Journal of Computational and Applied Mathematics 407 (2022) 113997

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

An anisotropic adaptive method for the numerical approximation of orthogonal maps

Alexandre Caboussat [a,*], Dimitrios Gourzoulidis [a,b], Marco Picasso [b]

[a] Geneva School of Business Administration, University of Applied Sciences and Arts Western Switzerland (HES-SO), 1227 Carouge, Switzerland
[b] Institute of Mathematics, Ecole Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland

ARTICLE INFO

Article history:
Received 10 April 2020
Received in revised form 20 October 2021

MSC:
65N30
65N50
65K10
49M20
35F30

Keywords:
Orthogonal maps
Eikonal equation
Origami
Operator splitting
Anisotropic adaptive mesh refinement

ABSTRACT

Orthogonal maps are two-dimensional mappings that are solutions of the so-called origami problem obtained when folding a paper. These mappings are piecewise linear, and the discontinuities of their gradient form a singular set composed of straight lines representing the folding edges. The proposed algorithm relies on the minimization of a variational principle discussed in Caboussat et al. (2019). A splitting algorithm for the corresponding flow problem derived from the first-order optimality conditions alternates between local nonlinear problems and linear elliptic variational problems at each time step. Anisotropic adaptive techniques allow to obtain refined triangulations near the folding edges while keeping the number of vertices as low as possible. Numerical experiments validate the accuracy and efficiency of the adaptive method in various situations. Appropriate convergence properties are exhibited, and solutions with sharp edges are recovered.

1. Introduction

Orthogonal maps are the solutions of the analytical model of paper-folding, also called origami problem [1], in which a paper is folded along creases lines, but neither stretched nor torn. Mathematical aspects of such problems have been addressed, e.g., in [1–4]. Numerical methods for related problems have been proposed, e.g., in [5–9], and initially for Eikonal equations, e.g., in [10,11].

Originally, the motivation of this work comes from the works of Dacorogna et al. [2,4] on theoretical aspects of orthogonal maps, and first order fully nonlinear equations in general [12,13]. The solution being piecewise linear, the key aspect is to approximate the discontinuity lines (the so-called singular set), and this has been the reason to introduce anisotropic adaptive methods.

Besides the original origami application [4], such orthogonal maps formulation is or will be used for some applications, e.g., deformations of prestrained plates [14,15], aerospace engineering [16], cartography [17], differential geometry [18] and, to some extend, robotics [19,20]. More recently, it has been applied to several fields of material science, such as 3D printing and material composites [21–24] (e.g. for self folding materials).

* Corresponding author.
E-mail addresses: alexandre.caboussat@hesge.ch (A. Caboussat), dimitrios.gourzoulidis@hesge.ch, dimitrios.gourzoulidis@epfl.ch (D. Gourzoulidis), marco.picasso@epfl.ch (M. Picasso).

https://doi.org/10.1016/j.cam.2021.113997

In terms of numerical methods, several algorithms can be found for the solution of the scalar Eikonal equation (see [10] and references therein), or the Hamilton–Jacobi–Bellman (HJB) equation [25]. However, for the vectorial problem, the literature is rather scarce, besides related problems such as [14,15].

The Dirichlet problem addressed in this work consists of a degenerate system of first order fully nonlinear equations. Namely, it consists in finding a vector-valued function such that its gradient is an orthogonal matrix-valued function, together with essential, piecewise linear, Dirichlet boundary conditions. The solution to this problem is piecewise linear, with a singular set composed of straight lines representing the folding edges.

A variational approach relying on the minimization of a variational principle to enforce the uniqueness of the solution has been presented in [7], and will be summarized in the sequel. In particular, a time-splitting approach allows to decouple a sequence of local nonlinear problems from a global elliptic variational problem at each time iteration.

Anisotropic adaptive techniques have been reported to be very efficient in the presence of boundary layers or singularities [26–30]. Indeed, the use of finite element meshes with high aspect ratio allows the number of degrees of freedom to be reduced, while keeping the same accuracy as the classical isotropic finite elements. The goal of this article is to apply these techniques to paper-folding. Since the underlying mathematical problem is fully nonlinear, and the solution is not smooth, a posteriori estimates are difficult to derive and our error indicator is the one corresponding to a simplified problem.

Compared to [7], the novelty of this work lies in the adaptive mesh refinement algorithm. Since the solution is piecewise linear, adaptive techniques not only allow to drastically improve the accuracy and efficiency of the method, but are even necessary to obtain the convergence of the time-stepping algorithm to some admissible solution in some stringent cases. We introduce here a non-standard adaptive method and, in the final examples, couple it with domain decomposition approach for the most difficult test cases. Adaptive methods for first order fully nonlinear equations are scarce in the literature. In the particular case of orthogonal maps, the solutions are piecewise linear, and thus the whole difficulty is to approximate the set of discontinuities for the solution. Thus anisotropic adaptive methods are very well adapted, and provide a significant improvement to finite element methods, as they allow to accurately track the set of discontinuity lines of the solutions.

Numerical experiments are presented to validate the accuracy and the efficiency of the method, on a set of examples with exact solutions. Adaptive meshes show appropriate convergence properties, and allow to recover the sharp edges of the solutions. In particular, singular points at the intersection of several lines of singularity are well-tracked.

In the last section, the case of the homogeneous Dirichlet problem is discussed, as it exhibits a stringent fractal behavior near the boundary. A domain-decomposition approach, together with the anisotropic adaptive mesh refinement, is advocated to appropriately take into account the boundary conditions, and to converge to a fractal solution.

2. Mathematical formulation

Let Ω be a open bounded domain of $\mathbb{R}^2$ (typically $(0,1)^2$), which represents a paper to fold. The orthogonal maps problem consists in finding $\mathbf{u} : \Omega \subset \mathbb{R}^2 \to \mathbb{R}^2$ satisfying

$$\begin{cases} \nabla \mathbf{u} \in \mathcal{O}(2) & \text{a.e. in } \Omega, \\ \mathbf{u} = \mathbf{g} & \text{on } \partial\Omega, \end{cases} \tag{1}$$

where $\mathcal{O}(2)$ denotes the set of orthogonal 2×2 matrix-valued functions, and $\mathbf{g} : \partial\Omega \to \mathbb{R}^2$ is a sufficiently smooth, piecewise linear, given function. Writing $\mathbf{u} = [u_1, u_2]^T$, (1) is equivalent to finding $[u_1, u_2]^T : \Omega \subset \mathbb{R}^2 \to \mathbb{R}^2$ such that

$$\begin{cases} |\nabla u_1| = |\nabla u_2| = 1 & \text{a.e in } \Omega, \\ \nabla u_1 \cdot \nabla u_2 = 0 & \text{a.e in } \Omega, \\ [u_1, u_2]^T = \mathbf{g} & \text{on } \partial\Omega. \end{cases} \tag{2}$$

3. Numerical methods

3.1. Regularization and penalization

The existence of a solution to (1) has been studied, e.g., in [1], and the uniqueness of the solution is not guaranteed. Thus, in order to enforce some kind of uniqueness, we consider a variational principle and the problem: Find $\mathbf{u} \in \mathbf{E_g}$ satisfying

$$\min_{\mathbf{v}\in\mathbf{E_g}} \left[\frac{C}{2} \int_\Omega |\mathbf{v} - \mathbf{f}|^2 \, d\mathbf{x} + \frac{1}{2} \int_\Omega |\nabla \mathbf{v}|^2 d\mathbf{x} \right] \tag{3}$$

where

$$\mathbf{E_g} = \{\mathbf{v} \in (H^1(\Omega))^2,\ \mathbf{v}|_{\partial\Omega} = \mathbf{g},\ \nabla\mathbf{v} \in \mathcal{O}(2)\ a.e \text{ in } \Omega\}, \tag{4}$$

and where $C \geq 0$, and $\mathbf{g}$ and $\mathbf{f}$ are given and sufficiently regular functions. The data $\mathbf{f}$ corresponds to a target function, corresponding to some a priori information on the solution and set to $\mathbf{0}$ by default.

In order to solve (3), one advocates an appropriate combination of penalization of the first order optimality conditions, together with a splitting algorithm for the corresponding flow problem. The splitting approach allows to decouple the nonlinearities (which are local) and a global linear variational problem. A complete description of the algorithm can be found in [7] and is briefly sketched in the sequel — in a somehow simplified version. The first step is to penalize the nonlinear constraints in $\mathbf{E_g}$ and regularize the objective function in (3), so that it becomes

$$\min_{\mathbf{v}\in\mathbf{V_g}}\left[\frac{C}{2}\int_\Omega|\mathbf{v}-\mathbf{f}|^2\,d\mathbf{x}+\frac{1}{2}\int_\Omega|\nabla\mathbf{v}|^2d\mathbf{x}+\frac{\varepsilon_1}{2}\int_\Omega\left|\nabla^2\mathbf{v}\right|^2d\mathbf{x}\right.$$
$$\left.+\frac{1}{4\varepsilon_2}\int_\Omega\left[(|\nabla v_1|^2-1)^2+(|\nabla v_2|^2-1)^2+|\nabla v_1\cdot\nabla v_2|^2\right]d\mathbf{x}.\right] \tag{5}$$

where $\mathbf{V_g}=\{\mathbf{v}\in(H^2(\Omega))^2,\ \mathbf{v}|_{\partial\Omega}=\mathbf{g}\}$, and where $\varepsilon_1,\varepsilon_2>0$ are given. We then compute the corresponding first order optimality conditions (Euler–Lagrange equations): Find $\mathbf{u}=[u_1,u_2]\in\mathbf{V_g}$ such that:

$$\int_\Omega\nabla\mathbf{u}:\nabla\mathbf{v}d\mathbf{x}+C\int_\Omega(\mathbf{u}-\mathbf{f})\cdot\mathbf{v}d\mathbf{x}+\varepsilon_1\int_\Omega\nabla^2\mathbf{u}\cdot\nabla^2\mathbf{v}d\mathbf{x}$$
$$+\frac{1}{\varepsilon_2}\int_\Omega\left[(|\nabla u_1|^2-1)\nabla u_1\cdot\nabla v_1+(|\nabla u_2|^2-1)\nabla u_2\cdot\nabla v_2\right. \tag{6}$$
$$\left.+\frac{1}{2}\nabla u_1\cdot\nabla u_2(\nabla u_2\cdot\nabla v_1+\nabla u_1\cdot\nabla v_2)\right]d\mathbf{x}=0,\quad\forall\mathbf{v}=[v_1,v_2]\in\mathbf{V_0},$$

where $\mathbf{V_0}=(H^2(\Omega)\cap H_0^1(\Omega))^2$.

3.2. Flow problem and time splitting algorithm

The next step of the algorithm is to introduce a flow problem (in the dynamical systems sense), and look for the stationary solution. In order to do so, this initial value problem is defined as follows: Find $\mathbf{u}(t)\in\mathbf{V_g}$ for a.e. $t\in(0,+\infty)$ satisfying

$$\int_\Omega\frac{\partial\nabla\mathbf{u}(t)}{\partial t}:\nabla\mathbf{v}d\mathbf{x}+\int_\Omega\nabla\mathbf{u}(t):\nabla\mathbf{v}d\mathbf{x}+C\int_\Omega(\mathbf{u}(t)-\mathbf{f})\cdot\mathbf{v}d\mathbf{x}+\varepsilon_1\int_\Omega\nabla^2\mathbf{u}(t)\cdot\nabla^2\mathbf{v}d\mathbf{x}$$
$$+\frac{1}{\varepsilon_2}\int_\Omega\left[(|\nabla u_1(t)|^2-1)\nabla u_1(t)\cdot\nabla v_1(t)+(|\nabla u_2(t)|^2-1)\nabla u_2(t)\cdot\nabla v_2\right. \tag{7}$$
$$\left.+\frac{1}{2}\nabla u_1(t)\cdot\nabla u_2(t)(\nabla u_2(t)\cdot\nabla v_1+\nabla u_1(t)\cdot\nabla v_2)\right]d\mathbf{x}=0,\quad\forall\mathbf{v}=[v_1,v_2]\in\mathbf{V_0}.$$

together with the initial condition $\mathbf{u}(0):=\mathbf{u}_0$, given. The choice of $\mathbf{u}_0$ has been discussed in [7]. This flow problem is not classical, in the sense that it involves $\dfrac{\partial\nabla\mathbf{u}(t)}{\partial t}$, which will allow to obtain the right balance between operators in the splitting algorithm that follows. We apply an operator-splitting strategy to solve (7) (namely a first-order Marchuk–Yanenko scheme). Let $\Delta t>0$ be a constant given time step, $t^n=n\Delta t$, $n=1,2,\ldots$, to define the approximations $\mathbf{u}^n\simeq\mathbf{u}(t^n)$. Starting from the initial condition $\nabla\mathbf{u}^0:=\nabla\mathbf{u}_0$, the Marchuk–Yanenko scheme allows, using $\mathbf{u}^n$ for all $n\geq0$, to compute successively $\mathbf{u}^{n+1/2}$ and $\mathbf{u}^{n+1}$ using the two following intermediate steps.

1. Prediction step: Find $\nabla\mathbf{u}^{n+1/2}\in\nabla\mathbf{V_g}$ satisfying

$$\int_\Omega\frac{\nabla\mathbf{u}^{n+1/2}-\nabla\mathbf{u}^n}{\Delta t}:\nabla\mathbf{v}d\mathbf{x}+\int_\Omega\nabla\mathbf{u}^{n+1/2}:\nabla\mathbf{v}d\mathbf{x}$$
$$+\frac{1}{\varepsilon_2}\int_\Omega\left[(\left|\nabla u_1^{n+1/2}\right|^2-1)\nabla u_1^{n+1/2}\cdot\nabla v_1+(\left|\nabla u_2^{n+1/2}\right|^2-1)\nabla u_2^{n+1/2}\cdot\nabla v_2\right. \tag{8}$$
$$\left.+\frac{1}{2}\nabla u_1^{n+1/2}\cdot\nabla u_2^{n+1/2}(\nabla u_2^{n+1/2}\cdot\nabla v_1+\nabla u_1^{n+1/2}\cdot\nabla v_2)\right]d\mathbf{x}=0,$$

for all $\nabla\mathbf{v}\in\nabla\mathbf{V_0}$. Problem (8) is a *local optimization problem* for $\nabla\mathbf{u}^{n+1/2}$, as it does not involve any derivative of the unknown variable $\nabla\mathbf{u}^{n+1/2}$. The local optimization problem leads to a nonlinear system solved by a Newton method without safeguarding. In practice, it can be solved on each element of the finite element discretization. Details can be found in [7].

