Omega\}, \tag{3.3}$$

where $L_\epsilon[s](x) := x(\epsilon) := x + \epsilon s(x)$, $s(x) : \Omega \to \mathbb{R}^n, \epsilon > 0$. With these definitions, we define the shape derivative as

$$\mathrm{d}\hat{J}(\Omega)[s] := \lim_{\epsilon \to 0^+} \frac{\hat{J}(\Omega(\epsilon)[s]) - \hat{J}(\Omega)}{\epsilon}. \tag{3.4}$$

We will use the notation $u(\epsilon, x)$ to denote the evaluation of the PDE solution in the perturbed domain, that is $u(\Omega(\epsilon)[s])(x)$. We will further use the notation u to denote $u(0,\Omega(0)[s])$. The material and local shape derivatives of u are defined as

$$\delta_m\left(u(x(0))\right)[s] := \lim_{\epsilon\to 0^+} \frac{u(\epsilon, x(\epsilon)) - u(0,x(0))}{\epsilon}, \quad u'[s] := \delta_m(u)[s] - Dus, \tag{3.5}$$

where Du is the Jacobian. With these definitions, one can use the method of mappings [30, 37] to represent the shape derivative of the functional J as an integral over the unperturbed domain.

THEOREM 3.1 (The method of mappings). *For a general volume objective function $k : [0,\delta] \times \Omega(\epsilon)[s] \to \mathbb{R}$ with $\delta > 0$,*

$$K(\Omega(\epsilon)[s]) = \int\limits_{\Omega(\epsilon)[s]} k(\epsilon, x)\ \mathrm{d}x, \tag{3.6}$$

the shape derivative is given by

$$\mathrm{d}K(\Omega)[s] = \int\limits_{\Omega} div(s)\,k + \delta_m(k)[s]\ \mathrm{d}x, \tag{3.7}$$

Similarly, for a surface objective function $h : [0,\delta] \times \partial\Omega(\epsilon)[s] \to \mathbb{R}$

$$H(\partial\Omega(\epsilon)[s]) = \int\limits_{\partial\Omega(\epsilon)[s]} h(\epsilon, x)\ \mathrm{d}S, \tag{3.8}$$

the shape derivative is given by

$$\mathrm{d}H(\partial\Omega)[s] = \int\limits_{\partial\Omega} h(div(s) - n^T Dsn) + \delta_m(h)[s]\ \mathrm{d}S, \tag{3.9}$$

where n is the outwards pointing normal of $\partial\Omega$. Please note that we omit the ϵ argument when $\epsilon = 0$ fixed.

The method of mappings is discretely consistent. In other words, when the problem is discretized, the gradient computed with method of mappings is the exact gradient of the discretized problem.

Next we apply the method of mappings to the multimesh Poisson problem (2.3). Perturbing the j-th predomain $\hat{\Omega}_j$, implicitly change the integration domain Ω_0. We therefore consider each summand of (2.3) independently. Denoting the i-th summand as a_i, we have $a_0 := \int_{\Omega_0} (\nabla T_0, \nabla v_0)\ \mathrm{d}x$. Using Theorem 3.1 we obtain the shape derivative

$$
\begin{aligned}
\mathrm{d}a_0[s_j] = &\int_{\Omega_0} \mathrm{div}\,(\bar{s}_j)\,(\nabla T_0, \nabla v_0) - ((D\bar{s}_j)^T \nabla T_0, \nabla v_0) - (\nabla T_0, (D\bar{s}_j)^T \nabla v_0)\ \mathrm{d}x \\
&+ \int_{\Omega_0} (\nabla T_0, \nabla(\delta_m\,(v_0)\,[\bar{s}_j])) + (\nabla(\delta_m\,(T_0)\,[\bar{s}_j]), \nabla v_0)\ \mathrm{d}x,
\end{aligned}
\tag{3.10}
$$

where $\bar{s}_j$ is an extension of the movement of the domain Ω_j to Ω_0. Since we assume that Ω_j is not dependent of $\Omega_k, j \neq k, j, k = 1, \ldots, N$, we obtain the following shape derivative for $a_j = \int_{\Omega_j} (\nabla T_j, \nabla v_j)\ \mathrm{d}x$:

$$
\begin{aligned}
\mathrm{d}a_j[s_j] = &\int_{\Omega_j} \mathrm{div}\,(s_j)\,(\nabla T_j, \nabla v_j) - (\nabla T_j, (Ds_j)^T \nabla v_j) - ((Ds_j)^T \nabla T_j, \nabla v_j)\ \mathrm{d}x \\
&+ \int_{\Omega_j} (\nabla T_j, \nabla(\delta_m\,(v_j)\,[s_j])) + (\nabla(\delta_m\,(T_j)\,[s_j]), \nabla v_j)\ \mathrm{d}x.
\end{aligned}
\tag{3.11}
$$

Since $\mathcal{O}_i, i = 0, \ldots, N$ depends on the position $\hat{\Omega}_j,\ j = 1, \ldots, N$, we write each term in (2.6) as $a_{\mathcal{O}_j} := \int_{\mathcal{O}_j} \beta_1 [\![\nabla T]\!] : [\![\nabla \lambda]\!]\ \mathrm{d}x$. Using Theorem 3.1 we obtain the shape derivative

$$
\begin{aligned}
\mathrm{d}a_{\mathcal{O}_j}[s_j] = &\int_{\mathcal{O}_j} \beta_1 \mathrm{div}\,(\bar{s}_j)\,([\![\nabla T]\!], [\![\nabla \lambda]\!])\ \mathrm{d}x \\
&- \int_{\mathcal{O}_j} \beta_1((D\bar{s}_j)^T [\![\nabla T]\!], [\![\nabla \lambda]\!]) + \beta_1([\![\nabla T]\!], (D\bar{s}_j)^T [\![\nabla \lambda]\!])\ \mathrm{d}x \\
&+ \int_{\mathcal{O}_j} \beta_1([\![\nabla(\delta_m\,(T)\,[\bar{s}_j])]\!], [\![\nabla \lambda]\!]) + (\beta_1 [\![\nabla T]\!], [\![\nabla(\delta_m\,(\lambda)\,[\bar{s}_j])]\!])\ \mathrm{d}x.
\end{aligned}
\tag{3.12}
$$

Similarly, we can split the interior penalty terms (2.5) into N integrals, $a_{IP_j}, j = 1, \ldots, N$ with $a_{IP_j} = \int_{\Lambda_j} -(\langle DT \rangle n_j, [\![v]\!]) - ([\![T]\!], \langle Dv \rangle n_j) + \frac{\beta_0}{\langle h \rangle}([\![T]\!], [\![v]\!])\ \mathrm{d}S$ to obtain the shape derivative

$$
\begin{aligned}
\mathrm{d}a_{IP_j}[s_j] = \int\limits_{\Lambda_j} &(\operatorname{div}(s_j) - n_j^T D s_j n_j) \\
&- (\langle DT\rangle n_j, [\![v]\!]) - ([\![T]\!], \langle Dv\rangle n_j) + \frac{\beta_0}{\langle h\rangle}([\![T]\!], [\![v]\!]) \\
&+ ((\langle DT\rangle D s_j) n_j, [\![v]\!]) - (\langle DT\rangle \delta_m(n_j)[s_j], [\![v]\!]) \\
&+ ([\![T]\!], (\langle Dv\rangle D s_j) n_j) - ([\![T]\!], \langle Dv\rangle \delta_m(n_j)[s_j]) \\
&- \frac{\beta_0}{\langle h\rangle^2}\delta_m(\langle h\rangle)[s_j]([\![T]\!], [\![v]\!]) \\
&- (\langle D\delta_m(T)[s_j]\rangle n_j, [\![v]\!]) - (\langle DT\rangle n_j, [\![\delta_m(v)[s_j]]\!]) \\
&- ([\![\delta_m(T)[s_j]]\!], \langle Dv\rangle n_j) - ([\![T]\!], \langle D\delta_m(v)[s_j]\rangle n_j) \\
&+ \frac{\beta_0}{\langle h\rangle}([\![\delta_m(T)[s_j]]\!], [\![v]\!]) + \frac{\beta_0}{\langle h\rangle}([\![T]\!], [\![\delta_m(v)[s_j]]\!])\,\mathrm{d}S.
\end{aligned}
\tag{3.13}
$$

Let's study the extensions $\bar{s}_j$ in more detail. In order to evaluate the shape derivatives above, we need to evaluate and represent the smooth extension $\bar{s}_j$ on $\mathcal{K}_{h,0}$. Mesh deformations of the j-th mesh, $j > 0$ can be expressed as piece-wise continuous finite element functions. Hence it seems natural to represent $\bar{s}_j$ as a finite element function. As illustrated in Figure 4 the multimesh finite element function spaces are not rich enough to describe this movement. An alternative option is to resolve the interfaces between the meshes, would which however defeat the purpose of multimesh FEM. A third option is to approximate $\bar{s}_j$ as a finite element function on the background mesh, for instance with a projection scheme. Numerical experiments showed that the quality of the resulting shape derivative is poor.

For these reasons the method of mappings is not used for the multimesh FEM and the Hadamard formulation [46] is considered instead.

