ADAPTIVE ISOGEOMETRIC ANALYSIS USING OPTIMAL TRANSPORT

Mustapha Bahari, Abderrahmane Habbal, Ahmed Ratnani, Eric Sonnendrücker

MATHEMATICS OF COMPUTATION
Volume 00, Number 0, Pages 000–000
S 0025-5718(XX)0000-0

ADAPTIVE ISOGEOMETRIC ANALYSIS USING OPTIMAL TRANSPORT

MUSTAPHA BAHARI, ABDERRAHMANE HABBAL, AHMED RATNANI, AND ERIC SONNENDRÜCKER

Abstract.

Contents

Received by the editor 26 October 2021.
1991 *Mathematics Subject Classification.* Primary .
Key words and phrases. B-spline discretization, Optimal Transport, Isogeometric Analysis, multigrid, mesh generation, r-refinement.

1. Introduction

The use of adaptive mesh methods is fundamental to the numerical solution of systems of partial differential equations that involve large solution variations or with different scales. Adaptive mesh methods are also a major tool for solving problems with high anisotropy, such those encountered in Computational Plasmas Physics. Grid Adaptation and moving meshes are used in different fields and the litterature is quite rich [11, 34, 27, 35, 31, 23, 22, 30, 29, 28, 39, 9]. They are also used to generate Anisotropic meshes [41, 44, 25, 26, 24].
The proposed methods in this work aim to construct a one-to-one mapping $\mathbf{F}$ that maps a logical domain (patch, computational domain) with the physical domain as shown in Fig. 1. The function $\mathbf{F}$ is constructed using B-splines or NURBS surfaces which are widely used in the Computer Aided Design (CAD) community. The use of these tools in numerical simulations was made popular thanks to the introdution of the IsoGeometric Analysis paradigm by Hughes [37].
Because of the geometric interpretation of the control points, B-spline curves and surfaces have become very popular in CAD. We are interested in these features as they allow us to construct a set of mappings, where each of them will map the unit square onto a sub-domain of the physical domain. Using the geometric properties of B-splines curves and surfaces, it is easy to stick these mappings together, in order to have a global $\mathcal{C}^1$ or even $\mathcal{C}^2$ mapping. Local regularity of each mapping is ensured by construction.

Motivations. Many physical phenomena can be numerically simulated by the resolution of the associated partial differential equations. However, many numerical methods suffer from the local singularity of the solution of these equations. Such as shock waves, boundary layer in compressible flow problems. Thus, it is very interesting and advantageous to use a mesh that complies to the behavior of each phenomenon, and of course, because the refinement of a uniform mesh directly consumes a lot of energy the local refinement is mandatory.

Recently, using moving mesh methods, some authors are generally addressed the application to the field of weather prediction and climate simulation, which represent small-scale problems [46]; thus, based on the optimal transport problem the process requires the resolution of the Monge-Ampere equation on the sphere domain (earth shape), the investigation in this direction is given in the papers [53, 46, 21]. Moreover, the global adaptive meshes and conservation of the connection between the grid points are the most attractive properties of optimal transport problem [49, 38].

Equidistribution mesh generation using Monge-Kantorovich Problem. The idea behind the equidistribution is very simple. It aims to generate a grid that equidistributes a given quantity (a density or a monitor function) along a surface (2D) or a volume (3D). More details about the equidistribution and minimization of total error can be found in [40], [3]. An interesting introduction to equidistribution grid adaptation based on the Monge-Kantorovich problem (MKP) can be found in [20] and the references therein. The connection between MKP and the resolution of the Monge-Ampère equation (MA) is now quite well-known [20, 33]. Recently, many authors proposed equidistribution grid generation methods using the MA [20, 19, 12, 51, 50, 13, 10, 55, 54]

The numerical solution of the elliptic Monge-Ampère equation has been a subject of increasing interest. Many methods have been proposed, but only few of them converge for singular solutions. In this work, we use the Benamou-Froese-Oberman (BFO) method that we have implemented using the IGA approach. More details can be found in [4].
In [15], the authors solve the MA using an augmented Lagrangian approach. In [16], they proposed a least square approach. Much more details can be found in [17]. In [43], the authors proposed an iterative solver based on the divergence form and Newton method. While, in [20] the authors used an iterative Newton-Krylov solver with preconditioning coupled with a Finite Difference method. In [2] [1], the author solves the Monge-Ampère equation as the limit of the solution of a singular perturbation problem, using triangular B-splines, which by the way can treat complex geometries. Another way of solving the Monge-Ampère equation, is to consider it as the steady state of a parabolic equation namely the Parabolic Monge-Ampère equation. This has been studied in [8, 51, 50].
In [56], the authors proposed three new methods based on Fourier integral, second-order divergence and a convolution forms.

Problem formulation. Now, we consider a fixed computational domain Ω_c, and Ω_p the physical domain where the underlying PDE is posed and the initial mapping $\mathbf{F} : \Omega_c \ni \xi \longrightarrow x \in \Omega_p$. The main idea behind an optimum mesh is that it should be closest to a uniform mesh in a suitable norm [32]. The equidistributed grids are constructed by solving the Monge-Ampere equation which is equivalent to the L^2 Monge-Kantorovich problem [18, 32].

Hence, the adaptive meshes are obtained using a new mapping $\mathbf{F}' : \Omega_c \ni \xi \longrightarrow x' \in \Omega_p$, that maps the parametric domain Ω_c into the physical domain Ω_p, defined as the composition of initial mapping $\mathbf{F} : \Omega_c \ni \xi' \longrightarrow x' \in \Omega_p$ and B-spline geometrical map $\mathbf{\Psi} : \Omega_c \ni \xi \longrightarrow \xi' \in \Omega_c$ derived from a resolution of Monge-Ampere equation. Indeed, for our application, we have $\Omega_c = (0,1)^d$, $d = 2$ or 3. We give the illustration of these ideas in Fig. 1.

Our contributions. Our contributions are the following:

(1) A new technique of mesh adaptation illustrated in Fig. 1,
(2) A fast solver for Monge-Ampère equation, using fast diagonalization method,
(3) Demonstrate the effectiveness of our method on complex geometry,
(4) Derive a priori estimates.

The remainder of this paper is as follows. In Section 2, we describe the basic theory behind B-splines curves and surfaces. In Section 3, we describe the basic theory behind adaptive meshes methods and specially the optimal transport problem which is linked to the Monge-Ampere equation. In Section 4-5, we use the Benamou-Froese-Oberman (BFO) method to generate equidistributed meshes. In order to enhance the performance of the Picard algorithm, a multi grids method is used. In Section 6-7, we introduce a mixed Aligned-Equidistributed strategy to handle the boundary conservation problem and limitation of BFO method. With high accuracy and fast convergence using the multi grids method, two algorithms of the fast diagonalization method in two and three dimensions are developed for the mixed variational formulation. Concluding remarks are given in section 8.

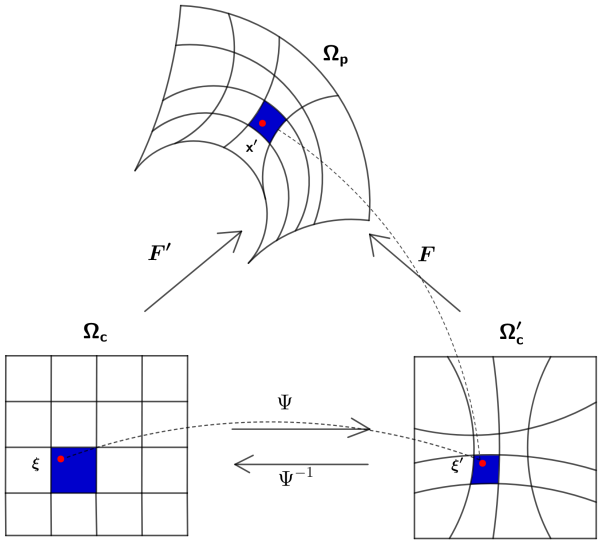

FIGURE 1. Illustration of the Adaptive Mapping idea. The old mapping $\mathbf{F} : \xi' \to x$ and the new mapping $\mathbf{F}' : \xi \to x'$ map the logical domain (patch) onto the same physical domain Ω_p. The mapping $\Psi : \xi \to \xi'$, maps the old (uniform) mesh onto the new one in parametric domain.

2. B-SPLINES AND GEOMETRICAL REPRESENTATION

IsoGeometric analysis (IGA) has been introduced in [36], the main idea was to improve the interoperability between Computer-Aided Design (CAD) and partial differential equation (PDE) solvers. Let's start by recalling the definition and some aspects of isogeometric analysis.

B-splines. Given two positive integers p and n, we introduce the (ordered) knot vector

$$\Xi = \{0 = \xi_1, \xi_2, ..., \xi_{n+p+1} = 1\} \tag{2.1}$$

where n is the number of basis functions necessary to describe it. Here we work only with open knot vectors, which means that first and last knots in Ξ have multiplicity $p+1$, so that

$$\Xi = \{\underbrace{\zeta_1, .., \zeta_1}_{p+1}, \underbrace{\zeta_2, .., \zeta_2}_{r_2}, ..., \underbrace{\zeta_m, .., \zeta_m}_{p+1}\} \tag{2.2}$$

where $1 \leq r_j \leq p+1$ for $j = 2, .., m-1$ and $n = p + 1 + \sum_{j=2}^{m-1} r_j$. Univariate B-spline basis functions B_i^p, $i = 1, ..., n$ are defined recursively by the well known Cox-de Boor recursion formula:

$$B_i^0(\xi) = \begin{cases} 1 & \text{if } \xi \in [\xi_i, \xi_{i+1}) \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

$$B_i^p(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} B_i^{p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} B_{i+1}^{p-1}(\xi) \tag{2.4}$$

where in 2.4, we adopt the convention $0/0 = 0$. These basis functions are piecewise polynomials of degree p on the subdivision $\{\zeta_1, .., \zeta_m\}$, where at ζ_j they have $\mu_j := p - r_j$ continuous derivative. Therefore, the vector $\boldsymbol{\mu} = \{\mu_1, ..., \mu_m\}$ collects the regularity at the internal knots, with $\mu_1 = \mu_m = -1$ for the boundary knots associated with the open knot vector.

An example of cubic B-splines is presented in Fig. 2. In this case $\boldsymbol{\mu} = \{-1, 2, 1, 0, -1\}$.

Respectively, by means of tensor products, a multi-dimensional B-spline can be constructed as $B^{p_1,p_2,...,p_d}_{j_1,j_2,...,j_d} = \otimes_{i=1}^{d} B^{p_i}_{j_i}$ [36, 14].

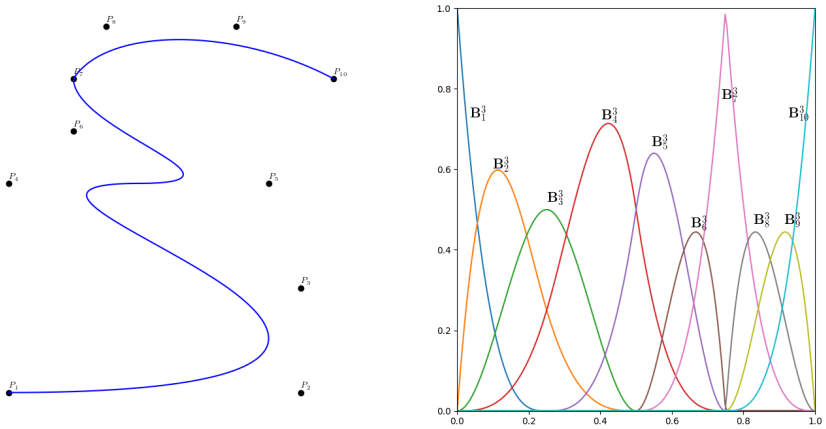

FIGURE 2. (left) A cubic B-Spline curve and its control points using the knot vector $\Xi = \{0000\ \frac{1}{4}\ \frac{1}{2}\frac{1}{2}\ \frac{3}{4}\frac{3}{4}\frac{3}{4}\ 1111\}$, (right) the corresponding B-Splines.

Geometrical representation. A single patch domain Ω_p is a B-spline region associated with the control points $C_{j_1,j_2,...,j_d}$. We introduce the B-spline geometrical map $\mathbf{F} : \Omega_c \longrightarrow \Omega_p$ by

$$\mathbf{F}(\xi) = \sum_{j_1,j_2,...,j_d} B^{p_1,p_2,...,p_d}_{j_1,j_2,...,j_d}(\xi) C_{j_1,j_2,...,j_d}. \tag{2.5}$$

For our purpose we assume that the geometry mapping is continuous and bijective which is a natural assumption for CAD applications. We introduce $\mathcal{Q}_h$ a family of meshes on Ω_c by

$$\mathcal{Q}_h = \{Q = \otimes_{i=1}^{d} (\zeta_{j_i,i},\ \zeta_{j_i+1,i}),\ 1 \le j_i \le m_i - 1,\ d = 2,\ 3\} \tag{2.6}$$

each element $Q \in \mathcal{Q}_h$ is mapped into an element

$$k = \mathbf{F}(Q) = \{\mathbf{F}(\xi), \xi \in Q\}, \tag{2.7}$$

and analogously $\widetilde{Q}$, the support extension of Q, is mapped into

$$\widetilde{k} = \mathbf{F}(\widetilde{Q}). \tag{2.8}$$

We then introduce the mesh $\mathcal{T}_h$ in the physical domain Ω_p, defined by:

$$\mathcal{T}_h := \{k = \mathbf{F}(Q), Q \in \mathcal{Q}_h\}. \tag{2.9}$$

Finally, we define B-spline spaces needed in the sequel as follows:

$$\mathcal{S}^p_{\boldsymbol{\mu}} = span\{B^p_i, \text{ for } i = 1, ..n\} \tag{2.10}$$

and

$$\left\{ \frac{\mathrm{d}v}{\mathrm{d}\xi} : \ v \in \mathcal{S}^p_{\boldsymbol{\mu}} \right\} = \mathcal{S}^{p-1}_{\boldsymbol{\mu}-1} \tag{2.11}$$

where we adopt the notation $\boldsymbol{\mu} - 1 = \{\mu_1, \mu_2 - 1, ..., \mu_{m-1} - 1, \mu_m\}$.

Then, we consider the discrete subspace of $H^s(\Omega_c)$, given by

$$\mathcal{V}_h = \otimes_{i=1}^d \mathcal{S}^{p_i}_{\boldsymbol{\mu}_i}. \tag{2.12}$$

Let us introduce the discrete subspace of $\boldsymbol{H}(\mathrm{div}, \Omega)$ and $L^2(\Omega)$ denoted by $\boldsymbol{V}_h(\mathrm{div}, \Omega)$ and $V_h(L^2, \Omega)$ successively by

-For 2D case:

$$\begin{cases} \boldsymbol{V}_h(\mathrm{div}, \Omega) & = \begin{pmatrix} \mathcal{S}^{p_1}_{\boldsymbol{\mu}_1} \otimes \mathcal{S}^{p_2-1}_{\boldsymbol{\mu}_2-1} \\ \mathcal{S}^{p_1-1}_{\boldsymbol{\mu}_1-1} \otimes \mathcal{S}^{p_2}_{\boldsymbol{\mu}_2} \end{pmatrix} \\ V_h(L^2, \Omega) & = \mathcal{S}^{p_1-1}_{\boldsymbol{\mu}_1-1} \otimes \mathcal{S}^{p_2-1}_{\boldsymbol{\mu}_2-1} \end{cases} \tag{2.13}$$

-For 3D case:

$$\begin{cases} \boldsymbol{V}_h(\mathrm{div}, \Omega) & = \begin{pmatrix} \mathcal{S}^{p_1}_{\boldsymbol{\mu}_1} \otimes \mathcal{S}^{p_2-1}_{\boldsymbol{\mu}_2-1} \otimes \mathcal{S}^{p_3-1}_{\boldsymbol{\mu}_3-1} \\ \mathcal{S}^{p_1-1}_{\boldsymbol{\mu}_1-1} \otimes \mathcal{S}^{p_2}_{\boldsymbol{\mu}_2} \otimes \mathcal{S}^{p_3-1}_{\boldsymbol{\mu}_3-1} \\ \mathcal{S}^{p_1-1}_{\boldsymbol{\mu}_1-1} \otimes \mathcal{S}^{p_2-1}_{\boldsymbol{\mu}_2-1} \otimes \mathcal{S}^{p_3}_{\boldsymbol{\mu}_3} \end{pmatrix} \\ V_h(L^2, \Omega) & = \mathcal{S}^{p_1-1}_{\boldsymbol{\mu}_1-1} \otimes \mathcal{S}^{p_2-1}_{\boldsymbol{\mu}_2-1} \otimes \mathcal{S}^{p_3-1}_{\boldsymbol{\mu}_3-1} \end{cases} \tag{2.14}$$

The functional spaces that we choose are because they form an exact sequence (called DeRham sequence) needed for the mixed variational formulation. So, we recall that in 2d, we have

$$V_h(\mathrm{grad}, \Omega) \xrightarrow{\nabla\times} \boldsymbol{V}_h(\mathrm{div}, \Omega) \xrightarrow{\nabla\cdot} V_h(L^2, \Omega)$$

While in 3d, we have the following sequence

$$V_h(\mathrm{grad}, \Omega) \xrightarrow{\boldsymbol{\nabla}} \boldsymbol{V}_h(\mathrm{curl}, \Omega) \xrightarrow{\nabla\times} \boldsymbol{V}_h(\mathrm{div}, \Omega) \xrightarrow{\nabla\cdot} V_h(L^2, \Omega)$$

where

$$\boldsymbol{V}_h(\mathrm{curl}, \Omega) = \begin{pmatrix} \mathcal{S}^{p_1-1}_{\boldsymbol{\mu}_1-1} \otimes \mathcal{S}^{p_2}_{\boldsymbol{\mu}_2} \otimes \mathcal{S}^{p_3}_{\boldsymbol{\mu}_3} \\ \mathcal{S}^{p_1}_{\boldsymbol{\mu}_1} \otimes \mathcal{S}^{p_2-1}_{\boldsymbol{\mu}_2-1} \otimes \mathcal{S}^{p_3}_{\boldsymbol{\mu}_3} \\ \mathcal{S}^{p_1}_{\boldsymbol{\mu}_1} \otimes \mathcal{S}^{p_2}_{\boldsymbol{\mu}_2} \otimes \mathcal{S}^{p_3-1}_{\boldsymbol{\mu}_3-1} \end{pmatrix} \tag{2.15}$$

is the discrete subspace of $\boldsymbol{H}(\mathrm{curl}, \Omega)$.

3. Equidistribution mesh generation based on Monge-Ampère equation

3.1. The Monge-Kantorovich Problem.

Definition 3.1 (L^2 Monge-Kantorovich problem)**.** *Let ρ_0 and ρ_1 be two given densities of equal masses, defined in $\mathbb{R}^d$.*
Find a mapping $\mathbf{x}' = \mathbf{T}(\mathbf{x})$, $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$, that transfers the density ρ_0 to ρ_1 and minimizes the transport cost

$$\mathcal{J}[\mathbf{T}] = \int_{\mathbb{R}^d} |\mathbf{T}(\mathbf{x}) - \mathbf{x}'|^2 \rho_0(\mathbf{x}) \, d\mathbf{x} \tag{3.1}$$

The density transfer means that, for all Borelian set B in $\mathbb{R}^d$

$$\int_{\mathbf{T}^{-1}(B)} \rho_0(x) \, \mathrm{d}x = \int_B \rho_1(x) \, \mathrm{d}x.$$

Then, it follows from a change of variable that

$$\rho_1(\mathbf{T}) \det(\nabla \mathbf{T}) = \rho_0.$$

Brenier's theorem [6], shows that $\mathbf{T} = \nabla\phi$ for given convex potential function ϕ. Hence, we get the Monge-Ampere equation:

$$\rho_1(\nabla\phi) \det(H(\phi)) = \rho_0, \tag{3.2}$$

where $H(\phi)$ is the Hessian matrix of ϕ.