2. Correction step: Find $\mathbf{u}^{n+1}\in\mathbf{V_g}$ satisfying

$$\int_\Omega\frac{\nabla\mathbf{u}^{n+1}-\nabla\mathbf{u}^{n+1/2}}{\Delta t}:\nabla\mathbf{v}d\mathbf{x}+C\int_\Omega(\mathbf{u}^{n+1}-\mathbf{f})\cdot\mathbf{v}d\mathbf{x}+\varepsilon_1\int_\Omega\nabla^2\mathbf{u}^{n+1}\cdot\nabla^2\mathbf{v}d\mathbf{x}=0, \tag{9}$$

for all $\mathbf{v}\in\mathbf{V_0}$. This problem is an elliptic linear variational problem, whose solution will be detailed in Section 3.3.

3.3. Linear variational problems

Problem (9) is a fourth-order linear variational problem of the biharmonic type. Via the introduction of an auxiliary variable $\mathbf{w}^{n+1} := -\nabla^2\mathbf{u}^{n+1}$, it is rewritten as a coupled second order linear system that reads as follows: Find $(\mathbf{u}^{n+1}, \mathbf{w}^{n+1}) \in \{\mathbf{v} \in H^1(\Omega)^2, \mathbf{v}|_{\partial\Omega} = \mathbf{g}\} \times \left(H_0^1(\Omega)\right)^2$ such that

$$\begin{aligned}
&\varepsilon_1 \Delta t \int_\Omega \nabla\mathbf{w}^{n+1} : \nabla\mathbf{v} d\mathbf{x} + \int_\Omega \nabla\mathbf{u}^{n+1} : \nabla\mathbf{v} d\mathbf{x} + C\Delta t \int_\Omega \mathbf{u}^{n+1}\cdot\mathbf{v} d\mathbf{x}\\
&\quad = C\Delta t \int_\Omega \mathbf{f}\cdot\mathbf{v} d\mathbf{x} + \int_\Omega \nabla\mathbf{u}^{n+1/2} : \nabla\mathbf{v} d\mathbf{x},\\
&\int_\Omega \nabla\mathbf{u}^{n+1} : \nabla\mathbf{q} d\mathbf{x} - \int_\Omega \mathbf{w}^{n+1}\cdot\mathbf{q} d\mathbf{x} = 0,
\end{aligned}\tag{10}$$

for all $(\mathbf{v}, \mathbf{q}) \in \left(H_0^1(\Omega)\right)^2 \times \left(H_0^1(\Omega)\right)^2$.

It follows from (10) that the pair $(\mathbf{u}^{n+1}, \mathbf{w}^{n+1})$ satisfies the strong formulation:

$$-\varepsilon_1\Delta t \nabla^2\mathbf{w}^{n+1} + \mathbf{w}^{n+1} + C\Delta t\mathbf{u}^{n+1} = C\Delta t\mathbf{f} - \nabla^2\mathbf{u}^{n+1/2}, \text{ in } \Omega. \tag{11}$$

Anticipating on the space discretization addressed in Section 4, boundary layer thickness considerations suggest taking $\varepsilon_1\Delta t$ of the order of h^2 in (11), that is taking ε_1 of the order of $h^2/\Delta t$.

When $\varepsilon_1 = 0$, the decoupling of (9) into (10) is not required, as (9) is a classical, second-order, linear variational problem. Considering that the coefficient $\varepsilon_1\Delta t \simeq h^2$ is small, the adaptive strategy we advocate in the sequel is based on the assumption $\varepsilon_1 = 0$ (without biharmonic regularization). The error estimate derived in that case is then applied, by extension, to the case when $\varepsilon_1 \neq 0$. The space adaptivity algorithm is applied at each time step of the time iteration method. As a consequence, there will be no adaptivity treatment using directly the auxiliary variable $\mathbf{w}^{n+1}$.

4. Finite element discretization and anisotropic adaptive algorithm

4.1. Discrete operator splitting algorithm

Let us denote by $h > 0$ a space discretization step, together with an associated *triangulation* $\mathcal{T}_h$ of Ω that satisfies the usual compatibility conditions (see, e.g., [31]). Let us denote by Σ_h the (finite) set of the vertices of $\mathcal{T}_h$, by N_h the number of elements in Σ_h, and by Σ_{0h} the subset of those elements in Σ_h not located on $\partial\Omega$ (with $N_{0h} := \text{card}(\Sigma_{0h})$). From the triangulation $\mathcal{T}_h$, we define the following finite element spaces:

$$\begin{aligned}
\mathbf{V}_h &= \{\mathbf{v}_h \in (C^0(\overline{\Omega}))^2, \mathbf{v}_h|_K \in (\mathbb{P}_1)^2, \forall K \in \mathcal{T}_h\},\\
\mathbf{V}_{\mathbf{g},h} &= \{\mathbf{v}_h \in \mathbf{V}_h, \mathbf{v}_h = \mathbf{g}_h \text{ on } \partial\Omega\},\\
\mathbf{Q}_h &= \{\mathbf{q}_h \in (L^\infty(\Omega))^{2\times 2}, \mathbf{q}_h|_K \in \mathbb{R}^{2\times 2}, \forall K \in \mathcal{T}_h\},
\end{aligned}$$

where $\mathbb{P}_1$ is the space of two-variable polynomials of degree ≤ 1, and $\mathbf{g}_h$ is the piecewise linear approximation of the boundary data. Note that $\nabla\mathbf{V}_h \subset \mathbf{Q}_h$. The use of low degree (piecewise linear) finite elements is justified by the low regularity of the solution (typically Lipschitz continuous, with jumps of the gradient). Next, we equip $\mathbf{V}_h$, and its sub-space $\mathbf{V}_{\mathbf{g},h}$, with a discrete inner product (based on classical quadrature formulas) $(\mathbf{v}, \mathbf{w})_{0h}$. The quadrature formulas we used are implemented in the library `libmesh` [32]. The corresponding norm is $\|\mathbf{v}\|_{0h} := \sqrt{(\mathbf{v}, \mathbf{v})_{0h}}$, for all $\mathbf{v} \in \mathbf{V}_h$. In a similar fashion, we equip the space $\mathbf{Q}_h$ with the inner product and norm respectively defined by: $((\mathbf{p}, \mathbf{q}))_{0h} = \sum_{K\in\mathcal{T}_h} |K|\, \mathbf{p}|_K : \mathbf{q}|_K$ and $|||\mathbf{q}|||_{0h} = \sqrt{((\mathbf{q}, \mathbf{q}))_{0h}}$ (with $|K| =$ area of K).

We denote by $\mathbf{u}_h^n$ the approximation of $\mathbf{u}_h(t^n)$, and we derive a finite dimensional approximation of the operator-splitting strategy (8) (9) as follows. Starting from the initial condition $\mathbf{u}_h^0 = \mathbf{u}_{0,h}$, we compute successively $\mathbf{u}_h^{n+1/2}$ and $\mathbf{u}_h^{n+1}$ via the two following intermediate steps, which are the discretized versions of (8) and (9):

1. Prediction step: Find $\nabla\mathbf{u}_h^{n+1/2} \in \mathbf{Q}_h$ satisfying

$$\begin{aligned}
&\left(\left(\frac{\nabla\mathbf{u}_h^{n+1/2} - \nabla\mathbf{u}_h^n}{\Delta t}, \nabla\mathbf{v}_h\right)\right)_{0h} + ((\nabla\mathbf{u}_h^{n+1/2}, \nabla\mathbf{v}_h))_{0h}\\
&+\frac{1}{\varepsilon_2}\int_\Omega \Big[(|\nabla u_{1,h}^{n+1/2}|^2 - 1)\nabla u_{1,h}^{n+1/2}\cdot\nabla v_{1,h} + (|\nabla u_{2,h}^{n+1/2}|^2 - 1)\nabla u_{2,h}^{n+1/2}\cdot\nabla v_{2,h}\\
&+\frac{1}{2}\nabla u_{1,h}^{n+1/2}\cdot\nabla u_{2,h}^{n+1/2}(\nabla u_{2,h}^{n+1/2}\cdot\nabla v_{1,h} + \nabla u_{1,h}^{n+1/2}\cdot\nabla v_{2,h})\Big]\, d\mathbf{x} = 0,
\end{aligned}\tag{12}$$

for all $\nabla\mathbf{v}_h \in \nabla\mathbf{V}_{\mathbf{0},h}$. The finite dimensional local nonlinear problem (12) is solved *triangle-wise* on each element of $\mathcal{T}_h$.

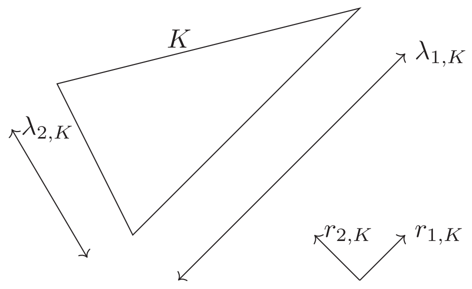

Fig. 1. Reference element, main directions, and element dimensions.

2. Correction step: Find $(\mathbf{u}_h^{n+1}, \mathbf{w}_h^{n+1}) \in \mathbf{V}_{\mathbf{g},h} \times \mathbf{V}_{\mathbf{0},h}$ such that

$$\left\{\begin{array}{l} \varepsilon_1 \Delta t((\nabla \mathbf{w}_h^{n+1}, \nabla \mathbf{v}_h))_{0h} + ((\nabla \mathbf{u}_h^{n+1}, \nabla \mathbf{v}_h))_{0h} + C\Delta t(\mathbf{u}_h^{n+1}, \mathbf{v}_h)_{0h} \\ = C\Delta t(\mathbf{f}, \mathbf{v}_h)_{0h} + ((\nabla \mathbf{u}_h^{n+1/2}, \nabla \mathbf{v}_h))_{0h}, \\ ((\nabla \mathbf{u}_h^{n+1}, \nabla \mathbf{q}_h))_{0h} - (\mathbf{w}_h^{n+1}, \mathbf{q}_h)_{0h} = 0, \end{array}\right. \tag{13}$$

for all $(\mathbf{v}_h, \mathbf{q}_h) \in \mathbf{V}_{\mathbf{0},h} \times \mathbf{V}_{\mathbf{0},h}$. The adaptive strategy incorporated into the solution to (13) is described in the next section.

The stopping criterion we use to decide on the flow stationarity is either $n \leq 1000$ or $||\mathbf{u}_h^{n+1} - \mathbf{u}_h^n||_{L^2(\Omega)} \leq 10^{-7}$ (unless stated otherwise).

4.2. Adaptive algorithm with anisotropic meshes

Following [33,34], at each step of (12) (13), our goal is now to build an anisotropic mesh such that the estimated relative error is close to a preset tolerance *TOL*, namely:

$$0.75\ TOL \leq \frac{\eta^{A,n+1}}{\left\|\nabla \mathbf{u}_h^{n+1}\right\|_{L^2(\Omega)}} \leq 1.25\ TOL, \tag{14}$$

where the anisotropic error estimate $\eta^{A,n+1}$ is based on the finite element mesh.

In order to describe the mesh anisotropy, let us first recall some required geometrical definitions. For any triangle K of the discretization $\mathcal{T}_h$, let $T_K : \hat{K} \to K$ be the affine transformation which maps the reference triangle $\hat{K}$ into K. Let M_K be the Jacobian of the mapping T_K. Since M_K is invertible, it admits a singular value decomposition $M_K = R_K^T \Lambda_K P_K$, where R_K and P_K are orthogonal and where Λ_K is diagonal with positive entries. In the following, we set

$$\Lambda_K = \begin{pmatrix} \lambda_{1,K} & 0 \\ 0 & \lambda_{2,K} \end{pmatrix}, \quad R_K = \begin{pmatrix} \mathbf{r}_{1,K}^T \\ \mathbf{r}_{2,K}^T \end{pmatrix}, \tag{15}$$

with the choice $\lambda_{1,K} \geq \lambda_{2,K}$. These geometrical quantities are illustrated in Fig. 1.

This error estimate is then re-distributed on the elements as

$$\eta^{A,n+1} = \left(\sum_{K\in\mathcal{T}_H} (\eta_K^{A,n+1})^2\right)^{1/2}.$$

Let us now focus on the derivation of the estimator $\eta_K^{A,n+1}$ on each triangle. The estimator contains one contribution from the residual of (13), and another contribution due to the anisotropic metric. The latter relies here on the Zienkiewicz–Zhu approach [35,36]. More precisely, we have:

$$(\eta_K^{A,n+1})^2 = \rho_K(\mathbf{u}_h^{n+1}) \times \omega_K(\mathbf{u}(t^{n+1}) - \mathbf{u}_h^{n+1}), \tag{16}$$

where, following the steps of [34], the problem-dependent residual is given on each element K by:

$$\begin{aligned} \rho_K(\mathbf{u}_h^{n+1}) &= ||C\Delta t\mathbf{f} - \nabla^2 \mathbf{u}_h^{n+1/2} + \nabla^2 \mathbf{u}_h^{n+1} - C\Delta t \mathbf{u}_h^{n+1}||_{L^2(K)} \\ &\quad + \frac{1}{2\lambda_{2,K}^{1/2}} \left|\left|\left[\frac{\partial(\mathbf{u}_h^{n+1} - \mathbf{u}_h^{n+1/2})}{\partial n}\right]\right|\right|_{L^2(\partial K)}, \end{aligned} \tag{17}$$

where $[\cdot]$ denotes the jump of the bracketed quantity across an internal edge ($[\cdot] = 0$ for an edge on the boundary $\partial\Omega$). On the other hand, the geometrical contribution is given by

$$\omega_K(\mathbf{u}(t^{n+1}) - \mathbf{u}_h^{n+1})^2 = \lambda_{1,K}^2(\mathbf{r}_{1,K}^T G_K(\mathbf{u}(t^{n+1}) - \mathbf{u}_h^{n+1})\mathbf{r}_{1,K}) + \lambda_{2,K}^2(\mathbf{r}_{2,K}^T G_K(\mathbf{u}(t^{n+1}) - \mathbf{u}_h^{n+1})\mathbf{r}_{2,K}),$$

where the error gradient matrix $G_K(\cdot)$ is defined as: $G_K(v) = \sum_{T\in\Delta K} \begin{pmatrix} \int_T \left(\frac{\partial v}{\partial x_1}\right)^2 d\mathbf{x} & \int_T \frac{\partial v}{\partial x_1}\frac{\partial v}{\partial x_2} d\mathbf{x} \\ \int_T \frac{\partial v}{\partial x_1}\frac{\partial v}{\partial x_2} d\mathbf{x} & \int_T \left(\frac{\partial v}{\partial x_2}\right)^2 d\mathbf{x} \end{pmatrix}$, and ΔK is the set of triangles having a vertex common with K.