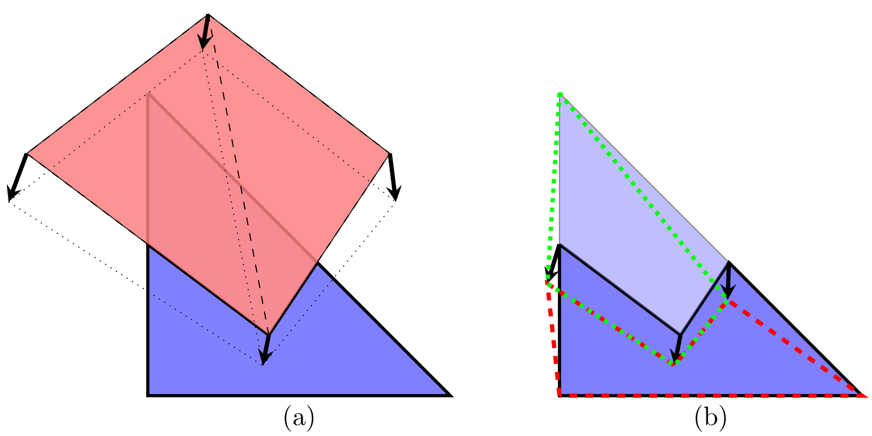


FIG. 4. *(a) A perturbation of the upper mesh with two elements (red) implicitly changes the visible integration domain of the bottom cell (blue). (b) The integration domains $\mathcal{O}_0$ (dashed green line) and $\mathcal{C}_0$ (dashed red line) after perturbing the top domain. Note that these changes of integration domains (black arrows) cannot be described by a finite element function on the background mesh.*

THEOREM 3.2 (The Hadamard formulation of the shape derivative). *For a general volume objective function $k : [0,\delta] \times \Omega(\epsilon)[s] \to \mathbb{R}$ where $\delta > 0$,*

$$K(\Omega(\epsilon)[s]) = \int\limits_{\Omega(\epsilon)[s]} k(\epsilon, x) \, \mathrm{d}x, \tag{3.14}$$

the shape derivative is given by

$$\mathrm{d}K(\Omega)[s] = \int\limits_{\partial\Omega} (n, s)k(x) \, \mathrm{d}S + \int\limits_{\Omega} k'[s] \, \mathrm{d}x. \tag{3.15}$$

Similarly, for a surface objective function $h : (\epsilon, \phi, \zeta) \to h(\epsilon, \phi, \zeta)$ involving the normal vector,

$$H(\partial\Omega(\epsilon)[s]) = \int\limits_{\partial\Omega(\epsilon)[s]} h(\epsilon, \phi(\epsilon, x), n(\epsilon, x)) \, \mathrm{d}S, \tag{3.16}$$

the shape derivative is given by

$$\mathrm{d}H(\partial\Omega))[s] = \int\limits_{\partial\Omega} (s, n) \left(\frac{\partial h}{\partial \phi} D\phi n + div_\Gamma \left(\frac{\partial h}{\partial \zeta} \right)^T + \kappa \left(h - \frac{\partial h}{\partial \zeta} n \right) \right) + \frac{\partial h}{\partial \phi} \phi'[s] \, \mathrm{d}S, \tag{3.17}$$

where $\frac{\partial h}{\partial \phi}, \frac{\partial h}{\partial \zeta}$ are the partial derivatives of h with respect to ϕ, ζ, respectively, $div_\Gamma(a) = div(a) - (n^T D a n)$ is the tangential divergence and κ is the additive mean curvature of $\partial\Omega$. The ϵ argument is omitted when $\epsilon = 0$.

Proof. The generalized Hadamard formulation with normal variation can be found in [41]. ∎

The Hadamard formulation alleviates the use of the projection $\bar{s}_j$ of movement s_j to Ω_0. However, as opposed to the method of mappings, the Hadamard formulas requires higher smoothness. The main drawback of the Hadamard formulation is that it is discretely inconsistent, which might slow down the convergence of the optimization algorithm. In subsection 5.2, we will investigate the impact of the discrete inconsistency. Using a sufficiently fine mesh, the Hadamard variational form converges to the discretely consistent gradient.

In order to derive the shape derivatives with the Hadamard formulation, we consider multidomain problem (2.2), where we have introduced an artificial interface with corresponding boundary conditions. For brevity, we consider T_i, $i = 0, \ldots, N$, to be scalar valued. In the following analysis, we will consider the functional $J(T) = \sum_{i=0}^{N} \int\limits_{\Omega_i} T_i^2 \, \mathrm{d}x$. We create the Lagrangian

$$\begin{aligned} \mathcal{L}(\Omega_0, \ldots, \Omega_N) := \sum_{i=0}^{N} \left(\int\limits_{\Omega_i} T_i^2 + \lambda_i(-\Delta T_i - f) \, \mathrm{d}x + \int\limits_{\partial\Omega \cap \partial\Omega_i} p_i(T_i - g) \, \mathrm{d}S \right) \\ + \sum_{j=1}^{N} \int\limits_{\Lambda_j} q_j(T_j - T_0) + w_j D(T_j - T_0) n_j \, \mathrm{d}S, \end{aligned} \tag{3.18}$$

where p_i, q_j and w_j are Lagrange multipliers that enforce the boundary conditions. Using Theorem 3.2 we obtain

(3.19)
$$
\begin{aligned}
\mathrm{d}\mathcal{L}(\Omega)[s] = \sum_{i=0}^{N} \Bigg(& \int_{\partial\Omega_i} (s, n_i)\Big(T_i^2 - \lambda_i \Delta T_i - \lambda_i f\Big) \\
&+ \int_{\Omega_i} 2T_i'[s]T_i - \lambda_i'[s]\Delta T_i - \lambda_i \Delta T_i'[s] - \lambda_i'[s] f - \lambda_i f'[s] \,\mathrm{d}x \\
&+ \int_{\partial\Omega\cap\partial\Omega_i} (s, n_i)\Big(\kappa p_i (T_i - g) + p_i \frac{\partial (T_i - g)}{\partial n_i} + \frac{\partial p_i}{\partial n_i}(T_i - g)\Big) \,\mathrm{d}S \\
&+ \int_{\partial\Omega\cap\partial\Omega_i} p_i'[s](T_i - g) + p_i T_i'[s] - p_i g'[s] \,\mathrm{d}S \Bigg) \\
&+ \sum_{j=1}^{N} \Bigg(\int_{\Lambda_j} (s, n_j)\Big(\kappa q_j (T_j - T_0) + q_j \frac{\partial (T_j - T_0)}{\partial n_j} + \frac{\partial q_j}{\partial n_j}(T_j - T_0)\Big) \,\mathrm{d}S \\
&+ \int_{\Lambda_j} (s, n_j)\Big(D(T_j - T_0) n_j D w_j n_j + w_j n_j^T D^2 (T_j - T_0) n_j\Big) \,\mathrm{d}S \\
&+ \int_{\Lambda_j} (s, n_j)\Big(\mathrm{div}_\Gamma \left(w_j \nabla (T_j - T_0)\right)\Big) + q_j'[s](T_j - T_0) + q_j T_j'[s] - q_j T_0'[s] \,\mathrm{d}S \\
&+ \int_{\Lambda_j} D(T_j - T_0) n_j w_j'[s] + w_j D((T_j - T_0)'[s] n_j) \,\mathrm{d}S \Bigg).
\end{aligned}
$$

To derive the Hadamard expression for surface integrals involving the normal from Theorem 3.1, a tubular extension of the normal is needed, for which we chose $Dnn = 0$. We observe that the Lagrangian above contains local shape derivatives T', λ', p', q', w', of both the state variable and the Lagrange multipliers. When these are assembled for all test functions, each local shape derivative is a dense matrix which is prohibited to compute. Instead, we use the adjoint method [21] to avoid explicit computations of these terms.

To obtain the adjoint equation we split $\partial\Omega_0$ into $N+1$ disjoint sets, namely $\partial\Omega\cap\partial\Omega_0, \Lambda_1, \ldots, \Lambda_N$. Similarly, $\partial\Omega_j$ can be split into two disjoint domains, $\partial\Omega\cap\partial\Omega_j$ and Λ_j for each $j = 1\ldots, N$. Carefully integrating the terms involving $\Delta T_i'[s]$ in

(3.19) by parts yields the following adjoint equation

$$
\begin{aligned}
0 = \quad & \sum_{i=0}^{N}\Bigg(\int\limits_{\Omega_i} 2T_i'[s]T_i - \lambda_i'[s]\Delta T_i - \Delta\lambda_i T_i'[s] - \lambda_i'[s]f \;\mathrm{d}x \\
& + \int\limits_{\partial\Omega\cap\partial\Omega_i} p_i'[s](T_i - g) + p_i T_i'[s] - \lambda_i \frac{\partial T_i'[s]}{\partial n_i} + \frac{\partial \lambda_i}{\partial n_i} T_i'[s] \;\mathrm{d}S\Bigg) \\
+ \quad & \sum_{j=1}^{N}\Bigg(\int\limits_{\Lambda_j} q_j'[s](T_j - T_0) + q_j T_j'[s] - q_j T_0'[s] \\
& + D(T_j - T_0)n_j w_j'[s] + w_j D(T_j'[s] - T_0'[s])n_j \\
& + \lambda_0 \frac{\partial T_0'[s]}{\partial n_j} - \frac{\partial \lambda_0}{\partial n_j} T_0'[s] \\
& - \lambda_j \frac{\partial T_j'[s]}{\partial n_j} + \frac{\partial \lambda_j}{\partial n_j} T_j'[s] \;\mathrm{d}S\Bigg)
\end{aligned}
\tag{3.20}
$$