Remark 3.1. *For arbitrary ρ_0 and ρ_1 defined in the physical domain Ω_p, let*

$$\sigma = \int_{\Omega_p} \rho_1(x) \, \mathrm{d}x \Big/ \int_{\Omega_p} \rho_0(x) \, \mathrm{d}x, \tag{3.3}$$

we rewrite the Monge-Ampere equation as follows

$$\rho_1(\nabla\phi) \det(H(\phi)) = \sigma. \tag{3.4}$$

3.2. **New Mesh Equidistribution Technique.** Since both density functions are defined in the physical domain, we explain in the following the mathematical background of the technique presented in Fig. 1.

Proposition 3.1. *In terms of mesh generation, we assume that the initial mapping $\mathbf{F}$ is bijective, so we take the two densities defined in Ω_c as follows*

1. *$\sigma \det(\nabla \mathbf{F})$: corresponding to a uniform mesh in Ω_c,*
2. *$\rho_1 \circ \mathbf{F} \ \det(\nabla \mathbf{F})$: corresponding to what we call the inverse image by the initial mapping $\mathbf{F}$ of the adapted mesh in Ω_p,*

where σ is given by 3.3, the two densities are of equal masses. Then, according to Brenier's theorem [7] *there exists an optimal mapping $\mathbf{\Psi}$ defined in the parametric domain Ω_c which is the gradient of a given convex potential function $\tilde{\phi}$, such that*

$$\rho_1(\mathbf{F} \circ \nabla\tilde{\phi}) \det(\nabla \mathbf{F}) \circ \nabla\tilde{\phi} \ \det(H(\tilde{\phi})) = \sigma \det(\nabla \mathbf{F}). \tag{3.5}$$

Moreover, our mesh adaptation technique and the one existing in the literature [18] *are equivalent.*

Proof. Using $\rho_0 = 1$ correspond to uniform mesh in Ω_p and assuming that Ω_p is a convex domain, then there exists $\mathbf{T} : \Omega_p \longrightarrow \Omega_p$ satisfying the Monge-Ampere equation 3.4, which corresponds to the adaptation in the physical domain, hence we have

$$\rho_1(\mathbf{T}) \det(\nabla \mathbf{T}) = \sigma. \tag{3.6}$$

using the fact that $\mathbf{F}$ is bijective, let $\mathbf{T}_0 : \Omega_c \longrightarrow \Omega_c$ such that

$$\mathbf{T} = \mathbf{F} \circ \mathbf{T}_0 \circ \mathbf{F}^{-1}, \tag{3.7}$$

as a consequence, by a change of variable in 3.6, we have

$$\rho_1(\mathbf{F} \circ \mathbf{T}_0 \circ \mathbf{F}^{-1}) \det\left(\nabla(\mathbf{F} \circ \mathbf{T}_0 \circ \mathbf{F}^{-1})\right) = \sigma \tag{3.8}$$

using the fact that

$$\begin{aligned}\nabla\big(\mathbf{F} \circ \mathbf{T}_0 \circ \mathbf{F}^{-1}\big) &= \big(\nabla\mathbf{F}^{-1}\big)^T \nabla\big(\mathbf{F} \circ \mathbf{T}_0\big) \circ \mathbf{F}^{-1} \\ &= \big(\nabla\mathbf{F}^{-1}\big)^T \big(\nabla\mathbf{T}_0\big)^T \circ \mathbf{F}^{-1} \big(\nabla\mathbf{F}\big) \circ \mathbf{T}_0 \circ \mathbf{F}^{-1},\end{aligned}$$

and $\det(AB) = \det(A)\det(B)$, we get

$$\rho_1(\mathbf{F}\circ\mathbf{T}_0\circ\mathbf{F}^{-1}) \det\left((\nabla\mathbf{F}^{-1})^T\right) \det\left((\nabla\mathbf{T}_0)^T\right) \circ\mathbf{F}^{-1} \det\left(\nabla\mathbf{F}\right) \circ\mathbf{T}_0\circ\mathbf{F}^{-1} = \sigma \tag{3.9}$$

and therefore, by applying $\mathbf{F}$ we have

$$\rho_1(\mathbf{F} \circ \mathbf{T}_0) \det\left((\nabla\mathbf{F}^{-1}) \circ \mathbf{F}\right) \ \det\left(\nabla\mathbf{T}_0\right) \ \det\left(\nabla\mathbf{F}\right) \circ \mathbf{T}_0 = \sigma. \tag{3.10}$$

From differential calculus, we know that $(\nabla\mathbf{F}^{-1}) \circ \mathbf{F} = \big(\nabla\mathbf{F}\big)^{-1}$, then

$$\rho_1(\mathbf{F} \circ \mathbf{T}_0) \ \det\left(\nabla\mathbf{F}\right) \circ \mathbf{T}_0 \ \det\left(\nabla\mathbf{T}_0\right) = \sigma \det\left(\nabla\mathbf{F}\right). \tag{3.11}$$

Finally, from the uniqueness of the optimal mapping we conclude that $\mathbf{T}_0 = \mathbf{\Psi}$. □

Corollary 3.1. *Assuming the initial mapping is uniform, we suppose that*

$$\det\left(\nabla\mathbf{F}\right) \circ \nabla\tilde{\phi} \approx \det\left(\nabla\mathbf{F}\right), \tag{3.12}$$

then, we simplify the equation 3.5 by considering the new form:

$$\rho_1(\mathbf{F} \circ \nabla\tilde{\phi}) \det(H(\tilde{\phi})) = \sigma. \tag{3.13}$$

The boundary conditions are obtained by requiring that $\mathbf{\Psi}$ *maps* $\partial\Omega_c$ *to* $\partial\Omega_c$*, i.e*

$$\nabla\tilde{\phi}(\partial\Omega_c) = \partial\Omega_c$$

which writes:

$$\nabla\tilde{\phi} \cdot \vec{n} = x \cdot \vec{n}. \tag{3.14}$$

Finally, we denote

$$f = \frac{\sigma}{\rho_1(\mathbf{F})}, \tag{3.15}$$

the desired convex potential function is the solution of Monge-Ampere system:

$$\begin{cases} \det(H(\tilde{\phi})) & = f(\nabla\tilde{\phi}) \quad \text{in } \Omega_c, \\ \nabla\tilde{\phi} \cdot \vec{n} & = x \cdot \vec{n} \quad \text{on } \partial\Omega_c, \\ \tilde{\phi} \text{ is convex.} & \end{cases} \tag{3.16}$$

A solution of 3.16 exists and is unique up to an additive constant [42]. *From now on, the notation* ϕ *stands for* $\tilde{\phi}$ *solution of the latest system 3.16 and* ρ *stands for* ρ_1*.*

4. Benamou-Froese-Oberman method

4.1. **Standard Monge Ampère Solver.** We adopt the second semi-implicit method proposed by Benamou Froese and Oberman [5] which guarantees convergence when a second member is strictly positive and regular. The authors noticed that a solution of the Monge-Ampere equation is a fixed point of the operator T.

$$\begin{aligned} T : H^2(\Omega_c) &\longrightarrow H^2(\Omega_c) \\ v &\longrightarrow T(v) = (\Delta)^{-1}\mathcal{G}(v) \end{aligned} \tag{4.1}$$

where $\mathcal{G}(v) = \Big((\Delta v)^d + d!(f(\nabla v)) - \det(H(v))\Big)^{\frac{1}{d}}$, d is spatial dimension and f is given by 3.15, that rewrite as

$$\begin{cases} -\Delta\phi &= -\mathcal{G}(\phi) & \text{in } \Omega_c, \\ \nabla\phi \cdot \vec{n} &= x \cdot \vec{n} & \text{on } \partial\Omega_c, \\ \phi & \text{is convex.} \end{cases} \tag{4.2}$$

Thus, the solution is given by Picard iteration as follow:

(1) Given an initial value ϕ^0.
(2) Compute ϕ^{n+1} as the solution of the problem:

$$\begin{cases} -\Delta\phi^{n+1} + \varepsilon\phi^{n+1} &= -\mathcal{G}(\phi^n) & \text{in } \Omega_c, \\ \nabla\phi^{n+1} \cdot \vec{n} &= x \cdot \vec{n} & \text{on } \partial\Omega_c, \\ \phi^n & \text{is convex,} \end{cases} \tag{4.3}$$

where $\varepsilon = 10^{-8}$ is a penalization parameter.
(3) Repeat the iteration number (2) until L^2 residual is smaller than a given tolerance.

Remark 4.1. *- Firstly, we note that the initial guess for the Picard algorithm can be computed using a multi-grids (MG) method.*

- We have tested different methods that exist in the literature such as the Lagrange multiplier, the Newton method, and the projection method, but we obtain the same convergence results as for the penalization method. Then, for reasons of simplicity and fast convergence, the penalization method is used for a robust numerical resolution of the Poisson equation with pure Neumann boundary conditions.

4.2. **Variational Formulation and A priori error estimate.** The goal of this subsection is to validate our solver by two examples in both a two and three-dimensional cases.

The variational formulation of 4.3, is:

$$\int_{\Omega_c} \nabla\phi^{n+1}\nabla v \,\mathrm{d}x + \varepsilon \int_{\Omega_c} \phi^{n+1} v \,\mathrm{d}x = -\int_{\Omega_c} \mathcal{G}(\phi^n) v \,\mathrm{d}x + \int_{\partial\Omega_c} x\cdot\vec{n}\, v \,\mathrm{d}\sigma, \text{ for all } v \in H^2(\Omega_c) \tag{4.4}$$

and a Galerkin approximation of 4.3 is given by

$$\int_{\Omega_c} \nabla\phi_h^{n+1}\nabla v^h \,\mathrm{d}x + \varepsilon \int_{\Omega_c} \phi_h^{n+1} v^h \,\mathrm{d}x = -\int_{\Omega_c} \mathcal{G}(\phi_h^n) v^h \,\mathrm{d}x + \int_{\partial\Omega_c} x\cdot\vec{n}\, v^h \,\mathrm{d}\sigma, \text{ for all } v \in \mathcal{V}_h \cap H^2(\Omega_c) \tag{4.5}$$

where $\mathcal{V}_h$ is given by 2.12.

Without losing the generalization, we neglect the penalization part of the variational form in the following estimation.

Proposition 4.1. *Let $\{\phi^n\}_n$ be a solution sequence of Picard algorithm in $H^2(\Omega_c)$, then it holds that:*

$$|e^{n,h}|_{H^1(\Omega_c)} \leq Ch^{l-1}, \tag{4.6}$$

where $e^{n,h} = \phi^n - \phi_h^n$ is the approximation error at iteration n and l is a Sobolev regularity exponent of ϕ^n $\left(\phi^n \in H^l(\Omega_c),\ l \geq 2\right)$.

Proof. By recurrence, we have for $n = 0$

$$|\phi^0 - \phi_h^0|_{H^1(\Omega_c)} \leq Ch^{l-1},$$

holds by choice. So, we first prove that we have the following implication

$$|\phi^n - \phi_h^n|_{H^1(\Omega_c)} \leq Ch^{l-1} \implies ||\mathcal{G}(x,\phi^n) - \mathcal{G}(x,\phi_h^n)||_{L^2(\Omega_c)} \leq Ch^{l-2}, \text{ for } n \in \mathbb{N}, \tag{4.7}$$

recalling that

$$||H(\phi^n)||^2 = (\Delta\phi^n)^d - d!\det(H(\phi^n)) \geq 0,$$

and, the triangular inequality leads to

$$\begin{aligned}
|\mathcal{G}(x,\phi^n) - \mathcal{G}(x,\phi_h^n)|^2 &= \left|\sqrt{||H(\phi^n)||^2 + d!\widehat{f}(\nabla\phi^n)} - \sqrt{||H(\phi_h^n)||^2 + d!\widehat{f}(\nabla\phi_h^n)}\right|^2 \\
&\leq 2\left|\sqrt{||H(\phi^n)||^2 + d!\widehat{f}(\nabla\phi^n)} - \sqrt{||H(\phi^n)||^2 + d!\widehat{f}(\nabla\phi_h^n)}\right|^2 \\
&+ 2\left|\sqrt{||H(\phi^n)||^2 + d!\widehat{f}(\nabla\phi_h^n)} - \sqrt{||H(\phi_h^n)||^2 + d!\widehat{f}(\nabla\phi_h^n)}\right|^2 \\
&:= 2A^2 + 2B^2.
\end{aligned}$$

Using the inequality $\left|\sqrt{a+b} - \sqrt{c+b}\right| \leq |\sqrt{a} - \sqrt{c}|$ for $a, b, c \geq 0$, we have

$$B \leq \left|||H(\phi^n)|| - ||H(\phi_h^n)||\right| \tag{4.8}$$

and

$$\begin{aligned}
A &\leq \sqrt{d!}\left|\sqrt{\widehat{f}(\nabla\phi^n)} - \sqrt{\widehat{f}(\nabla\phi_h^n)}\right| \\
&\leq \sqrt{d!}\left|\frac{\sqrt{\sigma}}{\sqrt{\rho(\nabla\phi^n)}} - \frac{\sqrt{\sigma}}{\sqrt{\rho(\nabla\phi_h^n)}}\right| \\
&\leq \sqrt{d!\sigma}\frac{\left|\sqrt{\rho(\nabla\phi^n)} - \sqrt{\rho(\nabla\phi_h^n)}\right|}{\sqrt{\rho(\nabla\phi^n)\rho(\nabla\phi_h^n)}}
\end{aligned}$$

since that $\rho \geq \gamma > 0$, and $t \longrightarrow \sqrt{t}$ is 1-Lipschitz function in $[1, +\infty)$ then

$$A \leq \sqrt{\frac{d!\sigma}{\gamma}}\left|\rho(\nabla\phi^n) - \rho(\nabla\phi_h^n)\right|$$

if we assume that ρ is L_ρ-Lipschitz

$$A \leq L_\rho\sqrt{\frac{d!\sigma}{\gamma}}\left|\nabla\phi^n - \nabla\phi_h^n\right|. \tag{4.9}$$

As a consequence of 4.8 and 4.9

$$||\mathcal{G}(x,\phi^n) - \mathcal{G}(x,\phi_h^n)||_{L^2(\Omega_c)} \leq 2|\phi^n - \phi_h^n|_{H^2(\Omega_c)} + 2L_\rho\sqrt{\frac{d!\sigma}{\gamma}}|\phi^n - \phi_h^n|_{H^1(\Omega_c)}.$$

Now, taking the Projector $\Pi_{\mathcal{V}_h} : L^2(\Omega_c) \longrightarrow \mathcal{V}_h$, we get

$$\begin{aligned}|\phi^n - \phi_h^n|_{H^2(\Omega_c)} &\leq |(\phi^n - \phi_h^n) - \Pi_{\mathcal{V}_h}(\phi^n - \phi_h^n)|_{H^2(\Omega_c)} + |\Pi_{\mathcal{V}_h}(\phi^n - \phi_h^n)|_{H^2(\Omega_c)} \\ &= I + \Pi.\end{aligned}$$

With C^2 regularity in B-spline function, from Proposition 3.2 [52]

$$\begin{aligned}I &= |(\phi^n - \phi_h^n) - \Pi_{\mathcal{V}_h}(\phi^n) + \phi_h^n|_{H^2(\Omega_c)} \\ &= |\phi^n - \Pi_{\mathcal{V}_h}(\phi^n)|_{H^2(\Omega_c)} \\ &\leq Ch^{l-2}||\phi^n||_{H^l(\Omega_c)}.\end{aligned}$$

By the usual inverse inequality for polynomials, we have

$$\begin{aligned}\Pi^2 &= \sum_{k\in\mathcal{T}_h} |\Pi_{\mathcal{V}_h}(\phi^n - \phi_h^n)|^2_{H^2(k)} \\ &\leq C_{shap} \sum_{k\in\mathcal{T}_h} h_K^{-2} |\Pi_{\mathcal{V}_h}(\phi^n - \phi_h^n)|^2_{H^1(k)} \\ &\leq C_{shap} \underline{h}^{-2} |\phi^n - \phi_h^n|^2_{H^1(\Omega_c)}, \qquad (4.10)\end{aligned}$$

where $\underline{h} = \min(h_k)$, so if we assume that for a positive constant θ,

$$h \leq \theta \underline{h}.$$

According to the hypothesis of 4.7, we have

$$I + II \leq Ch^{l-2},$$

where C depend only on $|\phi^n|_{H^l(\Omega_c)}$ and θ.

Returning now to the proof of our proposition, let ϕ^{n+1} be the solution at $n+1$ iteration in Picard algorithm and $v_h \in \mathcal{V}_h$, we have for some constant c_{n+1}

$$\begin{aligned}|\phi^{n+1} - \phi_h^{n+1}|^2_{H^1(\Omega_c)} &= \int_{\Omega_c} \nabla(\phi^{n+1} - \phi_h^{n+1}) \nabla(\phi^{n+1} - v_h + c_{n+1})\, \mathrm{d}x \\ &\quad + \int_{\Omega_c} \nabla(\phi^{n+1} - \phi_h^{n+1}) \nabla(v_h - c_{n+1} - \phi_h^{n+1})\, \mathrm{d}x \\ &= \int_{\Omega_c} \nabla(\phi^{n+1} - \phi_h^{n+1}) \nabla(\phi^{n+1} - v_h)\, \mathrm{d}x \\ &\quad + \int_{\Omega_c} (\mathcal{G}(x, \phi_h^n) - \mathcal{G}(x, \phi^n))(v_h - c_{n+1} - \phi_h^{n+1})\, \mathrm{d}x,\end{aligned}$$

by Holder's inequality, we have

$$\begin{aligned}|\phi^{n+1} - \phi_h^{n+1}|^2_{H^1(\Omega_c)} &\leq |\phi^{n+1} - \phi_h^{n+1}|_{H^1(\Omega_c)} |\phi^{n+1} - v_h|_{H^1(\Omega_c)} \\ &\quad + ||\mathcal{G}(x, \phi^n) - \mathcal{G}(x, \phi_h^n)||_{L^2(\Omega_c)} ||v_h - c_{n+1} - \phi_h^{n+1}||_{L^2(\Omega_c)},\end{aligned}$$

taking to account 4.7, we get by the triangular inequality

$$\begin{aligned}|\phi^{n+1} - \phi_h^{n+1}|^2_{H^1(\Omega_c)} &\leq |\phi^{n+1} - \phi_h^{n+1}|_{H^1(\Omega_c)} |\phi^{n+1} - v_h|_{H^1(\Omega_c)} + Ch^{l-2} ||v_h - c_{n+1} - \phi_h^{n+1}||_{L^2(\Omega_c)} \\ &\leq |\phi^{n+1} - \phi_h^{n+1}|_{H^1(\Omega_c)} |\phi^{n+1} - v_h|_{H^1(\Omega_c)} + Ch^{l-2} ||\phi^{n+1} - v_h||_{L^2(\Omega_c)} \\ &\quad + Ch^{l-2} ||\phi^{n+1} - \phi_h^{n+1} - c_{n+1}||_{L^2(\Omega_c)}\end{aligned}$$

where $v_h \in \mathcal{V}_h$, according to Proposition 3.2 [52], for $v_h = \Pi_{\mathcal{V}_h}(\phi^{n+1})$ we have

$$|\phi^{n+1} - \Pi_{\mathcal{V}_h}(\phi^{n+1})|_{H^1(\Omega_c)} \leq Ch^{l-1},$$

and

$$||\phi^{n+1} - \Pi_{\mathcal{V}_h}(\phi^{n+1})||_{L^2(\Omega_c)} \leq Ch^l.$$

Then,

(4.11)
$$|\phi^{n+1} - \phi_h^{n+1}|^2_{H^1(\Omega_c)} \leq Ch^{l-1}|\phi^{n+1} - \phi_h^{n+1}|_{H^1(\Omega_c)} + Ch^{2l-2} + Ch^{l-2}||\phi^{n+1} - \phi_h^{n+1} - c_{n+1}||_{L^2(\Omega_c)}.$$