In practice, $G_K(\mathbf{u}(t^{n+1}) - \mathbf{u}_h^{n+1})$ in (18) is approximated by a post-processed matrix $\tilde{G}_K(\mathbf{u}(t^{n+1}) - \mathbf{u}_h^{n+1})$, obtained by recovering the gradients according to the procedure described in [35,36].

In (17), the first term of the residual could practically become negligible when using piecewise linear finite elements, and when the time step is small (recall that $\Delta t = \varepsilon_2/2$, and the penalization parameter ε_2 is very small), meaning that the most important contribution comes from the jump terms.

The BL2D mesh generator [37] is used to reconstruct an adapted mesh at each iteration. It requires a metric to be given at the vertices of $\mathcal{T}_h$ for the update of the mesh, and thus the anisotropic error estimate on the elements is translated into an error estimate on each node of the mesh, as detailed, e.g., in [33].

4.3. General algorithm

The time splitting algorithm (12) (13) is revisited with additional mesh adaptivity techniques. Unlike what has been done in [38,39], the algorithm relies on the fact that the mesh is refined at each time step. When the stationary solution is reached, it is stabilized by performing additional iterations with the final adapted mesh. Let us consider given values of Δt, ε_1 and ε_2; and let us denote by n_{mesh} the number of time iterations achieved with mesh adaptation at each step, and n_{final} the (maximal) number of additional time steps performed with the given final adapted mesh. The general time stepping algorithm is sketched as follows:

```
▷ Set given initial conditions, with the initial finite element mesh $\mathcal{T}_h^0$.
▷ For $n = 0, 1, \dots, n_{mesh}$, do
      1 (Local optimization) Solve the local algebraic optimization problems (12) at each grid point of $\mathcal{T}_h^n$;
      2 (Variational) Solve the global linear variational problem (13) with the current discretization $\mathcal{T}_h^n$;
      3 (Adapt) Update the finite element mesh $\mathcal{T}_h^n \to \mathcal{T}_h^{n+1}$;
▷ For $n = 0, 1, \dots, n_{final}$ (or until $\left\|\mathbf{u}_h^{n+1} - \mathbf{u}_h^n\right\|_{0h} < 10^{-7}$), solve (12) (13) on the fixed final adapted mesh$\mathcal{T}_h^{n_{mesh}}$.
```

Numerical experiments have shown that adapting the finite element mesh at each time step helps to converge to a stationary solution. This effect is documented in the next section. Adapting the mesh at each pseudo-time step may create spurious mesh adaptations in some cases. The only purpose of the introduction of stabilization steps is to safeguard the algorithm, and it acts merely as a smoother for small oscillations that may be introduced. However, the presented optimal convergence orders are identical without this additional loop, and the convergence behavior is similar when replacing this stopping criterion with a fixed number of additional steps, and when using, e.g., $\left\|\mathbf{u}_h^{n+1} - \mathbf{u}_h^n\right\|_{0h} < 10^{-7}$.

5. Numerical experiments

We present some test cases to illustrate the convergence of our algorithm, and to perform a sensitivity analysis. The numerical implementation has been done with `libmesh` [32], which uses the libraries Lapack and Blas (for the solution of the algebraic nonlinear systems), and Petsc (for the solution of linear systems). The mesh is adapted with `bl2d` [37]. For all examples considered, the computational domain is the unit square $\Omega = (0, 1)^2$, and the parameters in (8) (9) are given by $\varepsilon_2 = 10^{-11}$ and $\Delta t = \varepsilon_2/2$. The value of the parameter ε_1 will be discussed further. The parameter ε_2 ensures that the orthogonality conditions are satisfied, see (5). The condition $\Delta t \leq \varepsilon_2$ ensures the Newton algorithm for the local nonlinear problems to converge. Numerical experiments have shown that taking larger values of those parameters (up to 10^{-6}) does not impact the number of pseudo-time steps and thus the speed of the algorithm.

Numerical experiments have been obtained with Intel$^{\circledR}$ Xeon(R) CPU E5-1650 v3 @ 3.50 GHz $\times$ 12 processor with 64 GB ram.

Because of the low regularity of the solution, small oscillations are created when the mesh is adapted and the stationary solution is difficult to catch. Therefore, the stopping criterion for the iterative method reads as follows: the number of time steps is fixed as $n_{mesh} = 500$ for all test cases. All error indicators are averaged over the last 200 time steps in the tables detailing the convergence behavior of the algorithm. Finally, the results on the effectivity index are given with a margin of error ($\pm$ standard deviation).

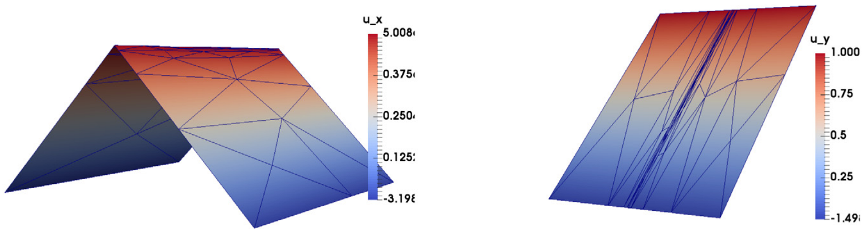

Fig. 2. Single folding. Snapshots of the approximated solution (left: first component $u_{1,h}$; right: second component $u_{2,h}$), with illustration of the final adapted mesh.

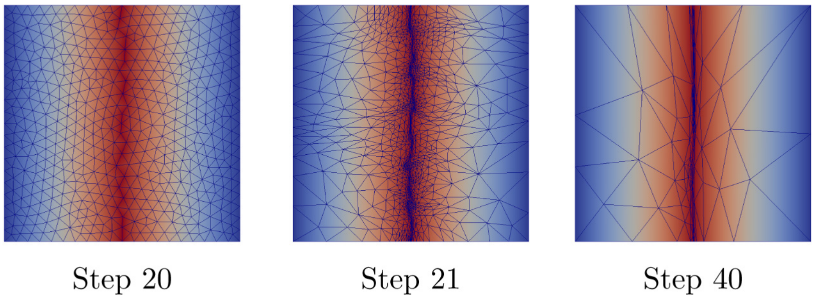

Fig. 3. Iterative mesh adaptation within the time stepping algorithm ($\Delta t = 0.5 \cdot 10^{-11}$, $\varepsilon_1 = 0.0$, $TOL = 0.03125$). Left: mesh at time step $n = 20$; middle: mesh at time step $n = 21$; right: mesh at time step $n = 40$.

5.1. Single folding

The first example corresponds to a single fold, for which the singular set consists of the segment $\{1/2\} \times [0, 1]$. The exact solution of this problem is

$$\begin{aligned} u_1(x_1, x_2) &= \begin{cases} x_1 & \text{if } x_1 < 0.5, \\ 1 - x_1 & \text{if } x_1 \geq 0.5, \end{cases} \qquad \forall (x_1, x_2) \in \Omega, \\ u_2(x_1, x_2) &= x_2, \end{aligned}$$

and the boundary conditions are defined accordingly. Fig. 2 illustrates a snapshot of the stationary solution, together with an illustration of an adaptive mesh (in that case, $h_{\min} = 4.53 \cdot 10^{-3}$, $h_{\max} = 9.81 \cdot 10^{-1}$, $TOL = 0.03125$, $\varepsilon_1 = 0$). We can observe that the orthogonality conditions are accurately satisfied ($\int_\Omega |\nabla u_{1,h}| = 1.000407$, $\int_\Omega |\nabla u_{2,h}| = 1.000000$, and $\int_\Omega \nabla u_{1,h} \cdot \nabla u_{2,h} = 0.0001$).

The method we advocate adapts the mesh at each time iteration. Fig. 3 illustrates the evolution of the finite element discretization after 20,21 and 40 time iterations. Even though the transient solution at one given time step is not accurate and has not converged yet to the stationary solution, the mesh refinement allows to track for the singularities and obtain more robust convergence properties of the global outer loop algorithm.

Figs. 4 and 5 illustrate the snapshots of the solution for various values of TOL ($\varepsilon_1 = 0$ fixed), and various values of ε_1 (TOL fixed). The conclusions are the following: i) the first figure shows that the smaller the tolerance, the thinner the region where elements are generated along the discontinuity line, and the larger the number of those elements; ii) the second figure shows that, for a given tolerance, if ε_1 is too large the solution becomes very smooth and thus the anisotropic mesh refinement is not accurate anymore as there is no privileged direction in the solution. Actually, when ε_1 becomes smaller, the mesh converges to the same mesh as when $\varepsilon_1 = 0$. Numerical results confirm that the jump terms in $\rho_K(\mathbf{u}_h)$ in (17) are the crucial ones for mesh adaptation, as results are very similar even when $C = 0$.

Table 1 shows the numerical behavior of the algorithm for varying parameters, namely the values of the parameters and final adapted meshes for all tolerances and for $\varepsilon_1 = 2 \cdot 10^{-6}$ and $\varepsilon_1 = 0$. The mesh sizes are defined as $h_{min} = \min_{K \in \mathcal{T}_h} \lambda_{1,K}$ and $h_{max} = \max_{K \in \mathcal{T}_h} \lambda_{2,K}$. The aspect ratio AR represents the maximal aspect ratio defined as $AR = \max_{K \in \mathcal{T}_h} \frac{\lambda_{1,K}}{\lambda_{2,K}}$.

As illustrated, the aspect ratio, the number of elements and the number of nodes increase appropriately when the tolerance decreases. The number of elements and nodes is larger when $\varepsilon_1 = 0$, showing that the convergence of the mesh adaptive algorithm is more difficult to reach. On the other hand, for the same number of time iterations, the L^2-error is smaller when $\varepsilon_1 = 0$.

We observe that the effectivity index becomes smaller than one when the tolerance decreases, meaning that the H^1-error is not bounded by the estimator η_K^A. This effect will be observed in all numerical experiments in the sequel. This behavior means that a contribution is missing in the estimator to have optimal convergence orders. This effect is actually

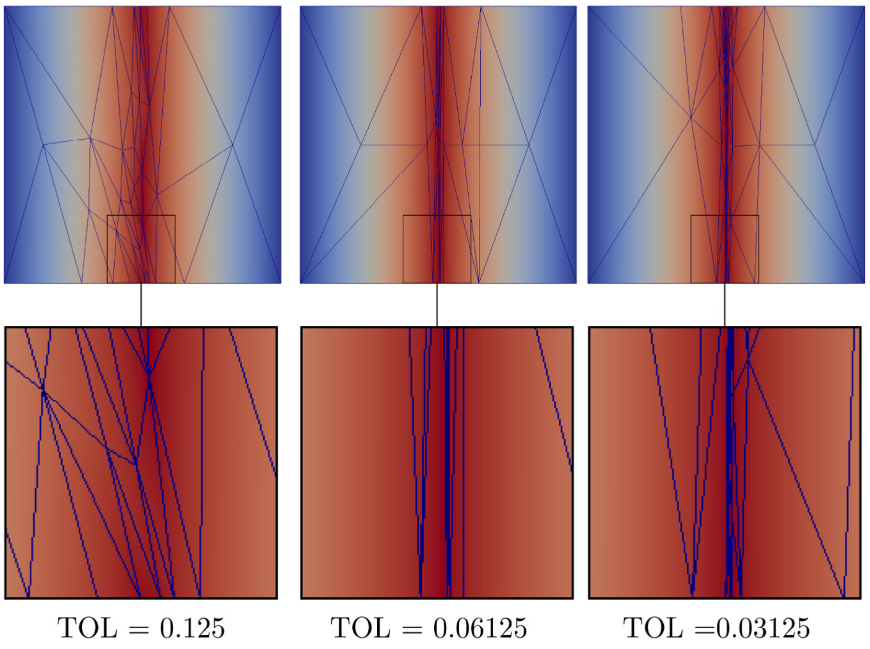

Fig. 4. Single folding. Snapshots of the final adapted mesh after 500 time iterations, for various values of the tolerance TOL ($\varepsilon_1 = 0.0$). The colormap represents the values of the first component $u_{1,h}$. The second row corresponds to a zoom in the squared region indicated in the first row.