The corresponding strong for of the adjoint equation(3.20) is

$$
\begin{aligned}
-\Delta\lambda_i &= -2T_i && \text{in } \Omega_i, \\
p_i &= -\frac{\partial\lambda_i}{\partial n_i}, \quad \lambda_i = 0, && \text{on } \partial\Omega\cap\partial\Omega_i, \\
\frac{\partial(\lambda_j - \lambda_0)}{\partial n_j} &= 0, \quad \lambda_j - \lambda_0 = 0, \quad w_j = \lambda_j, \quad q_i = -\frac{\partial\lambda_j}{\partial n_j} && \text{on } \Lambda_j,
\end{aligned}
\tag{3.21}
$$

where $i = 0, \ldots, N$ and $j = 1, \ldots, N$. Using the state (2.2) and adjoint equation (3.21), the shape derivative (3.19) can be simplified to

$$
\begin{aligned}
\mathrm{d}\mathcal{L}(\Omega)[s] = & \sum_{i=0}^{N}\Bigg(\int\limits_{\partial\Omega_i\cap\partial\Omega} (s, n_i)\Big(T_i^2 - \frac{\partial\lambda_i}{\partial n_i}\frac{\partial(T_i - g)}{\partial n_i}\Big) - p_i g'[s] \;\mathrm{d}S - \int\limits_{\Omega_i} \lambda_i f'[s] \;\mathrm{d}x\Bigg) \\
& + \sum_{j=1}^{N}\Bigg(\int\limits_{\Lambda_j} (s, n_j)\Big([\![T_j^2]\!] - \lambda_j\Delta(T_j - T_0) - \lambda_j[\![f]\!]\Big) \\
& + \int\limits_{\Lambda_j} (s, n_j)\Big(\lambda_j n_j^T D^2(T_j - T_0)n_j + \mathrm{div}_\Gamma\left(\lambda_j\nabla(T_j - T_0)\right)\Big) \;\mathrm{d}S\Bigg).
\end{aligned}
\tag{3.22}
$$

Since $\frac{\partial T_j - T_0}{\partial n_j} = 0$ on Λ_j, $\nabla(T_j - T_0) = \nabla_\Gamma(T_j - T_0)$ where ∇_Γ is the tangential gradient. Here, we can note that since $T_j = T_0$ on Λ_j, $\nabla_\Gamma(T_j - T_0) = 0$. We can therefore transform the last term in (3.22) to

(3.23)
$$
\begin{aligned}
\operatorname{div}_\Gamma\left(\lambda_j \nabla (T_j - T_0)\right) &= \operatorname{div}_\Gamma\left(\lambda_j \nabla_\Gamma (T_j - T_0)\right)\\
&= \lambda_j \operatorname{div}_\Gamma\left(\nabla_\Gamma (T_j - T_0)\right) + \nabla_\Gamma \lambda_j \nabla_\Gamma (T_j - T_0)\\
&= \lambda_j \Delta_\Gamma (T_j - T_0) + \nabla_\Gamma \lambda_j \nabla_\Gamma (T_j - T_0)\\
&= \lambda_j \Delta (T_j - T_0) - \lambda_j \kappa \frac{\partial (T_j - T_0)}{\partial n_j} - \lambda_j n_j^T D^2 (T_j - T_0) n_j)\\
&= \lambda_j \Delta (T_j - T_0) - \lambda_j n_j^T D^2 (T_j - T_0) n_j).
\end{aligned}
$$

In addition to (3.23), we have that $T_j^2 = T_0^2$ on Λ_j since $T_j = T_0$ on Λ_j. Thus, for the Λ_j-terms in (3.22), the only term remaining is the jump of the source term f across the interface Λ_j.

If f is continuous at the interface Λ_j then the internal multidomain interface Λ_j does not contribute to the shape derivative. In addition, if the right hand side of the Dirichlet condition g is "moving along" with the deformation then $g'[s] = -Dgs$ [7]. Thus if g is constant on each boundary and f is a function fixed to the computational domain we obtain

$$
\mathrm{d}\mathcal{L}(\Omega)[s] = \mathrm{d}J(\Omega)[s] = \sum_{i=0}^{N} \int\limits_{\partial\Omega_i \cap \partial\Omega} (s, n_i)\Big(T_i^2 - \frac{\partial \lambda_i}{\partial n_i}\frac{\partial T_i}{\partial n_i}\Big)\,\mathrm{d}S. \tag{3.24}
$$

We realize that this gradient is equivalent to the traditional shape derivative for a Poisson problem. This result also holds for arbitrary many overlapping meshes since one has the same interface conditions.

4. Optimization algorithm and mesh deformation.

In general, we would like to use the shape sensitivity of the functional to update the domain. At iteration k, we have the domain Ω^k. The functional sensitivity at the current iterand is denoted $\mathrm{d}J(\Omega^k)$. The discretized domain used in the next iteration, will be written as

$$
\Omega^{k+1} = \mathcal{F}(\,\mathrm{d}J(\Omega^k), \xi), \tag{4.1}
$$

where $\mathcal{F}$ represents an optimization strategy with step-length ξ. For a steepest descent algorithm, we can write

$$
\Omega^{k+1} = \Omega^k(\xi)[-\mathcal{R}(\,\mathrm{d}J(\Omega^k))], \tag{4.2}
$$

where $\mathcal{R}(\cdot)$ is a Riesz representation of the shape derivative.

The choice of the Riesz representer is important to retain a high mesh quality during the optimization process. The $H^1(\partial\Omega_i)$ Riesz representer would be natural, since the i-th term of the shape derivative of (3.24) is $\int\limits_{\partial\Omega_i} (s_i, n_i) g_i(x)\,\mathrm{d}S$. However, such a Riesz representer only deform the boundary mesh nodes, and hence quickly result in degenerated meshes. Hence a Riesz representation which extends into the volume is needed. Since a $H^1(\Omega_i)$ representation often results in compression effects, we consider an approach adapted from [45].

As in [45], we use the elasticity equations to represent the mesh deformation,

$$
\begin{aligned}
&\operatorname{div}(\sigma) = 0 \text{ in } \Omega_j,\\
&\frac{\partial r_j}{\partial n_j} = \begin{cases} g_j(x) \text{ on } \partial\Omega_j \cap \partial\Omega,\\ 0 \text{ on } \Lambda_j,\end{cases}
\end{aligned} \tag{4.3}
$$

where the solution r_j, $j = 1, \dots, N$ is used a replacement for the Riesz representer in (4.1), and

$$\sigma(r_j) = \lambda_{elas}\mathrm{Tr}(\epsilon(r_j)) + 2\mu_{elas}\epsilon(r_j), \tag{4.4}$$
$$\epsilon(r_j) = \frac{1}{2}(\nabla r_j + \nabla r_j^T).$$

For our numerical experiments, we use $\lambda_{elas} = 0$, $\mu_{elas} = 400$. In the traditional finite element method, a homogeneous Dirichlet condition is often imposed on the outer boundaries of the domain. However, with the use of multiple domains, $\hat{\Omega}_j$, $j = 1, \dots, N$, we do not need to impose Dirichlet conditions on the boundary Λ_j. Therefore, we impose a no-stress condition at the interfaces Λ_j. Also in contrast to [45], we choose μ_{elas} to be constant. To obtain a unique solution of (4.3), we have to remove rigid motions from the solution space, as they are in the null-space of the operator. We can write our deformation formulation as, find $r_j \in H^1(\hat{\Omega}_j), j = 1, \dots, N$ such that

$$\int_{\hat{\Omega}_j} (\sigma(r_j), \epsilon(s_j))\ \mathrm{d}x = \int_{\partial\hat{\Omega}_j \cap \partial\Omega} (s_j, n_j) g_j(x)\ \mathrm{d}S \quad \forall s_j \in H^1(\hat{\Omega}_j). \tag{4.5}$$

In subsection 5.3, we show that this method yields good mesh quality properties for large deformations. Also, since we do not employ this algorithm on the full computational domain Ω, but on the subdomains $\hat{\Omega}_i$, this method is computationally cheaper than traditional mesh deformation.

As opposed to deforming the computational domain, one could use re-meshing as an approach to update the computational domain. Re-meshing adds a similar discrete inconsistency as using the Hadamard formula, as the new positioning of interior cells are arbitrary. We have not employed the method of re-meshing in this article. However, note that by employing the multimesh FEM approach, meshes can be re-meshed independent of each other, possibly saving some computational effort.

A common case in practical problems is that the domains $\hat{\Omega}_j$, $j = 1, \dots, N$ are parameterized by its position and rotational angle, see Figure 1. Using the chain rule, we obtain the shape sensitivities of the centroid c_j as

$$\frac{\mathrm{d}J(\hat{\Omega}_j(c_j))}{\mathrm{d}c_j} = \mathrm{d}J(\hat{\Omega}_j)\left[\frac{\mathrm{d}\hat{\Omega}_j}{\mathrm{d}c_j}\right], \tag{4.6}$$

where $\frac{\mathrm{d}\hat{\Omega}_j}{\mathrm{d}c_j} = (e_1, e_2)$ where e_k is the k-th unit vector in $2D$. For the multimesh FEM this approach does not require any deformation, since the gradient corresponds to a translation of the j-th mesh.