Let's now introduce the following problem:

$$\begin{cases} -\Delta\psi = \phi^{n+1} - \phi_h^{n+1} - c_{n+1} & \text{in } \Omega_c, \\ \nabla\psi \cdot \vec{n} = 0 & \text{on } \partial\Omega_c. \end{cases} \tag{4.12}$$

The compatibility condition holds for $c_{n+1} = \int_{\Omega_c} \left(\phi^{n+1} - \phi_h^{n+1}\right) \mathrm{d}x$, and the solution exists and unique up to an additive constant. Moreover, the variational formulation of 4.12 is written as follows :

$$\int_{\Omega_c} \nabla\psi\nabla v \,\mathrm{d}x = \int_{\Omega_c} (\phi^{n+1} - \phi_h^{n+1} - c_{n+1}) v \,\mathrm{d}x \quad \text{for all } v \in H^1(\Omega_c) \tag{4.13}$$

by assuming that

$$\int_{\Omega_c} (\mathcal{G}(x, \phi^n) - \mathcal{G}(x, \phi_h^n)) \,\mathrm{d}x \neq 0 \tag{4.14}$$

we introduce the constant c such that

$$\int_{\Omega_c} (\mathcal{G}(x, \phi^n) - \mathcal{G}(x, \phi_h^n))(\Pi_{\mathcal{V}_h}(\psi) - c) \,\mathrm{d}x = 0. \tag{4.15}$$

Hence, from the variational formulation 4.5, for $v^h = \Pi_{\mathcal{V}_h}(\psi) - c$, we have

$$\begin{aligned} \int_{\Omega_c} \nabla(\phi^{n+1} - \phi_h^{n+1} - c_{n+1})\nabla(\Pi_{\mathcal{V}_h}(\psi) - c) \,\mathrm{d}x &= \int_{\Omega_c} (\mathcal{G}(x, \phi^n) - \mathcal{G}(x, \phi_h^n))(\Pi_{\mathcal{V}_h}(\psi) - c) \,\mathrm{d}x \\ &= 0. \end{aligned} \tag{4.16}$$

As a consequence, by replacing the results in 4.13, we get for $v = \phi^{n+1} - \phi_h^{n+1} - c_{n+1}$

(4.17)
$$\int_{\Omega_c} (\phi^{n+1} - \phi_h^{n+1} - c_{n+1})^2 \,\mathrm{d}x = \int_{\Omega_c} \nabla\psi\nabla(\phi^{n+1} - \phi_h^{n+1}) \,\mathrm{d}x$$

$$= \int_{\Omega_c} \nabla(\psi - (\Pi_{\mathcal{V}_h}(\psi) - c))\nabla(\phi^{n+1} - \phi_h^{n+1}) \,\mathrm{d}x \tag{4.18}$$

$$\leq |\psi - (\Pi_{\mathcal{V}_h}(\psi) - c)|_{H^1(\Omega_c)} |\phi^{n+1} - \phi_h^{n+1}|_{H^1(\Omega_c)}. \tag{4.19}$$

Furthermore, because $\phi^{n+1} - \phi_h^{n+1} \in H^l(\Omega_c)$ at least $\psi \in H^{l+2}(\Omega_c)$, then from the Proposition 3.2 [52] we have the following inequality :

$$\begin{aligned} |\psi - (\Pi_{\mathcal{V}_h}(\psi) - c)|_{H^1(\Omega_c)} &= |\psi - \Pi_{\mathcal{V}_h}(\psi)|_{H^1(\Omega_c)} \\ &\leq Ch^{l+1}||\psi||_{H^{l+2}(\Omega_c)} \end{aligned}$$

which is equivalent to saying that we have :

$$\int_{\Omega_c} (\phi^{n+1} - \phi_h^{n+1} - c_{n+1})^2 \,\mathrm{d}x \leq Ch^{l+1}||\psi||_{H^{l+2}(\Omega_c)} |\phi^{n+1} - \phi_h^{n+1}|_{H^1(\Omega_c)} \tag{4.20}$$

as a consequence, replacing the results in inequality 4.11, we obtain

(4.21)
$$|\phi^{n+1}-\phi_h^{n+1}|^2_{H^1(\Omega_c)} \leq Ch^{l-1}|\phi^{n+1}-\phi_h^{n+1}|_{H^1(\Omega_c)} + Ch^{2l-2} + Ch^{\frac{3(l-1)}{2}}|\phi^{n+1}-\phi_h^{n+1}|^{\frac{1}{2}}_{H^1(\Omega_c)},$$

with Young's inequality ($ab \leq a^2 + \frac{b^2}{4}$) in the first and ($ab \leq \frac{3a^{\frac{4}{3}}}{4} + \frac{b^4}{4}$) in the last term of 4.21, we have

$$(1-\frac{1}{2})|\phi^{n+1}-\phi_h^{n+1}|^2_{H^1(\Omega_c)} \leq Ch^{2l-2} + Ch^{2l-2} + Ch^{2l-2}, \tag{4.22}$$

finally, the estimation is given by

$$|\phi^{n+1}-\phi_h^{n+1}|^2_{H^1(\Omega_c)} \leq Ch^{2(l-1)},$$

where C constant depend on $||\phi^{n+1}||_{H^l(\Omega_c)}$. □

Corollary 4.1. *The error estimation for Monge-Ampere equation in H^2-norm fulfills*

$$|\phi^{n+1}-\phi_h^{n+1}|_{H^2(\Omega_c)} \leq Ch^{l-2}. \tag{4.23}$$

Proof. By taking the Projector $\Pi_{\mathcal{V}_h} : L^2(\Omega_c) \longrightarrow \mathcal{V}_h$, we get

$$|\phi^{n+1}-\phi_h^{n+1}|_{H^2(\Omega_c)} \leq |\phi^{n+1}-\phi_h^{n+1} - \Pi_{\mathcal{V}_h}(\phi^{n+1}-\phi_h^{n+1})|_{H^2(\Omega_c)} + |\Pi_{\mathcal{V}_h}(\phi^{n+1}-\phi_h^{n+1})|_{H^2(\Omega_c)}$$

by the proposition 3.2 [52]

$$\begin{aligned}|\phi^{n+1}-\phi_h^{n+1} - \Pi_{\mathcal{V}_h}(\phi^{n+1}-\phi_h^{n+1})|_{H^2(\Omega_c)} &= |\phi^{n+1} - \Pi_{\mathcal{V}_h}(\phi^{n+1})|_{H^2(\Omega_c)} \\ &\leq Ch^{l-2}||\phi^{n+1}||_{H^l(\Omega_c)}.\end{aligned}$$

By the usual inverse inequality as in 4.10, we have

$$\begin{aligned}|\Pi_{\mathcal{V}_h}(\phi^{n+1}-\phi_h^{n+1})|_{H^2(\Omega_c)} &\leq C_\theta h^{-1}|\Pi_{\mathcal{V}_h}(\phi^{n+1}-\phi_h^{n+1})|_{H^1(\Omega_c)} \\ &\leq C_\theta h^{-1}|\phi^{n+1}-\phi_h^{n+1}|_{H^1(\Omega_c)}.\end{aligned}$$

Therefore, according to the last proposition our result is achieved. □

Remark 4.2. *If we denote by $\mathcal{K}_T$ the constant of the contraction T, such that $\mathcal{K}_T < 1$, we have:*

$$|\phi-\phi_h^n|_{H^2(\Omega_c)} \leq C(\left(\mathcal{K}_T\right)^n + h^{l-2}). \tag{4.24}$$

4.3. **Fast Diagonalization method.** In order to devise a fast solver the Poisson and Laplace problems, we choose to follow the work of Sangalli and Tani [48], we describe in the sequel the fast diagonalization method in the case of Isogeometric Analysis. This method was first introduced in [45].
For the seek of simplicity, we shall consider the following Laplace problem,

$$\begin{cases} -\nabla^2 u + \tau u &= f, \quad \Omega \\ u &= 0, \quad \partial\Omega \end{cases} \tag{4.25}$$

The Poisson problem and its solver shall be retrieved with $\tau = 0$. After discretizing the Laplace problem, we get the following linear system

(4.26)
$$\mathcal{L}_\tau x := (K_1 \otimes M_2 \otimes M_3 + M_1 \otimes K_2 \otimes M_3 + M_1 \otimes M_2 \otimes K_3 + \tau M_1 \otimes M_2 \otimes M_3)\, x = b$$

where M and K are unidimensional parametric mass and stiffness matrices respectively, $\otimes$ is a Kronecker product.

We first consider the generalized eigendecompositions problems

$$K_1U_1 = M_1U_1D_1, \quad K_2U_2 = M_2U_2D_2, \quad K_3U_3 = M_3U_3D_3, \tag{4.27}$$

where D_1, D_2 and D_3 are diagonal matrices such that U_1, U_2 and U_3 fulfills

$$U_1^T M_1 U_1 = I_1, \quad U_2^T M_2 U_2 = I_2, \quad U_3^T M_3 U_3 = I_3 \tag{4.28}$$

Therefor, (4.26) can be written as

$$(U_1 \otimes U_2 \otimes U_3)^{-1} \left(D_1 \otimes I_2 \otimes I_3 + I_1 \otimes D_2 \otimes I_3 + I_1 \otimes I_2 \otimes D_3 + \tau I_1 \otimes I_2 \otimes I_3\right) (U_1 \otimes U_2 \otimes U_3)^{-T} x = b \tag{4.29}$$

The direct solver for the Laplace problem (4.26), is then given by the following algorithm, where we omit the initialization step achieved by solving the generalized eigendecompositions in (4.27):

Algorithm 1: `fast_diag`: Fast diagonalization method for Laplace problem

Input : $\mathcal{L}_\tau, b$
Output: x

1 $\tilde{b} \leftarrow (U_1 \otimes U_2 \otimes U_3)\, b$
2 $\tilde{x} \leftarrow \left(D_1 \otimes I_2 \otimes I_3 + I_1 \otimes D_2 \otimes I_3 + I_1 \otimes I_2 \otimes D_3 + \tau I_1 \otimes I_2 \otimes I_3\right)^{-1} \tilde{b}$
3 $x \leftarrow (U_1 \otimes U_2 \otimes U_3)^T\, \tilde{x}$

5. Numerical Results

In this section, we show the performance and a limit of standard method by

- presenting numerical convergence and CPU timing of our Monge-Ampère solver in 2D and 3D,
- showing examples of two-dimensional adaptive mesh generation,
- showing the limitation of using the monitor function with a variation near the boundary.

Our numerical code is written in Python, where intensive computations parts were accelerated using *Pyccel* [47], a static compiler for Python, based on Fortran and C. A laptop with a quad 2.6 GHz Intel Core i7, on ubuntu OS is used throughout the numerical tests.

5.1. **Monge-Ampère solver using BFO method.** The goal of this subsection is to validate our solver 4.1 by a manufactured convex solutions of Monge-Ampere equation. The L^2 residual error tolerance in the Picard iteration is fixed at 10^{-10}.

Example 5.1 (Two-dimensional case)**.** *We test our solver using a convex analytical solution in the unit square, defined as follows.*

$$\phi(\xi,\eta) = exp\Big(\frac{\xi^2+\eta^2}{2}\Big) \tag{5.1}$$

for which the source term is

$$f(\xi,\eta) = (1+\xi^2+\eta^2)exp(\xi^2+\eta^2). \tag{5.2}$$

TABLE 1. CPU-time needed to solve the Monge-Ampere equation and error evolution for various polynomial degrees p along each direction.

(A) Degree $p = 3$

#cells	*H^1-norm*	*H^1-norm MG*	*CPU-time (s)*	*CPU-time (s) MG*	*Nbr-iter*	*Nbr-iter MG*
8	*1.783e-04*	*1.783e-04*	*0.067*	*0.017*	*35*	*26*
16	*2.308e-05*	*2.308e-05*	*0.032*	*0.033*	*35*	*28*
32	*2.934e-06*	*2.934e-06*	*0.106*	*0.094*	*35*	*27*
64	*3.692e-07*	*3.694e-07*	*0.399*	*0.310*	*35*	*25*
128	*4.579e-08*	*4.595e-08*	*1.671*	*1.186*	*35*	*23*

(B) Degree $p = 4$

#cells	*H^1-norm*	*H^1-norm MG*	*CPU-time (s)*	*CPU-time (s) MG*	*Nbr-iter*	*Nbr-iter MG*
8	*6.037e-06*	*6.037e-06*	*0.075*	*0.020*	*35*	*26*
16	*4.020e-07*	*4.020e-07*	*0.046*	*0.041*	*35*	*26*
32	*2.571e-08*	*2.572e-08*	*0.148*	*0.121*	*35*	*24*
64	*2.187e-09*	*2.183e-09*	*0.564*	*0.393*	*35*	*22*
128	*1.693e-09*	*1.684e-09*	*2.297*	*1.404*	*35*	*20*

(C) Degree $p = 5$

#cells	*H^1-norm*	*H^1-norm MG*	*CPU-time (s)*	*CPU-time (s) MG*	*Nbr-iter*	*Nbr-iter MG*
8	*3.584e-07*	*3.584e-07*	*0.042*	*0.026*	*35*	*26*
16	*1.325e-08*	*1.325e-08*	*0.064*	*0.058*	*35*	*26*
32	*1.874e-09*	*1.875e-09*	*0.220*	*0.175*	*35*	*24*
64	*1.711e-09*	*1.712e-09*	*0.831*	*0.571*	*35*	*22*
128	*1.710e-09*	*1.710e-09*	*3.336*	*2.040*	*35*	*20*

Example 5.2 (Three-dimensional case)**.** *We test our solver by a convex manufactured solution on the cube case, defined as :*

$$\phi(\xi,\eta,\gamma) = exp\Big(\frac{\xi^2+\eta^2+\gamma^2}{3}\Big) \tag{5.3}$$

where the source term is

$$f(\xi,\eta,\gamma) = \frac{8}{81}(3+2(\xi^2+\eta^2+\gamma^2))\exp(\xi^2+\eta^2+\gamma^2). \tag{5.4}$$

TABLE 2. CPU-time needed to solve the Monge-Ampere equation and error evolution for various polynomial degrees p along each direction.

(A) Degree $p = 3$

#cells	*H^1-norm*	*H^1-norm MG*	*CPU-time (s)*	*CPU-time (s) MG*	*Nbr-iter*	*Nbr-iter MG*
8	*5.885e-05*	*5.885e-05*	*0.214*	*0.187*	*14*	*10*
16	*7.672e-06*	*7.672e-06*	*1.290*	*0.987*	*14*	*10*
32	*9.793e-07*	*9.793e-07*	*9.671*	*6.683*	*14*	*9*
64	*1.235e-07*	*1.236e-07*	*76.92*	*40.88*	*14*	*7*

(B) Degree $p = 4$

#cells	*H^1-norm*	*H^1-norm MG*	*CPU-time (s)*	*CPU-time (s) MG*	*Nbr-iter*	*Nbr-iter MG*
8	*1.758e-06*	*1.758e-06*	*0.528*	*0.472*	*14*	*10*
16	*1.184e-07*	*1.184e-07*	*3.547*	*2.680*	*14*	*10*
32	*7.658e-09*	*7.658e-09*	*27.35*	*16.42*	*14*	*8*
64	*9.609e-10*	*9.617e-10*	*216.1*	*98.91*	*14*	*6*

(C) Degree $p = 5$

#cells	*H^1-norm*	*H^1-norm MG*	*CPU-time (s)*	*CPU-time (s) MG*	*Nbr-iter*	*Nbr-iter MG*
8	*9.064e-08*	*9.064e-08*	*1.206*	*1.181*	*14*	*10*
16	*3.266e-09*	*3.266e-09*	*9.118*	*6.787*	*14*	*10*
32	*8.756e-10*	*8.757e-10*	*71.63*	*42.58*	*14*	*8*
64	*8.561e-10*	*8.563e-10*	*582.9*	*270.6*	*14*	*6*

Remark 5.1. *For numerical purpose d! is replaced by δ such that*

$$\mathcal{G}(\phi) = \Big((\Delta\phi)^d + \delta(f(\nabla\phi)) - \det(H(\phi))\Big)^{\frac{1}{d}} \tag{5.5}$$

Then, we see that a choice of δ in three-dimension affects the number of iterations of the Picard algorithm to achieve the same precision Fig. 3.

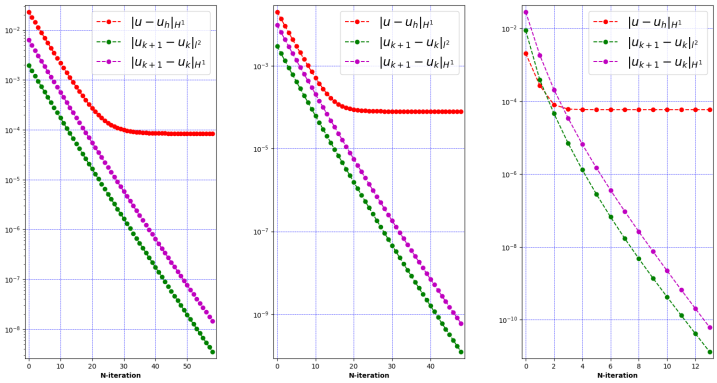

FIGURE 3. L^2 and H^1-residual and H^1-error profiles with different choices of $\delta = 6$ (left), $\delta = 9$ (middle) and $\delta = 27$ (right) using bi-cubic B-splines.

5.2. **Adaptive mesh generation.** In the following examples, we show in different figures two-dimensional adaptive meshes obtained using bi-cubic B-splines with $\#cells = 32$. Here, a tolerance of L^2 residual error in Picard's iteration is fixed at 10^{-8}.

As in [18], we define the total error in the scheme and the quality of the adapted grid by measuring the global displacement of the grid points.

(5.6)
$$Err^2 = \int_{\Omega_c} |\det(H(\phi)) - \widehat{f}(\nabla(\phi))|^2 \, \mathrm{d}x \qquad Qual^2 = \int_{\Omega_c} |\nabla(\phi(x)) - x|^2 / \widehat{f}(\nabla(\phi(x))) \, \mathrm{d}x.$$

Examples on the unit square.

Example 5.3. *In this example, the monitor function represents a circle centered in the middle of the square Fig. 4. We display CPU-time, the min-max value of the Jacobian function, and the accuracy in the Table 3.*

$$\rho(\xi, \eta) = 1 + 5\exp(-50\big|(\xi - 0.5)^2 + (\eta - 0.5)^2 - 0.09\big|),$$

TABLE 3. Grid convergence analysis for various polynomial degrees p along each direction.