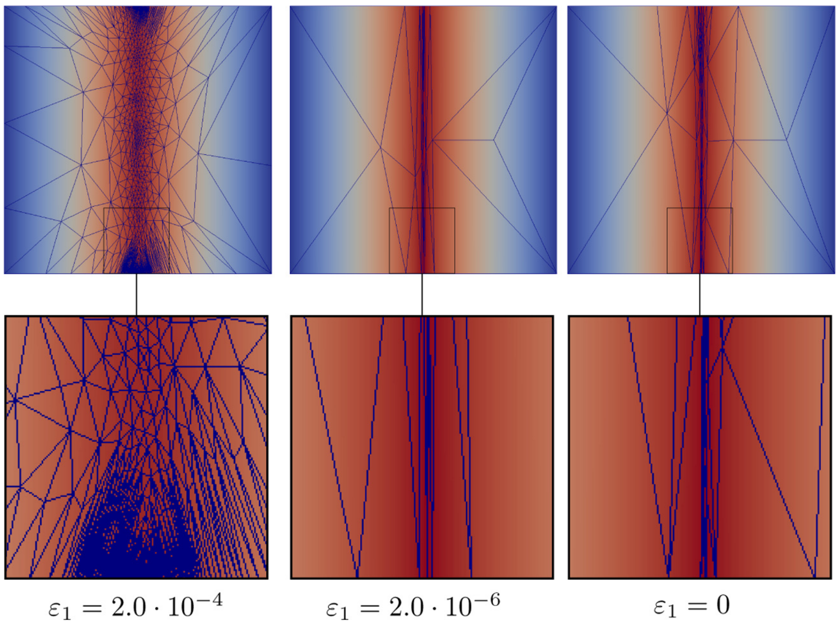
$\varepsilon_1 = 2.0 \cdot 10^{-4}$
$\varepsilon_1 = 2.0 \cdot 10^{-6}$
$\varepsilon_1 = 0$

Fig. 5. Simple folding. Snapshots of the final adapted mesh after 500 time iterations, for various values of the regularization parameter ε_1 ($TOL = 0.03125$). The colormap represents the values of the first component $u_{1,h}$. The second row corresponds to a zoom in the squared region indicated in the first row.

expected, since we are not incorporating either the splitting error, not the contribution from the nonlinear operators in the estimator η_K^A, but only the linear variational operator. Table 2 shows an appropriate convergence behavior for the orthogonality conditions, with even some super-convergence behavior in some cases.

Finally, Fig. 6 illustrates the iterative behavior of the time-stepping algorithm for various tolerances when $\varepsilon_1 = 0$; it shows that the time evolution of indicators is indeed oscillating when the mesh is adapted, due to the low regularity of the solution. The top left figure shows the time evolution of the number of elements; the top right figure shows the

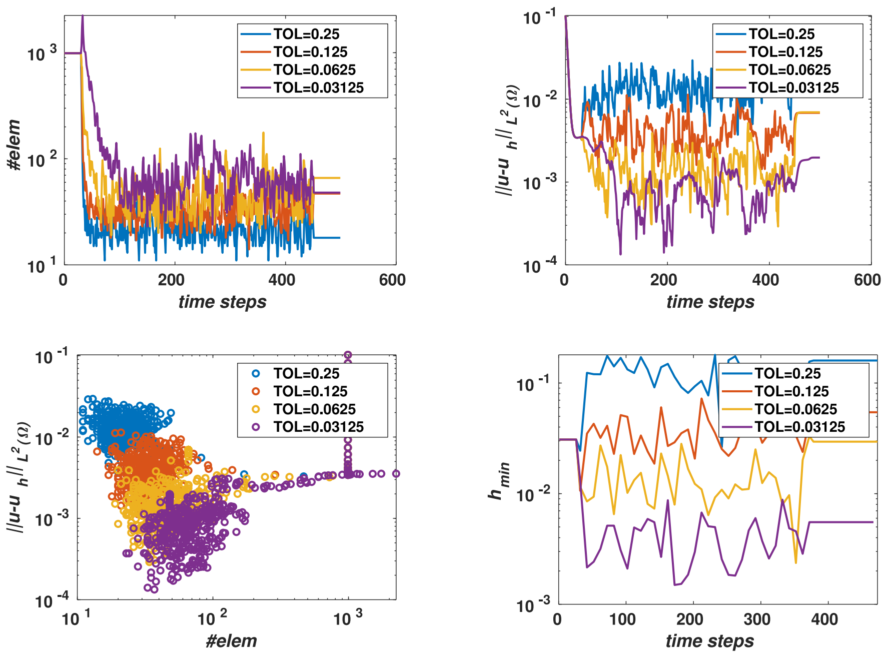

Fig. 6. Simple folding (case $\varepsilon_1 = 0.0$). Visualization of the behavior of the iterative algorithm. Top left: Visualization of the time evolution of the number of elements; Top right: Visualization of the time evolution of the error $\|\mathbf{u} - \mathbf{u}_h\|_{L^2(\Omega)}$; Bottom left: Visualization of the relationship between the error $\|\mathbf{u} - \mathbf{u}_h\|_{L^2(\Omega)}$ vs the number of elements; Bottom right: Visualization of the time evolution of $h_{\min}$.

Table 1
Simple folding. Convergence behavior of the algorithm for various values of parameter ε_1, as a function of the tolerance *TOL*. The columns contain the final minimal and maximal mesh sizes, the final numbers of elements and nodes, the maximal value of the aspect ratio, the value of the estimator, the effectivity index, and the L^2-norm on the approximation $\mathbf{u}_h$ of the solution map $\mathbf{u}$.

Regularization term: $\varepsilon_1 = 2 \cdot 10^{-6}$								
TOL	h_{min}	h_{max}	*AR*	# elem	# nodes	η_K^A	$\frac{\eta_K^A}{\|\mathbf{u}-\mathbf{u}_h\|_{H^1}}$	$\|\mathbf{u} - \mathbf{u}_h\|_{L^2}$
0.250000	7.86e−02	9.96e−01	10.57	24	19	4.49e−01	1.33 ± 0.4224	1.52e−02
0.125000	3.50e−02	1.00e+00	11.11	35	26	2.34e−01	1.18 ± 0.2705	4.56e−03
0.006250	1.09e−02	9.84e−01	29.27	40	28	1.38e−01	1.11 ± 0.3452	2.03e−03
0.031250	3.22e−03	9.83e−01	94.54	64	43	6.51e−02	0.86 ± 0.1584	8.88e−04
Regularization term: $\varepsilon_1 = 0.0$								
TOL	h_{min}	h_{max}	*AR*	# elem	# nodes	η_K^A	$\frac{\eta_K^A}{\|\mathbf{u}-\mathbf{u}_h\|_{H^1}}$	$\|\mathbf{u} - \mathbf{u}_h\|_{L^2}$
0.250000	1.09e−01	1.00e+00	5.10	21	17	4.46e−01	1.39 ± 0.3156	1.36e−02
0.125000	2.45e−02	9.95e−01	19.24	39	28	2.14e−01	1.03 ± 0.2432	4.89e−03
0.006250	1.20e−02	9.90e−01	26.58	53	36	1.34e−01	0.93 ± 0.2517	2.85e−03
0.031250	4.53e−03	9.81e−01	46.53	61	40	7.47e−02	0.86 ± 0.1453	1.12e−03

time evolution of the error $\|\mathbf{u} - \mathbf{u}_h\|_{L^2(\Omega)}$; both allow to conclude to the convergence of the algorithm when the tolerance decreases. The bottom left figure shows the relationship between the error $\|\mathbf{u} - \mathbf{u}_h\|_{L^2(\Omega)}$ and the number of elements (at each time iteration). It shows that, in average in time, a smaller tolerance leads to a smaller error and a larger number of elements, and the relationship seems to be linear. The bottom right figure illustrates the behavior of $h_{\min}$ versus time.

5.2. Double diagonal folding

The second example corresponds to a double folding, along the diagonals of the unit square domain. The exact solution of this problem is

$$\begin{aligned} u_1(x_1, x_2) &= d(\mathbf{x}, \partial\Omega), \\ u_2(x_1, x_2) &= \begin{cases} \min(x_2, 1 - x_1) & \text{if } x_1 < x_2, \\ \min(x_1, 1 - x_2) & \text{otherwise}, \end{cases} \qquad \forall \mathbf{x} = (x_1, x_2) \in \Omega, \end{aligned}$$

Table 2
Simple folding. Convergence behavior of the algorithm for various regularization parameters ε_1, as a function of the tolerance *TOL*. The columns contain the constraints for the orthogonality of the solution.

Regularization term: $\varepsilon_1 = 2 \cdot 10^{-6}$			
TOL	$\int_\Omega \lvert\nabla u_{1,h}\rvert d\mathbf{x}$	$\int_\Omega \lvert\nabla u_{2,h}\rvert d\mathbf{x}$	$\int_\Omega \lvert\nabla u_{1,h} \cdot \nabla u_{2,h}\rvert d\mathbf{x}$
0.2500	0.9914	1.0002	0.0417
0.1250	0.9975	1.0001	0.0192
0.0625	0.9950	1.0000	0.0101
0.03125	0.9982	1.0000	0.0049
Regularization term: $\varepsilon_1 = 0.0$			
TOL	$\int_\Omega \lvert\nabla u_{1,h}\rvert d\mathbf{x}$	$\int_\Omega \lvert\nabla u_{2,h}\rvert d\mathbf{x}$	$\int_\Omega \lvert\nabla u_{1,h} \cdot \nabla u_{2,h}\rvert d\mathbf{x}$
0.0312	0.9982	1.0000	0.0049
0.1250	0.9929	1.0000	0.0221
0.0625	0.9959	1.0000	0.0165
0.03125	1.0004	1.0000	0.0010

Table 3
Double diagonal folding. Convergence behavior of the algorithm for various values of parameter ε_1, as a function of the tolerance *TOL*. The columns contain the final minimal and maximal mesh sizes, the final numbers of elements and nodes, the maximal value of the aspect ratio, the value of the estimator, the effectivity index, and the L^2-norm on the approximation $\mathbf{u}_h$ of the solution map $\mathbf{u}$.

Regularization term: $\varepsilon_1 = 2 \cdot 10^{-6}$								
TOL	h_{min}	h_{max}	*AR*	# elem	# nodes	η_K^A	$\frac{\eta_K^A}{\lVert\mathbf{u}-\mathbf{u}_h\rVert_{H^1}}$	$\lVert\mathbf{u}-\mathbf{u}_h\rVert_{L^2}$
0.250000	4.48e−02	4.80e−01	8.72	82	54	5.28e−01	1.18 ± 0.2049	1.74e−02
0.125000	1.42e−02	4.96e−01	18.63	157	93	2.96e−01	0.97 ± 0.0853	6.75e−03
0.006250	3.65e−03	4.81e−01	66.06	309	173	1.48e−01	0.81 ± 0.0993	2.92e−03
0.031250	7.75e−04	4.97e−01	208.35	537	300	7.42e−02	0.61 ± 0.0518	1.09e−03
Regularization term: $\varepsilon_1 = 0.0$								
TOL	h_{min}	h_{max}	*AR*	# elem	# nodes	η_K^A	$\frac{\eta_K^A}{\lVert\mathbf{u}-\mathbf{u}_h\rVert_{H^1}}$	$\lVert\mathbf{u}-\mathbf{u}_h\rVert_{L^2}$
0.250000	4.42e−02	4.53e−01	7.69	86	56	5.25e−01	1.15 ± 0.1645	1.70e−02
0.125000	1.52e−02	4.71e−01	15.93	164	97	2.97e−01	1.03 ± 0.1109	6.60e−03
0.006250	4.82e−03	5.05e−01	50.25	265	150	1.56e−01	0.88 ± 0.1259	2.63e−03
0.031250	1.57e−03	5.14e−01	120.01	456	248	7.76e−02	0.69 ± 0.0767	1.15e−03

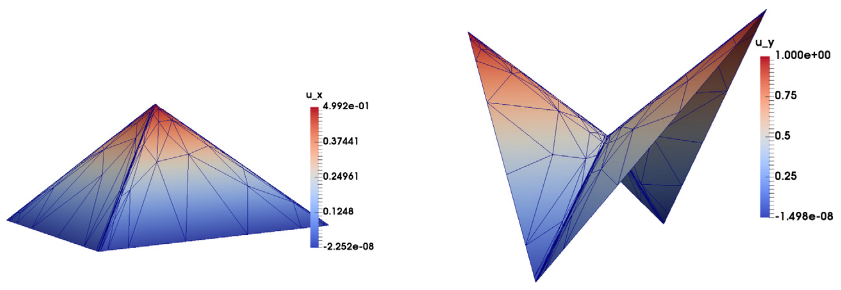

Fig. 7. Double diagonal folding. Snapshots of the approximated solution (left: first component $u_{1,h}$; right: second component $u_{2,h}$), with illustration of the final adapted mesh.

and the boundary conditions are set accordingly. The additional difficulty lies in the intersection of two lines of the singular set. Fig. 7 illustrates a snapshot of the stationary solution, together with an illustration of an adaptive mesh (in that case, $h_{\min} = 1.57 \cdot 10^{-3}$, $h_{\max} = 5.14 \cdot 10^{-1}$, $TOL = 0.03125$, $\varepsilon_1 = 0.0$).

Fig. 8 illustrates, for $\varepsilon_1 = 0.0$, the refined mesh when the tolerance decreases. Again, the number of elements increases in a narrow neighborhood around the line singularities. Table 3 numerically confirms this statement. It also emphasizes that the convergence properties are comparable when $\varepsilon_1 \neq 0$ and when $\varepsilon_1 = 0$ (both in terms of accuracy and convergence rate). Note that using smaller tolerances usually requires a larger number of iterations, and may require a continuation approach (namely starting the time iterations with a larger tolerance and decreasing it as the iterations go, as, e.g., in [7]).

Fig. 9 illustrates the influence of the regularization parameter ε_1. When too large, the singularities of the solution are lost, and the anisotropic mesh adaptation algorithm does not converge easily.

Table 3 numerically confirms that (i) the smaller the tolerance, the larger the number of elements, nodes and aspect ratio; (ii) the estimator η_K^A and the error in the L^2-norm are divided by $1.8 \sim 2.6$ when the tolerance is divided by two; (iii) however, the effectivity index does not remain constant.