Similarly, by parameterizing the j-th domain with respect to rotation θ_j around the point p_j, then

$$\frac{\mathrm{d}J(\hat{\Omega}_j)}{\mathrm{d}\theta_j} = \mathrm{d}J(\hat{\Omega}_j)\left[\frac{\mathrm{d}\hat{\Omega}_j}{\mathrm{d}\theta_j}\right], \tag{4.7}$$

where $\frac{\mathrm{d}\hat{\Omega}_j}{\mathrm{d}c_j} = (-y + p_j|_0, x - p_j|_1)$ is the first order approximation of rotation around the point p_j in $2D$. As for the case of parameterizing by the position of the meshes, the rotation parameterization alliviates the need for mesh deformation when using multimesh FEM, as one simply can rotate the j-th mesh around the point p_j.

5. Numerical examples.

This section discusses three numerical examples to demonstrate different aspects of multimesh FEM shape optimization. We optimize both unparameterized as well as parameterized shapes, such as the position and angle of objects as discussed in Section 4. We further investigate the impact of the Hadamard formulation in the accuracy of the shape derivative, and compare the performance of the multimesh strategy against traditional shape optimization.

5.1. Implementation.

The numerical experiments were implemented using the FEniCS project [1, 26], version 2018.1.0. Details on the multimesh implementation in FEniCS can be found in [22]. For this paper, we implemented additional FEniCS functionality that allows for automatically marking holes in domains (see subsection 2.2) as well as extending FEniCS' Python interface. These features were also released as part of FEniCS 2018.1.0. Since the current version of multimesh FEM in FEniCS does not support parallel execution, all experiments were performed on a single core. The meshes in this section were generated with GMSH, version 3.0.6 [17], the Python interfaces pygmsh, version 4.3.6 [39] and meshio, version 2.3.3 [40]. The implementation of the examples and installation instructions are available at https://github.com/jorgensd/MultiMeshShapeOpt_code.

5.2. Optimization of Current Carrying Multi-cables.

An important category of shape-optimization are problems where the position of individual objects are to be optimized [16, 19, 29]. In this section, we investigate such an example, namely, the design optimization of a multi-cable. The basic construction of a multi cable consists of a bundle of individual cables surrounded by a single outer jacket, as shown in Figure 5. A critical design goal of multi-cables is to position the internal cables to minimize the risk of overheating.

This multi-cable design problem has been formulated as a mathematical optimization problem in [19], where the design variables are the positions of each internal cable of the multi-cable. Since, each optimization iteration results in new cable positions, a re-meshing strategy was used to update the mesh to ensure that the internal cable boundaries are always resolved by the mesh. As we will see in this example, multimesh FEM allows to completely avoid re-meshing by describing each internal cable by a separate mesh.

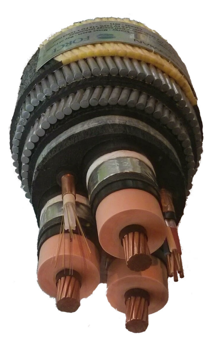
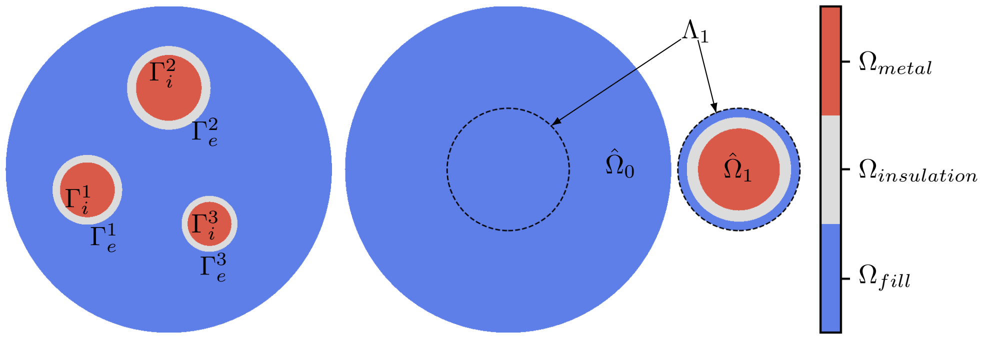


Fig. 5. *A current carrying multi-cable as studied in subsection 5.2.*

Motivated by [19, 27], we consider the multi-cable problem:

$$\min_{c_1,\dots,c_N,T} \int_\Omega \frac{1}{q}|T|^q \,\mathrm{d}x, \quad q > 1, \tag{5.1}$$



(a) (b)

FIG. 6. *(a) Illustration of the material composition of a multi-cable with annotated boundaries. (b) Illustration of how an internal cable is represented by a separate domain. Every domain includes an extra halo surrounding the cable.*

subject to

$$
\begin{aligned}
-\nabla \cdot (\lambda \nabla T) - 0.01T &= f \text{ in } \Omega, \\
\lambda \frac{\partial T}{\partial n} + (T - T^{\text{ex}}) &= 0 \text{ on } \partial\Omega,
\end{aligned} \tag{5.2}
$$

where $\Omega = \Omega_{fill} \cup \Omega_{insulation} \cup \Omega_{metal}$ describes a 2D slice through the multi-cable with N internal cables, as specified in Figure 6 (a). We define the normal vector n as the vector pointing in the outwards radial direction of each internal cable. The internal interface between the fill and insulation material of the k-th internal cable is denoted by Γ_e^k. Similarly, Γ_i^k denotes the interface between insulation and metal. The centroid of the k-th cable is denoted as c_k. The source-term f and heat-conductivity λ are constant in each material but discontinuous across the material boundaries. Therefore, these terms are dependent on the optimization variables $c_j, j = 1, \dots, N$. The linear source term in the state equation describes the rise of electrical resistivity for increasing temperatures in conductive material. The external boundary condition is a Robin-condition, related to the air surrounding the outer jacket, with temperature $T^{\text{ex}} = 3.2$. Furthermore, we set $q = 3$ to approximate the L^∞ norm, as done in [19]. Due to the discontinuities in f and λ, the temperature profile T is continuous but has kinks across the interface of the different materials. These kinks are important for the derivation of the Hadamard representation of the shape gradient [19]. Additional constraints must be added to (5.2) in order to avoid movement of internal cables outside the outer jacket and overlaps of internal cables.

For the multimesh FEM formulation, we chose to represent the domain Ω by one mesh for the outer jacket, and N meshes for the internal cables, as shown in Figure 6 (b). Following the strategy laid out in Section 2, we obtain the multidomain formulation of (5.1) and (5.2):

$$
\min_{c_1,\dots,c_N,T} J(c_1, \dots, c_N, T) = \sum_{i=0}^{N} \int_{\Omega_i} \frac{1}{q} |T_i|^q \, \mathrm{d}x, \tag{5.3}
$$

subject to

$$
\begin{aligned}
-\nabla\cdot(\lambda\nabla T_i) - 0.01T_i &= f \text{ in } \Omega_i, i=0,\dots,N,\\
\lambda\frac{\partial T_0}{\partial n} + (T_0 - T^{\text{ex}}) &= 0 \text{ on } \partial\Omega,\\
[\![T]\!] = \left[\!\!\left[\lambda\frac{\partial T}{\partial n_j}\right]\!\!\right] &= 0 \text{ on } \Lambda_j, j=1,\dots,N.
\end{aligned}
\tag{5.4}
$$

Note that the meshes for the internal cables, $\Omega_1,\dots,\Omega_N$ include a halo of filling material, which is sufficiently large so that the heat conductivity λ is constant over the cells categorized as overlapped. As a result, the derivation of the multimesh variational form of (5.4) is the same as in subsection 2.1. In the numerical test, we used continuous, piecewise linear finite elements and the penalty parameters in the interior penalty and overlap terms, (2.5) and (2.6), were set to $\beta_0=\beta_1=4$.

In the original problem formulation (5.1) and (5.2), the optimization variables $c_j, j=1,\dots,N$ appeared the in the source f and the heat-conductivity λ. In contrast, in the multimesh formulation (5.3) and (5.4), the optimization variables appear as a dependency of the sub-domains, $\Omega_0(c_1,\dots,c_N)$ and $\Omega_1(c_1),\dots,\Omega_N(c_N)$. This enables us to applying the Hadamard shape analysis as presented in Section 3, which results in the shape derivative

$$
\begin{aligned}
\mathrm{d}J(\Omega)[s] = \sum_{j=1}^{N}\int_{\Gamma_i^j\cup\Gamma_e^j}(s,n)\Big([\![-0.01Tp - fp]\!] - \lambda^+\frac{\partial p^+}{\partial n}\left[\!\!\left[\frac{\partial T}{\partial n}\right]\!\!\right]\\
+ [\![\lambda]\!](\nabla_\Gamma p^+, \nabla_\Gamma T^+)\Big)\,\mathrm{d}S.
\end{aligned}
\tag{5.5}
$$

Here, the super-script $+$ denotes the evaluation of a function from the fill side at Γ_e^j, and evaluation at the insulation side of Γ_i^j, $[\![\cdot]\!]$ denotes the jump over the interface Γ_i^j or Γ_e^j from the external side of the interface, and p is the solution of the adjoint equations of (5.3) and (5.4):

$$
\begin{aligned}
-\nabla\cdot(\lambda\nabla p_i) - 0.01p_i &= -T_i|T_i|^{q-2} && \text{in } \Omega_i,\\
\lambda\frac{\partial p_0}{\partial n_0} + p_0 &= 0 && \text{on } \partial\Omega,\\
[\![p]\!] = [\![\lambda\frac{\partial p}{\partial n_j}]\!] &= 0 && \text{on } \Lambda_j,
\end{aligned}
\tag{5.6}
$$

where $i=0,\dots,N$ and $j=1,\dots,N$.