(A) Degree $p = 3$

$\#cells$	Err	$CPU\text{-}time\ (s)$	$Qual$	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*4.195e-02*	*0.059*	*6.030e-02*	*2.784e-01*	*1.796e+00*
32	*1.124e-02*	*0.080*	*6.003e-02*	*2.759e-01*	*1.627e+00*
64	*3.023e-03*	*0.257*	*6.005e-02*	*2.707e-01*	*1.625e+00*
128	*1.090e-03*	*0.934*	*6.005e-02*	*2.711e-01*	*1.625e+00*

(B) Degree $p = 4$

$\#cells$	Err	$CPU\text{-}time\ (s)$	$Qual$	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*2.391e-02*	*0.079*	*5.993e-02*	*2.864e-01*	*1.674e+00*
32	*6.038e-03*	*0.123*	*6.005e-02*	*2.803e-01*	*1.626e+00*
64	*2.345e-03*	*0.393*	*6.005e-02*	*2.757e-01*	*1.625e+00*
128	*7.728e-04*	*1.419*	*6.005e-02*	*2.730e-01*	*1.625e+00*

(C) Degree $p = 5$

$\#cells$	Err	$CPU\text{-}time\ (s)$	$Qual$	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*2.494e-02*	*0.085*	*5.996e-02*	*2.945e-01*	*1.671e+00*
32	*6.633e-03*	*0.178*	*6.005e-02*	*2.795e-01*	*1.628e+00*
64	*2.119e-03*	*0.569*	*6.005e-02*	*2.755e-01*	*1.625e+00*
128	*8.364e-04*	*1.936*	*6.005e-02*	*2.733e-01*	*1.625e+00*

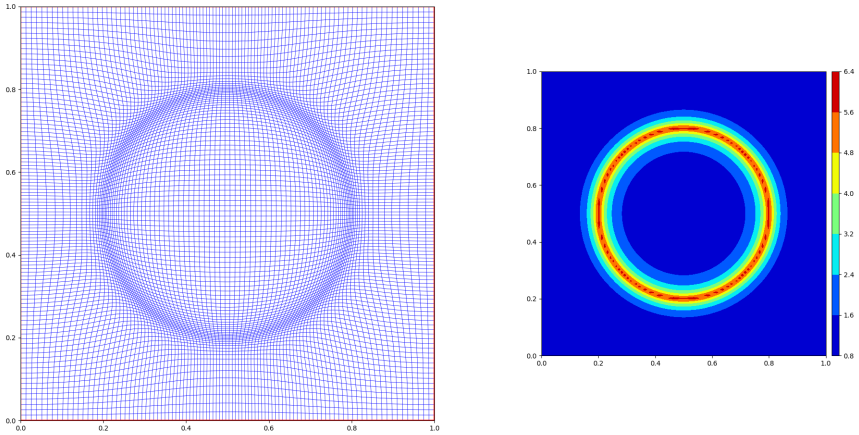

FIGURE 4. (left) Adaptive meshes and (right) the monitor function.

Example 5.4. *This example in the unit square corresponds to a spiral centered at (0.7, 0.5) with very tight arms Fig. 5 (right).*

$$\rho(\xi,\eta) = 1 + 9/\Big(1 + \big(10r(\xi,\eta)\cos(\theta(\xi,\eta) - 20r(\xi,\eta))\big)^2\Big), \tag{5.7}$$

where $\theta(\xi,\eta) = \arctan((\eta-0.5)/(\xi-0.7))$ *and* $r(\xi,\eta) = \sqrt{(\xi-0.7)^2+(\eta-0.5)^2}$.

A monitor function close to the boundary can generate a negative value in the Jacobian function Table 4 and boundary errors Fig 6.

TABLE 4. Grid convergence analysis for various polynomial degrees p along each direction.

(A) Degree $p = 3$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*4.816e-01*	*0.127*	*1.512e-01*	*-2.286e-02*	*3.587e+00*
32	*1.892e-01*	*0.303*	*1.201e-01*	*-7.120e-02*	*3.644e+00*
64	*4.322e-02*	*0.784*	*1.083e-01*	*1.936e-01*	*3.382e+00*
128	*9.048e-03*	*2.259*	*1.081e-01*	*3.464e-01*	*3.128e+00*

(B) Degree $p = 4$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*4.702e-01*	*0.190*	*1.414e-01*	*5.377e-02*	*4.142e+00*
32	*1.288e-01*	*0.461*	*1.103e-01*	*4.119e-02*	*3.384e+00*
64	*2.965e-02*	*0.978*	*1.081e-01*	*2.215e-01*	*3.300e+00*
128	*4.821e-03*	*3.194*	*1.081e-01*	*3.511e-01*	*3.136e+00*

(C) Degree $p = 5$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*4.640e-01*	*0.320*	*1.459e-01*	*-5.108e-02*	*5.528e+00*
32	*1.267e-01*	*0.549*	*1.099e-01*	*6.682e-02*	*3.461e+00*
64	*2.787e-02*	*1.355*	*1.081e-01*	*2.348e-01*	*3.302e+00*
128	*4.112e-03*	*4.430*	*1.081e-01*	*3.510e-01*	*3.140e+00*

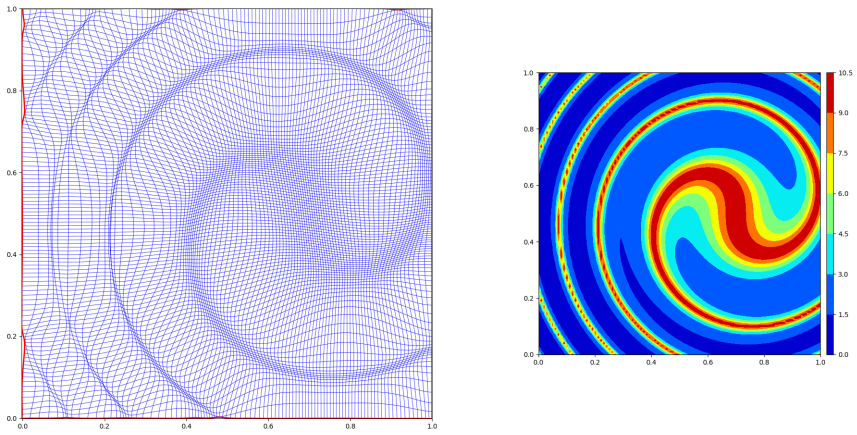

FIGURE 5. (left) Adaptive meshes and (right) the monitor function.

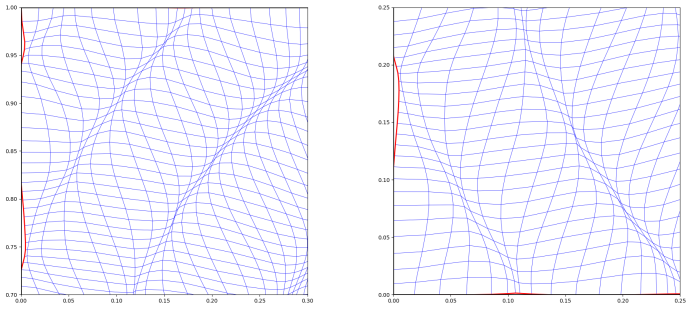

FIGURE 6. A close-up of the region near the boundary.

Examples on more general geometries.

Example 5.5 (Circle)**.** *In this test we show the Mickey Mouse example, the mesh quality using a new technique of mesh adaptation can be seen clearly in the Figs. 7-8. However, to clear the idea about a limit using the standard method, we add the last colon in the Table 5 showing the error at the boundary.*

(5.8)

$$\rho(x,y) = 1. + 5\exp(-100|(x-0.45)^2 + (y-0.4)^2 - 0.1|) + 5\exp(-100|x^2+y^2-0.2|) + 5\exp(-100|(x+0.45)^2 + (y-0.4)^2 - 0.1|).$$

TABLE 5. Grid convergence analysis for various polynomial degrees p along each direction.

(A) Degree $p = 3$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$	$\max_{\partial\Omega_c}\|(\mathbf{\Psi} - x)\cdot\vec{n}\|$
16	*2.720e-01*	*0.070*	*5.179e-02*	*9.727e-02*	*1.719e+00*	*2.426e-03*
32	*7.321e-02*	*0.146*	*5.786e-02*	*9.899e-02*	*1.412e+00*	*3.969e-04*
64	*1.765e-02*	*0.414*	*5.807e-02*	*1.338e-01*	*1.314e+00*	*4.189e-05*
128	*5.066e-03*	*1.516*	*5.811e-02*	*1.365e-01*	*1.280e+00*	*2.881e-07*

(B) Degree $p = 4$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$	$\max_{\partial\Omega_c}\|(\mathbf{\Psi} - x)\cdot\vec{n}\|$
16	*2.353e-01*	*0.090*	*5.217e-02*	*1.835e-01*	*1.613e+00*	*1.848e-03*
32	*5.832e-02*	*0.222*	*5.818e-02*	*1.554e-01*	*1.429e+00*	*4.023e-04*
64	*1.375e-02*	*0.617*	*5.807e-02*	*1.373e-01*	*1.307e+00*	*4.301e-05*
128	*3.558e-03*	*2.158*	*5.811e-02*	*1.353e-01*	*1.280e+00*	*3.245e-07*

(C) Degree $p = 5$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$	$\max_{\partial\Omega_c}\|(\mathbf{\Psi} - x)\cdot\vec{n}\|$
16	*2.205e-01*	*0.123*	*5.482e-02*	*1.490e-01*	*1.621e+00*	*2.606e-03*
32	*4.607e-02*	*0.292*	*5.812e-02*	*1.487e-01*	*1.367e+00*	*2.842e-04*
64	*1.273e-02*	*0.858*	*5.807e-02*	*1.414e-01*	*1.306e+00*	*3.999e-05*
128	*3.496e-03*	*3.093*	*5.811e-02*	*1.354e-01*	*1.280e+00*	*5.209e-07*

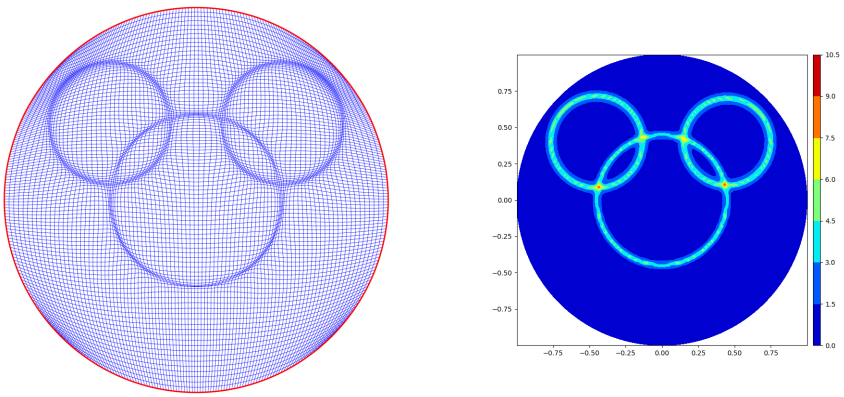

FIGURE 7. (left) Adaptive meshes and (right) the monitor function.

Example 5.6 (Circle)**.** *In this example we use a non-axisymmetric monitor function defined by:*

$$\rho(x,y) = 1 + 5/sech\Big(5\big((x-\frac{\sqrt{3}}{2})^2 + (y-0.5)^2 - \frac{\pi^2}{4}\big)\Big) + 5/sech\Big(5\big((x+\frac{\sqrt{3}}{2})^2 + (y-0.5)^2 - \frac{\pi^2}{4}\big)\Big). \tag{5.9}$$

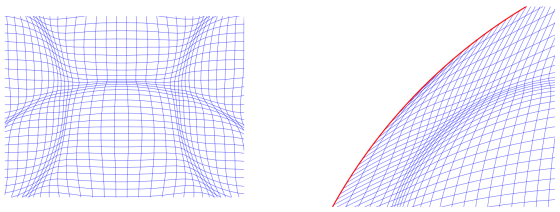

FIGURE 8. A close-up of the region where the two circles intersect (left) and on one of the singular points (right).

Results of adaptive meshes shown in Fig. 9, there is no mesh-distortion but there are some errors at the boundary Table 6 that vanish with refinement.

TABLE 6. Grid convergence analysis for various polynomial degrees p along each direction.

(A) Degree $p = 3$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$	$\max_{\partial\Omega_c}\|(\mathbf{\Psi} - x)\cdot\vec{n}\|$
16	*7.015e-02*	*0.070*	*9.650e-02*	*1.742e-01*	*2.167e+00*	*1.453e-03*
32	*1.646e-02*	*0.137*	*9.627e-02*	*1.706e-01*	*1.926e+00*	*3.079e-04*
64	*3.468e-03*	*0.446*	*9.620e-02*	*1.699e-01*	*1.869e+00*	*4.823e-05*
128	*8.166e-04*	*1.695*	*9.619e-02*	*1.701e-01*	*1.869e+00*	*6.217e-06*

(B) Degree $p = 4$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$	$\max_{\partial\Omega_c}\|(\mathbf{\Psi} - x)\cdot\vec{n}\|$
16	*6.839e-02*	*0.096*	*9.577e-02*	*1.671e-01*	*2.015e+00*	*1.665e-03*
32	*1.168e-02*	*0.214*	*9.619e-02*	*1.704e-01*	*1.899e+00*	*1.381e-04*
64	*9.126e-04*	*0.689*	*9.619e-02*	*1.701e-01*	*1.869e+00*	*1.590e-05*
128	*7.788e-05*	*2.609*	*9.619e-02*	*1.701e-01*	*1.868e+00*	*9.362e-07*

(C) Degree $p = 5$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$	$\max_{\partial\Omega_c}\|(\mathbf{\Psi} - x)\cdot\vec{n}\|$
16	*4.822e-02*	*0.131*	*9.631e-02*	*1.660e-01*	*1.992e+00*	*9.295e-04*
32	*9.567e-03*	*0.284*	*9.620e-02*	*1.702e-01*	*1.888e+00*	*1.254e-04*
64	*6.496e-04*	*0.945*	*9.619e-02*	*1.701e-01*	*1.869e+00*	*4.599e-06*
128	*1.833e-05*	*3.512*	*9.619e-02*	*1.701e-01*	*1.868e+00*	*2.851e-07*

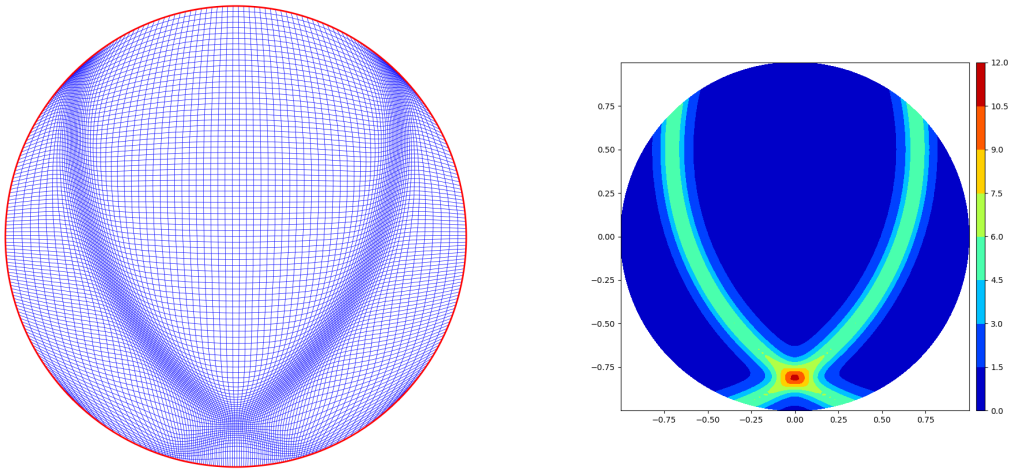

FIGURE 9. (left) Adaptive meshes and (right) the monitor function.

6. Mixed Finite Elements approach

6.1. **Mixed formulation of Benamou-Froese-Oberman system.** We can obtain a more accurate mapping with an exact normal boundary condition by using a mixed formulation that consists of taking into a count the boundary condition in strong form $\boldsymbol{\Psi}(\partial\Omega_c) = \partial\Omega_c$. Hence, returning to the system 4.2, by introducing $\boldsymbol{\Psi} = \nabla\phi$, we have

$$\left\{\begin{array}{lll} \nabla\cdot\boldsymbol{\Psi} & = \mathcal{G}(\boldsymbol{\Psi}) & \text{in } \Omega_c,\\ \boldsymbol{\Psi} & = \nabla\phi & \text{in } \Omega_c,\\ \boldsymbol{\Psi}\cdot\boldsymbol{n} & = \boldsymbol{x}\cdot\boldsymbol{n} & \text{on } \partial\Omega_c, \end{array}\right. \tag{6.1}$$

where

$$\mathcal{G}(\boldsymbol{\Psi}) = \Big((\nabla.\boldsymbol{\Psi})^d + d!(f(\boldsymbol{\Psi}) - \det\left(\nabla(\boldsymbol{\Psi})\right)\ )\Big)^{\frac{1}{d}}. \tag{6.2}$$

Hence, let's introduce the decomposition $\boldsymbol{\Psi} = \boldsymbol{\Psi}_0 + \boldsymbol{\Psi}_N$ such that $\boldsymbol{\Psi}_0\cdot\boldsymbol{n} = 0$ and $\boldsymbol{\Psi}_N\cdot\boldsymbol{n} = \boldsymbol{x}\cdot\boldsymbol{n}$. Therefor, the problem (6.1) writes

$$\left\{\begin{array}{lll} \nabla\cdot\boldsymbol{\Psi}_0 & = \mathcal{G}(\boldsymbol{\Psi}) - \nabla\cdot\boldsymbol{\Psi}_N & \text{in } \Omega_c,\\ \boldsymbol{\Psi}_0 & = \nabla\phi - \boldsymbol{\Psi}_N & \text{in } \Omega_c,\\ \boldsymbol{\Psi}_0\cdot\boldsymbol{n} & = 0 & \text{on } \partial\Omega_c. \end{array}\right. \tag{6.3}$$

Then, the variational formulation of 6.3, is :

Find $(\boldsymbol{\Psi},\phi)\in \boldsymbol{H}_0(\mathrm{div},\Omega)\times L^2_0(\Omega)$ *such that* $\forall(\boldsymbol{v},\varphi)\in \boldsymbol{H}_0(\mathrm{div},\Omega)\times L^2_0(\Omega)$

$$\left\{\begin{array}{lll} \int_\Omega \boldsymbol{\Psi}_0\cdot\boldsymbol{v} & + \int_\Omega \phi\ \nabla\cdot\boldsymbol{v} & = -\int_\Omega \boldsymbol{\Psi}_N\cdot\boldsymbol{v},\\ \int_\Omega (\nabla\cdot\boldsymbol{\Psi}_0)\,\varphi & & = \int_\Omega \mathcal{G}(\boldsymbol{\Psi})\varphi - \int_\Omega (\nabla\cdot\boldsymbol{\Psi}_N)\,\varphi, \end{array}\right. \tag{6.4}$$

6.2. **Fast Diagonalization method for Mixed Variational Formulation.** Following the same steps as in section 4.3, we derive a new algorithm of fast diagonalization in two and three dimensions.

We consider for simplicity Poisson equation with homogeneous Neumann boundary conditions.

$$\left\{\begin{array}{ll} -\nabla^2\phi = f & \text{in } \Omega,\\ \nabla\phi\cdot\boldsymbol{n} = 0 & \text{on } \partial\Omega, \end{array}\right. \tag{6.5}$$

As well known, the associated variational formulation can be written as a minimization problem over $H^1(\Omega)$. Let us now, introduce an auxiliary variable $\boldsymbol{u} = \boldsymbol{\nabla}\phi$. The Poisson problem is then equivalent to the following system

$$\left\{\begin{array}{lll} -\nabla\cdot\boldsymbol{u} = & f & \text{in } \Omega,\\ \boldsymbol{u} = & \boldsymbol{\nabla}\phi & \text{in } \Omega,\\ \boldsymbol{u}\cdot\boldsymbol{n} = & 0 & \text{on } \partial\Omega. \end{array}\right. \tag{6.6}$$

The mixed weak formulation for the Poisson equation is

Find $(\boldsymbol{u},\phi)\in \boldsymbol{H}_0(\mathrm{div},\Omega)\times L^2_0(\Omega)$ *such that* $\forall(\boldsymbol{\Psi},\varphi)\in \boldsymbol{H}_0(\mathrm{div},\Omega)\times L^2_0(\Omega)$

$$\left\{\begin{array}{lll} \int_\Omega \boldsymbol{u}\cdot\boldsymbol{\Psi} & + \int_\Omega \phi\ \nabla\cdot\boldsymbol{\Psi} & = 0,\\ \int_\Omega (\nabla\cdot\boldsymbol{u})\,\varphi & & = -\int_\Omega f\varphi. \end{array}\right. \tag{6.7}$$

Using the discrete spaces defined in 2.13 and 2.14, we get the following variational problem

Find $(\boldsymbol{u}_h, \phi_h) \in \boldsymbol{V}_h(\mathrm{div}, \Omega) \times V_h(L^2, \Omega)$ *such that* $\forall (\boldsymbol{\Psi}, \varphi) \in \boldsymbol{V}_h(\mathrm{div}, \Omega) \times V_h(L^2, \Omega)$

$$\begin{cases} \int_\Omega \boldsymbol{u}_h \cdot \boldsymbol{\Psi} & + \int_\Omega \phi_h \ \nabla \cdot \boldsymbol{\Psi} \quad = 0, \\ \int_\Omega (\nabla \cdot \boldsymbol{u}_h)\, \varphi & \qquad\qquad\quad = -\int_\Omega f \varphi. \end{cases} \tag{6.8}$$

We then recover a saddle point problem of the form

$$\mathcal{A} \begin{pmatrix} \boldsymbol{u}_h \\ \phi_h \end{pmatrix} = \begin{pmatrix} A & B \\ B^T & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_h \\ \phi_h \end{pmatrix} = \begin{pmatrix} 0 \\ F \end{pmatrix} \tag{6.9}$$

with

$$F_i = -\int_\Omega f \phi_i \, \mathrm{d}\mathbf{x}, \quad 1 \leq i \leq N_{V_h}$$

where N_{V_h} is a number of B-spline basis functions.