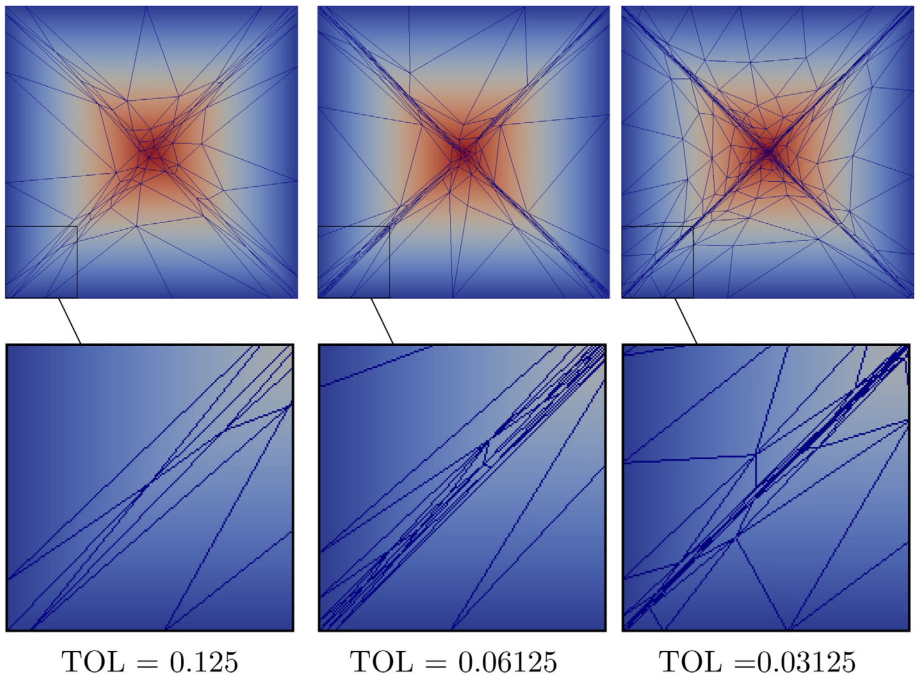

Fig. 8. Double diagonal folding. Snapshots of the final adapted mesh after 500 time iterations, for various values of the tolerance TOL ($\varepsilon_1 = 0.0$). The colormap represents the values of the first component $u_{1,h}$. The second row corresponds to a zoom in the squared region indicated in the first row.

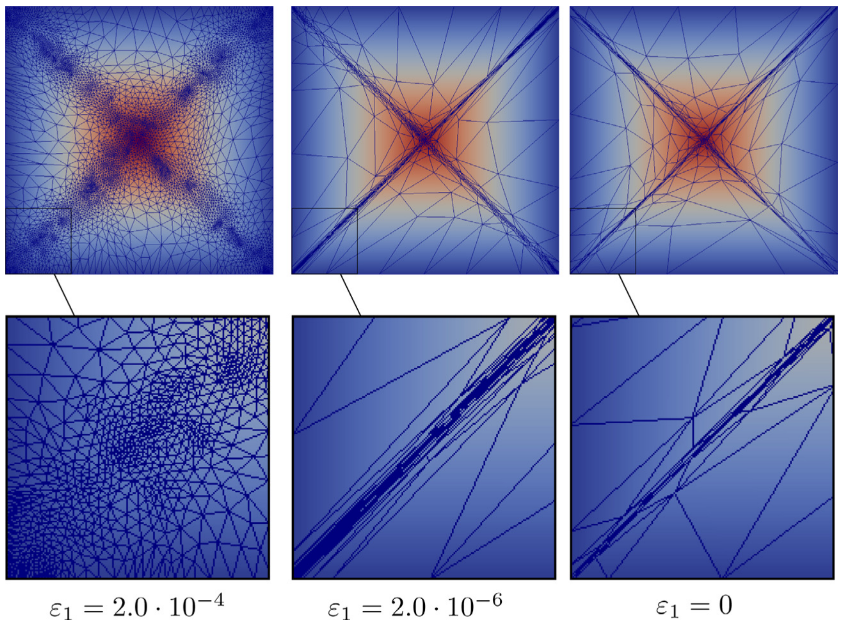

Fig. 9. Double diagonal folding. Snapshots of the final adapted mesh after 500 time iterations, for various values of the regularization parameter ε_1 ($TOL = 0.03125$). The colormap represents the values of the first component $u_{1,h}$. The second row corresponds to a zoom in the squared region indicated in the first row.

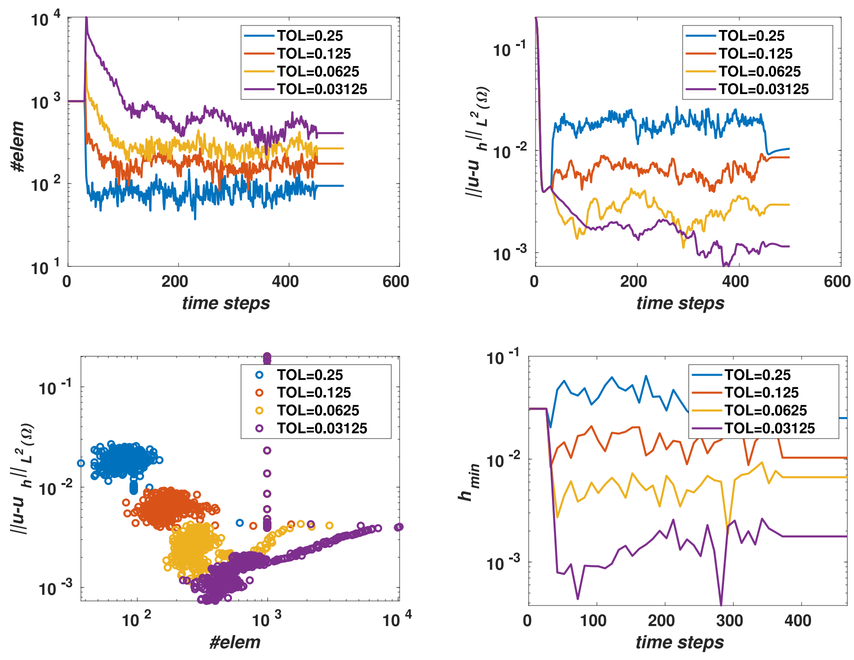

Fig. 10. Double diagonal folding (case $\varepsilon_1 = 0.0$). Visualization of the behavior of the iterative algorithm. Top left: Visualization of the time evolution of the number of elements; Top right: Visualization of the time evolution of the error $\|\mathbf{u} - \mathbf{u}_h\|_{L^2(\Omega)}$; Bottom left: Visualization of the relationship between the error $\|\mathbf{u} - \mathbf{u}_h\|_{L^2(\Omega)}$ vs the number of elements; Bottom right: Visualization of the time evolution of $h_{\min}$.

Table 4
Double diagonal folding. Convergence behavior of the algorithm for various regularization parameters ε_1, as a function of the tolerance *TOL*. The columns contain the constraints for the orthogonality of the solution.

Regularization term: $\varepsilon_1 = 2 \cdot 10^{-6}$			
TOL	$\int_\Omega \lvert\nabla u_{1,h}\rvert d\mathbf{x}$	$\int_\Omega \lvert\nabla u_{2,h}\rvert d\mathbf{x}$	$\int_\Omega \lvert\nabla u_{1,h} \cdot \nabla u_{2,h}\rvert d\mathbf{x}$
0.2500	0.9785	0.9899	0.1111
0.1250	0.9957	0.9960	0.0432
0.0625	0.9961	0.9957	0.0288
0.03125	0.9957	0.9946	0.0180
Regularization term: $\varepsilon_1 = 0.0$			
TOL	$\int_\Omega \lvert\nabla u_{1,h}\rvert d\mathbf{x}$	$\int_\Omega \lvert\nabla u_{2,h}\rvert d\mathbf{x}$	$\int_\Omega \lvert\nabla u_{1,h} \cdot \nabla u_{2,h}\rvert d\mathbf{x}$
0.2500	0.9896	0.9893	0.0763
0.1250	0.9976	0.9988	0.0409
0.0625	0.9988	0.9986	0.0237
0.03125	0.9997	0.9995	0.0100

Table 4 shows that the orthogonality constraints are satisfied and converge asymptotically. The accuracy in approximating the orthogonality conditions is better when $\varepsilon_1 = 0.0$.

Fig. 10 illustrates the iterative behavior of the time-stepping algorithm for various tolerances when $\varepsilon_1 = 0.0$; it shows that the time evolution of indicators is indeed oscillating when the mesh is adapted, due to the low regularity of the solution. The conclusions are similar to those of the single folding example.

Finally, Table 5 illustrates the computational cost of the algorithm for the approximation of the solution to this test case. We can observe that the algorithm is faster when considering no smoothing ($\varepsilon_1 = 0.0$), as it converges more rapidly to the solution with sharp edges. On the other hand, the introduction of the smoothing parameter $\varepsilon_1 = 2 \cdot 10^{-6}$ diminishes the oscillations but increases the total CPU time.

5.3. Comparison with a standard adaptive approach

We actually advocate here a non-standard adaptive method when compared to the literature about mesh adaptive methods for elliptic problems. Indeed, the more standard approach [38,39] would be to solve the entire time-stepping

Table 5
Double diagonal folding. Computational cost (total CPU time) of the algorithm for various values of parameter ε_1, and various tolerances *TOL*.

Regularization term	$\varepsilon_1 = 2 \cdot 10^{-6}$	$\varepsilon_1 = 0.0$
TOL	CPU time	CPU time
0.250000	1 min 55 s	1 min 57 s
0.125000	2 min 21 s	2 min 20 s
0.006250	2 min 59 s	2 min 55 s
0.031250	6 min 60 s	5 min 32 s

Table 6
Double diagonal folding. Convergence behavior of the algorithm for various values of parameter ε_1, as a function of the tolerance using the standard adaptive strategy. The columns contain the final minimal and maximal mesh sizes, the final numbers of elements and nodes, the maximal value of the aspect ratio, the value of the estimator, the effectivity index, and the L^2-norm on the approximation $\mathbf{u}_h$ of the solution map $\mathbf{u}$.

Regularization term: $\varepsilon_1 = 2 \cdot 10^{-6}$								
TOL	h_{min}	h_{max}	*AR*	# elem	# nodes	η_K^A	$\frac{\eta_K^A}{\|\mathbf{u}-\mathbf{u}_h\|_{H^1}}$	$\|\mathbf{u}-\mathbf{u}_h\|_{L^2}$
0.250000	5.00e−02	4.82e−01	7.34	72	47	5.51e−01	1.44 ± 0.2884	1.42e−02
0.125000	1.54e−02	5.20e−01	20.72	161	95	2.74e−01	1.16 ± 0.1154	4.61e−03
0.006250	4.42e−03	3.96e−01	27.64	447	242	1.51e−01	0.81 ± 0.0623	4.05e−03
0.031250	1.48e−03	2.86e−01	23.36	1564	815	7.43e−02	0.67 ± 0.0597	7.96e−04
Regularization term: $\varepsilon_1 = 0.0$								
TOL	h_{min}	h_{max}	*AR*	# elem	# nodes	η_K^A	$\frac{\eta_K^A}{\|\mathbf{u}-\mathbf{u}_h\|_{H^1}}$	$\|\mathbf{u}-\mathbf{u}_h\|_{L^2}$
0.250000	4.86e−02	4.56e−01	5.52	95	59	5.45e−01	1.37 ± 0.1456	1.35e−02
0.125000	1.65e−02	4.65e−01	13.18	157	92	2.90e−01	1.05 ± 0.1022	7.49e−03
0.006250	3.60e−03	3.98e−01	25.83	514	277	1.49e−01	0.84 ± 0.0618	2.35e−03
0.031250	1.07e−03	2.84e−01	30.35	1462	759	7.60e−02	0.55 ± 0.0357	1.62e−03

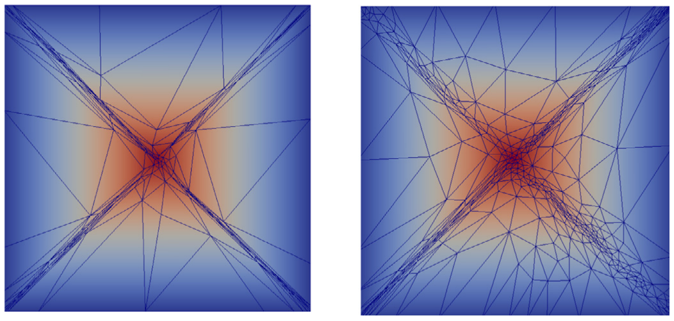
Fig. 11. Double diagonal folding. Snapshots of the final adapted mesh (the colormap represents the values of the first component $u_{1,h}$). Left: non-standard approach advocated here, after 500 time iterations; right: standard approach, after 4000 time iterations. ($TOL = 0.625$, $\varepsilon_1 = 0$).

problem, then to adapt the mesh and re-apply the whole solution method for the time-dependent problem. In the adaptive strategy described in Section 4.3, a few time iterations are performed without reaching the stationary solution with a fixed mesh, then the mesh is adapted at each time iteration. Actually the variations in the solution occur slowly and locally (on the edges where the singularities are formed), which favors a mesh adaptivity at each iteration. The other advantage of the adaptive approach described in Section 4.3 is that it allows to recover a suitable mesh faster compare to the standard strategy.

Fig. 11 illustrates a comparison between the different approaches (with $TOL = 0.625$ and $\varepsilon_1 = 0$). The figure on the left shows the numerical solution using the adaptive approach (Section 4.3) after 500 iterations (total of time iterations and adaptive remeshing steps at each time iteration). The figure on the right shows the numerical solution applying the standard approach after 4000 iterations (same total). Fig. 11 indicates that the mesh obtained with the adaptive approach tracks more efficiently the singularities with a smaller total number of iterations.

Table 6 shows the numerical behavior of the algorithm using the standard adaptive approach for varying parameters, namely the values of the parameters and final adapted meshes for all tolerances and $\varepsilon_1 = 2 \cdot 10^{-6}$ and $\varepsilon_1 = 0$.

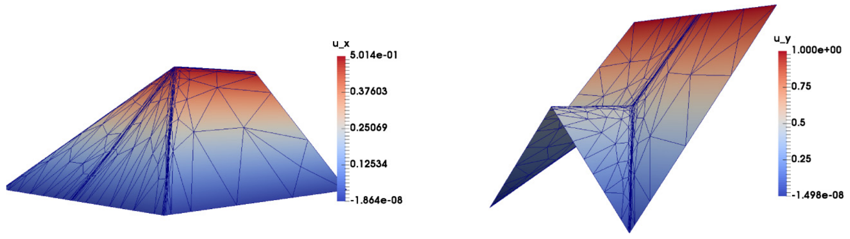

Fig. 12. Non-smooth folding with point singularity. Snapshots of the approximated solution (left: first component $u_{1,h}$; right: second component $u_{2,h}$), with illustration of the final adapted mesh.