5.2.1. Results. First, the adjoint equation and shape derivative were verified using a Taylor test. The test was performed on a multimesh with radius 1.2 and one internal cable placed at $(0,0.1)$ with 0.2 radius plus a 0.055 thick insulation. For the source term f and the heat diffusivity, we used the parameters:

	Ω_{fill}	$\Omega_{insulation}$	Ω_{metal}
f	0.0	0.0	10.0
λ	0.33	0.03	205.0

The convergence rates for the first order residual for different mesh resolutions are shown in Figure 7. We observe that the discrete inconsistencies of the Hadamard formulas are present on coarse meshes, which results in a decreased convergence rate for smaller perturbations. For finer meshes, the discrete inconsistency decreases. The

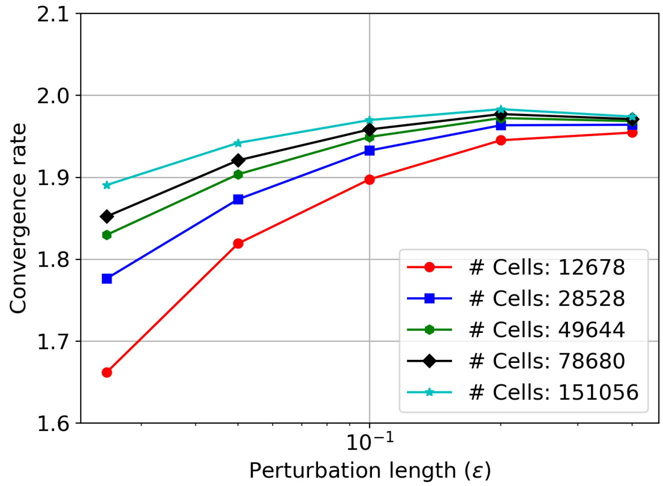


FIG. 7. *Results of the Taylor test of a multi-cable with a single internal cable placed at c_1, as described in subsection 5.2.1. The plot shows the convergence rates of the the first order residual $|J(c_1 + \epsilon s) - J(c_1) - \epsilon \, \mathrm{d}J(c_1)[s]|$ in direction $s = [0, 1]^T$ for different meshes and purturbation sizes ϵ. We observe that the expected convergence rate 2.0 is obtained on fine meshes.*

same behavior was also observed in [19]. Based on these results, we use the mesh with 151,056 cells for the following experiments.

Next, we test the optimization algorithm on a setup with a known optimal solution. For a multi-cable with three identical internal cables the heat in the domain is minimized when the cables are placed as far from each other as possible. Therefore, the optimal positions of the internal cables form an equilateral triangle [19]. Since the problem suffers from rotational symmetry, we fix the x-position of one of the cables on the y-axis. The initial guess is depicted in Figure 8(a). IPOPT [56] terminated with the default stopping criteria after 16 iterations. The functional has decreased from 180.4 to 135.4, and the optimized cable positions form a triangle with angles 59.94, 60.00 and 60.06 degrees which is in agreement with [19]. The final positioning is visualized in Figure 8(b).

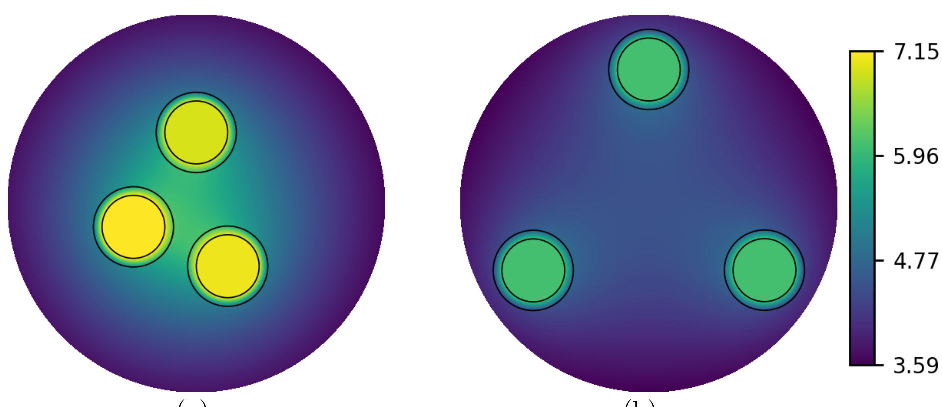
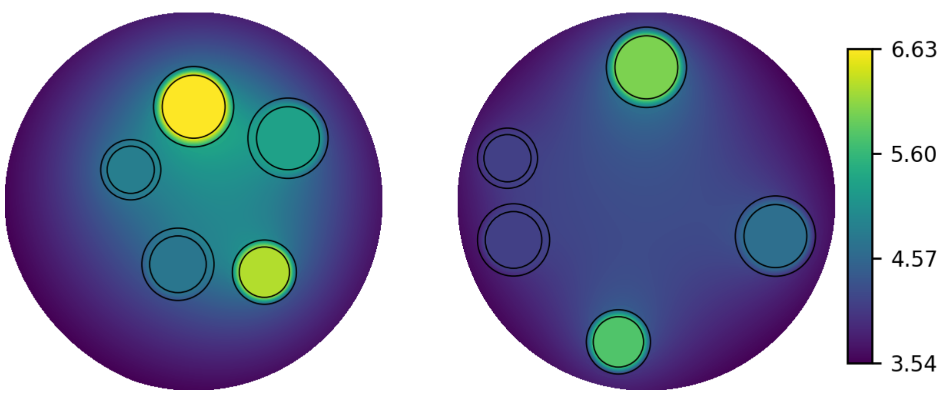


FIG. 8. *Design optimization of a multi-cable with three internal cables with common sizes and material parameters, as described in subsection 5.2.1. (a) The cable cable positions and temperature distribution before the optimization. (b) The cable cable positions and temperature distribution after the optimization. The inner cables form an equilateral triangle.*

We also considered the same minimization problem with five internal cables of different sizes and insulation parameters, as listed in Table 1. The initial and optimized cable configurations are shown in Figure 9. The IPOPT algorithm terminated after 22 iterations, when the functional decreased from 152 to 140.

TABLE 1
The setup for the 5 multi-cable optimization shown in Figure 9. The parameters $\lambda_{fill}, \lambda_{metal}$, are the same as for Figure 8. The scaling r_{scale} is the relative scale of the cables compared to those used in Figure 8.

	Cable 1	Cable 2	Cable 3	Cable 4	Cable 5
Init. Positions	$0, 0.6$	$-0.4, 0.2$	$-0.1, -0.4$	$0.6, 0.4$	$0.45, -0.45$
Opt. Positions	$0, 0.85$	$-0.88, 0.26$	$-0.85, -0.25$	$0.82, -0.22$	$-0.18, -0.89$
r_{scale}	1	0.75	0.9	1	0.8
λ_{iso}	0.03	0.12	0.06	0.04	0.02
f	10	5	2.5	5	10



(a) (b)

FIG. 9. *Design optimization of a multi-cable with five internal cables with different sizes and material parameters, as described in subsection 5.2.1. (a) The cable cable positions and temperature distribution before the optimization. (b) The cable cable positions and temperature distribution before the optimization. The smallest cable is placed as far away from the other cables since it has the lowest insulation and highest heat source.*

Finally, we compared the computational expense of the multimesh shape optimization approach against a traditional shape optimization strategy. For that, we implemented a solver for the multi-cable problem (5.2) and its gradient (5.5) using the traditional (single-mesh) FEM with FEniCS and benchmarked the problem with three identical internal cables, see Figure 8. The mesh for the single-mesh setup was created such that the total number of cells is similar to the total number of cells in the multimesh setup. The number of active cells (cut and uncut cells) in the multimesh was $227,746$ and $246,176$ if the covered cells are included. The single mesh had $211,008$ cells. At every optimization step, we re-meshed the domain to resolve the boundary of the internal cables. A more advanced setup could combine re-meshing with mesh-deformation techniques, but this was disregarded for simplicity. The optimization process was manually terminated after 16 iterations. Without manual termination, the re-meshing would eventually fail after 35 iterations due to an internal cable moving outside the outer jacket. The angles between the optimized cable positions were $59.84, 60.26$ and 59.91 degrees.

The computational costs of the multimesh and traditional FEM approaches are contrasted in Table 2. Each optimization iteration typically consists of assembling and solving one state equation and one adjoint equation, followed by a mesh update and a mesh building step. In the traditional FEM approach, the mesh update consists of triangulating the domain, while the build step prepares and converts the mesh data between the mesh generator and the finite element solver, as implemented in pygmsh. In the multimesh approach, the mesh update changes the mesh coordinates, while the build step determines the cut and uncut cells, computes intersections of cutting cells and create corresponding quadrature rules, see [22] for details.