Notations. Before expliciting the matrix forms in 2D and 3D, we start by introducing some 1D matrices that will be needed.

$$\begin{cases} (M_s)_{i_s, j_s} & = \int_0^1 B^{p_s}_{\boldsymbol{\mu}_s, i_s} B^{p_s}_{\boldsymbol{\mu}_s, j_s} \, \mathrm{d}x_s \\ (D_s)_{i_s, j_s} & = \int_0^1 B^{p_s - 1}_{\boldsymbol{\mu}_s - 1, i_s} B^{p_s - 1}_{\boldsymbol{\mu}_s - 1, j_s} \, \mathrm{d}x_s \\ (R_s)_{i_s, j_s} & = \int_0^1 B^{p_s - 1}_{\boldsymbol{\mu}_s - 1, i_s} \frac{\mathrm{d} B^{p_s}_{\boldsymbol{\mu}_s, j_s}}{\mathrm{d}\xi} \, \mathrm{d}x_s \end{cases} \tag{6.10}$$

where

$$B^{p_s}_{\boldsymbol{\mu}_s, i_s} = B^{p_s}_{i_s}, \text{ such that } B^{p_s}_{i_s} \in \mathcal{S}^{p_s}_{\boldsymbol{\mu}_s}, \ s = 1, 2.., d.$$

Matrix form in 2D. For the sake of simplicity we shall introduce the auxiliary scalar functions

$$\begin{cases} \Psi^1_{\boldsymbol{j}} = B^{p_1}_{\boldsymbol{\mu}_1, j_1} B^{p_2 - 1}_{\boldsymbol{\mu}_2 - 1, j_2} \\ \Psi^2_{\boldsymbol{j}} = B^{p_1 - 1}_{\boldsymbol{\mu}_1 - 1, j_1} B^{p_2}_{\boldsymbol{\mu}_2, j_2} \\ \phi_{\boldsymbol{j}} = B^{p_1 - 1}_{\boldsymbol{\mu}_1 - 1, j_1} B^{p_2 - 1}_{\boldsymbol{\mu}_2 - 1, j_2} \end{cases} \tag{6.11}$$

where $\boldsymbol{j} = (j_1, j_2)$, we also define the vectors $\boldsymbol{e}_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $\boldsymbol{e}_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. Therefor, the expression of $\boldsymbol{u}_h \in \boldsymbol{V}_h(\mathrm{div}, \Omega)$ becomes

$$\boldsymbol{u}_h = \sum_{\boldsymbol{j}} \left(u^1_{\boldsymbol{j}} \Psi^1_{\boldsymbol{j}} \boldsymbol{e}_1 + u^2_{\boldsymbol{j}} \Psi^2_{\boldsymbol{j}} \boldsymbol{e}_2 \right)$$

Because $\boldsymbol{e}_i \cdot \boldsymbol{e}_j = \delta_{ij}$, we find that A is a block diagonal matrix of the form

$$A = \begin{pmatrix} A_{11} & 0 \\ 0 & A_{22} \end{pmatrix} \tag{6.12}$$

where

$$A_{11\boldsymbol{i},\boldsymbol{j}} = \int_\Omega \Psi^1_{\boldsymbol{i}} \Psi^1_{\boldsymbol{j}} \,\mathrm{d}\mathbf{x} = (M_1 \otimes D_2)_{\boldsymbol{ij}}$$
$$A_{22\boldsymbol{i},\boldsymbol{j}} = \int_\Omega \Psi^2_{\boldsymbol{i}} \Psi^2_{\boldsymbol{j}} \,\mathrm{d}\mathbf{x} = (D_1 \otimes M_2)_{\boldsymbol{ij}}$$

While for B we have

$$B = \begin{pmatrix} B_1 \\ B_2 \end{pmatrix} \tag{6.13}$$

where

$$B_{1\boldsymbol{i},\boldsymbol{j}} = \int_\Omega \phi_{\boldsymbol{j}} \partial_{x_1} \Psi^1_{\boldsymbol{i}} \,\mathrm{d}\mathbf{x} = (R_1 \otimes D_2)_{\boldsymbol{ij}}$$
$$B_{2\boldsymbol{i},\boldsymbol{j}} = \int_\Omega \phi_{\boldsymbol{j}} \partial_{x_2} \Psi^2_{\boldsymbol{i}} \,\mathrm{d}\mathbf{x} = (D_1 \otimes R_2)_{\boldsymbol{ij}}$$

The final matrix form is

$$\mathcal{A} = \begin{pmatrix} M_1 \otimes D_2 & 0 & R_1 \otimes D_2 \\ 0 & D_1 \otimes M_2 & D_1 \otimes R_2 \\ (R_1 \otimes D_2)^T & (D_1 \otimes R_2)^T & 0 \end{pmatrix}$$

Matrix form in 3D. It is possible to extend the notation of the auxiliary scalar functions to the 3D case as follows:

$$\Psi^1_{\boldsymbol{j}} = B^{p_1}_{\boldsymbol{\mu}_1, j_1} B^{p_2-1}_{\boldsymbol{\mu}_2-1, j_2} B^{p_3-1}_{\boldsymbol{\mu}_3-1, j_3}$$
$$\Psi^2_{\boldsymbol{j}} = B^{p_1-1}_{\boldsymbol{\mu}_1-1, j_1} B^{p_2}_{\boldsymbol{\mu}_2, j_2} B^{p_3-1}_{\boldsymbol{\mu}_3-1, j_3}$$
$$\Psi^3_{\boldsymbol{j}} = B^{p_1-1}_{\boldsymbol{\mu}_1-1, j_1} B^{p_2-1}_{\boldsymbol{\mu}_2-1, j_2} B^{p_3}_{\boldsymbol{\mu}_3, j_3}$$

where $\boldsymbol{j} = (j_1, j_2, j_3)$, and

$$\phi_{\boldsymbol{j}} = B^{p_1-1}_{\boldsymbol{\mu}_1-1, j_1} B^{p_2-1}_{\boldsymbol{\mu}_2-1, j_2} B^{p_3-1}_{\boldsymbol{\mu}_3-1, j_3}$$

we also define the vectors $\boldsymbol{e}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$, $\boldsymbol{e}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$ and $\boldsymbol{e}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$. The expression of $\boldsymbol{u}_h \in \boldsymbol{V}_h(\mathrm{div}, \Omega)$ equivalent to

$$\boldsymbol{u}_h = \sum_{\boldsymbol{j}} \left(u^1_{\boldsymbol{j}} \Psi^1_{\boldsymbol{j}} \boldsymbol{e}_1 + u^2_{\boldsymbol{j}} \Psi^2_{\boldsymbol{j}} \boldsymbol{e}_2 + u^3_{\boldsymbol{j}} \Psi^3_{\boldsymbol{j}} \boldsymbol{e}_3 \right)$$

and as a consequence of $\boldsymbol{e}_i \cdot \boldsymbol{e}_j = \delta_{ij}$, we have

$$A = \begin{pmatrix} A_{11} & 0 & 0 \\ 0 & A_{22} & 0 \\ 0 & 0 & A_{33} \end{pmatrix} \tag{6.14}$$

where

$$A_{11\boldsymbol{i},\boldsymbol{j}} = \int_\Omega \Psi^1_{\boldsymbol{i}} \Psi^1_{\boldsymbol{j}} \,\mathrm{d}\mathbf{x} = (M_1 \otimes D_2 \otimes D_3)_{\boldsymbol{ij}}$$
$$A_{22\boldsymbol{i},\boldsymbol{j}} = \int_\Omega \Psi^2_{\boldsymbol{i}} \Psi^2_{\boldsymbol{j}} \,\mathrm{d}\mathbf{x} = (D_1 \otimes M_2 \otimes D_3)_{\boldsymbol{ij}}$$
$$A_{33\boldsymbol{i},\boldsymbol{j}} = \int_\Omega \Psi^3_{\boldsymbol{i}} \Psi^3_{\boldsymbol{j}} \,\mathrm{d}\mathbf{x} = (D_1 \otimes D_2 \otimes M_3)_{\boldsymbol{ij}}$$

While for B we have

$$B = \begin{pmatrix} B_1 \\ B_2 \\ B_3 \end{pmatrix} \tag{6.15}$$

where

$$\begin{aligned} B_{1\boldsymbol{i},\boldsymbol{j}} &= \int_\Omega \phi_{\boldsymbol{j}} \partial_{x_1} \Psi^1_{\boldsymbol{i}} \, \mathrm{d}\mathbf{x} = (R_1 \otimes D_2 \otimes D_3)_{\boldsymbol{ij}} \\ B_{2\boldsymbol{i},\boldsymbol{j}} &= \int_\Omega \phi_{\boldsymbol{j}} \partial_{x_2} \Psi^2_{\boldsymbol{i}} \, \mathrm{d}\mathbf{x} = (D_1 \otimes R_2 \otimes D_3)_{\boldsymbol{ij}} \\ B_{3\boldsymbol{i},\boldsymbol{j}} &= \int_\Omega \phi_{\boldsymbol{j}} \partial_{x_3} \Psi^3_{\boldsymbol{i}} \, \mathrm{d}\mathbf{x} = (D_1 \otimes D_2 \otimes R_3)_{\boldsymbol{ij}} \end{aligned}$$

Finally,

$$\mathcal{A} = \begin{pmatrix} M_1 \otimes D_2 \otimes D_3 & 0 & 0 & R_1 \otimes D_2 \otimes D_3 \\ 0 & D_1 \otimes M_2 \otimes D_3 & 0 & D_1 \otimes R_2 \otimes D_3 \\ 0 & 0 & D_1 \otimes D_2 \otimes M_3 & D_1 \otimes D_2 \otimes R_3 \\ (R_1 \otimes D_2 \otimes D_3)^T & (D_1 \otimes R_2 \otimes D_3)^T & (D_1 \otimes D_2 \otimes R_3)^T & 0 \end{pmatrix}$$

6.2.1. *Linear Solver.* Based on the problem 6.9, we consider the following saddle point problem

$$\mathcal{A} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} A & B \\ B^T & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \tag{6.16}$$

Let us define $\tilde{A} = B^T A^{-1} B$ as the Schur complement of (6.16). Using the Uzawa method leads to

$$\begin{cases} \tilde{A} x_2 = B^T A^{-1} b_1 - b_2 \\ A x_1 = b_1 - B x_2 \end{cases} \tag{6.17}$$

Fast Diagonalization method (2D). The matrix $\tilde{A}$ can be expressed as the sum of Kronecker matrices

$$\tilde{A} = D_1 \otimes \left(R_2^T M_2^{-1} R_2 \right) + \left(R_1^T M_1^{-1} R_1 \right) \otimes D_2 \tag{6.18}$$

We have

$$B^T A^{-1} b_1 = \begin{pmatrix} (R_1 \otimes D_2)^T & (D_1 \otimes R_2)^T \end{pmatrix} \begin{pmatrix} (M_1 \otimes D_2)^{-1} & 0 \\ 0 & (D_1 \otimes M_2)^{-1} \end{pmatrix} \begin{pmatrix} b_{11} \\ b_{12} \end{pmatrix}$$

$$= \begin{pmatrix} R_1^T \otimes D_2^T & D_1^T \otimes R_2^T \end{pmatrix} \begin{pmatrix} M_1^{-1} \otimes D_2^{-1} & 0 \\ 0 & D_1^{-1} \otimes M_2^{-1} \end{pmatrix} \begin{pmatrix} b_{11} \\ b_{12} \end{pmatrix}$$

$$= \begin{pmatrix} \left(R_1^T \otimes D_2^T\right)\left(M_1^{-1} \otimes D_2^{-1}\right) & \left(D_1^T \otimes R_2^T\right)\left(D_1^{-1} \otimes M_2^{-1}\right) \end{pmatrix} \begin{pmatrix} b_{11} \\ b_{12} \end{pmatrix}$$

$$= \begin{pmatrix} \left(R_1^T M_1^{-1}\right) \otimes I_2 & I_1 \otimes \left(R_2^T M_2^{-1}\right) \end{pmatrix} \begin{pmatrix} b_{11} \\ b_{12} \end{pmatrix}$$

$$= \left(\left(R_1^T M_1^{-1}\right) \otimes I_2\right) b_{11} + \left(I_1 \otimes \left(R_2^T M_2^{-1}\right)\right) b_{12}$$

Therefor

$$\tilde{A} x_2 = b \tag{6.19}$$

where

$$b = \left(\left(R_1^T M_1^{-1}\right) \otimes I_2\right) b_{11} + \left(I_1 \otimes \left(R_2^T M_2^{-1}\right)\right) b_{12} - b_2$$

On the other hand, we have

$$Bx_2 = \begin{pmatrix} (R_1 \otimes D_2)\, x_2 \\ (D_1 \otimes R_2)\, x_2 \end{pmatrix} \tag{6.20}$$

By introducing $x_1 := (x_{11}, x_{12})^T$, the equation $Ax_1 = b_1 - Bx_2$ is equivalent to

$$\begin{cases} x_{11} = \left(M_1^{-1} \otimes D_2^{-1}\right) b_{11} - \left(M_1^{-1} \otimes D_2^{-1}\right)(R_1 \otimes D_2)\, x_2 \\ x_{12} = \left(D_1^{-1} \otimes M_2^{-1}\right) b_{12} - \left(D_1^{-1} \otimes M_2^{-1}\right)(D_1 \otimes R_2)\, x_2 \end{cases}$$

which can be simplified as

$$\begin{cases} x_{11} = \left(M_1^{-1} \otimes D_2^{-1}\right) b_{11} - \left(M_1^{-1} R_1 \otimes I_2\right) x_2 \\ x_{12} = \left(D_1^{-1} \otimes M_2^{-1}\right) b_{12} - \left(I_1 \otimes M_2^{-1} R_2\right) x_2 \end{cases}$$

In order to devise a fast solver (6.17), we choose to follow the approach of Sangalli and Tani [48]. We describe in the sequel the fast diagonalization method in the case of Isogeometric Analysis. This method was first introduced in [45].
We consider the generalized eigendecompositions problems

$$\left(R_1^T M_1^{-1} R_1\right) U_1 = D_1 U_1 d_1, \quad \left(R_2^T M_2^{-1} R_2\right) U_2 = D_2 U_2 d_2 \tag{6.21}$$

where d_1 and d_2 are diagonal matrices such that U_1 and U_2 fulfills

$$U_1^T D_1 U_1 = I_1, \quad U_2^T D_2 U_2 = I_2 \tag{6.22}$$

Therefor,

$$(U_1 \otimes U_2)^{-T} (d_1 \otimes I_2 + I_1 \otimes d_2) (U_1 \otimes U_2)^{-1} x = b \tag{6.23}$$

The direct solver for (6.17), is then given by the following algorithm, where we omit the initialization step achieved by solving the generalized eigendecompositions in (6.21):

Algorithm 2: `fast_diag`: Fast diagonalization method for (6.17) in the 2D case

Input : b
Output: x

$(b_{11}, b_{12}), b_2 \leftarrow \texttt{unfold}(b)$
$r_2 \leftarrow \left(\left(R_1^T M_1^{-1}\right) \otimes I_2\right) b_{11} + \left(I_1 \otimes \left(R_2^T M_2^{-1}\right)\right) b_{12} - b_2$
$\tilde{r_2} \leftarrow \left(U_1 \otimes U_2\right)^T r_2$
$\tilde{x_2} \leftarrow \left(d_1 \otimes I_2 + I_1 \otimes d_2\right)^{-1} \tilde{r_2}$
$x_2 \leftarrow \left(U_1 \otimes U_2\right) \tilde{x_2}$
$x_{11} \leftarrow \left(M_1^{-1} \otimes D_2^{-1}\right) b_{11} - \left(M_1^{-1} R_1 \otimes I_2\right) x_2$
$x_{12} \leftarrow \left(D_1^{-1} \otimes M_2^{-1}\right) b_{12} - \left(I_1 \otimes M_2^{-1} R_2\right) x_2$
$x \leftarrow \texttt{fold}(x_{11}, x_{12}, x_2)$

Fast Diagonalization method (3D). The matrix $\tilde{A}$ in (3D) can be expressed as the sum of Kronecker matrices

(6.24)
$$\tilde{A} = D_1 \otimes D_2 \otimes \left(R_3^T M_3^{-1} R_3\right) + D_1 \otimes \left(R_2^T M_2^{-1} R_2\right) \otimes D_3 + \left(R_1^T M_1^{-1} R_1\right) \otimes D_2 \otimes D_3$$

And as in (2D) case we can show that we have

$$B^T A^{-1} b_1 = \left(\left(R_1^T M_1^{-1}\right) \otimes I_2 \otimes I_3\right) b_{11} + \left(I_1 \otimes \left(R_2^T M_2^{-1}\right) \otimes I_3\right) b_{12} + \left(I_1 \otimes I_2 \otimes \left(R_3^T M_3^{-1}\right)\right) b_{13}.$$

Therefor

(6.25)
$$\tilde{A} x_2 = \left(\left(R_1^T M_1^{-1}\right) \otimes I_2 \otimes I_3\right) b_{11} + \left(I_1 \otimes \left(R_2^T M_2^{-1}\right) \otimes I_3\right) b_{12} + \left(I_1 \otimes I_2 \otimes \left(R_3^T M_3^{-1}\right)\right) b_{13} - b_2.$$

On the other hand, we have

$$B x_2 = \begin{pmatrix} \left(R_1 \otimes D_2 \otimes D_3\right) x_2 \\ \left(D_1 \otimes R_2 \otimes D_3\right) x_2 \\ \left(D_1 \otimes D_2 \otimes R_3\right) x_2 \end{pmatrix} \tag{6.26}$$

By introducing $x_1 := \left(x_{11}, x_{12}, x_{13}\right)^T$, the equation $A x_1 = b_1 - B x_2$ is equivalent to

$$\begin{cases} x_{11} = \left(M_1^{-1} \otimes D_2^{-1} \otimes D_3^{-1}\right) b_{11} - \left(M_1^{-1} \otimes D_2^{-1} \otimes D_3^{-1}\right) \left(R_1 \otimes D_2 \otimes D_3\right) x_2 \\ x_{12} = \left(D_1^{-1} \otimes M_2^{-1} \otimes D_3^{-1}\right) b_{12} - \left(D_1^{-1} \otimes M_2^{-1} \otimes D_3^{-1}\right) \left(D_1 \otimes R_2 \otimes D_3\right) x_2 \\ x_{13} = \left(D_1^{-1} \otimes D_2^{-1} \otimes M_3^{-1}\right) b_{13} - \left(D_1^{-1} \otimes D_2^{-1} \otimes M_3^{-1}\right) \left(D_1 \otimes D_2 \otimes R_3\right) x_2 \end{cases}$$

which can be simplified to

$$\begin{cases} x_{11} = \left(M_1^{-1} \otimes D_2^{-1} \otimes D_3^{-1}\right) b_{11} - \left(M_1^{-1} R_1 \otimes I_2 \otimes I_3\right) x_2 \\ x_{12} = \left(D_1^{-1} \otimes M_2^{-1} \otimes D_3^{-1}\right) b_{12} - \left(I_1 \otimes M_2^{-1} R_2 \otimes I_3\right) x_2 \\ x_{13} = \left(D_1^{-1} \otimes D_2^{-1} \otimes M_3^{-1}\right) b_{13} - \left(I_1 \otimes I_2 \otimes M_3^{-1} R_3\right) x_2 \end{cases}$$