Table 7
Non-smooth folding with point singularity. Convergence behavior of the algorithm for various values of parameter ε_1, as a function of the tolerance *TOL*. The columns contain the final minimal and maximal mesh sizes, the final numbers of elements and nodes, the maximal value of the aspect ratio, the value of the estimator, the effectivity index, and the L^2-norm on the approximation $\mathbf{u}_h$ of the solution map $\mathbf{u}$.

Regularization term: $\varepsilon_1 = 2 \cdot 10^{-6}$								
TOL	h_{min}	h_{max}	*AR*	# elem	# nodes	η_K^A	$\frac{\eta_K^A}{\lVert\mathbf{u}-\mathbf{u}_h\rVert_{H^1}}$	$\lVert\mathbf{u}-\mathbf{u}_h\rVert_{L^2}$
0.250000	4.83e−02	6.66e−01	8.14	60	41	5.11e−01	1.15 ± 0.1605	2.00e−02
0.125000	1.55e−02	6.58e−01	11.97	136	82	2.86e−01	0.98 ± 0.0875	8.18e−03
0.006250	4.59e−03	6.81e−01	37.89	259	149	1.44e−01	0.81 ± 0.1469	3.39e−03
0.031250	6.55e−04	5.81e−01	225.45	545	301	7.71e−02	0.64 ± 0.0697	1.48e−03
Regularization term: $\varepsilon_1 = 0.0$								
TOL	h_{min}	h_{max}	*AR*	# elem	# nodes	η_K^A	$\frac{\eta_K^A}{\lVert\mathbf{u}-\mathbf{u}_h\rVert_{H^1}}$	$\lVert\mathbf{u}-\mathbf{u}_h\rVert_{L^2}$
0.250000	3.59e−02	6.92e−01	15.49	64	44	5.08e−01	1.08 ± 0.1142	2.16e−02
0.125000	1.61e−02	7.11e−01	15.97	122	75	2.88e−01	1.06 ± 0.1573	6.25e−03
0.006250	4.81e−03	6.82e−01	33.88	233	134	1.52e−01	0.86 ± 0.0989	2.59e−03
0.031250	1.78e−03	5.81e−01	49.02	427	234	7.71e−02	0.77 ± 0.1375	9.73e−04

5.4. Non-smooth example with a point singularity

The third example also corresponds to a double, re-entrant, folding. The exact solution of this problem is

$$\begin{aligned}
u_1(x_1,x_2) &= \begin{cases} x_1 & \text{if } x_2 \geq x_1 \text{ and } x_1 \leq 0.5, \\ 1-x_1 & \text{if } x_1 > 0.5 \text{ and } x_2 > -x_1+1, \\ x_2 & \text{if } x_2 \leq x_1 \text{ and } x_2 \leq -x_1+1, \end{cases} \\
u_2(x_1,x_2) &= \begin{cases} x_2 & \text{if } x_2 \geq x_1 \text{ and } x_1 \leq 0.5, \\ x_2 & \text{if } x_1 > 0.5 \text{ and } x_2 > -x_1+1, \\ x_1 & \text{if } x_2 \leq x_1 \text{ and } x_1 \leq 0.5, \\ 1-x_1 & \text{if } x_2 > x_1 \text{ and } x_2 \leq -x_1+1, \end{cases}
\end{aligned} \qquad \forall (x_1,x_2) \in \Omega,$$

and the boundary conditions are set accordingly. We consider a fixed number of $n_{mesh} = 500$ time steps, and, for this example, we take $C = 0$. The additional difficulty lies in the refolding with a re-entrant corner, which causes a new type of point singularity. For this specific test problem, in the case when $TOL = 0.03125$, the algorithm fails to converge for $n_{mesh} = 500$. In order to accelerate the convergence the adaptive algorithm uses gradually decreasing tolerances TOL (i.e. first a few iterations are performed with $TOL = 0.125$ then the tolerance decreases to $TOL = 0.0625$ and $TOL = 0.03125$ successively).

Fig. 12 illustrates a snapshot of the stationary solution, together with an illustration of an adaptive mesh (in that case, $h_{\min} = 1.78 \cdot 10^{-3}$, $h_{\max} = 5.81 \cdot 10^{-1}$, $TOL = 0.03125$, $\varepsilon_1 = 0.0$). Figs. 13 and 14 illustrate the adapted mesh for $\varepsilon_1 = 0.0$ when the tolerance decreases, and for $TOL = 0.03125$ when the penalization parameter varies. Similar conclusions to the previous test cases hold.

Table 7 numerically confirms the conclusions reached earlier. Table 8 shows an appropriate behavior for the orthogonality constraints that converge asymptotically when the tolerance decreases (except for one result for the smaller tolerance and $\varepsilon_1 \neq 0.0$). As expected, the accuracy in approximating the orthogonality conditions is higher when $\varepsilon_1 = 0.0$.

Fig. 15 illustrates the iterative behavior of the time-stepping algorithm for various tolerances when $\varepsilon_1 = 0.0$ respectively; it confirms the oscillatory behavior of the time evolution of indicators when the mesh is adapted, due to the low regularity of the solution. Fig. 16 illustrates the same iterative behavior when $\varepsilon_1 = 2.0 \cdot 10^{-6}$; it confirms that the oscillatory behavior of the algorithm can be decreased by the introduction of the smoothing parameter. Table 9 illustrates the numerical behavior of the approximated solution for various ε_1. Again, when ε_1 is too large, the algorithm

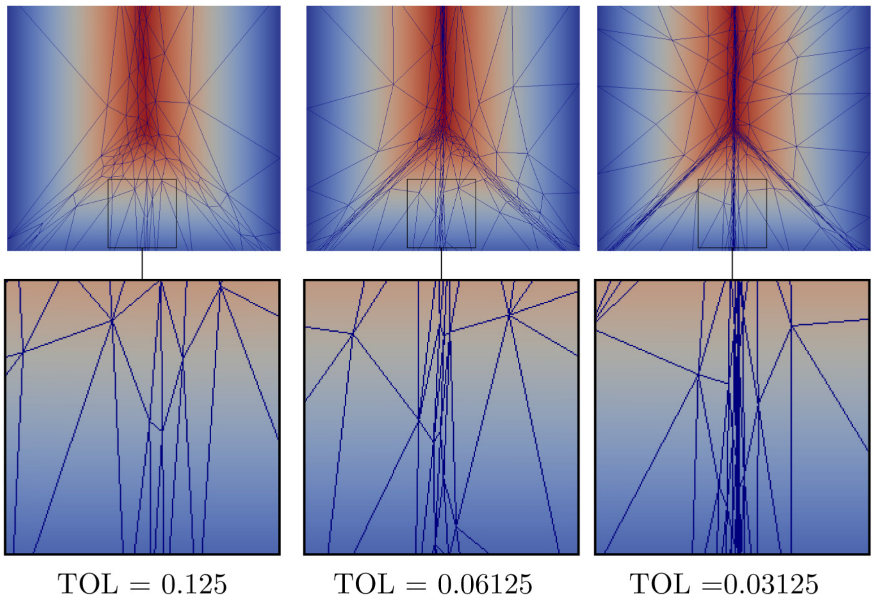

Fig. 13. Non-smooth folding with point singularity. Snapshots of the final adapted mesh after 500 time iterations, for various values of the tolerance TOL ($\varepsilon_1 = 0.0$). The colormap represents the values of the first component $u_{1,h}$. The second row corresponds to a zoom in the squared region indicated in the first row.

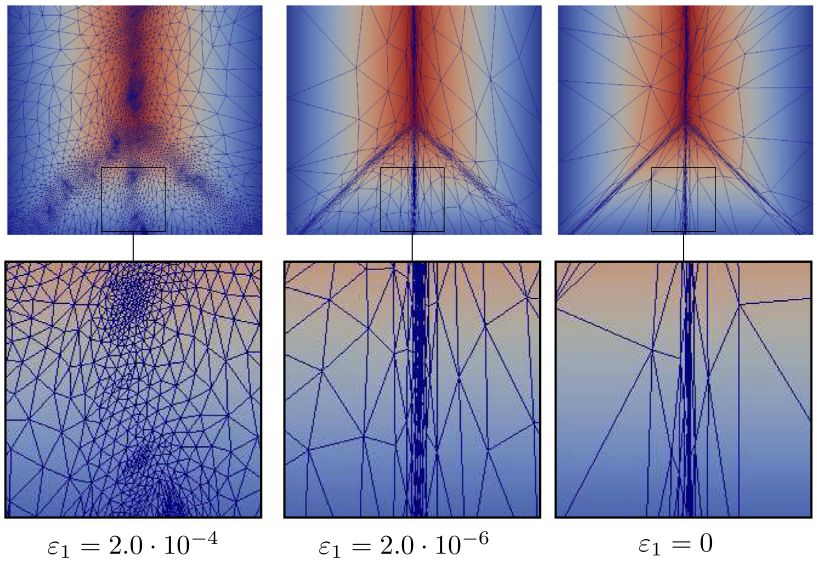

Fig. 14. Non-smooth folding with point singularity. Snapshots of the final adapted mesh after 500 time iterations, for various values of the regularization parameter ε_1 ($TOL = 0.03125$). The colormap represents the values of the first component $u_{1,h}$. The second row corresponds to a zoom in the squared region indicated in the first row.

under-performs. Then, we can observe that, when $\varepsilon_1 \to 0.0$, the number of elements and nodes remains bounded despite the loss of regularity of the solution. Similarly the orthogonality conditions become more accurate when $\varepsilon_1 \to 0.0$.

5.5. Comparison of results with and without the adaptive mesh refinement algorithm

This test case is a practical application of foldable compliant mechanism device using thick material, see, e.g., [40]. Here we emphasize and quantify the added value of the adaptive mesh refinement algorithm for such a practical application. Table 10 first illustrates the computational cost of the adaptive algorithm. The same remarks as for the previous example hold: the algorithm is faster without the introduction of the smoothing term, but its behavior is more oscillatory.

Table 8
Non-smooth folding with point singularity. Convergence behavior of the algorithm for various regularization parameters ε_1, as a function of the tolerance *TOL*. The columns contain the constraints for the orthogonality of the solution.

Regularization term: $\varepsilon_1 = 2 \cdot 10^{-6}$			
TOL	$\int_\Omega \lvert\nabla u_{1,h}\rvert d\mathbf{x}$	$\int_\Omega \lvert\nabla u_{2,h}\rvert d\mathbf{x}$	$\int_\Omega \lvert\nabla u_{1,h} \cdot \nabla u_{2,h}\rvert d\mathbf{x}$
0.2500	0.9780	0.9907	0.0826
0.1250	0.9933	0.9945	0.0489
0.0625	0.9968	0.9979	0.0174
0.03125	0.9959	0.9951	0.0092
Regularization term: $\varepsilon_1 = 0.0$			
TOL	$\int_\Omega \lvert\nabla u_{1,h}\rvert d\mathbf{x}$	$\int_\Omega \lvert\nabla u_{2,h}\rvert d\mathbf{x}$	$\int_\Omega \lvert\nabla u_{1,h} \cdot \nabla u_{2,h}\rvert d\mathbf{x}$
0.2500	0.9754	0.9947	0.1126
0.1250	0.9899	0.9956	0.0493
0.0625	0.9984	0.9994	0.0187
0.0312	0.9992	0.9995	0.0074

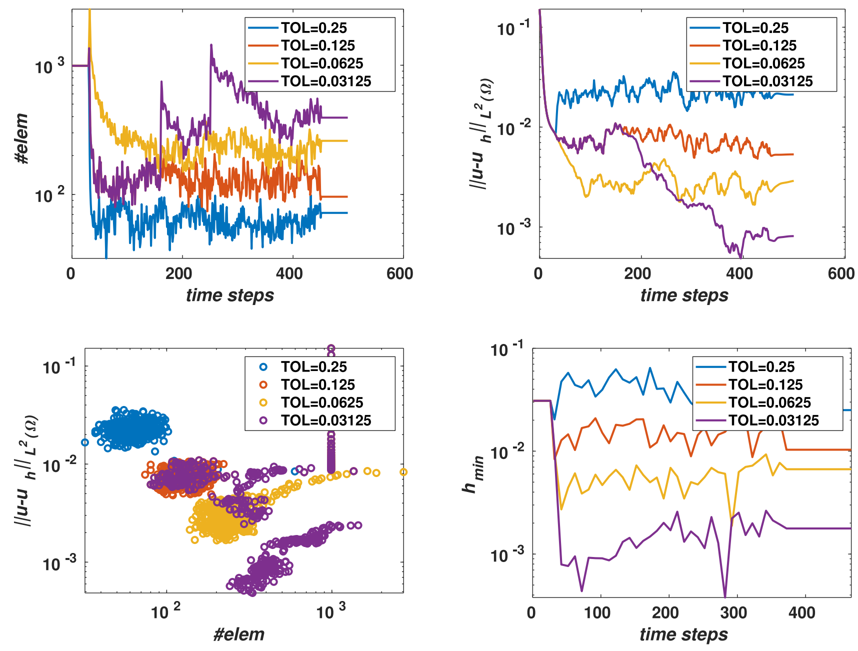

Fig. 15. Non-smooth folding with point singularity (case $\varepsilon_1 = 0.0$). Visualization of the behavior of the iterative algorithm. Top left: Visualization of the time evolution of the number of elements; Top right: Visualization of the time evolution of the error $\|\mathbf{u} - \mathbf{u}_h\|_{L^2(\Omega)}$; Bottom left: Visualization of the relationship between the error $\|\mathbf{u} - \mathbf{u}_h\|_{L^2(\Omega)}$ vs the number of elements; Bottom right: Visualization of the time evolution of $h_{\min}$.

Table 9
Non-smooth folding with point singularity. Convergence behavior of the algorithm as a function of the regularization parameter ε_1 (tolerance: *TOL* = 0.3125).