The timings show that the assembly of the multimesh system is slower than with traditional FEM, primarily caused by the additional overlap and interior integrals in the multimesh variational form. The resulting linear systems were solved using a direct LU decomposition, and no significant differences in time was observed. In contrast, the mesh update and building steps differ significantly between the two approaches. When combining both steps the multimesh FEM is about 48 times faster than the traditional FEM approach. Overall, the estimated runtime for a single optimization iteration for the multimesh FEM approach ($2,530$ ms) is about five times faster compared to the traditional FEM approach ($12,895$ ms).

It should be noted that these benchmark runtimes can likely be improved. For instance, the traditional FEM approach does not require an expensive re-mesh step at every optimization iteration. Instead, a common strategy is to deform the domain with respect to a deformation equation as described Section 4. However, even a simple deformation equations, such as computing a smoothed H^1-Riesz representation, will have approximately the same assembly and solve time as the state equation of this problem. Thus, assuming the same assembly and solve time, the runtime of a single iteration with the traditional FEM would be $3,232ms$. Since this Riesz representation does not preserve mesh quality, re-meshing would still be required after every few optimization iterations.

TABLE 2

The timing results of the traditional FEM versus multimesh FEM, as described in subsection 5.2.1. The table states the average time that different operations (assembly of linear systems, solver time using a LU decomposition, mesh update and build steps, and one optimization iteration) took during optimizing a multi-cable with three identical internal cables.

	Cells	Assembly	Solve	Mesh Update	Build	Opt iter.
MultiMesh FEM	22,7746	406 ms	749 ms	0.94 ms	222 ms	2,530 ms
Traditional FEM	21,1008	270 ms	807 ms	6,368 ms	4,372 ms	12,895 ms

5.3. Shape Optimization of an Obstacle in Stokes Flow. This example considers the drag minimization of an object subject to a Stokes flow in two dimensions. In contrast to the previous example, the shape to be optimized is here not parameterized. This problem has a known analytical solution consisting of a rugby-ball shaped object, which was first presented in [36]. The drag is measured by the dissipation of kinetic energy into heat, that is

$$J_S(\Omega, u) = \int_\Omega \sum_{i,j=1}^{2} \left(\frac{\partial u_i}{\partial x_j} \right)^2 \,\mathrm{d}x, \tag{5.7}$$

where $\Omega = [0,1]^2$ is the computational domain, u is the velocity vector and $\frac{\partial u_i}{\partial x_j}$ denotes the derivative of the i-th velocity component in the j-th direction. The trivial solution to this problem would be to remove the object from the Stokes-flow completely. This is avoided by introducing additional constraints on the area and centroid of the obstacle. Denoting the target centroid of the obstacle as $(c_{x0}, c_{y0}) = (0.5, 0.5)$ and the target area as $V_O = 0.047$, we enforce these constraints with quadratic penalty terms. This yields the cost functional

$$
\begin{aligned}
J(\Omega, u) &= J_S(\Omega, u) + J_V(\Omega) + J_{Cx}(\Omega) + J_{Cy}(\Omega),\\
J_V(\Omega) &= \gamma_1 \left(V(\Omega) - V_0\right)^2,\\
J_C(\Omega) &= \gamma_2 \left((c_x - c_{x0})^2 + (c_y - c_{y0})^2\right),
\end{aligned}
\tag{5.8}
$$

with penalty parameters $\gamma_1 > 0$ and $\gamma_2 > 0$ and actual area of the object $V(\Omega)$. The actual object area can be computed as $V = 1 - \int_\Omega 1 \, \mathrm{d}x$, and the coordinate of the obstacle's centroid with $c_x = (0.5 - \int\limits_\Omega x \, \mathrm{d}x)/V$ and $c_y = (0.5 - \int\limits_\Omega y \, \mathrm{d}x)/V$.

The complete shape optimization problem is then:

$$
\min_{\Omega, u} J(\Omega, u)
\tag{5.9}
$$

subject to

$$
\begin{aligned}
-\Delta u + \nabla p &= 0 && \text{in } \Omega,\\
\nabla \cdot u &= 0 && \text{in } \Omega,\\
u &= 0 && \text{on } \Gamma_2,\\
u &= u_0 && \text{on } \Gamma_1 \cup \Gamma_3,\\
\frac{\partial u}{\partial n} + pn &= 0 && \text{on } \Gamma_4,
\end{aligned}
\tag{5.10}
$$

where p is the fluid pressure, u_0 a prescribed boundary velocity. Further, Γ_1 is the left boundary, Γ_2 is the boundary of the obstacle, Γ_3 is the top and bottom boundary and Γ_4 is the right boundary.

The multimesh variational formulation of the Stokes equations for two overlapping domains has been derived and analyzed in [24]. We used this formulation in our experiments with the penalty value $\beta = 6$. The system was discretized using the Taylor-Hood element pair, that is second order piece-wise continuous polynomials for the velocity and first order piece-wise continuous polynomials for the pressure. The arising linear systems were solved using the direct solver MUMPS [2], which is sufficient for the problem sizes considered. For finer discretizations, the options of using iterative solver should be explored.

The Hadamard formulation of the shape sensitivity of J_S has been derived in [28] and is

$$
\mathrm{d}J_S(\Omega, u, p)[s] = \int\limits_{\Gamma_2} -(s, n) \left(\frac{\partial u}{\partial n}, \frac{\partial u}{\partial n}\right) \, \mathrm{d}S.
\tag{5.11}
$$

The shape sensitivity of J_V and J_S is obtained by applying the product rule and quotient rule, respectively, and then Theorem 3.2:

$$
\mathrm{d}J_V(\Omega)[s] = -2\gamma_1 (V(\Omega) - V_0) \int\limits_{\Gamma_2} (s, n) \, \mathrm{d}S.
\tag{5.12}
$$

$$\text{(5.13)} \qquad \mathrm{d}J_{Cx}(\Omega)[s] = 2\gamma_2(1 - V(\Omega))^{-1}(c_x - c_{x0}) \int_{\Gamma_2} (s, n)(c_x - x)\ \mathrm{d}S.$$

More details can be found in [45]. Similar result can be derived for $\mathrm{d}J_{Cy}$. Combining (5.11)–(5.13) and obtain the shape sensitivity

(5.14)

$$\mathrm{d}J(\Omega, u, p)[s] = \int_{\Gamma_2} (s, n)\Bigg(- \left(\frac{\partial u}{\partial n}, \frac{\partial u}{\partial n}\right) - 2\gamma_1(V(\Omega) - V_0) + 2\gamma_2(1 - V(\Omega))^{-1}\Big[(c_x - x)(c_x - c_{x0}) + (c_y - y)(c_y - c_{y0})\Big]\Bigg)\ \mathrm{d}S.$$

We note that (5.14) does not depend on the adjoint solution. This is due to the fact that with the given functional, the adjoint solution λ can be expressed through the state variable u, see for instance [41].

5.3.1. Results. We decided to describe the domain using two meshes: one fixed background covering the domain $[0, 1]^2$ and one top mesh that represents the obstacle. This is visualized in Figure 11(a). Similar to [6], the top mesh has a circular geometry with a front and back wedge. To create the hole to represent the flow obstructing object, the background cells inside the hole of the top mesh were marked as inactive, as described in subsection 2.2.

The steepest descent method with an Armijo linesearch was employed as optimization algorithm. The mesh deformation was preformed using (4.5). To ensure that the volume and centroid constraints are sufficiently satisfied, we increased the penalty coefficients γ_1 and γ_2 every $8th$ iteration, starting with $\gamma_1 = \gamma_2 = 5 \cdot 10^4$.

Figure 10 visualizes the initial mesh and the mesh after 24 iterations and the velocity magnitude. The solution inside the object is set 0, since the associated element are marked as inactive. During the optimization, the functional reduced from initially 21.5 to 18.2. The final volume were 2.29% smaller less than the desired volume and the offsets in the barycenter were 0.005% and 0.000004%. Note that the front mesh contains much fewer elements $(2, 545)$ than the background mesh $(8, 223)$. The deformation scheme is only solved on the top mesh, and hence significantly more efficient than if the entire domain had to be deformed.

Figure 11 shows close-ups of the top mesh after 0, 8, 16 and 24 optimization iterations. The shape of the object after 8 iterations is visually in agreement with the results published in [6, 10]. After iteration 8, the volume and barycenter penalty increases, but causes only minor changes to the geometry. This figure also indicates that the scheme conserves mesh quality during the optimization. Indeed, the initial top mesh has a maximum element radius ratio of 1.57, while the top mesh after 24 iterations has nearly identical a maximum element radius ration of 1.53. The reason why the mesh quality can be conserved, is because the movement of the (physical) boundary of the top mesh is well transferred to the outer (non-physical) boundary of the top mesh.