Following the same steps, we consider the generalized eigendecompositions problems

(6.27)
$$\left(R_1^T M_1^{-1} R_1\right) U_1 = D_1 U_1 d_1, \quad \left(R_2^T M_2^{-1} R_2\right) U_2 = D_2 U_2 d_2, \quad \left(R_3^T M_3^{-1} R_3\right) U_3 = D_3 U_3 d_3$$

where d_1, d_2, and d_3 are diagonal matrices such that U_1, U_2 and U_2 fulfills

$$U_1^T D_1 U_1 = I_1, \quad U_2^T D_2 U_2 = I_2, \quad U_3^T D_3 U_3 = I_3 \tag{6.28}$$

Therefor

(6.29)
$$\left(U_1 \otimes U_2 \otimes U_3\right)^{-T} \left(d_1 \otimes I_2 \otimes I_3 + I_1 \otimes d_2 \otimes I_3 + I_1 \otimes I_2 \otimes d_3\right) \left(U_1 \otimes U_2 \otimes U_3\right)^{-1} x = b$$

The direct solver for (6.17), is then given by the following algorithm:

Algorithm 3: `fast_diag`: Fast diagonalization method for (6.17) in the 3D case

Input : b
Output: x

$\left(b_{11}, b_{12}, b_{13}\right), b_2 \leftarrow \texttt{unfold}(b)$
$r_2 \leftarrow \left(\left(R_1^T M_1^{-1}\right) \otimes I_2 \otimes I_3\right) b_{11} + \left(I_1 \otimes \left(R_2^T M_2^{-1}\right) \otimes I_3\right) b_{12} + \left(I_1 \otimes I_2 \otimes \left(R_3^T M_3^{-1}\right)\right) b_{13} - b_2$
$\tilde{r_2} \leftarrow \left(U_1 \otimes U_2 \otimes U_3\right)^T r_2$
$\tilde{x_2} \leftarrow \left(d_1 \otimes I_2 \otimes I_3 + I_1 \otimes d_2 \otimes I_3 + I_1 \otimes I_2 \otimes d_3\right)^{-1} \tilde{r_2}$
$x_2 \leftarrow \left(U_1 \otimes U_2 \otimes U_3\right) \tilde{x_2}$
$x_{11} \leftarrow \left(M_1^{-1} \otimes D_2^{-1} \otimes D_3^{-1}\right) b_{11} - \left(M_1^{-1} R_1 \otimes I_2 \otimes I_3\right) x_2$
$x_{12} \leftarrow \left(D_1^{-1} \otimes M_2^{-1} \otimes D_3^{-1}\right) b_{12} - \left(I_1 \otimes M_2^{-1} R_2 \otimes I_3\right) x_2$
$x_{13} \leftarrow \left(D_1^{-1} \otimes D_2^{-1} \otimes M_3^{-1}\right) b_{13} - \left(I_1 \otimes I_2 \otimes M_3^{-1} R_3\right) x_2$
$x \leftarrow \texttt{fold}(x_{11}, x_{12}, x_{13}, x_2)$

Remark 6.1. *For the steps 6-7 in Algorithm 2 and steps 6-7-8 in Algorithm 3, we use Algorithm 1 instead of inverse matrices.*

7. Numerical Results

To assess the efficiency of our method, the present section

- shows numerical convergence and CPU timing of our Monge-Ampère solver in 2D and 3D,
- compares CPU timing in 3D for the standard solver 4.1 and Mixed variational formulation solver 6.2,
- shows that we have solved the problem of errors at the boundary,
- shows examples of adaptive mesh generation using Mixed variational formulation in 2D, 2.5D and 3D.

7.1. **Monge-Ampère solver using mixed variational formulation.** We adopt the same test in both two and three dimension as in section 5.1. We also keep the L^2 residual error tolerance in the Picard iteration which is fixed at 10^{-10}.

Example 7.1 (Two-dimensional case)**.** *In the following Table 7 we remark that we obtain more accuracy in the calculation using the mixed formulation compared to the standard method Table 1.*

Table 7. CPU-time needed to solve the Monge-Ampere equation and error evolution for various polynomial degrees p along each direction.

(a) Degree $p = 3$

#cells	*H^1-norm*	*H^1-norm MG*	*CPU-time (s)*	*CPU-time (s) MG*	*Nbr-iter*	*Nbr-iter MG*
8	*5.232e-05*	*5.232e-05*	*0.029*	*0.036*	*41*	*32*
16	*6.508e-06*	*6.507e-06*	*0.061*	*0.067*	*41*	*35*
32	*8.155e-07*	*8.153e-07*	*0.178*	*0.200*	*35*	*32*
64	*1.018e-07*	*1.017e-07*	*0.602*	*0.625*	*35*	*31*
128	*1.279e-08*	*1.273e-08*	*2.434*	*2.343*	*35*	*30*

(b) Degree $p = 4$

#cells	*H^1-norm*	*H^1-norm MG*	*CPU-time (s)*	*CPU-time (s) MG*	*Nbr-iter*	*Nbr-iter MG*
8	*1.786e-06*	*1.786e-06*	*0.032*	*0.042*	*38*	*29*
16	*1.150e-07*	*1.150e-07*	*0.069*	*0.068*	*35*	*26*
32	*7.383e-09*	*7.369e-09*	*0.222*	*0.204*	*35*	*24*
64	*8.965e-10*	*9.467e-10*	*0.763*	*0.598*	*35*	*22*
128	*7.297e-10*	*8.337e-10*	*2.937*	*2.074*	*35*	*20*

(c) Degree $p = 5$

#cells	*H^1-norm*	*H^1-norm MG*	*CPU-time (s)*	*CPU-time (s) MG*	*Nbr-iter*	*Nbr-iter MG*
8	*8.527e-08*	*8.527e-08*	*0.049*	*0.047*	*35*	*26*
16	*3.057e-09*	*3.056e-09*	*0.085*	*0.085*	*35*	*26*
32	*7.407e-10*	*7.475e-10*	*0.281*	*0.254*	*35*	*24*
64	*7.293e-10*	*7.368e-10*	*0.983*	*0.757*	*35*	*22*
128	*7.292e-10*	*7.368e-10*	*3.846*	*2.641*	*35*	*20*

Example 7.2 (Three-dimensional case)**.** *The table 8 shows high accuracy and fast convergence in terms of the number of iterations performed using a multi-grid initial estimate. However, we show in the figure 10 CPU-time behavior for each method where we fix a tolerance of H^1-norm at 10^{-5}.*

TABLE 8. CPU-time needed to solve the Monge-Ampere equation and error evolution for various polynomial degrees p along each direction..

(A) Degree $p = 3$

#cells	*H^1-norm*	*H^1-norm MG*	*CPU-time (s)*	*CPU-time (s) MG*	*Nbr-iter*	*Nbr-iter MG*
8	*1.966e-05*	*1.966e-05*	*0.238*	*0.268*	*14*	*10*
16	*2.519e-06*	*2.519e-06*	*1.398*	*1.309*	*14*	*13*
32	*3.190e-07*	*3.190e-07*	*9.747*	*8.134*	*14*	*12*
64	*4.011e-08*	*4.016e-08*	*74.71*	*57.22*	*14*	*11*

(B) Degree $p = 4$

#cells	*H^1-norm*	*H^1-norm MG*	*CPU-time (s)*	*CPU-time (s) MG*	*Nbr-iter*	*Nbr-iter MG*
8	*6.622e-07*	*6.622e-07*	*0.521*	*0.583*	*14*	*10*
16	*4.322e-08*	*4.321e-08*	*3.686*	*2.702*	*14*	*10*
32	*2.750e-09*	*2.754e-09*	*27.55*	*15.50*	*14*	*8*
64	*1.965e-10*	*4.997e-10*	*214.5*	*90.37*	*14*	*6*

(C) Degree $p = 5$

#cells	*H^1-norm*	*H^1-norm MG*	*CPU-time (s)*	*CPU-time (s) MG*	*Nbr-iter*	*Nbr-iter MG*
8	*2.294e-08*	*2.294e-08*	*1.300*	*1.312*	*14*	*10*
16	*8.125e-10*	*8.118e-10*	*8.569*	*6.011*	*14*	*10*
32	*1.199e-10*	*1.202e-10*	*65.68*	*35.52*	*14*	*8*
64	*1.104e-10*	*1.112e-10*	*507.8*	*210.6*	*14*	*6*

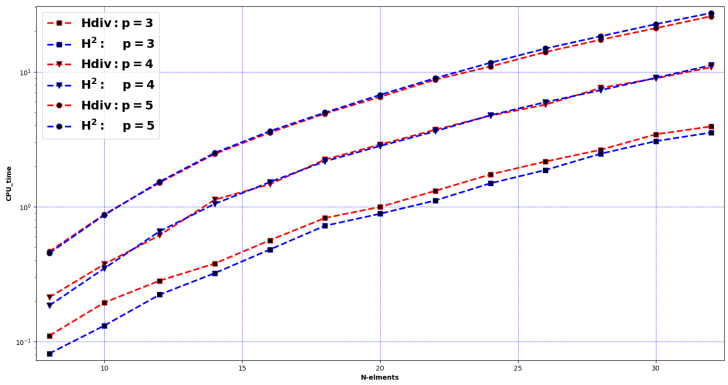

FIGURE 10. CPU time profiles between standard method ($\mathbf{H^2}$) and mixed variational formulation (**Hdiv**).

7.2. **Adaptive mesh generation.**

Examples on the unit square. As it is for the standard method, in order to compare the result, we keep the same tolerance and $\#cell = 32$ for all figures in the following examples.

Example 7.3. *We test with the same monitor function as in* [18] *defined as :*

$$\rho(\xi,\eta) = \frac{1}{2+\cos(8\pi\sqrt{(\xi-0.5)^2+(\eta-0.5)^2})}. \tag{7.1}$$

Fig. 11 shows the adaptive meshes results and Table 9 shows a high quality of mesh adaptation.

TABLE 9. Grid convergence analysis for various polynomial degrees p along each direction.

(A) Degree $p = 3$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*4.949e-02*	*0.086*	*1.905e-02*	*5.017e-01*	*1.769e+00*
32	*4.665e-03*	*0.142*	*1.902e-02*	*5.681e-01*	*1.722e+00*
64	*5.005e-04*	*0.430*	*1.902e-02*	*5.697e-01*	*1.713e+00*
128	*1.128e-04*	*1.516*	*1.902e-02*	*5.699e-01*	*1.710e+00*

(B) Degree $p = 4$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*3.221e-02*	*0.101*	*1.903e-02*	*4.997e-01*	*1.748e+00*
32	*2.744e-03*	*0.174*	*1.902e-02*	*5.679e-01*	*1.718e+00*
64	*8.635e-05*	*0.534*	*1.902e-02*	*5.699e-01*	*1.710e+00*
128	*2.102e-05*	*1.879*	*1.902e-02*	*5.699e-01*	*1.710e+00*

(C) Degree $p = 5$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*4.077e-02*	*0.161*	*1.903e-02*	*5.005e-01*	*1.721e+00*
32	*2.289e-03*	*0.206*	*1.902e-02*	*5.673e-01*	*1.713e+00*
64	*3.299e-05*	*0.650*	*1.902e-02*	*5.699e-01*	*1.710e+00*
128	*3.737e-05*	*2.168*	*1.902e-02*	*5.699e-01*	*1.710e+00*

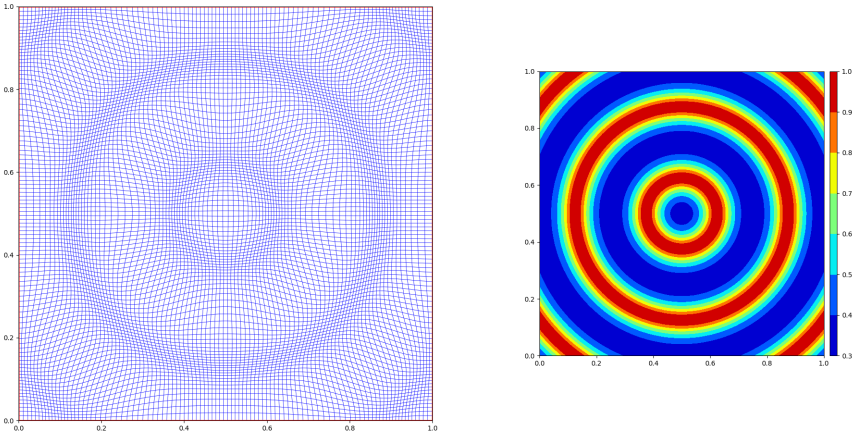

FIGURE 11. (left) Adaptive meshes and (right) the monitor function.

Example 7.4. *In order to show that we have solved the problem of boundary errors and negative values, we adopt the same example 5.4 used in the standard method. As expected, the errors at the boundary are eliminated Fig. 12 with high accuracy validated by Table 10.*

TABLE 10. Grid convergence analysis for various polynomial degrees p along each direction.

(A) Degree $p = 3$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*4.165e-01*	*0.196*	*1.428e-01*	*1.077e-01*	*2.950e+00*
32	*1.231e-01*	*0.600*	*1.111e-01*	*3.123e-02*	*3.413e+00*
64	*3.020e-02*	*1.279*	*1.082e-01*	*2.209e-01*	*3.323e+00*
128	*4.589e-03*	*3.782*	*1.081e-01*	*3.509e-01*	*3.132e+00*

(B) Degree $p = 4$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*4.249e-01*	*0.216*	*1.435e-01*	*-3.402e-03*	*2.963e+00*
32	*1.182e-01*	*0.620*	*1.101e-01*	*8.919e-02*	*3.567e+00*
64	*2.747e-02*	*1.539*	*1.081e-01*	*2.483e-01*	*3.324e+00*
128	*3.411e-03*	*4.250*	*1.081e-01*	*3.513e-01*	*3.134e+00*

(C) Degree $p = 5$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*3.764e-01*	*0.344*	*1.407e-01*	*-6.914e-02*	*3.361e+00*
32	*1.065e-01*	*0.764*	*1.092e-01*	*8.918e-02*	*3.269e+00*
64	*2.213e-02*	*1.832*	*1.081e-01*	*2.296e-01*	*3.308e+00*
128	*3.138e-03*	*5.358*	*1.081e-01*	*3.511e-01*	*3.137e+00*

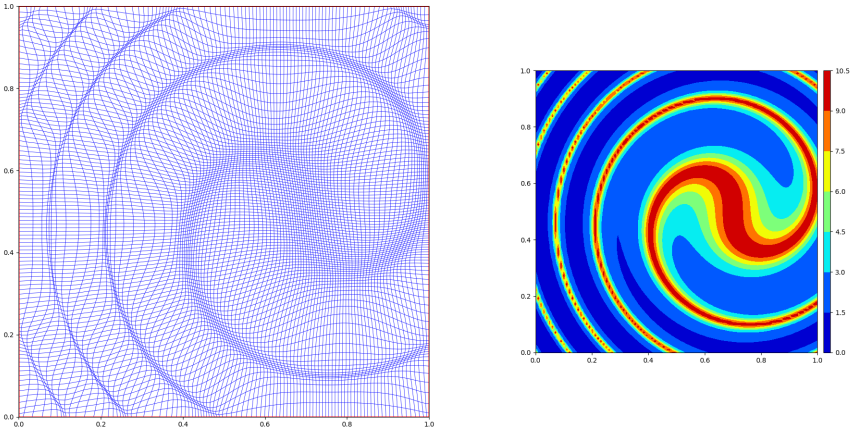

FIGURE 12. (left) Adaptive meshes and (right) the monitor function.

Example 7.5 (Examples of adapted mesh on more general geometries (circle))**.** *We add to the example 5.5 another circle touching the boundary like an unhappy mouth for Mickey Mouse. A good performance in the computation is shown in table 11, with no self-intersection Figs. 13-14.*

$$\rho(x,y) = 1. + 5\exp(-100|(x-0.45)^2 + (y-0.4)^2 - 0.1|) + 5\exp(-100|x^2+y^2-0.2|) + 5\exp(-100|(x+0.45)^2 + (y-0.4)^2 - 0.1|) + 7\exp(-100|x^2 + (y+1.25)^2 - 0.4|). \tag{7.2}$$

TABLE 11. Grid convergence analysis for various polynomial degrees p along each direction.

(A) Degree $p = 3$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*3.258e-01*	*0.119*	*4.369e-02*	*1.831e-01*	*1.574e+00*
32	*8.837e-02*	*0.373*	*4.479e-02*	*4.882e-02*	*1.508e+00*
64	*2.684e-02*	*0.785*	*4.392e-02*	*1.274e-01*	*1.408e+00*
128	*5.374e-03*	*2.342*	*4.389e-02*	*1.358e-01*	*1.333e+00*

(B) Degree $p = 4$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*3.325e-01*	*0.142*	*4.553e-02*	*1.556e-01*	*1.658e+00*
32	*7.359e-02*	*0.419*	*4.440e-02*	*9.047e-02*	*1.504e+00*
64	*2.246e-02*	*0.930*	*4.390e-02*	*1.420e-01*	*1.430e+00*
128	*5.135e-03*	*2.777*	*4.389e-02*	*1.362e-01*	*1.338e+00*

(C) Degree $p = 5$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*2.955e-01*	*0.176*	*4.407e-02*	*1.692e-01*	*1.613e+00*
32	*7.086e-02*	*0.610*	*4.437e-02*	*7.912e-02*	*1.492e+00*
64	*2.491e-02*	*1.108*	*4.392e-02*	*1.333e-01*	*1.405e+00*
128	*4.987e-03*	*3.461*	*4.389e-02*	*1.359e-01*	*1.341e+00*

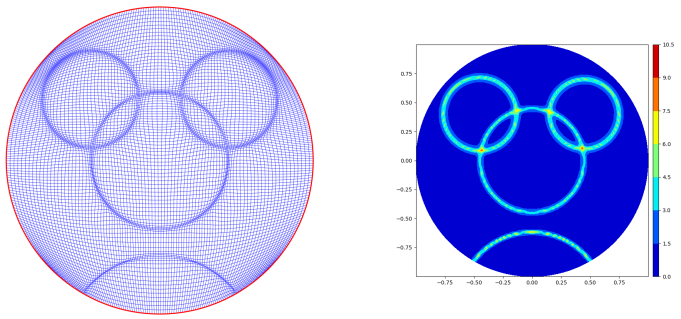

FIGURE 13. (left) Adaptive meshes and (right) the monitor function.

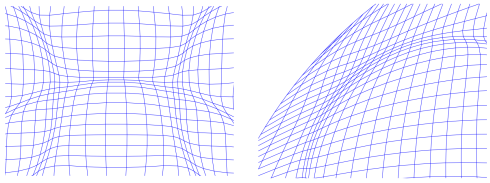

FIGURE 14. A close-up of the region where the two circles intersect (left) and on one of the singular points (right).

Example 7.6 (Circle)**.** *With no-errors at the boundary we adopt the example of non-axisymmetric monitor function used in 5.6 Fig. 15. On other hand, Table 12 shows fast convergence, high accuracy and positive Jacobian function.*

TABLE 12. Grid convergence analysis for various polynomial degrees p along each direction.