ε_1	h_{min}	h_{max}	# elem	# nodes	$\|\mathbf{u} - \mathbf{u}_h\|_{L^2}$
2.0e−04	7.66e−04	2.35e−01	3869	2095	1.25e−04
2.0e−06	6.55e−04	5.81e−01	545	301	1.48e−03
2.0e−08	1.70e−03	4.70e−01	422	232	4.16e−04
0.0	1.78e−03	5.81e−01	427	234	9.73e−04
ε_1	$\int_\Omega \lvert\nabla u_{1,h}\rvert d\mathbf{x}$	$\int_\Omega \lvert\nabla u_{2,h}\rvert d\mathbf{x}$	$\int_\Omega \lvert\nabla u_{1,h} \cdot \nabla u_{2,h}\rvert d\mathbf{x}$		
2.0e−04	0.9529	0.9496	0.0641		
2.0e−06	0.9960	0.9954	0.0113		
2.0e−08	0.9992	0.9996	0.0117		
0.0	0.9991	0.9998	0.0074		

Tables 11 and 12 show the numerical results for the algorithm without mesh adaptation. In order to compare the algorithms with and without adaptive mesh refinement, we consider the appropriate tolerance that leads to the right minimal mesh size considered without the adaptive algorithm. Comparing Tables 8 and 12, we can see that the accuracy

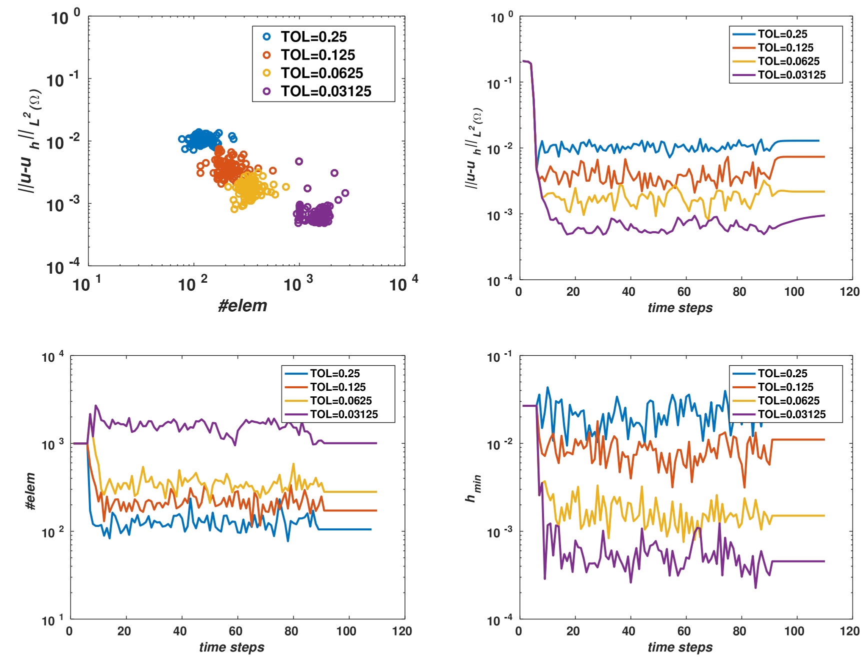

Fig. 16. Non-smooth folding with point singularity (case $\varepsilon_1 = 2.0{\cdot}10^{-6}$). Visualization of the behavior of the iterative algorithm. Top left: Visualization of the time evolution of the number of elements; Top right: Visualization of the time evolution of the error $\|\mathbf{u}-\mathbf{u}_h\|_{L^2(\Omega)}$; Bottom left: Visualization of the relationship between the error $\|\mathbf{u}-\mathbf{u}_h\|_{L^2(\Omega)}$ vs the number of elements; Bottom right: Visualization of the time evolution of $h_{\min}$.

Table 10
Non-smooth folding with point singularity. Computational cost (total CPU time) of the algorithm for various values of parameter ε_1, and various tolerances *TOL*.

Regularization term	$\varepsilon_1 = 2 \cdot 10^{-6}$	$\varepsilon_1 = 0.0$
TOL	CPU time	CPU time
0.250000	2 min 2 s	1 min 53 s
0.125000	2 min 9 s	2 min 10 s
0.006250	2 min 50 s	2 min 50 s
0.031250	3 min 51 s	3 min 24 s

Table 11
Non-smooth folding with point singularity and without adaptive mesh refinement. Convergence behavior of the algorithm for various values of parameter ε_1, as a function of the mesh size. The columns contain the fixed minimal and maximal mesh sizes, the fixed numbers of elements and nodes, the L^2-norm on the approximation $\mathbf{u}_h$ of the solution map $\mathbf{u}$, the number of iterations, and the computational time.

Regularization term: $\varepsilon_1 = h^2_{min}$						
$h_{\min}$	$h_{\max}$	# elem	# nodes	$\|\mathbf{u}-\mathbf{u}_h\|_{L^2}$	iter	CPU time
4.00e−02	5.65e−02	1250	676	1.54e−02	147	0 min 10 s
2.00e−02	2.82e−02	5000	2601	5.71e−03	332	1 min 57 s
1.00e−02	2.88e−02	20000	10201	2.88e−03	675	129 min 56 s
5.00e−03	7.07e−03	80000	40401	1.51e−03	1438	1118 min 58 s
Regularization term: $\varepsilon_1 = 0$						
$h_{\min}$	$h_{\max}$	# elem	# nodes	$\|\mathbf{u}-\mathbf{u}_h\|_{L^2}$	iter	CPU time
4.00e−02	5.65e−02	1250	676	1.06e−02	255	0 min 17 s
2.00e−02	2.82e−02	5000	2601	5.93e−03	290	1 min 36 s
1.00e−02	2.88e−02	20000	10201	5.76e−03	580	24 min 11 s
5.00e−03	7.07e−03	80000	40401	5.86e−03	2077	692 min 17 s

in the fulfillment of orthogonality conditions is very similar. Comparing Tables 7 and 11, we remark that the accuracy of the algorithm (expressed by $\|\mathbf{u}-\mathbf{u}_h\|_{L^2}$) is about 10% better with adaptive mesh refinement (for a comparable $h_{\min}$), and we obtain better convergence behavior, especially when $\varepsilon_1 = 0.0$ and small values of $h_{\min}$. Finally comparing the computational costs in Tables 10 and 11, we observe a significant computational gain when introducing the adaptive mesh algorithm. This illustrates the improved efficiency of the algorithm.

Table 12
Non-smooth folding with point singularity and without adaptive mesh refinement. Convergence behavior of the algorithm for various regularization parameters ε_1, as a function of the fixed mesh size $h_{\min}$. The columns contain the constraints for the orthogonality of the solution.

Regularization term: $\varepsilon_1 = h^2_{min}$			
$h_{\min}$	$\int_\Omega \lvert\nabla u_{1,h}\rvert d\mathbf{x}$	$\int_\Omega \lvert\nabla u_{2,h}\rvert d\mathbf{x}$	$\int_\Omega \lvert\nabla u_{1,h} \cdot \nabla u_{2,h}\rvert d\mathbf{x}$
4.00e−02	0.9703	0.9667	0.0422
2.00e−02	0.9852	0.9831	0.0219
1.00e−02	0.9926	0.9915	0.0113
5.00e−03	0.9963	0.9957	0.0058
Regularization term: $\varepsilon_1 = 0.0$			
$h_{\min}$	$\int_\Omega \lvert\nabla u_{1,h}\rvert d\mathbf{x}$	$\int_\Omega \lvert\nabla u_{2,h}\rvert d\mathbf{x}$	$\int_\Omega \lvert\nabla u_{1,h} \cdot \nabla u_{2,h}\rvert d\mathbf{x}$
4.00e−02	0.9886	0.9895	0.0476
2.00e−02	0.9948	0.9851	0.0202
1.00e−02	0.9971	0.9967	0.0182
5.00e−03	0.9982	0.9977	0.0155

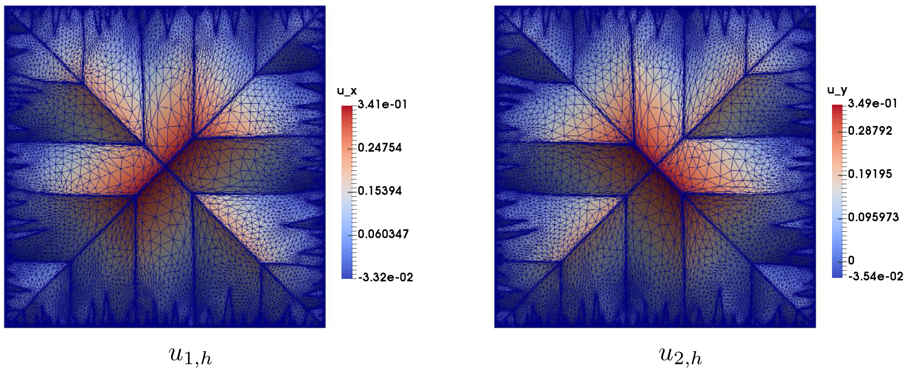

Fig. 17. Homogeneous Dirichlet test case. Snapshots of the approximated solution (left: first component $u_{1,h}$; right: second component $u_{2,h}$), with illustration of the final adapted mesh consisting of an unstructured adapted triangulation with 22,871 vertices and 42,987 triangles ($C = 0$, $\Delta t = 5 \cdot 10^{-12}$, $\varepsilon_1 = 0$, $TOL = 0.625$, approx. 3000–5000 time steps).

6. A decomposition approach for the homogeneous Dirichlet problem

In this final section, let us consider the, purely academic, homogeneous Dirichlet problem:

$$\begin{cases} \nabla \mathbf{u} \in \mathcal{O}(2) & \text{a.e. in } \Omega, \\ \mathbf{u} = \mathbf{0} & \text{on } \partial\Omega. \end{cases} \tag{18}$$

In this particular case, the solution becomes fractal near the boundary (see, e.g., [1], but also [11] for a similar behavior for a scalar Eikonal equation). This test case with homogeneous boundary conditions is mainly an academic example. It is addressed last as it is the most stringent case for the orthogonal maps problem, as the solution maps the whole boundary into a single point that is the origin.

Preliminary numerical results reported in [7] have shown that adaptive mesh refinement is required to obtain the convergence of the time-stepping algorithm, and recover a (nearly-) admissible solution. Fig. 17 illustrates the snapshot of the numerical approximation $\mathbf{u}_h$ of $\mathbf{u}$, obtained with the anisotropic mesh adaptivity algorithm. However, sharp edges cannot be recovered exactly near the boundary where strong oscillations arise.

One property of the solution is that every singular point should be adjacent to an even number of edges, and this number is at least four. This property is not satisfied for the numerical approximation in Fig. 17 due to these instabilities near the boundary.

In order to overcome the introduction of such instabilities and numerically capture one solution, we advocate a *domain decomposition algorithm* to approximate one given solution of (18). Using the geometric information about the expected oscillatory behavior of the solution near the boundary of $\Omega = (0, 4)^2$, we define a sequence of domains Ω_k, $k \geq 0$, such that

$$\overline{\Omega} = \bigcup_{k\geq 0} \overline{\Omega_k},$$

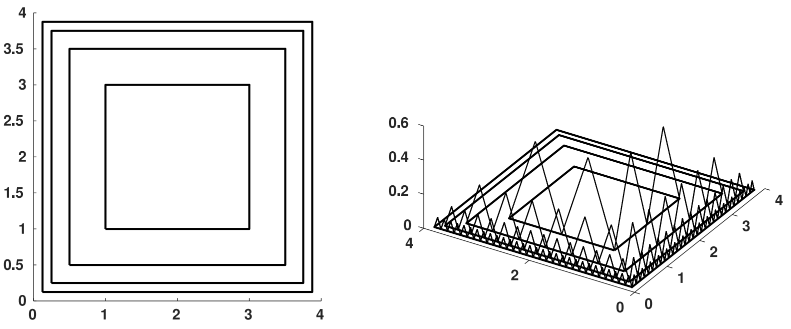

Fig. 18. Decomposition approach for the solution of the homogeneous Dirichlet problem. Left: sketch of the sequence of domains Ω_k; Right: sketch of the shape of (piecewise linear) boundary conditions.

with

$$\Omega_0 = (1,3)^2, \quad \text{and } \Omega_k = \left(1 - \frac{1}{k}, 3 + \frac{1}{k}\right)^2 \backslash \overline{\Omega_{k-1}}, \quad k \geq 1.$$

Numerically, the sequence is truncated to M domains, and $\Omega_M = (0,4)^2 \backslash \overline{\Omega_{M-1}}$. Fig. 18 (left) illustrates the situation for a decomposition in four domains. This decomposition allows to enforce a fractal behavior through boundary conditions. Fig. 18 (right) illustrates schematically the shape of successive boundary conditions (only on half of the boundary).

The algorithm reads as follows: for each k, the orthogonal maps problem is solved on Ω_k with given boundary conditions. We use the following notation: $\Omega_k = (a_k, b_k)^2$, $N_k = 2N_{k-1} + 4$ for $k \geq 1$, $N_0 = 4$, $h_k = \dfrac{b_k - a_k}{N_k}$ and $p_j = a_k + jh_k$, where $j = 2\ell + 1$, $\ell = 0, \dots, \frac{N_k}{2} - 1$.

The boundary conditions are written, for all $\mathbf{x} = (x_1, x_2)$ on $\partial\Omega_k$ as

$$g_{1,k}(x_1, x_2) = 0 \tag{19}$$

and

$$g_{2,k}(x_1, x_2) = \begin{cases} y - p_{2\ell}, & \text{if } p_{2\ell} \leq y \leq p_{2\ell+1}, \\ p_{2\ell+2} - y, & \text{if } p_{2\ell+1} \leq y \leq p_{2\ell+2}, \end{cases} \qquad \forall \ell = 0, 1, 2, \dots, \frac{N_k}{2} - 1, \tag{20}$$

where

$$y = \begin{cases} x_1 \text{ when } x_2 = a_k \text{ or } x_2 = b_k, \\ x_2 \text{ when } x_1 = a_k \text{ or } x_1 = b_k. \end{cases}$$

On the internal boundary, the external boundary conditions $\mathbf{g}_{k-1}$ are reproduced.

Fig. 19 illustrates the snapshot of the numerical approximation $\mathbf{u}_h$ of $\mathbf{u}$ obtained by coupling the anisotropic adaptive algorithm with this domain decomposition method and $M = 4$. Results show that the symmetries are perfectly respected, and that the spurious oscillations are controlled. As a side effect, let us note that this approach allows us to observe that the proposed adapted algorithm behaves well when solving the orthogonal maps problem in non-convex domains, as emphasized in Fig. 20. Parallelization of this domain decomposition approach can be investigated in the future for speed-up purposes.