5.4. Orientation of 9 objects in Stokes-flow. As a final example, we considered the problem of optimally rotating nine obstacles in Stokes flow to minimize dissipation of energy. This time, we parameterize the domain, a channel with 9 obstacles, through the angles of the obstacles, as shown in Section 4. We consider 9 identical objects placed in a structured fashion, as shown in Figure 12(a), with two inlets on

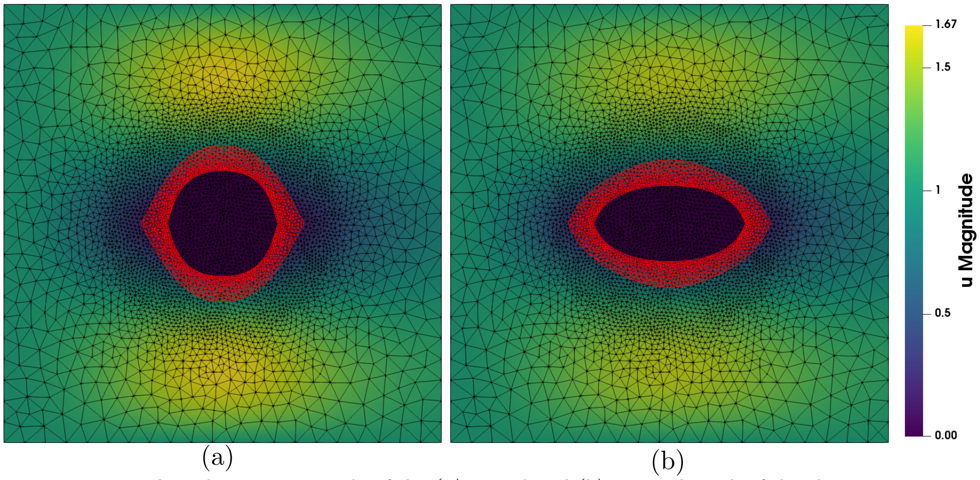


Fig. 10. *The velocity magnitude of the (a) initial and (b) optimal mesh of the shape optimization of an obstacle in stokes flow, see* subsection 5.3. *Notice that the number of cells on the front mesh* $(2,545)$ *is considerably less than the background mesh* $(8,223)$. *Thus deformation of the top domain is not as computationally expensive as deforming the full domain in a traditional finite element method with similar mesh size.*

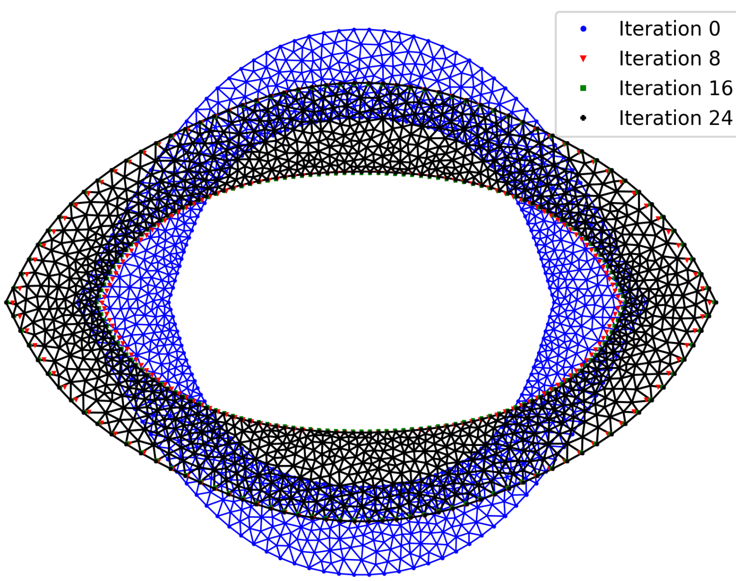


Fig. 11. *The initial mesh describing the obstacle is compared to the mesh after* 8, 16 *and* 24 *iterations. The volume and barycenter penalization factor was doubled every eighth iteration. We observe that increasing the volume and barycenter penalization only creates minor changes in the geometry. The deformed mesh does not experience distortion in the same way as with a traditional mesh, as the outer boundaries are not subject to a homogeneous Dirichlet Condition, but are free to deform.*

the left wall of the domain, with different sizes and inlet profiles, and one outlet on the top right of the domain. Using the chain rule, we get that functional sensitivity with respect to the j-th rotation angle is $\frac{\mathrm{d}J}{\mathrm{d}\theta^j} = \mathrm{d}J(\Omega)[\frac{\partial\Omega}{\partial s_\theta^j}]$, where $s_\theta^j = (-y + c_y^j, x + c_x^j)$ is the first order approximation of the rotation vector.

The optimization was performed using a multimesh consisting of a total of 10 meshes, where each obstacle was represented by a separate mesh. The number of cells in the background mesh was $33,283$ and in each front mesh $1,900$. Using Scipy [35] and its Newton-CG method, we optimized the angles of the nine obstacles. The stopping criterion was that the average change in the angle of the obstacle was less than 0.1 degrees. This criterion was reached after 18 iterations, when the functional had decreased to 5.59 from 5.85. The optimal angles were 16.48, 13.05, 34.13, 12.80, 20.23, 52.72, 13.33, 13.00 and 47.02. The velocity magnitudes for the initial and optimized configuration is shown in Figure 12.

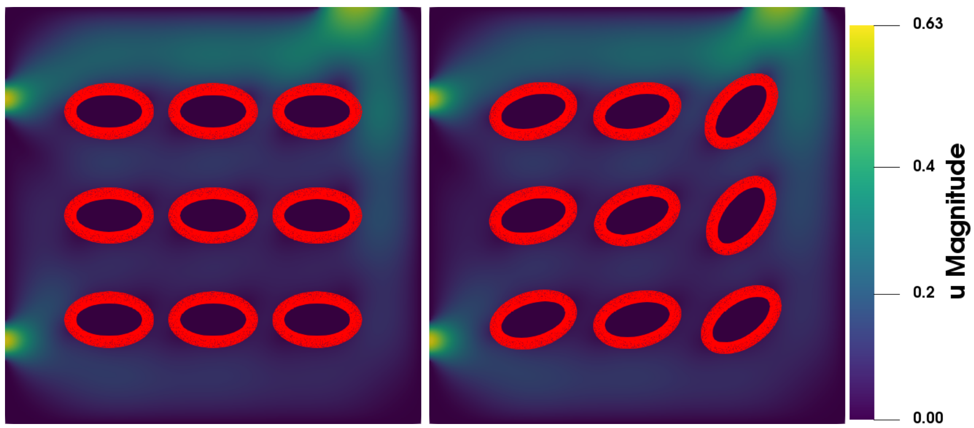


FIG. 12. *The initial and optimal configuration of the 9 objects in Stokes-flow. The initial functional value was* 5.85 *and the final value was* 5.59.

6. Concluding remarks. The main purpose of this work is analyzing how the multimesh FEM influences the computation of shape sensitivities in the shape optimization setting. For this analysis, we consider the method of mappings and the Hadamard formulation. In the numerical examples, we illustrate that for shape optimization problems parameterized by rigid motions, re-meshing and deformation equations are not required, as we can move meshes independently of each other. For traditional shape optimization problems, we presented a new robust deformation scheme, where we described the design boundaries on a separate mesh, which can be moved independently of the fixed domain boundaries. Since we deform subdomains, our deformation scheme yields a speed-up compared to similar schemes for single-mesh problems.

Nevertheless, since the multimesh FEM is a fairly new method, further study of Nitsche enforcement of interface conditions is required to obtain stable finite element methods for other equations than the Poisson and Stokes-equations.

In conclusion, the results reported in this paper, shows that the combination of shape optimization and the multimesh FEM holds great promise as a powerful method. In a later paper, we will extend this approach to time-dependent problems, with more complex state-equations.

Acknowledgments. The authors would like to acknowledge all FEniCS developers for feedback, and especially those involved in the MultiMesh project.