(A) Degree $p = 3$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*5.900e-02*	*0.114*	*9.608e-02*	*1.665e-01*	*1.982e+00*
32	*1.096e-02*	*0.226*	*9.611e-02*	*1.702e-01*	*1.893e+00*
64	*2.048e-03*	*0.671*	*9.611e-02*	*1.699e-01*	*1.867e+00*
128	*1.819e-03*	*2.490*	*9.611e-02*	*1.699e-01*	*1.867e+00*

(B) Degree $p = 4$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*4.695e-02*	*0.138*	*9.629e-02*	*1.676e-01*	*1.955e+00*
32	*8.832e-03*	*0.265*	*9.611e-02*	*1.703e-01*	*1.882e+00*
64	*1.922e-03*	*0.774*	*9.611e-02*	*1.699e-01*	*1.867e+00*
128	*1.816e-03*	*2.819*	*9.611e-02*	*1.699e-01*	*1.867e+00*

(C) Degree $p = 5$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*4.790e-02*	*0.204*	*9.617e-02*	*1.662e-01*	*1.942e+00*
32	*9.055e-03*	*0.366*	*9.611e-02*	*1.696e-01*	*1.882e+00*
64	*1.867e-03*	*1.057*	*9.611e-02*	*1.699e-01*	*1.867e+00*
128	*1.815e-03*	*3.612*	*9.611e-02*	*1.699e-01*	*1.867e+00*

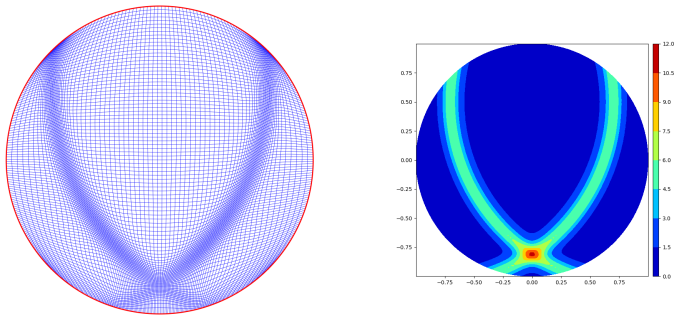

FIGURE 15. (left) Adaptive meshes and (right) the monitor function.

Example 7.7 (Examples of adapted mesh on more general geometries (quart-annulus))**.** *In this test, we provide the quality of mesh adaptation in the quart-annulus domain as shown in the Tab. 13. However, there is no distortion in the mesh using a density function as two sin-waves, illustrated in Figs. 16-17.*

$$\rho(x,y) = 1 + 5\exp\Big(-50|y - 0.5 - 0.25\sin(2\pi x)\sin(\frac{2\pi}{3})|\Big) + 5\exp\Big(-50|y - 0.25 - 0.25\sin(2\pi x)|\Big). \tag{7.3}$$

TABLE 13. Grid convergence analysis for various polynomial degrees p along each direction.

(A) Degree $p = 3$

#*cells*	*Err*	*CPU-time (s)*	*Qual*	min *Jac*($\mathbf{\Psi}$)	max *Jac*($\mathbf{\Psi}$)
16	*6.810e-02*	*0.117*	*7.786e-02*	*1.880e-01*	*1.795e+00*
32	*1.765e-02*	*0.230*	*7.752e-02*	*2.531e-01*	*1.678e+00*
64	*4.754e-03*	*0.682*	*7.754e-02*	*2.740e-01*	*1.614e+00*
128	*1.763e-03*	*2.397*	*7.754e-02*	*2.715e-01*	*1.614e+00*

(B) Degree $p = 4$

#*cells*	*Err*	*CPU-time (s)*	*Qual*	min *Jac*($\mathbf{\Psi}$)	max *Jac*($\mathbf{\Psi}$)
16	*5.320e-02*	*0.145*	*7.794e-02*	*1.696e-01*	*1.702e+00*
32	*1.721e-02*	*0.278*	*7.751e-02*	*2.570e-01*	*1.660e+00*
64	*4.538e-03*	*0.821*	*7.754e-02*	*2.739e-01*	*1.616e+00*
128	*1.707e-03*	*2.806*	*7.754e-02*	*2.718e-01*	*1.614e+00*

(C) Degree $p = 5$

#*cells*	*Err*	*CPU-time (s)*	*Qual*	min *Jac*($\mathbf{\Psi}$)	max *Jac*($\mathbf{\Psi}$)
16	*5.087e-02*	*0.174*	*7.779e-02*	*1.737e-01*	*1.784e+00*
32	*1.612e-02*	*0.367*	*7.751e-02*	*2.660e-01*	*1.670e+00*
64	*4.386e-03*	*1.027*	*7.754e-02*	*2.739e-01*	*1.614e+00*
128	*1.678e-03*	*3.737*	*7.754e-02*	*2.716e-01*	*1.614e+00*

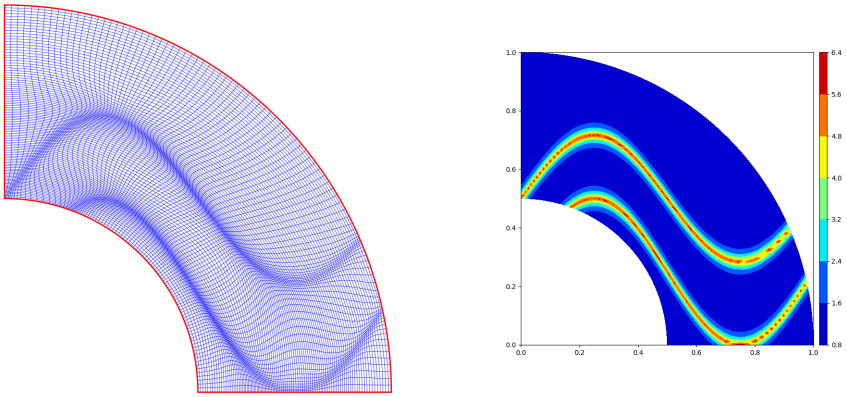

FIGURE 16. (left) Adaptive meshes and (right) the monitor function.

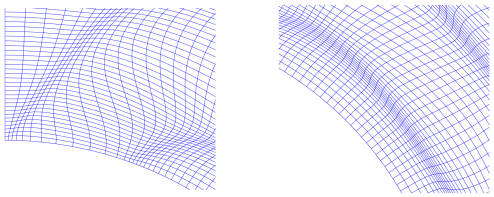

FIGURE 17. A close-up of the region where the mesh is very fine.

Examples of two and half-dimensional adaptive meshes. Solving the Monge-Ampere equation directly in the torus, Möbius, and other 2.5-dimensional surfaces is complicated because of convexity and other assumptions necessary to ensure convergence. However, in what follows, we give a starting point for the moving mesh method into this complicated geometry.

Example 7.8 (Empty-Torus subdomain)**.** *In this example, we present a mesh adaptation in an extracted part of the torus domain. The geometry is represented in a three-dimensional space as shown in Fig. 18, but thinks to our new technique 1, the mesh adaptation is done in the square with a high quality of mesh adaption Table 14. Where the monitor function is defined in the torus by :*

$$\rho(\xi,\eta,\gamma) = \frac{1}{2+\cos(8\pi\sqrt{(x-0.5)^2+(y-0.5)^2+(z-0.5)^2})}. \tag{7.4}$$

TABLE 14. Grid convergence analysis for various polynomial degrees p along each direction.

(A) Degree $p = 3$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*3.626e-01*	*0.104*	*7.822e-03*	*3.264e-01*	*1.652e+00*
32	*2.698e-01*	*0.275*	*8.649e-03*	*3.806e-01*	*1.702e+00*
64	*7.026e-02*	*0.845*	*6.952e-03*	*4.616e-01*	*1.780e+00*
128	*8.173e-03*	*2.870*	*6.952e-03*	*5.763e-01*	*1.759e+00*

(B) Degree $p = 4$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*3.560e-01*	*0.133*	*8.141e-03*	*3.137e-01*	*1.975e+00*
32	*2.700e-01*	*0.423*	*8.663e-03*	*3.650e-01*	*1.786e+00*
64	*7.102e-02*	*1.160*	*6.953e-03*	*4.550e-01*	*1.755e+00*
128	*6.492e-03*	*3.619*	*6.952e-03*	*5.712e-01*	*1.749e+00*

(C) Degree $p = 5$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*3.599e-01*	*0.159*	*7.748e-03*	*2.696e-01*	*1.733e+00*
32	*2.713e-01*	*0.534*	*8.511e-03*	*3.011e-01*	*1.791e+00*
64	*7.167e-02*	*1.480*	*6.953e-03*	*4.484e-01*	*1.754e+00*
128	*5.981e-03*	*4.582*	*6.952e-03*	*5.696e-01*	*1.742e+00*

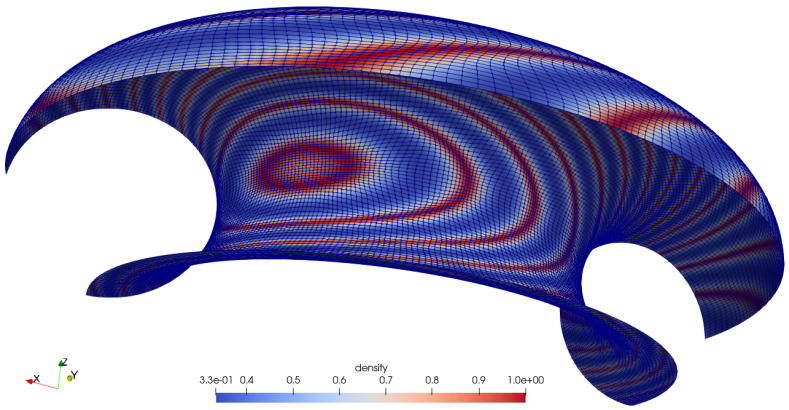

FIGURE 18. Adaptive meshes and the monitor function.

Example 7.9 (Non-Orientable Möbius Surface)**.** *From the application viewpoint Möbius strip is well-known as a tool for building more efficient mechanical belts. Hence, we defend a mesh adaptation in a non-orientable Möbius surface represented in a three-dimensional space Fig. 7.9. Table 15 shows that accurate and bijective mapping is always achieved using various regularities.. Therefore, here we use the exact form of non-axisymmetric function used in* [46] *into the Möbius strip as*

follows.

$$\rho(x,y) = 1 + 5/sech\Big(5\big((x-\frac{\sqrt{3}}{2})^2 + (y-0.5)^2 + (z-0.5)^2 - \frac{\pi^2}{4}\big)\Big) + 5/sech\Big(5\big((x+\frac{\sqrt{3}}{2})^2 + (y-0.5)^2 + (z-0.5)^2 - \frac{\pi^2}{4}\big)\Big). \tag{7.5}$$

TABLE 15. Grid convergence analysis for various polynomial degrees p along each direction.

(A) Degree $p = 3$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*2.202e-01*	*0.236*	*9.284e-02*	*4.959e-02*	*2.023e+00*
32	*7.748e-02*	*0.332*	*8.802e-02*	*1.635e-01*	*1.944e+00*
64	*1.367e-02*	*0.915*	*8.745e-02*	*1.627e-01*	*1.853e+00*
128	*1.149e-03*	*3.073*	*8.743e-02*	*1.626e-01*	*1.790e+00*

(B) Degree $p = 4$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*2.173e-01*	*0.175*	*9.307e-02*	*7.994e-02*	*2.169e+00*
32	*8.067e-02*	*0.420*	*8.812e-02*	*1.576e-01*	*1.954e+00*
64	*1.486e-02*	*1.181*	*8.745e-02*	*1.627e-01*	*1.856e+00*
128	*1.057e-03*	*3.552*	*8.743e-02*	*1.626e-01*	*1.789e+00*

(C) Degree $p = 5$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*2.082e-01*	*0.268*	*9.200e-02*	*2.356e-02*	*2.060e+00*
32	*7.130e-02*	*0.498*	*8.794e-02*	*1.599e-01*	*1.949e+00*
64	*1.312e-02*	*1.517*	*8.745e-02*	*1.628e-01*	*1.841e+00*
128	*8.070e-04*	*4.622*	*8.743e-02*	*1.626e-01*	*1.792e+00*

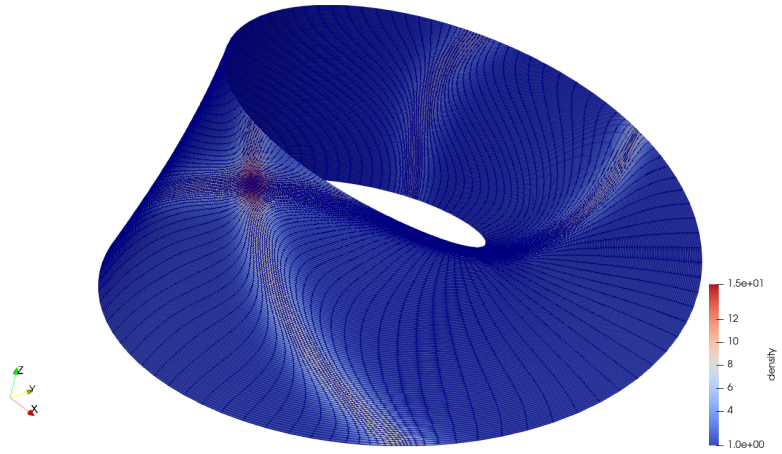

FIGURE 19. Adaptive meshes and the monitor function.

Examples of three-dimensional adaptive meshes. In this subsection, we shall give some examples of three-dimensional adaptive meshes using different analytical density functions defined in the physical domain. However, To show the quality of meshes, we use the last formula 5.6. The tolerance is set to 10^{-8} throughout the coming tests.

Example 7.10. *We use the generalized form of the monitor function in 7.3 defined as :*

$$\rho(\xi,\eta,\gamma) = \frac{1}{2+\cos(8\pi\sqrt{(\xi-0.5)^2+(\eta-0.5)^2+(\gamma-0.5)^2})}. \tag{7.6}$$

We can see a fast convergence and the accuracy of B-spline mapping in Table 16. Fig. 20 shows high quality of mesh adaptation.

TABLE 16. Grid convergence analysis for various polynomial degrees p along each direction.

(A) Degree $p = 3$

#cells	*Err*	*CPU-time (s)*	*Qual*	min *Jac*($\mathbf{\Psi}$)	max *Jac*($\mathbf{\Psi}$)
16	*5.303e-03*	*1.503*	*4.925e-03*	*4.922e-01*	*1.791e+00*
32	*3.662e-04*	*8.520*	*3.329e-03*	*5.718e-01*	*1.746e+00*
64	*2.451e-05*	*55.52*	*2.322e-03*	*5.733e-01*	*1.725e+00*

(B) Degree $p = 4$

#cells	*Err*	*CPU-time (s)*	*Qual*	min *Jac*($\mathbf{\Psi}$)	max *Jac*($\mathbf{\Psi}$)
16	*3.763e-03*	*3.769*	*4.922e-03*	*5.166e-01*	*1.779e+00*
32	*1.934e-04*	*21.40*	*3.330e-03*	*5.720e-01*	*1.732e+00*
64	*4.271e-06*	*129.6*	*2.322e-03*	*5.736e-01*	*1.721e+00*

(C) Degree $p = 5$

#cells	*Err*	*CPU-time (s)*	*Qual*	min *Jac*($\mathbf{\Psi}$)	max *Jac*($\mathbf{\Psi}$)
16	*2.780e-03*	*9.072*	*4.917e-03*	*5.229e-01*	*1.763e+00*
32	*1.252e-04*	*45.97*	*3.330e-03*	*5.720e-01*	*1.725e+00*
64	*3.368e-06*	*270.2*	*2.322e-03*	*5.736e-01*	*1.721e+00*

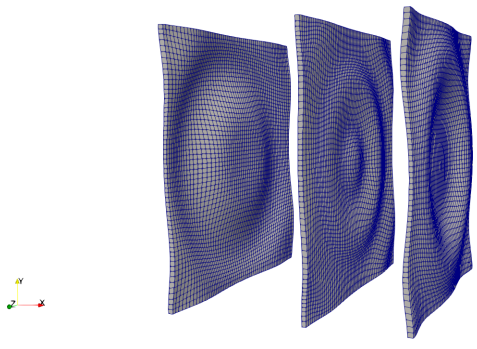

FIGURE 20. Location of the planes in the cube corresponding to $x = 0.1,\ 0.6,\ 0.9$.

Example 7.11 (Examples of adapted mesh on more general geometries: sphere)**.** *We assess the quality of mesh adaptation in the sphere case, see Fig. 21 and no mesh-distortion with high quality as it is demonstrated in Tab. 17 using a density function as cos-waves defined by*

$$\rho(x,y) = 3/\Big(2+\cos\big(8\pi\sqrt{(x-0.5)^2+(y-0.5)^2+(z-0.5)^2}\big)\Big). \tag{7.7}$$

TABLE 17. Grid convergence analysis for various polynomial degrees p along each direction.

(A) Degree $p=3$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*2.148e-02*	*1.716*	*3.108e-03*	*3.825e-01*	*1.920e+00*
32	*2.005e-03*	*11.26*	*2.198e-03*	*5.141e-01*	*1.796e+00*
64	*1.195e-04*	*74.83*	*1.533e-03*	*5.725e-01*	*1.748e+00*

(B) Degree $p=4$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*2.030e-02*	*4.208*	*3.116e-03*	*3.668e-01*	*2.139e+00*
32	*1.580e-03*	*29.18*	*2.198e-03*	*5.167e-01*	*1.786e+00*
64	*4.936e-05*	*166.7*	*1.533e-03*	*5.731e-01*	*1.732e+00*

(C) Degree $p=5$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*1.969e-02*	*9.441*	*3.115e-03*	*3.566e-01*	*1.919e+00*
32	*1.315e-03*	*58.24*	*2.198e-03*	*5.306e-01*	*1.760e+00*
64	*3.708e-05*	*349.7*	*1.533e-03*	*5.724e-01*	*1.729e+00*

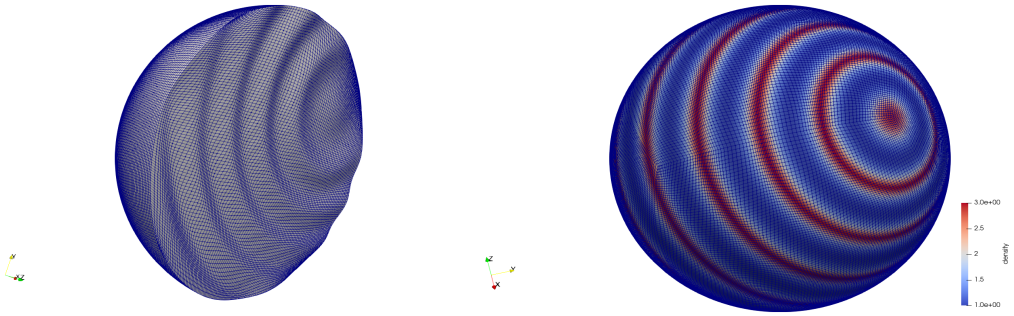

FIGURE 21. (left) A close-up of one extract part inside the sphere (right) Adaptive meshes and the monitor function.

Example 7.12 (Examples of adapted mesh on more general geometries: cylinder)**.** *Using the same density function as in Example 7.7, we show numerical results for the case of the cylinder that has four singular points on each side of the boundary. The adaptive meshes in Fig. 22 (left) and a close-up Fig.22 (right) illustrates the quality of mesh adaptation. However, even if the error does not change for different degrees, the regularity may be mandatory requirement by the end user, so we can guarantee that bijectivity is preserved with a high-quality of mesh adaptation, see Tab. 18*

TABLE 18. Grid convergence analysis for various polynomial degrees p along each direction.