7. Conclusions and perspectives

We have extended the operator-splitting/finite element methodology for the numerical solution of the Dirichlet problem for orthogonal maps presented in [7] by introducing an anisotropic space adaptivity method to track more accurately the singularities of the gradient of the solution. The adaptive criterion is that of a linear variational problem at each time step of the splitting algorithm, regardless of the local nonlinearities of the problem. Anisotropic adaptive techniques are based on a posteriori error estimates for a Laplace operator, and on a Zienkiewicz–Zhu estimator. Numerical experiments have illustrated an accurate tracking of the line singularities of the gradient of the solution, but have exhibited convergence properties that are sub-optimal. The effectivity index decreases when the tolerance goes to zero, implying that our estimator is not optimal. This was to be expected since it relies only on the linear variational part of the fully nonlinear equation, without incorporating neither the nonlinear operator, nor the operator splitting error. For some stringent instances (e.g. with homogeneous boundary data), the fractal behavior of the solution near the domain boundary requires a more sophisticated, heuristics-based, approach. A domain decomposition strategy has been presented to capture one particular solution.

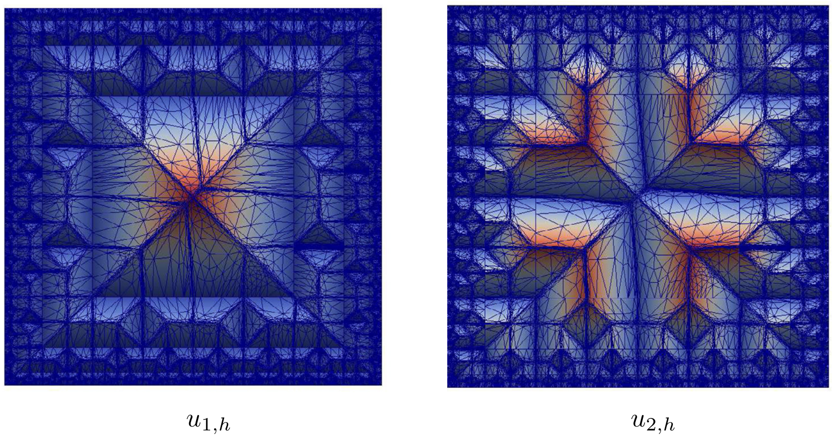

Fig. 19. Decomposition approach for the solution of the homogeneous Dirichlet problem. Snapshots of the approximated solution (left: first component $u_{1,h}$; right: second component $u_{2,h}$), with illustration of the final adapted mesh after 4150 time iterations on each subdomain, superimposed after individual computation on each subdomain. The resulting mesh obtained by superimposition of the several meshes is non-conforming.

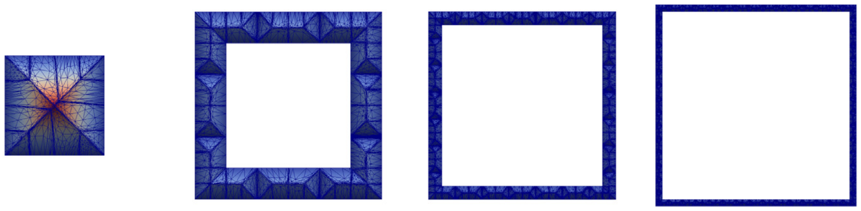

Fig. 20. Decomposition approach for the solution of the homogeneous Dirichlet problem. Snapshots of the individual solution computed on each subdomain. The colormap represents the values of the first component $u_{1,h}$.

Future work will include the investigation of a posteriori error estimates based on a more complete operator. First, error estimates for the bi-harmonic operator will be investigated, in order to incorporate the case $\varepsilon_1 \neq 0$; then the investigation should include the operator splitting error, before addressing the case of the nonlinear operator. We can study the extension of those adaptive mesh refinement methods to other fully nonlinear equations in two space dimensions, such as the second-order elliptic Monge–Ampère equation. Extensions to adaptive methods in three space dimensions can also be investigated using algorithms described in the literature, e.g., `mmg3d` [38,39], `meshadapt` [41] or `feflo` [42].

Acknowledgments

This work was partially supported by the Swiss National Science Foundation (SNF Grant 165785). The authors would like to thank Prof. R. Glowinski (Univ. of Houston, and Hong Kong Baptist University), Prof. B. Dacorogna (EPFL), and Prof. Marcellini (University of Firenze) for fruitful discussions, and the two anonymous referees for insightful comments.

References

[1] B. Dacorogna, P. Marcellini, E. Paolini, On the n-dimensional Dirichlet problem for isometric maps, J. Funct. Anal. 255 (2018) 3274–3280, http://dx.doi.org/10.1016/j.jfa.2008.10.010.
[2] B. Dacorogna, P. Marcellini, Implicit Partial Differential Equations, Birkhaüser, Basel, 1999, http://dx.doi.org/10.1007/978-1-4612-1562-2.
[3] B. Dacorogna, P. Marcellini, E. Paolini, Functions with orthogonal hessian, Differential Integral Equations 23 (2010) 51–60.
[4] B. Dacorogna, P. Marcellini, E. Paolini, Origami and partial differential equations, Notices Amer. Math. Soc. 57 (2010) 598–606.
[5] S. Bartels, Handbook of numerical analysis, in: Ch. Finite Element Simulation of Nonlinear Bending Models for Thin Elastic Rods and Plates, Elsevier, 2019, http://dx.doi.org/10.1016/bs.hna.2019.06.003.
[6] S. Bartels, A. Bonito, A.H. Muliana, R.H. Nochetto, Modeling and simulation of thermally actuated bilayer plates, J. Comput. Phys. 354 (2018) 512–528, http://dx.doi.org/10.1016/j.jcp.2017.10.044.
[7] A. Caboussat, R. Glowinski, D. Gourzoulidis, M. Picasso, Numerical approximation of orthogonal maps, SIAM J. Sci. Comput. 41 (2019) B1341–B1367, http://dx.doi.org/10.1137/19M1243683.
[8] R. Glowinski, T. Lueng, H. Liu, J. Qian, A penalization-regularization-operator splitting method for eikonal based traveltime tomography, SIAM J. Imaging Sci. 8 (2015) 1263–1292, http://dx.doi.org/10.1137/140992072.
[9] R. Glowinski, S. Osher, W. Yin, Splitting methods in communication, imaging, science, and engineering, Sci. Comput. (2018) http://dx.doi.org/10.1007/978-3-319-41589-5.

[10] A. Caboussat, R. Glowinski, A penalty-regularization-operator splitting method for the numerical solution of a scalar Eikonal equation, Chinese Ann. Math. Ser. B 36 (5) (2015) 659–688, http://dx.doi.org/10.1007/s11401-015-0930-8.
[11] A. Caboussat, R. Glowinski, T.W. Pan, On the numerical solution of some Eikonal equations: An elliptic solver approach, Chinese Ann. Math. Ser. B 36 (5) (2015) 689–702, http://dx.doi.org/10.1007/s11401-015-0971-z.
[12] S. Basterrechea, B. Dacorogna, Existence of solutions for Jacobian and Hessian equations under smallness assumptions, Numer. Funct. Anal. Optim. 35 (2014) 868–892, http://dx.doi.org/10.1080/01630563.2014.895746.
[13] B. Dacorogna, J. Moser, On a partial differential equation involving the Jacobian determinant, Ann. Inst. Henri PoincarÉ, Analyse Non LinÉaire 7 (1990) 1–26.
[14] A. Bonito, D. Guignard, R.H. Nochetto, S. Yang, Numerical analysis of the LDG method for large deformations of prestrained plates, 2021, arXiv:2106.13877.
[15] A. Bonito, D. Guignard, R.H. Nochetto, S. Yang, LDG approximation of large deformations of prestrained plates, J. Comput. Phys. 448 (2022) 110719, http://dx.doi.org/10.1016/j.jcp.2021.110719.
[16] S.W. Grey, F. Scarpa, M. Schenk, Embedded actuation for shape-adaptive origami, J. Mech. Des. 143 (8) (2021) 02, http://dx.doi.org/10.1115/1.4049880.
[17] J. Bernhard, J. Gilgen, R. Geisthövel, B.E. Marston, L. Hurni, Design principles for swiss-style rock drawing, Cartograp. J. 51 (2014) 360–371.
[18] S. Janbaz, R. Hedayati, A.A. Zadpoor, Programming the shape-shifting of flat soft matter: from self-rolling/self-twisting materials to self-folding origami, Mater. Horiz. 3 (2016) 536–547, http://dx.doi.org/10.1039/C6MH00195E.
[19] D.D. Santis, A framework for optimizing co-adaptation in body-machine interfaces, Front. Neurorobot. 15 (2021) 40, http://dx.doi.org/10.3389/fnbot.2021.662181.
[20] V. Nguyen, A. Harati, R. Siegwart, A lightweight slam algorithm using orthogonal planes for indoor mobile robotics, in: 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007, pp. 658–663, http://dx.doi.org/10.1109/IROS.2007.4399512.
[21] S.J.P. Callens, A.A. Zadpoor, From flat sheets to curved geometries: Origami and kirigami approaches, Mater. Today 21 (2018).
[22] S. Janbaz, N. Noordzij, D.S. Widyaratih, C.W. Hagen, L.E. Fratila-Apachitei, A.A. Zadpoor, Origami lattices with free-form surface ornaments, Sci. Adv. 3 (2017).
[23] T. van Manen, S. Janbaz, K. Jansen, A.A. Zadpoor, 4D printing of reconfigurable metamaterials and devices, Commun. Mater. 2 (2021) 56.
[24] L. Winkless, Origami inspires shape-shifting microelectronics, Mater. Today 31 (2019) 3–4, http://dx.doi.org/10.1016/j.mattod.2019.10.018.
[25] M. Jensen, I. Smears, On the convergence of finite element methods for Hamilton-Jacobi-Bellman equations, SIAM J. Numer. Anal. 51 (1) (2013) 137–162.
[26] L. Billon, Y. Mesri, E. Hachem, Anisotropic boundary layer mesh generation for immersed complex geometries, Eng. Comput. 33 (2) (2017) 249–260, http://dx.doi.org/10.1007/s00366-016-0469-7.
[27] A. Laadhari, P. Saramito, C. Misbah, An adaptive finite element method for the modeling of the equilibrium of red blood cells, Internat. J. Numer. Methods Fluids 80 (2015) 397–428.
[28] A. Loseille, R. Löhner, On 3D anisotropic local remeshing for surface, volume and boundary layers, in: B.W. Clark (Ed.), Proceedings of the 18th International Meshing Roundtable, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 611–630.
[29] J.L. Prieto, J. Carpio, A-SLEIPNNIR: A multiscale, anisotropic adaptive, particle level set framework for moving interfaces. transport equation applications, J. Comput. Phys. 377 (2019) 89–116, http://dx.doi.org/10.1016/j.jcp.2018.10.031.
[30] M. Shakoor, C.H. Park, A higher-order finite element method with unstructured anisotropic mesh adaption for two phase flows with surface tension, Comput. & Fluids 230 (2021) 105154.
[31] R. Glowinski, Numerical Methods for Nonlinear Variational Problems, second ed., Springer-Verlag, New York, NY, 2008, http://dx.doi.org/10.1007/978-3-662-12613-4.
[32] B.S. Kirk, J.W. Peterson, R.H. Stogner, G.F. Carey, libMesh: A C++ library for parallel adaptive mesh refinement/coarsening simulations, Eng. Comput. 22 (3–4) (2006) 237–254, http://dx.doi.org/10.1007/s00366-006-0049-3.
[33] M. Picasso, An anisotropic error indicator based on Zienkiewicz-Zhu error estimator: Application to elliptic and parabolic problems, SIAM J. Sci. Comput. 24 (4) (2003) 1328–1355, http://dx.doi.org/10.1137/S1064827501398578.
[34] M. Picasso, Numerical study of the effectivity index for an anisotropic error indicator based on Zienkiewicz-Zhu error estimator, Commun. Numer. Methods. Eng. 19 (1) (2003) 13–23, http://dx.doi.org/10.1002/cnm.546.
[35] O.C. Zienkiewicz, J.Z. Zhu, A simple error estimator and adaptive procedure for practical engineering analysis, Internat. J. Numer. Methods Engrg. 24 (1987) 337–357, http://dx.doi.org/10.1002/nme.1620240206.
[36] O.C. Zienkiewicz, J.Z. Zhu, The superconvergent patch recovery and a posteriori error estimates. I. The recovery technique, Internat. J. Numer. Methods Engrg. 33 (1992) 1331–1364, http://dx.doi.org/10.1002/nme.1620330702.
[37] P. Laug, H. Borouchaki, the BL2D Mesh Generator: Beginner's Guide, User's and Programmer's Manual, Tech. Rep. RT-0194, 1996, INRIAhttps://hal.inria.fr/inria-00069977.
[38] Y. Bourgault, M. Picasso, F. Alauzet, A. Loseille, On the use of anisotropic a posteriori error estimators for the adaptative solution of 3D inviscid compressible flows, Internat. J. Numer. Methods Fluids 59 (2009) 47–74, http://dx.doi.org/10.1002/fld.1797.
[39] W. Hassan, M. Picasso, An anisotropic adaptive finite element algorithm for transonic viscous flows around a wing, Comput. & Fluids 111 (2015) 33–45, http://dx.doi.org/10.1016/j.compfluid.2015.01.002.
[40] A. Yellowhorse, R.J. Lang, K. Tolman, L.L. Howell, Creating linkage permutations to prevent self-intersection and enable deployable networks of thick-origami, Sci. Rep. 8 (1) (2018) 12936, http://dx.doi.org/10.1038/s41598-018-31180-4.
[41] M. Picasso, Adaptive finite elements with large aspect ratio based on an anisotropic error estimator involving first order derivatives, Comput. Methods Appl. Mech. Engrg. 196 (1) (2006) 14–23, http://dx.doi.org/10.1016/j.cma.2005.11.018.
[42] M. Picasso, A. Loseille, Anisotropic, Adaptive Finite Elements for a Thin 3D Plate, Springer International Publishing, Cham, 2015, pp. 217–230, http://dx.doi.org/10.1007/978-3-319-06053-8_11.