REFERENCES

[1] M. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, and G. N. Wells, *The fenics project version 1.5*, Archive of Numerical Software, 3 (2015), pp. 9–23.
[2] P. R. Amestoy, I. S. Duff, J. Koster, and J.-Y. L'Excellent, *A fully asynchronous multifrontal solver using distributed dynamic scheduling*, SIAM Journal on Matrix Analysis and Applications, 23 (2001), pp. 15–41.
[3] K. Bandara, T. Rüberg, and F. Cirak, *Shape optimisation with multiresolution subdivision surfaces and immersed finite elements*, Computer Methods in Applied Mechanics and Engineering, 300 (2016), pp. 510–539.
[4] D. M. Belk, *The role of overset grids in the development of the general purpose cfd code*, (1995).
[5] J. Benek, J. Steger, and F. C. Dougherty, *A flexible grid embedding technique with application to the euler equations*, in 6th Computational Fluid Dynamics Conference Danvers, 1983, p. 1944.
[6] J. Benk, H.-J. Bungartz, M. Mehl, and M. Ulbrich, *Immersed boundary methods for fluid-structure interaction and shape optimization within an FEM-based PDE toolbox*, in Advanced Computing, Springer, 2013, pp. 25–56.
[7] M. Berggren, *A unified discrete–continuous sensitivity analysis method for shape optimization*, in Applied and numerical partial differential equations, Springer, 2010, pp. 25–39.
[8] A. Bernland, E. Wadbro, and M. Berggren, *Acoustic shape optimization using cut finite elements*, International Journal for Numerical Methods in Engineering.
[9] P. T. Boggs, A. Althsuler, A. R. Larzelere, E. J. Walsh, R. L. Clay, and M. F. Hardwick, *Dart system analysis.*, tech. report, Sandia National Laboratories, 2005.
[10] T. Borrvall and J. Petersson, *Topology optimization of fluids in Stokes flow*, International journal for numerical methods in fluids, 41 (2003), pp. 77–107.
[11] F. Brezzi, J.-L. Lions, and O. Pironneau, *Analysis of a chimera method*, Comptes Rendus de l'Academie des Sciences Series I Mathematics, 332 (2001), pp. 655–660.
[12] E. Burman, S. Claus, P. Hansbo, M. G. Larson, and A. Massing, *CutFEM: Discretizing geometry and partial differential equations*, International Journal for Numerical Methods in Engineering, 104 (2015), pp. 472–501.
[13] E. Burman, D. Elfverson, P. Hansbo, M. G. Larson, and K. Larsson, *Shape optimization using the cut finite element method*, Computer Methods in Applied Mechanics and Engineering, 328 (2018), pp. 242–261.
[14] W. Chan, R. Gomez, S. Rogers, and P. Buning, *Best practices in overset grid generation*, in 32nd AIAA Fluid Dynamics Conference and Exhibit, 2002, p. 3191.
[15] M. C. Delfour and J.-P. Zolésio, *Shapes and geometries: metrics, analysis, differential calculus, and optimization*, SIAM, 2011.
[16] S. W. Funke, P. E. Farrell, and M. Piggott, *Tidal turbine array optimisation using the adjoint approach*, Renewable Energy, 63 (2014), pp. 658–673.
[17] C. Geuzaine and J.-F. Remacle, *Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities*, International journal for numerical methods in engineering, 79 (2009), pp. 1309–1331.
[18] C. V. Hansen, A. Logg, and C. Lundholm, *Simulation of flow and view with applications in computational design of settlement layouts*, arXiv preprint arXiv:1610.02277, (2016).
[19] H. Harbrecht and F. Loos, *Optimization of current carrying multicables*, Computational Optimization and Applications, 63 (2016), pp. 237–271, https://doi.org/10.1007/s10589-015-9764-2.
[20] W. Henshaw, *Overture: An object-oriented framework for overlapping grid applications*, in 32nd AIAA Fluid Dynamics Conference and Exhibit, 2002, p. 3189.
[21] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich, *Optimization with PDE constraints*, vol. 23, Springer Science & Business Media, 2008.
[22] A. Johansson, B. Kehlet, M. G. Larson, and A. Logg, *MultiMesh Finite Element Methods: Solving PDEs on Multiple Intersecting Meshes*, Computer Methods in Applied Mechanics and Engineering, (2018), https://doi.org/10.1016/j.cma.2018.09.009.
[23] A. Johansson and M. G. Larson, *A high order discontinuous Galerkin Nitsche method for elliptic problems with fictitious boundary*, Numerische Mathematik, 123 (2013), pp. 607–628.
[24] A. Johansson, M. G. Larson, and A. Logg, *High order cut finite element methods for the Stokes problem*, Advanced Modeling and Simulation in Engineering Sciences, 2 (2015), p. 1.
[25] W. Liao and H. M. Tsai, *Aerodynamic design optimization by the adjoint equation method on overset grids*, in 44th AIAA Aerospace Sciences Meeting and Exhibit, 2006, p. 54.
[26] A. Logg and G. N. Wells, *Dolfin: Automated finite element computing*, ACM Transactions

on Mathematical Software (TOMS), 37 (2010), p. 20.

[27] F. Loos, *Joule Heating in Connecting Structures of Automotive Electric Devices–Modelling, Simulation and Optimization*, PhD thesis, München, Univ. der Bundeswehr, Diss., 2014, 2014.

[28] B. Mohammadi and O. Pironneau, *Applied shape optimization for fluids*, Oxford university press, 2010.

[29] G. Mosetti, C. Poloni, and B. Diviacco, *Optimization of wind turbine positioning in large windfarms by means of a genetic algorithm*, Journal of Wind Engineering and Industrial Aerodynamics, 51 (1994), pp. 105–116.

[30] F. Murat and J. Simon, *Etude de problèmes d'optimal design*, in IFIP Technical Conference on Optimization Techniques, Springer, 1975, pp. 54–62.

[31] A. R. Najafi, M. Safdari, D. A. Tortorelli, and P. H. Geubelle, *A gradient-based shape optimization scheme using an interface-enriched generalized fem*, Computer Methods in Applied Mechanics and Engineering, 296 (2015), pp. 1 – 17, https://doi.org/https://doi.org/10.1016/j.cma.2015.07.024.

[32] E. J. Nielsen and M. A. Park, *Using an adjoint approach to eliminate mesh sensitivities in computational design*, AIAA journal, 44 (2006), p. 948.

[33] J. Nitsche, *Über ein Variationsprinzip zur Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind*, in Abhandlungen aus dem mathematischen Seminar der Universität Hamburg, vol. 36, Springer, 1971, pp. 9–15.

[34] L. Noël and P. Duysinx, *Shape optimization of microstructural designs subject to local stress constraints within an xfem-level set framework*, Structural and Multidisciplinary Optimization, 55 (2017), pp. 2323–2338.

[35] T. E. Oliphant, *Python for scientific computing*, Computing in Science & Engineering, 9 (2007).

[36] O. Pironneau, *On optimum design in fluid mechanics*, Journal of Fluid Mechanics, 64 (1974), pp. 97–110.

[37] O. Pironneau, *Optimal shape design for elliptic systems*, Springer Science & Business Media, 2012.

[38] J. J. Reuther, A. Jameson, J. J. Alonso, M. J. Rimlinger, and D. Saunders, *Constrained Multipoint Aerodynamic Shape Optimization Using an Adjoint Formulation and Parallel Computers, part 1*, Journal of Aircraft, 36 (1999), pp. 51–60.

[39] N. Schlömer, G. Cervone, Antonio amd McBain, F. Gokstorp, M. Bussonnier, S. Chen, N. Wagner, and N. Sime, *Pygmsh*. Zenodo, 2018, https://doi.org/10.5281/zenodo.1173105.

[40] N. Schlömer, N. Wagner, L. Dalcin, T. Li, L. Schwarz, C. Coutinho, G. McBain, A. Cervone, T. Langlois, S. Peak, M. Bussonnier, G. Jacquenot, G. A. Vaillant, and A. Croucher, *Meshio*. Zenodo, 2018, https://doi.org/10.5281/zenodo.1173115.

[41] S. Schmidt, *Efficient large scale aerodynamic design based on shape calculus*, 2010.

[42] S. Schmidt, *A Two Stage CVT/Eikonal Convection Mesh Deformation Approach for Large Nodal Deformations*, arXiv preprint arXiv:1411.7663, (2014).

[43] S. Schmidt, C. Ilic, V. Schulz, and N. R. Gauger, *Three-dimensional large-scale aerodynamic shape optimization based on shape calculus*, AIAA journal, (2013).

[44] S. Schmidt, E. Wadbro, and M. Berggren, *Large-scale three-dimensional acoustic horn optimization*, SIAM Journal on Scientific Computing, 38 (2016), pp. B917–B940.

[45] V. Schulz and M. Siebenborn, *Computational comparison of surface metrics for PDE constrained shape optimization*, Computational Methods in Applied Mathematics, 16 (2016), pp. 485–496.

[46] J. Sokolowski and J.-P. Zolesio, *Introduction to shape optimization*, in Introduction to Shape Optimization, Springer, 1992, pp. 5–12.

[47] R. Stangl, *Ein Euler-Verfahren zur Berechnung der Strömung um einen Hubschrauber im Vorwärtsflug*, Herbert Utz Verlag, 1996.

[48] G. Starius, *Composite mesh difference methods for elliptic boundary value problems*, Numerische Mathematik, 28 (1977), pp. 243–258.

[49] J. Steger, *The chimera method of flow simulation*, in Workshop on applied CFD, Univ of Tennessee Space Institute, vol. 188, 1991.

[50] J. L. Steger, F. C. Dougherty, and J. A. Benek, *A chimera grid scheme.[multiple overset body-conforming mesh system for finite difference adaptation to complex aircraft configurations]*, (1983).

[51] K. Stein, T. Tezduyar, and R. Benney, *Mesh moving techniques for fluid-structure interactions with large displacements*, Journal of applied mechanics, 70 (2003), pp. 58–63.

[52] E. Tekin and I. Sabuncuoglu, *Simulation optimization: A comprehensive review on theory*

and applications, IIE transactions, 36 (2004), pp. 1067–1081.
[53] L. Van Miegroet, N. Moës, C. Fleury, and P. Duysinx, *Generalized shape optimization based on the level set method*, in Proceedings of the 6th World Congress of Structural and Multidisciplinary Optimization (WCSMO6), 2005.
[54] C. H. Villanueva and K. Maute, *Cutfem topology optimization of 3d laminar incompressible flow problems*, Computer Methods in Applied Mechanics and Engineering, 320 (2017), pp. 444–473.
[55] E. Volkov, *The method of composite meshes*, Automatic Programming, Numerical Methods and Functional Analysis, 96 (1970), p. 145.
[56] A. Wächter and L. T. Biegler, *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming*, Mathematical programming, 106 (2006), pp. 25–57.