(A) Degree $p = 3$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*3.196e-02*	*2.083*	*2.769e-03*	*3.252e-01*	*1.845e+00*
32	*5.415e-03*	*10.90*	*1.891e-03*	*4.868e-01*	*1.818e+00*
64	*3.313e-04*	*61.59*	*1.313e-03*	*5.751e-01*	*1.760e+00*

(B) Degree $p = 4$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*2.847e-02*	*4.734*	*2.753e-03*	*3.562e-01*	*1.863e+00*
32	*4.518e-03*	*28.79*	*1.890e-03*	*5.060e-01*	*1.785e+00*
64	*1.945e-04*	*176.1*	*1.313e-03*	*5.747e-01*	*1.741e+00*

(C) Degree $p = 5$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*2.599e-02*	*11.62*	*2.780e-03*	*3.034e-01*	*1.894e+00*
32	*4.037e-03*	*70.16*	*1.890e-03*	*4.848e-01*	*1.774e+00*
64	*1.518e-04*	*359.5*	*1.313e-03*	*5.738e-01*	*1.737e+00*

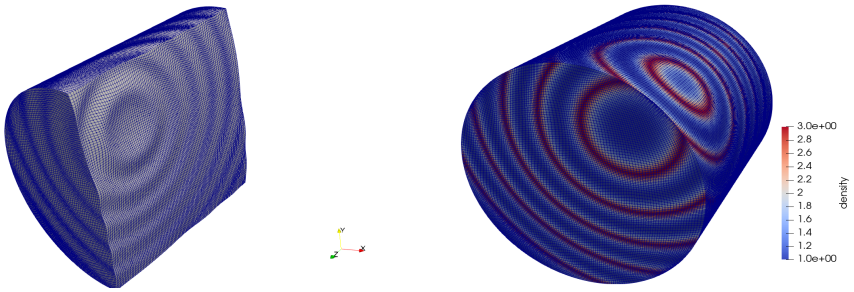

FIGURE 22. (left) A close-up of one extract part inside the sphere (right) Adaptive meshes and the monitor function.

Example 7.13 (Examples of adapted mesh on more general geometries: half–torus)**.** *A torus domain is one of the popular geometries used in many papers for the three-dimensional simulation of plasma physic problems. Therefore, we study this geometry by adding mesh adaptation. However, we use the same density function 7.7 presented in Fig. 23 (right) and to see what happens inside the domain, see Fg. 23 (left), and for the different degrees we validate the bijectivity and quality with fast convergence results in Table 19.*

TABLE 19. Grid convergence analysis for various polynomial degrees p along each direction.

(A) Degree $p = 3$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*4.801e-02*	*2.637*	*2.029e-03*	*3.284e-01*	*1.893e+00*
32	*7.936e-03*	*12.84*	*1.338e-03*	*4.300e-01*	*1.829e+00*
64	*4.406e-04*	*76.00*	*9.252e-04*	*5.721e-01*	*1.774e+00*

(B) Degree $p = 4$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*4.680e-02*	*7.165*	*2.095e-03*	*2.872e-01*	*2.030e+00*
32	*7.310e-03*	*34.75*	*1.339e-03*	*4.481e-01*	*1.841e+00*
64	*2.774e-04*	*214.9*	*9.252e-04*	*5.705e-01*	*1.755e+00*

(C) Degree $p = 5$

#cells	*Err*	*CPU-time (s)*	*Qual*	min $Jac(\mathbf{\Psi})$	max $Jac(\mathbf{\Psi})$
16	*4.321e-02*	*15.52*	*2.097e-03*	*2.005e-01*	*2.032e+00*
32	*6.685e-03*	*76.55*	*1.338e-03*	*4.481e-01*	*1.799e+00*
64	*2.335e-04*	*454.7*	*9.252e-04*	*5.699e-01*	*1.743e+00*

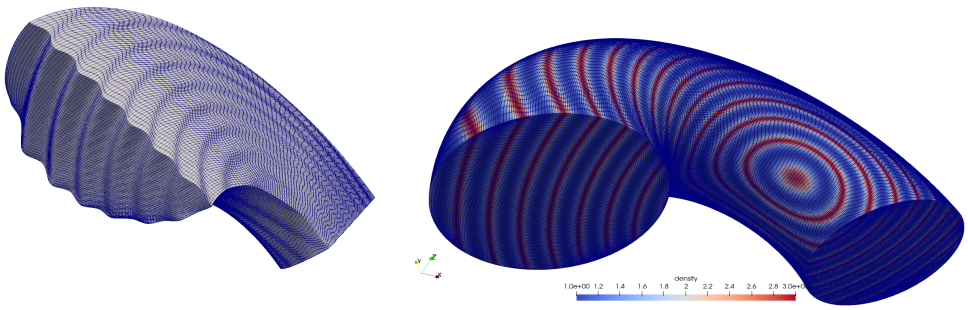

FIGURE 23. (left) A close-up of one extract part inside the sphere (right) Adaptive meshes and the monitor function.

8. Conclusions

We have presented the original technique of the adaptive meshes, and it works efficiently. The L^2 Monge-Kantorovich problem gave us a good strategy of mesh adaptation in the Isogeometric Analysis; solving the Monge-Ampere equation by the BFO method that is robust and reliable as shown in section 6, our method constructs a global and regular one-to-one mapping that maps the unit square into the physical domain, while ensuring the equidistribution property. Therefore, the bijectivity from the unit square to the physical domain is assured when we use an initial bijective mapping.

We started our study with a standard method which solves the BFO system directly by means of the Galerkin variational formulation, a fast convergence is always obtained using the fast diagonalization method Algorithm 1. However, the accuracy is provided for some examples, but unfortunately, it generates boundary errors due to the weak integration of the Neumann boundary conditions, which may lead to the loss of the geometry. We fixed this problem by introducing a mixed variational formulation, and the boundary errors are fully resolved. Moreover, for fast convergence, two new fast Alg. 2 and Alg. 3 based on the fast diagonalization method are proposed, and we obtain a more accurate bijective mapping, as shown in the tables for various B-spline regularities and geometric complexities in two- and three-dimensional cases.

The performance and stability of the technique of adaptation are demonstrated by several examples. The algorithm is simple to implement, accelerated by the fast diagonalization method, and has no free parameters. Moreover, it works for all geometry that we have tested.

Current limitations and future work.

Acknowledgments. We warmly thank G. Latu, G. Dif-Pradalier, V. Grandgirard (CEA-Cadarache), B. Nkonga (University of Nice), G. Huysmans (Iter Organization), B. Mourain (INRIA Sophi-Antipolis) and A. Buffa (EPFL), for useful discussions and advices. This project started during the ANEMOS-ANR and EFDA-fellowship and IPAM long program *Computational Methods in High Energy Density Plasmas* in 2012.

References

1. G Awanou, *Pseudo transient continuation and time marching methods for monge-ampère type equations*, Submitted.
2. ———, *Spline element method for the monge-ampère equation*, Submitted.
3. M. J. Baines, *Least squares and approximate equidistribution in multidimensions*, Numerical Methods for Partial Differential Equations **15** (1999), no. 5, 605–615.
4. Jean-David Benamou, Brittany D. Froese, and Adam M. Oberman, *Two numerical methods for the elliptic monge-ampère equation*, ESAIM: Mathematical Modelling and Numerical Analysis **44** (2010), 737–758.
5. Jean-David Benamou, Brittany D Froese, and Adam M Oberman, *Two numerical methods for the elliptic monge-ampere equation*, ESAIM: Mathematical Modelling and Numerical Analysis **44** (2010), no. 4, 737–758.
6. Yann Brenier, *Polar factorization and monotone rearrangement of vector-valued functions*, Communications on pure and applied mathematics **44** (1991), no. 4, 375–417.
7. ———, *Polar factorization and monotone rearrangement of vector-valued functions*, Communications on Pure and Applied Mathematics **44** (1991), no. 4, 375–417.
8. C J Budd and J F Williams, *Parabolic monge¿ampère methods for blow-up problems in several spatial dimensions*, Journal of Physics A: Mathematical and General **39** (2006), no. 19, 5425.
9. Chris J. Budd, Weizhang Huang, and Robert D. Russell, *Adaptivity with moving grids*, Acta Numerica **18** (2009), 111–241. MR 2506041 (2010c:65165)
10. C.J. Budd, M.J.P. Cullen, and E.J. Walsh, *Monge-ampère based moving mesh methods for numerical weather prediction, with applications to the eady problem*, Journal of Computational Physics **236** (2013), no. 0, 247 – 270.
11. Weiming Cao, Weizhang Huang, and Robert D. Russell, *Approaches for generating moving adaptive meshes: location versus velocity*, Applied Numerical Mathematics **47** (2003), no. 2, 121 – 138, Second International Workshop on Numerical Linear Algebra - Numerical Methods for Partial Differential Equations and Optimization.
12. L. Chacòn, G.L. Delzanno, and J.M. Finn, *Robust, multidimensional mesh-motion based on monge-kantorovich equidistribution*, Journal of Computational Physics **230** (2011), no. 1, 87 – 103.
13. Jean-François Cossette and Piotr K. Smolarkiewicz, *A monge-ampère enhancement for semi-lagrangian methods*, Computers & Fluids **46** (2011), no. 1, 180 – 185, 10th ICFD Conference Series on Numerical Methods for Fluid Dynamics (ICFD 2010).
14. J Austin Cottrell, Thomas JR Hughes, and Yuri Bazilevs, *Isogeometric analysis: toward integration of cad and fea*, John Wiley & Sons, 2009.
15. Edward J Dean and Roland Glowinski, *Numerical solution of the two-dimensional elliptic monge-ampère equation with dirichlet boundary conditions: an augmented lagrangian approach*, Comptes Rendus Mathematique **336** (2003), no. 9, 779 – 784.
16. Edward J. Dean and Roland Glowinski, *Numerical solution of the two-dimensional elliptic monge-ampère equation with dirichlet boundary conditions: a least-squares approach*, Comptes Rendus Mathematique **339** (2004), no. 12, 887 – 892.
17. E.J. Dean and R. Glowinski, *Numerical methods for fully nonlinear elliptic equations of the monge-ampère type*, Computer Methods in Applied Mechanics and Engineering **195** (2006), no. 13-16, 1344 – 1386, A Tribute to Thomas J.R. Hughes on the Occasion of his 60th Birthday.
18. Gian Luca Delzanno, Luis Chacón, John M Finn, Y Chung, and Giovanni Lapenta, *An optimal robust equidistribution method for two-dimensional grid adaptation based on monge–kantorovich optimization*, Journal of Computational Physics **227** (2008), no. 23, 9841–9864.
19. Gian Luca Delzanno and John M. Finn, *The fluid dynamic approach to equidistribution methods for grid adaptation*, Computer Physics Communications **182** (2011), no. 2, 330 – 346.
20. G.L. Delzanno, L. Chacòn, J.M. Finn, Y. Chung, and G. Lapenta, *An optimal robust equidistribution method for two-dimensional grid adaptation based on monge-kantorovich optimization*, Journal of Computational Physics **227** (2008), no. 23, 9841 – 9864.
21. Brittany Froese Hamfeldt and Axel GR Turnquist, *A convergence framework for optimal transport on the sphere*, Numerische Mathematik (2022), 1–31.
22. Weizhang Huang, *Practical aspects of formulation and solution of moving mesh partial differential equations*, Journal of Computational Physics **171** (2001), no. 2, 753 – 775.

23. ———, *Variational mesh adaptation: Isotropy and equidistribution*, Journal of Computational Physics **174** (2001), no. 2, 903 – 924.
24. ———, *Metric tensors for anisotropic mesh generation*, Journal of Computational Physics **204** (2005), no. 2, 633 – 665.
25. Weizhang Huang, Lennard Kamenski, and Jens Lang, *A new anisotropic mesh adaptation method based upon hierarchical a posteriori error estimates*, Journal of Computational Physics **229** (2010), no. 6, 2179 – 2198.
26. Weizhang Huang and Xianping Li, *An anisotropic mesh adaptation method for the finite element solution of variational problems*, Finite Elements in Analysis and Design **46** (2010), no. 1-2, 61 – 73, Mesh Generation - Applications and Adaptation.
27. Weizhang Huang, Jingtang Ma, and Robert D. Russell, *A study of moving mesh pde methods for numerical simulation of blowup in reaction diffusion equations*, Journal of Computational Physics **227** (2008), no. 13, 6532 – 6552.
28. Weizhang Huang, Yuhe Ren, and Robert D. Russell, *Moving mesh methods based on moving mesh partial differential equations*, Journal of Computational Physics **113** (1994), no. 2, 279 – 290.
29. Weizhang Huang and Robert D. Russell, *A moving collocation method for solving time dependent partial differential equations*, Applied Numerical Mathematics **20** (1996), no. 1-2, 101 – 116, Method of Lines for Time-Dependent Problems.
30. ———, *A high dimensional moving mesh strategy*, Applied Numerical Mathematics **26** (1998), no. 1-2, 63 – 76.
31. ———, *Adaptive mesh movement - the mmpde approach and its applications*, Partial Differential Equations (D. Sloan, S. Vandewalle, and E. Süli, eds.), Elsevier, Amsterdam, 2001, pp. 383 – 398.
32. Weizhang Huang and Robert D Russell, *Adaptive moving mesh methods*, vol. 174, Springer Science & Business Media, 2010.
33. Weizhang Huang and Robert D. Russell, *Adaptive moving mesh methods*, Applied mathematical sciences, Springer, New York, Heidelberg, London, 2011.
34. Weizhang Huang and Forrest Schaeffer, *An stability analysis for the finite-difference solution of one-dimensional linear convection-diffusion equations on moving meshes*, Journal of Computational and Applied Mathematics **236** (2012), no. 13, 3338 – 3348.
35. Weizhang Huang and Weiwei Sun, *Variational mesh adaptation ii: error estimates and monitor functions*, Journal of Computational Physics **184** (2003), no. 2, 619 – 648.
36. Thomas JR Hughes, John A Cottrell, and Yuri Bazilevs, *Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement*, Computer methods in applied mechanics and engineering **194** (2005), no. 39-41, 4135–4195.
37. T.J.R. Hughes, J.A. Cottrell, and Y. Bazilevs, *Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement*, Computer Methods in Applied Mechanics and Engineering **194** (2005), no. 39-41, 4135 – 4195.
38. Milad Khosravi and Mitra Javan, *Three-dimensional features of the lateral thermal plume discharge in the deep cross-flow using dynamic adaptive mesh refinement*, Theoretical and Computational Fluid Dynamics (2022), 1–18.
39. Jens Lang, Weiming Cao, Weizhang Huang, and Robert D. Russell, *A two-dimensional moving finite element method with local refinement based on a posteriori error estimates*, Applied Numerical Mathematics **46** (2003), no. 1, 75 – 94.
40. Giovanni Lapenta, *Variational grid adaptation based on the minimization of local truncation error: time-independent problems*, Journal of Computational Physics **193** (2004), no. 1, 159 – 179.
41. Xianping Li and Weizhang Huang, *An anisotropic mesh adaptation method for the finite element solution of heterogeneous anisotropic diffusion problems*, Journal of Computational Physics **229** (2010), no. 21, 8072 – 8094.
42. P-L Lions, Neil Sydney Trudinger, and John IE Urbas, *The neumann problem for equations of monge-ampère type*, Communications on pure and applied mathematics **39** (1986), no. 4, 539–563.
43. Grégoire Loeper and Francesca Rapetti, *Numerical solution of the monge-ampère equation by a newton's algorithm*, Comptes Rendus Mathematique **340** (2005), no. 4, 319 – 324.

44. Changna Lu, Weizhang Huang, and Erik S. Van Vleck, *The cutoff method for the numerical computation of nonnegative solutions of parabolic pdes with application to anisotropic diffusion and lubrication-type equations*, Journal of Computational Physics (2013), no. 0, –.
45. Robert E Lynch, John R Rice, and Donald H Thomas, *Direct solution of partial difference equations by tensor product methods*, Numerische Mathematik **6** (1964), no. 1, 185–199.
46. Andrew TT McRae, Colin J Cotter, and Chris J Budd, *Optimal-transport–based mesh adaptivity on the plane and sphere using finite elements*, SIAM Journal on Scientific Computing **40** (2018), no. 2, A1121–A1148.
47. Ahmed Ratnani, Said Hadjout, Yaman Guclu, and Emily Bourne, *Pyccel, python static compiler for high performance computing*, 2019.
48. Giancarlo Sangalli and Mattia Tani, *Isogeometric preconditioners based on fast solvers for the sylvester equation*, SIAM Journal on Scientific Computing **38** (2016), no. 6, A3644–A3671.
49. Deepti Shakti, Jugal Mohapatra, Pratibhamoy Das, and Jesus Vigo-Aguiar, *A moving mesh refinement based optimal accurate uniformly convergent computational method for a parabolic system of boundary layer originated reaction–diffusion problems with arbitrary small diffusion terms*, Journal of Computational and Applied Mathematics **404** (2022), 113167.
50. Mohamed Sulman, J.F. Williams, and R.D. Russell, *Optimal mass transport for higher dimensional adaptive grid generation*, Journal of Computational Physics **230** (2011), no. 9, 3302 – 3330.
51. Mohamed M. Sulman, J.F. Williams, and Robert D. Russell, *An efficient approach for the numerical solution of the monge-ampère equation*, Applied Numerical Mathematics **61** (2011), no. 3, 298 – 307.
52. Anna Tagliabue, Luca Dede, and Alfio Quarteroni, *Isogeometric analysis and error estimates for high order partial differential equations in fluid dynamics*, Computers & Fluids **102** (2014), 277–303.
53. Hilary Weller, Philip Browne, Chris Budd, and Mike Cullen, *Mesh adaptation on the sphere using optimal transport and the numerical solution of a monge–ampère type equation*, Journal of Computational Physics **308** (2016), 102–123.
54. Xuefei Yuan, Stephen C. Jardin, and David E. Keyes, *Moving grids for magnetic reconnection via newton-krylov methods*, Computer Physics Communications **182** (2011), no. 1, 173 – 176, Computer Physics Communications Special Edition for Conference on Computational Physics Kaohsiung, Taiwan, Dec 15-19, 2009.
55. ———, *Numerical simulation of four-field extended magnetohydrodynamics in dynamically adaptive curvilinear coordinates via newton-krylov-schwarz*, Journal of Computational Physics **231** (2012), no. 17, 5822 – 5853.
56. V. Zheligovsky, O. Podvigina, and U. Frisch, *The monge-ampère equation: Various forms and numerical solution*, Journal of Computational Physics **229** (2010), no. 13, 5043 – 5061.

Modeling, Simulation and Data Analysis, Mohammed VI Polytechnic University, Benguerir 43150, Lot 660, Hay Moulay Rachid Morocco
Current address: Modeling, Simulation and Data Analysis, Mohammed VI Polytechnic University, Benguerir 43150, Lot 660, Hay Moulay Rachid Morocco
Current address: Laboratoire J.A.Dieudonné, Université de Nice Sophia-Antipolis, France
Email address: mustapha.bahari@um6p.ma

Laboratoire J.A.Dieudonné, Université de Nice Sophia-Antipolis, France
Current address: Laboratoire J.A.Dieudonné, Université de Nice Sophia-Antipolis, France
Email address: abderrahmane.habbal@unice.fr

Modeling, Simulation and Data Analysis, Mohammed VI Polytechnic University, Benguerir 43150, Lot 660, Hay Moulay Rachid Morocco
Current address: Modeling, Simulation and Data Analysis, Mohammed VI Polytechnic University, Benguerir 43150, Lot 660, Hay Moulay Rachid Morocco
Email address: ahmed.ratnani@um6p.ma

Max Planck Institute for Plasma Physics, Boltzmannstr. 2, 85748 Garching, Germany
Current address: Max Planck Institute for Plasma Physics, Boltzmannstr. 2, 85748 Garching, Germany
Email address: Eric.Sonnendruecker@ipp.mpg.de